A Multiparadigm Intelligent Tutoring System for Robotic Arm Training

Philippe Fournier-Viger, Roger Nkambou, Engelbert Mephu Nguifo, André Mayers, and Usef Faghihi

Abstract—To assist learners during problem-solving activities, an intelligent tutoring system (ITS) has to be equipped with domain knowledge that can support appropriate tutoring services. Providing domain knowledge is usually done by adopting one of the following paradigms: building a cognitive model, specifying constraints, integrating an expert system, and using data mining algorithms to learn domain knowledge. However, for some ill-defined domains, each single paradigm may present some advantages and limitations in terms of the required resources for deploying it, and tutoring support that can be offered. To address this issue, we propose using a multiparadigm approach. In this paper, we explain how we have applied this idea in CanadarmTutor, an ITS for learning to operate the Canadarm2 robotic arm. To support tutoring services in this ill-defined domain, we have developed a multiparadigm model combining: 1) a cognitive model to cover well-defined parts of the task and spatial reasoning, 2) a data mining approach for automatically building a task model from user solutions for ill-defined parts of the task, and 3) a 3D path-planner to cover other parts of the task for which no user data are available. The multiparadigm version of CanadarmTutor allows providing a richer set of tutoring services than what could be offered with previous single paradigm versions of CanadarmTutor.

Index Terms—Computer-assisted instruction, intelligent tutoring systems, ill-defined domains, tutoring feedback

1 Introduction

To assist learners during problem-solving activities, an intelligent tutoring system (ITS) needs to be equipped with domain knowledge that can support appropriate tutoring services [1]. However, modeling the domain knowledge can be quite time-consuming and difficult, especially for *ill-defined domains* [2]. An ill-defined domain can generally be viewed as a domain that poses new challenges, i.e., where classical approaches for building tutoring systems are not applicable or do not work well [20]. Because many domains are ill-defined, there has been considerable research interest on building ITS for ill-defined domains in recent years [2], [20], [47]. The notion of ill-defined domains is partly ill-defined and various criteria have been proposed to characterize ill-defined domains [2], [20], [47], which will be reviewed in the next section. To provide domain knowledge to an ITS, three paradigms have been widely used in the ITS community. The first one is to build a cognitive task model such as the ones used in Cognitive Tutors [4], [5]. This is often done by observing expert and novice users (e.g., [4], [5]) to produce effective problem spaces or task models by hand. However, this process can be very time-consuming [5]. Furthermore, for ill-defined domains, it is not always possible to define a complete or partial task model by hand. The second paradigm is constraint-based modeling (CBM) [6], which consists of specifying sets of constraints on a correct behavior instead of providing a complete task description. Though this approach was shown to be effective for some ill-defined domains, it can be very challenging to design a complete set of constraints for some domains. The third paradigm consists of integrating an expert system into an ITS (e.g., [7], [8]). However, developing an expert system can be difficult, time-consuming, and expensive especially for ill-defined domains, and expert systems sometimes do not generate explanations in a form that is appropriate for learning. Recently, a fourth paradigm [9], [10] used data mining algorithms to automatically extract partial task models from recorded user demonstrations of the task. The partial task models can then be used to assist learners. Even though this approach was proven efficient in procedural ill-defined domains, the task models extracted are partial and are not useful for unseen situations.

To address these limitations of each paradigm, we assume that a good integration of these different paradigms could help to maximize the benefits associated with each of them, especially for ill-defined domains. To validate this hypothesis, we have developed a multiparadigm ITS named CanadarmTutor to train astronauts how to manipulate the Canadarm2 robot in various situations. In this ITS, we have integrated three paradigms: 1) using a cognitive model, 2) using an expert system, and 3) using a data mining approach. This allows providing tutoring services to learners that are much richer than what is offered in single-paradigm versions of CanadarmTutor.

- *P. Fournier-Viger is with the Department of Computer Science, University of Moncton, 18 Avenue Antonine-Maillet, Moncton, New Brunswick E1A 3E9, Canada. E-mail: philippe.fournier-viger@umoncton.ca.*
- *R. Nkambou is with the Department of Computer Science, University of Quebec in Montreal, 201 Avenue Président-Kennedy, Local PK-4150, Montreal, Quebec H2X 3Y7, Canada. E-mail: nkambou.roger@uqam.ca.*
- *E.M. Nguifo is with the Department of Computer Science, Clermont Université, Université Blaise-Pascal, LIMOS, F-63000 Clermont-Ferrand, 5 CNRS, UMR 6158, LIMOS, F-63173 Aubière. E-mail: engelbert.mephu_nguifo@univ-bpclermont.fr.*
- *A. Mayers is with the Department of Computer Science, University of Sherbrooke, 2500 Boulevard de l'Université, Sherbrooke, Québec J1K 2R1, Canada. E-mail: andre.mayers@usherbrooke.ca.*
- *U. Faghihi is with the Department of Computer Science, Sull Ross State University, Texas. E-mail: jfaghihi@yahoo.com.*

Manuscript received 13 Aug. 2012; revised 24 Apr. 2013; accepted 17 July 2013; published online 16 Aug. 2013.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2012-08-0110.
Digital Object Identifier no. 10.1109/TLT.2013.27.

This paper is organized as follows: Section 2 presents related work on ill-defined domains. Sections 3, 4, 5, and 6 introduce CanadarmTutor and the three paradigms we have implemented into it for representing the domain expertise. Section 7 explains how we have integrated them to build a multiparadigm version of CanadarmTutor. Section 8 presents a pilot study with the multiparadigm version of CanadarmTutor. Section 9 discusses the lessons learned and what could be generalized to other domains. Finally, Section 10 draws conclusions.

2 Related Works on Ill-Defined Domains

In this section, we briefly explain what is an ill-defined domain and the main types of solutions that have been adopted for building ITS for ill-defined domains.

2.1 What Is an Ill-Defined Domain?

The definition of what is an ill-defined domain is still under debate [17], [18], [19]. Generally, an ill-defined domain is one that poses new challenges, i.e., where classical approaches for building tutoring systems are not applicable or do not work well [20]. Note that a complex domain is not necessarily ill-defined. A domain is said to be ill-defined because of its structure or content that makes it less suitable for supporting tutoring services [20]. Examples of ill-defined domains are ethics, argumentation, essay writing, and the design of UML diagrams [2]. It is generally agreed that there is no clear boundary between ill-defined and well-defined domains. Instead, there is a continuum ranging from well-defined to ill-defined [2], [20], [21]. For identifying ill-defined domains, comparing domains on the basis of their "ill-definedness," or categorizing strategies and approaches for supporting tutoring services in these domains, researchers have identified characteristics of ill-defined domains. Lynch et al. [2] have performed a literature review and concluded that domains having one of the following characteristics are ill-defined:

1. domains having problems with many arguable solutions and no clear procedure for evaluating a solution,
2. domains that do not have a clear or complete domain theory for determining a problem's outcome and test its validity,
3. domains where tasks involve the design of new artifacts or the analysis of incomplete and potentially incorrect information about a changing environment for taking decisions,
4. domains incorporating abstract concepts that are partially indeterminate or do not have an absolute definition, and
5. domains including complex problems that cannot be divided into smaller independent subproblems that are easier to solve.

A definition of what is an ill-defined task was proposed by Simon [3] based on the study of human problem-solving and artificial problem-solvers. Simon states that a problem is ill-structured if it possesses one or more of the following features [3]: 1) Instructions or information necessary for solving the problem are incomplete or vague, 2) criterion that determines whether the goal has been attained is complex and imprecise, and 3) there are no clear strategies for finding solutions at each step of the problem-solving process. Recently, Le et al. [47] proposed a five-classes progressive scale to classify problems according to their "ill-definedness," from well-defined (Class 1) to ill-defined (Class 5). Class 1 are problems with a single solution. Class 2 are problems with a single problem-solving strategy but some implementation variants. Class 3 are problems with a known number of typical problem-solving strategies. Class 4 are problems with a great variety of strategies beyond the anticipation of a teacher and where solution correctness can be verified automatically. Finally, Class 5 are problems whose solution correctness cannot be verified automatically.

2.2 How to Build an ITS for Ill-Defined Domains

To build ITS in ill-defined domains, two main types of solutions have been adopted. The first type of solution is to adopt a suitable teaching model that will facilitate the conception of the ITS for the ill-defined domain. Several teaching models have been used and have achieved great success. For example, CATO [22], CATO-Dial [23], and GUIDON [7] are ITS where learning is structured around the study of cases. A second teaching model is to provide metacognitive support to learners. That is, to help learners being efficient during training sessions, while giving a limited support to learners for learning domain knowledge. An example is the ITS created by Walker et al. [24] for teaching intercultural competence skills. A third teaching model used in ITS for ill-defined domains is to support inquiry-learning, a constructivist approach to learning [1, p. 312]. An example of ITS for inquiry learning is Rashi [25], a generic ITS applied to several domains including forestry, history, and geology. A fourth teaching model is to offer "interactive narratives," where the learner is put in a story and has to take decision that will affect the outcomes (e.g., AEINS [26]). A fifth teaching model for ill-defined domains is to structure learning around collaboration. The goal is to make the student learn by working and exchanging information and ideas with peers. The benefit of this strategy for ill-defined domains is that the collaboration can replace the need to build a domain model, for example, Walker et al. [24] (collaboration with humans) and Chou et al. [27] (collaboration with virtual companions). More conventional teaching models can also be adopted (e.g., [29]).

The second type of solution to build an ITS operating in ill-defined domains consist of choosing an appropriate paradigm for supporting tutoring services. As mentioned in Section 1, there exists several paradigms for representing expertise in ITS. Each of them has advantages and disadvantages for ill-defined domains. We here review them in details.

The first paradigm is to build a cognitive model representing the various steps or solution paths to solve a problem(s). A cognitive task model can take various forms. For example, in Cognitive Tutors, task models are represented by rules corresponding to problem-solving steps [4], [5]. In Example-Tracing Tutors [44], task models are represented as state-spaces where states represent problem states and transitions represent problem-solving operations

[4]. The process of comparing task models with learner solutions to detect errors is called *model-tracing* [1], [5]. Producing a detailed cognitive model such as the one used in Cognitive Tutors is usually done by observing expert and novice users performing the task [4], [5] to capture different ways of solving problems. The advantage of this paradigm is that reasoning processes of learners can be represented with great details (in fact, some authors even claim to model cognitive processes of learners), and that the models obtained can support a wide variety of tutoring services such as 1) generating hints and demonstrations to a learner during problem-solving activities, 2) evaluating the learners' knowledge in terms of applied skills, and 3) inferring learner goals.

The second paradigm is constraint-based modeling [6]. It consists of specifying sets of constraints on what is a correct behavior or solution instead of providing an explicit task model. During a task, when a learner violates a constraint, a constraint-based tutor diagnoses that an error has been made, and can provide hints about the violated constraint. A limitation to this paradigm is that it does not support tutoring services such as the generation of demonstrations or the suggestion of next steps to perform to learners. Moreover, for some tasks, states/solutions are sometimes not informative enough to permit specifying a set of relevant constraints, or a very large number of constraints would be required, because there are too many dissimilar solutions [28]. Another limitation is that the constraint-based approach does not take into account the solution path leading to a constraint violation. This can have two negative implications. First, generated help can be inadequate, especially when a solution path largely differs from ideal solution(s) [1, p. 98]. Second, if the reasoning that led to a solution is not evaluated, a constraint-based tutor may not be able to distinguish that a learner answered a question correctly on purpose or by mistake.

The third paradigm consists of integrating an expert system in an ITS [7], [8], [11]. This approach is very broad, because many forms of expert systems can be used such as rule-based, neural networks, and case-based reasoning systems. The advantage of this approach is that tailored expert systems are especially well suited for some domains, unlike previous paradigms that are general approaches. There are two main and complimentary ways of using an expert system in an ITS. First, an expert system can be used to generate expert solutions. The ITS then compare these solutions with learner solutions, use them as demonstrations or for suggesting next problem-solving steps to learners (e.g., Guidon [7]). The second main way of using an expert system in an ITS is for comparing ideal solution(s) with learner solutions (e.g., AutoTutor [7], Mycin [8], and DesignFirst-ITS [29]). In several cases, the expert systems approach allows providing rich tutoring services that would be hard to offer with the previous paradigms. However, limitations of the expert system approach are that 1) developing or adapting an expert system can be costly and difficult, especially for ill-defined domains, and that 2) some expert systems cannot justify their inferences, or cannot provide explanations that are appropriate for learning.

The fourth paradigm is to apply data mining or machine learning techniques for automatically learning partial task models from user solutions [9], [10]. The rationale of this approach is that a partial task model can be a satisfying substitute to an exhaustive task model for domains where a task model is hard to define. The approach of learning partial task models is appropriate for problem-solving tasks where the initial state and the goal state are clear, there are a large number of possibilities, no clear strategy for finding the best solution, and solution paths can be expressed as sequences of actions. A benefit of the approach is that it does not require specifying any background knowledge by a domain expert, and the system can enrich its knowledge base with each new solution. In addition, unlike the expert system approach, it is based on human solutions. However, no help can be offered to learners if the solution path was previously completely unexplored [10]. Based on the observation that each paradigm has some limitations in ill-defined domains, we assume that combining several paradigms would provide better results. In the next sections, we explain how we have implemented a multi-paradigm version of CanadarmTutor. The hypothesis is that a good integration of several paradigms could help to maximize the benefits associated with each of them for ill-defined domains. We first present CanadarmTutor and explain why it operates in an ill-defined domain. We then describe how we have applied three paradigms separately and how they are now integrated.

3 The CanadarmTutor Tutoring System

CanadarmTutor [11], [12], [43] (see Fig. 1) is a simulation-based tutoring system to train astronauts how to operate Canadarm2 (see Fig. 2), a 7-degrees-of-freedom robotic arm deployed on the International Space Station (ISS). CanadarmTutor offers a 3D simulation of the arm and the space station, which let learners to operate the arm in a safe environment. The main learning activity in CanadarmTutor is to move the arm from an initial configuration to a goal configuration by operating the arm. During an exercise, CanadarmTutor acts as a virtual coach in a manner that is similar to human coaches at the Canadian Space Agency [43]. CanadarmTutor offers the following tutoring services (described in more details in the next sections):

1. assessing a learner solution to see if he is following a correct solution path,
2. providing hints and proactive help to guide the learner in a correct solution path,
3. generating demonstrations to show the learner how to operate the arm,
4. assessing the skills of a learner, and
5. generating personalized problems based on the skill assessment.

Besides the main learning activity of arm manipulation, CanadarmTutor also offers an interactive questionnaire named the "Space Quiz" to learn declarative knowledge about the space station and how to operate the arm [12].

The CanadarmTutor interface (see Fig. 2) reproduces parts of Canadarm2's control panel (see Fig. 3). It has three monitors. The interface buttons and scroll wheels

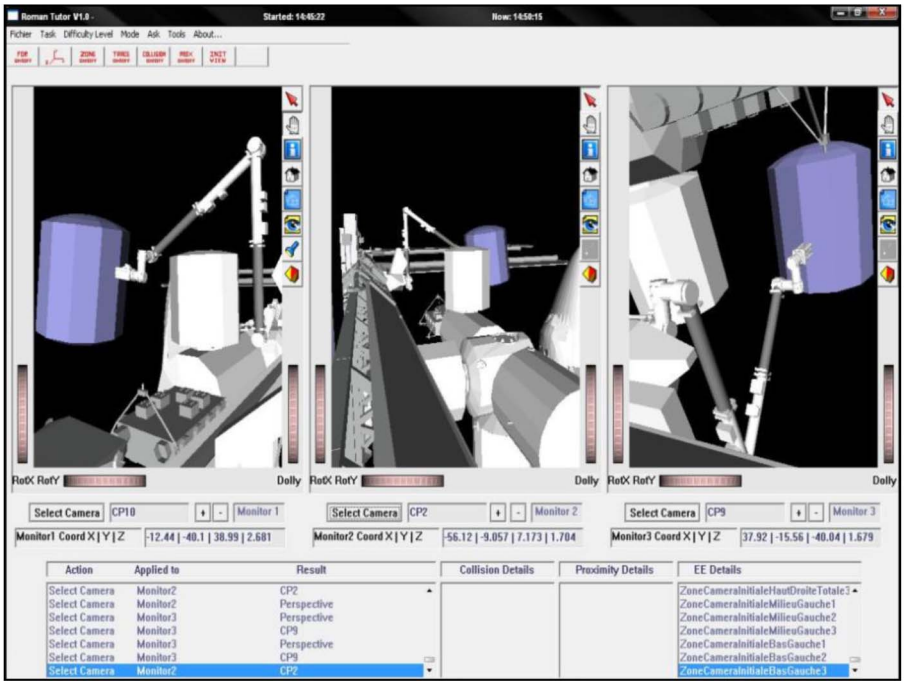

Fig. 1. The CanadarmTutor interface.

(see Fig. 2) allow users to associate a camera with each monitor and adjust the zoom, pan, and tilt of the selected cameras. The arm is controlled with a keyboard in reverse kinematics or joint-by-joint mode. The text fields at the bottom part of the window display the state of the simulator. The menus make it possible to set preferences and request tutor feedback and demonstrations.

Operating Canadarm2 is difficult. The reason is that arm movements are performed by astronauts inside the ISS, with a limited view of the environment. The environment is rendered through three monitors (see Fig. 3), each showing the view obtained from a single camera while about 10 cameras are mounted at different locations on the ISS and on the arm. To move the arm, the operator must select at every moment the best cameras for viewing the scene of operation. Moreover, an operator has to select and perform appropriate joint rotations for moving the arm, while avoiding collisions and dangerous configurations. Operators also have to follow a security protocol that comprises numerous steps because a single mistake, such as neglecting to lock the arm into position can lead to catastrophic and costly consequences. The task of manipulating Canadarm2 can be considered ill-defined. We here explain why by reviewing criteria from Section 2 [47]. A first criterion is that the initial state is unclear (criteria 1 by Simon). In CanadarmTutor, the main exercise is to move the arm from one configuration to another by operating the arm. The initial state is clear. It is the initial configuration of the robotic arm.

A second criterion is that the goal state is unclear or that there is no procedure for evaluating if the goal has been attained. In CanadarmTutor, the goal state is clear. It is to bring a payload attached to Canadarm2 in a colored cube, displayed in the 3D environment. Verifying if the goal state is met consists of checking if the payload is in the cube and if faults have been committed such as hitting the ISS, while moving the arm.

A third criterion is that the task requires creativity. This is not the case of CanadarmTutor because the user is not asked to create new artifacts.

A fourth criterion is that there is no clear strategy at each problem-solving step to move from the initial state to the goal state. In CanadarmTutor, there are some clear

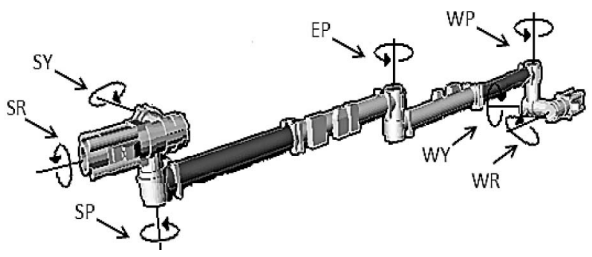

Fig. 2. Three-dimensional representation of Canadarm2 illustrating its seven joints.

Fig. 3. Astronaut L. Chiao operating Canadarm2 (courtesy of NASA).

metasteps that an astronaut has to follow to operate the arm. For instance, if an astronaut adjusts the cameras, he must always set the middle, left, and right monitors, in that order. Another example is that the brakes must be removed before moving the arm and be put back after. However, for selecting the joint rotations to move the arm to the goal, there is no clear procedure. To select an appropriate sequence of joint rotations for moving the arm to the goal state, one could think that the best sequence of joint rotations is the shortest in terms of the number of rotations or distance. However, this is generally not the case. The main criterion to evaluate a sequence of joint rotations is safety. But what is a safe sequence? It depends on several criteria that are hard to formalize. First, one should consider the familiarity of the user with the movements. Second, one should take into account the context in terms of camera views, the relative positions of the obstacles to the arm, and zones/configurations that are considered desirable. Third, one should consider the risks related to the surrounding environment and the arm itself with respect to movements. For instance, one should stay away from configurations where the arm can get jammed (singularities). Fourth, the cognitive load and effort required for performing the movements should be assessed because it has a direct impact on the risk of accident. For instance, sequences of movements that require too much concentration or dexterity, or are too difficult in terms of spatial skills (mental rotations, etc.) should be avoided. For the above reasons, the skills to operate the arm are mainly learned by practice, and experts often disagree on what is the best sequence of arm movements. Note that the lack of formal procedure for selecting the joint rotations does not mean that it is impossible to define one. However, the current state of the knowledge in this domain is that there is none and it would be difficult and costly to try to define one. For this reason, we can assert that according to the current knowledge, the task of operating Canadarm2 is ill-defined. More precisely, we can say that how to select joint rotations is ill-defined and that other parts of the task are well defined (the metasteps previously mentioned).

A fifth criterion is that the domain includes abstract concepts that are partially indeterminate or do not have an absolute definition. CanadarmTutor meet this criterion. Some concepts are abstract such as the safety, effort, and difficulty of a sequence of joint rotations.

A sixth criterion is that problems are complex and cannot be divided into smaller independent problems. CanadarmTutor meet this criterion for the subtask of joint selections. As mentioned previously, there are several aspects that need to be considered to select an appropriate sequence of joint rotations and one cannot consider these aspects independently.

A seventh criterion is that there is a very large number of possible solutions. Operating Canadarm2 meet this criterion as there are a huge number of possibilities for solving each problem. Although this criterion does not guarantee that a domain is ill-defined, it poses an additional challenge, which is that it would be impossible to specify all the possibilities for a problem by hand.

Finally, the task of operating Canadarm2 is classified by Le et al. [47] as a representative example of Class 4 problems in their five-classes scale. Class 4 are problems with a great variety of problem-solving strategies beyond the anticipation of a teacher where solution correctness can be verified automatically. This definition agrees with the analysis of the task that we have presented in previous paragraphs. It is interesting to note that Class 4 is the second most difficult class of the five-classes scale. In Class 5, the focus moves from supporting learning domain knowledge to mainly supporting learners at being efficient during training sessions [47].

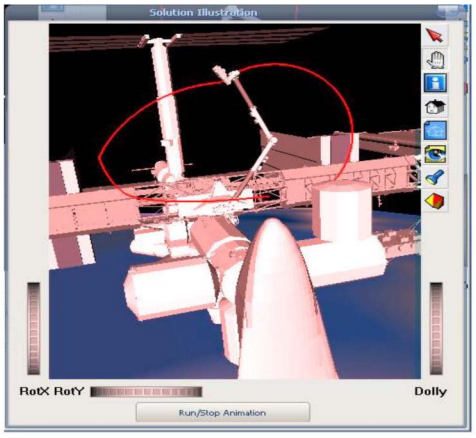

Fig. 4. A demonstration generated by the path-planner.

In the next sections, we explained how we have applied three different paradigms to provide tutoring services in CanadarmTutor. We then present their integration in a multiparadigm version of CanadarmTutor.

4 Integrating a Path-Planner for Automatic Path Generation

To implement the domain expertise in CanadarmTutor, we first applied the expert system paradigm. A custom path-planner named FADPRM was developed and integrated into CanadarmTutor (see [11] for full details).

FADPRM is an efficient algorithm for robot path-planning in constrained-based environments. It lets the user specify different zones in the environment with arbitrary geometrical forms. A degree of desirability *dd* can be assigned to each zone (a real in [0, 1]). A *dd* value of 1 means a highly desirable zone while a *dd* of 0 means a zone that should be avoided at all cost. In CanadarmTutor, the number, the form, and the placement of zones reflect the disposition of cameras and zones that are considered safer. For example, a zone that is not visible by any camera will be considered as a nondesired zone with a *dd* near 0 and will take an arbitrary polygonal shape. The FADPRM algorithm uses a probabilistic roadmap approach to calculate a trajectory (see Fig. 4) between any two robotic arm configurations while avoiding obstacles and considering constraints such as obstacles and dangerous and desirable zones. FADPRM works as follows: Based on the probabilistic roadmap approach for path-planning (PRM) [45], it builds a roadmap by choosing robot configurations probabilistically, with a probability that is biased by the density of obstacles. A path is then a sequence of collision free edges in the roadmap, connecting the initial and goal

configurations. Furthermore, FADPRM also integrates characteristics of the Anytime Dynamic A* (AD*) approach for path-planning [46], to get new paths when the conditions defining the environment have dynamically changed, FADPRM can quickly plan a new path by exploiting a previous roadmap. Moreover, for efficiency purposes, paths are computed through incremental improvements so that the planner can be called at anytime to provide a collision-free path and the more time it is given, the better the path optimizes moves through desirable zones. Therefore, the planner is a combination of the traditional PRM approach [45] and AD* [46] and it is flexible in that it takes into account zones with degrees of desirability. This explains why it is called flexible anytime dynamic PRM (FADPRM).

Integrating FADPRM in CanadarmTutor provides the following benefits. First, in a training session, CanadarmTutor uses FADPRM to automatically produce demonstrations of correct arm maneuver on the ISS by generating safe path(s) between two arm configurations, while considering the obstacles (the ISS modules), cameras and other constraints. Second, for a given task, CanadarmTutor automatically generates paths and estimates the distance with the learner solution to evaluate it.

Although the path-planner can provide useful tutoring services, our experiments with learners show that the generated paths are not always realistic, as they are not based on human experience. Moreover, they do not cover some important aspects of the task such as selecting cameras and adjusting their parameters. Furthermore, given that the path-planner has no representation of knowledge and skills, it cannot support important tutoring services such as estimating learners' knowledge gaps.

5 Integrating a Cognitive Model to Assess Skills and Spatial Reasoning

Facing these problems, we applied the paradigm of building a cognitive model [12]. To understand how astronauts operate Canadarm2, we attended two-week training at the Canadian Space Agency and interviewed the training staff. To encode how users operate the robotic arm, we used a custom rule-based cognitive model [12], which is similar to what is used in CTAT [1] and the Cognitive Tutors [5], which are the reference cognitive models in the field of ITS. The main difference between our model and the one used in the Cognitive Tutors is that ours is designed to also evaluate spatial reasoning, because it is a key issue for manipulating Canadarm2.

5.1 Literature Review on Spatial Cognition

To take into account the spatial dimension, we performed an extensive literature review on spatial cognition. We found that most researchers in psychology and neurosciences agree that there are two main types of spatial knowledge [14], [15], [30]. *Egocentric representations* describe the position of objects from a person's perspective, whereas *allocentric representations* represent spatial relationship(s) between objects independently of any point of view. For example, in the context of route navigation, egocentric representations describe relative landmarks along a route to follow, whereas an allocentric representation could be a map describing the relative position of landmarks to each other [15]. According to Tversky [15], egocentric representations are sufficient to perform tasks such as navigating through an environment, but they are inadequate to perform complex spatial reasoning. For reasoning that requires inference, humans build allocentric representations that do not preserve measurements, but instead retain the main relationships between elements. Such representations do not encode a single perspective, yet they make it possible to adopt several perspectives. This allows performing complex spatial reasoning. Strong evidences in neurosciences support the distinction between allocentric egocentric representations. That is when the human being performs spatial task different regions of the brain activates between [14], [15], [30]. For example, "place cells" have been discovered in human hippocampi [31], and head-direction cells [32], [33], speed modulated place cells [34], and cells with periodic place fields [35] in rats.

Because allocentric representations are crucial for complex reasoning, we, therefore, wanted to assess these representations in CanadarmTutor. The goal is to see if the learner can not only use egocentric representation but also build and use allocentric representations. To see how we could represent allocentric representations in a computational model, we reviewed computational models in spatial cognition research. We have found that most of the models replicate a specific phenomenon of spatial cognition, such as visual perception and motion recognition [36], navigation in 3D environments [31], [37] and mental imagery and inference from spatial descriptions [38]. Few models attempt to provide a more general explanation of spatial reasoning [39]. In our case, we are interested in a general model of reasoning to not only evaluate spatial reasoning and representations but also other necessary skills to operate Canadarm2, in an integrated way. Furthermore, several existing models of spatial cognition are neural network-based (e.g., [40]). We dismissed these latter because knowledge has to be explicit to support detailed feedback to the learner in ITS. We, therefore, based our work on symbolic models of spatial cognitions [36], [37], [38], [39]. Those latter represents allocentric representations as relations of the form "*a r b*," where "*a*" and "*b*" are symbols designating objects and "*r*," a spatial relationship between the objects [36], [37], [38]. But which cognitive model should we use in CanadarmTutor to evaluate spatial reasoning and reasoning in general? We had two options: 1) adapt a model of spatial cognition to be used in ITS or 2) adapt a cognitive model used in ITS to also evaluate spatial reasoning. We chose the second option because models of spatial cognition in cognitive science are not designed to be used in ITS and would require major changes to be used that way.

5.2 The Cognitive Model

We, therefore, based our work on a rule-based cognitive model that we had developed in previous works. We here only give the main idea of this model and how it has been adapted. The interested reader can refer to a complete article on this cognitive model for full details [12]. The cognitive model is based on the ACT-R [41] and MIACE [42] cognitive theories. It allows describing how to perform a procedural

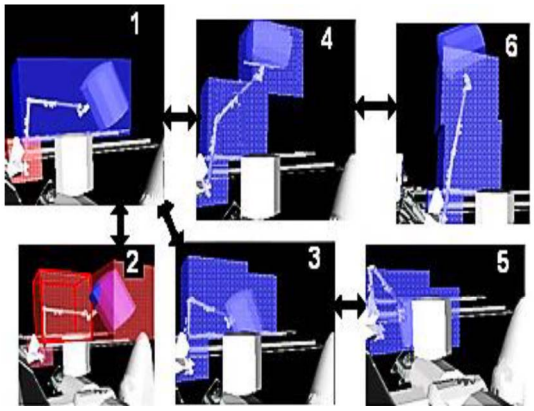

Fig. 5. Six elementary spaces.

task by specifying goals, rules for satisfying goals and how the declarative knowledge (facts) is manipulated by the rules to accomplish goals. A rule represents the procedural knowledge of how to achieve a goal. A rule has parameters and some rules have subgoals. Several rules can be used to solve the same goal (representing different correct or erroneous ways of achieving the goal). This system allows performing model-tracing like in the Cognitive Tutors to detect procedural errors made by learners. Moreover, it also includes an extension to the model-tracing approach to indirectly assess the declarative knowledge that is recalled and in particular the spatial knowledge [12].

5.3 Applying the Cognitive Model in CanadarmTutor

To model the spatial knowledge with the aforementioned model, we discretized the 3D space into 3D subspaces that we name elementary spaces (ESP). This allows us to represent the continuous space as discrete symbols. In Canadarm2 manipulation, it was determined that the most realistic types of ESP for mental processing are ESPs configured with an arm shape. Fig. 5 illustrates 6 of the 30 ESPs that we defined. For example, one can move the arm from ESP 1 to ESP 2, ESP 3, and ESP 4. ESP 5 can be reached from ESP 3, and ES6 can be reached from ES4. Each ESP is represented by three cubes. Spatial knowledge was then encoded as four types of relationships of the form "*a r b*" such as

1. a camera can see an ESP or an ISS module,
2. an ESP contains an ISS module,
3. an ESP is next to another ESP, and
4. a camera is attached to an ISS module.

We then modeled the procedural knowledge of how to move the arm to a goal configuration as a set of rules. These rules perform a loop where the learner, before any arm movements, must recall a set of cameras for viewing the ESPs containing the arm, select the correct cameras, adjust their parameters in the correct order (pan, tilt, and then zoom), retrieve a sequence of ESPs to go from the current ESP to the goal, and then start moving the arm to the next ESP [12]. An example of rule is the rule of adjusting cameras in the correct order. It is named "PAdjustCameras," takes as parameter the current ESP and it consists of three subgoals ordered as follows: GAdjustCenterCamera, GAdjustLeftCamera, and GAdjustRightCamera, which takes the current ESP as parameter.

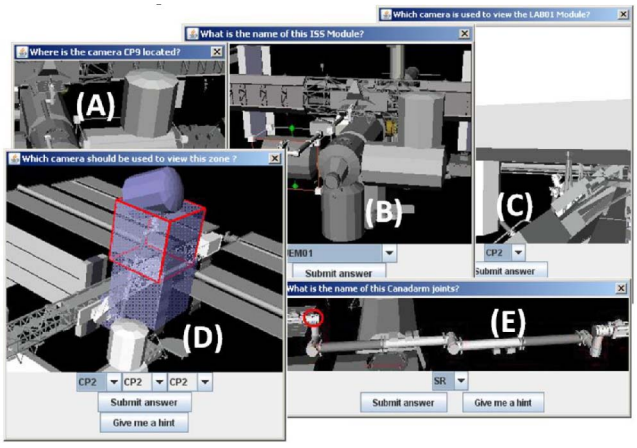

Fig. 6. Generated questions about declarative knowledge.

The task model allowed us to integrate six new tutoring services in CanadarmTutor [12]. First, a learner can explore the task model to learn how to operate the arm and learn about the properties of the ISS, the cameras and Canadarm2.

Second, model-tracing capability allows the system to evaluate the learner's knowledge during arm manipulation exercises. After a few exercises, CanadarmTutor automatically builds a detailed learner profile that shows the strength and weakness of the learner in terms of mastered, missing and buggy knowledge. This is done by comparing the task model with a learner solution to see which knowledge (rules and declarative knowledge) is used by the learner and which one is not.

Third, CanadarmTutor uses the declarative knowledge linked to the task model to generate and provide the learner with direct questions. These questions are parts of a questionnaire named the "Space Quiz" that the learner can take to test his declarative knowledge, or can be asked by CanadarmTutor. Several types of questions can be generated from the declarative knowledge. The four main types are illustrated in Fig. 6. Fig. 6a asks where a given camera is located. Fig. 6b asks to name an ISS module. Fig. 6c asks which camera was used to obtain a given view. Fig. 6d asks to name a given joint of Canadarm 2.

The fourth tutoring service is to assist the learners by providing useful hints and demonstrations during arm manipulation exercises. Suggesting the next step and generating demonstrations is done thanks to the model-tracing capability of this paradigm.

The fifth tutoring service is to generate personalized exercises based on the student model. By using the student model, CanadarmTutor can generate exercises that involve knowledge not yet mastered by the learner. For example, if a learner had not shown that he can use a given camera, CanadarmTutor can generate an exercise that requires to use this camera.

The sixth and last tutoring service is to offer proactive help to the learner. For instance, if Canadarm2 is moved without performing camera adjustment, CanadarmTutor warns the learner to check if cameras are well adjusted. This type of help is also implemented based on model-tracing. It is particularly appreciated by beginners and intermediate learners.

TABLE 1
An Example Toy Database Containing Six User Solutions

ID	Dimensions					Sequence of actions
	Solution state	Expertise	Skill_1	Skill_2	Skill_3	
S1	successful	novice	no	yes	no	<(0,a),(1,bc)>
S2	successful	expert	yes	yes	yes	<(0,d) >
S3	successful	novice	no	yes	yes	<(0,a),(1,bc)>
S4	buggy	intermediate	no	yes	yes	<(0,a),(1,b), (2,d)>
S5	buggy	novice	no	yes	yes	<(0,d), (1,c)>
S6	successful	expert	yes	yes	yes	<(0,c), (1,d)

However, this cognitive model has some limitations. Although it models the main steps of the manipulation task in detail, it does not go into detail about how to select joint rotations for moving Canadarm2. The reason is that for a given arm movement exercise, there does not exist a clear and complete formal procedure to decide how to select the best joint rotations (as discussed in Section 2). Choosing one of the possibilities requires considering several abstract criteria such as the safety and ease of maneuvers (as discussed in Section 3). It is, thus, not possible to define a complete and explicit task model for selecting appropriate joint rotations, which makes this part of the Canadarm2 manipulation ill-defined. Moreover, because there are a huge number of different possibilities, it would not be feasible to just define all the possibilities by hand. The path-planner described in Section 4 can generate paths to provide complementary help at the level of joint rotations. But these paths are sometimes too complex and difficult to be executed by users, as they are not based on human solutions.

6 Using Data Mining Techniques to Learn Partial Task Models

Given the aforementioned drawbacks with other paradigms, we applied the fourth paradigm, which is to apply data mining techniques for automatically learning partial task models from user solutions [10]. It consists of applying data mining algorithms on user solutions to automatically extract a partial task model instead of defining it by hand. The goal is to provide tutoring services for parts of the task of operating the arm that are ill-defined and could not be represented easily with the cognitive model (e.g., how to select the joint rotations to move Canadarm2). An advantage of this approach over the path-planner (see Section 4) is that the data mining approach is based on real user data. Note that the approach that we have developed according to this paradigm is inspired by work on learning domain knowledge by demonstration in ITS for well-defined domains (e.g., [9], [58], [59]). But, to our knowledge it is the first approach to learn domain knowledge for an ill-defined domain [10]. What makes our approach applicable to the ill-defined domain of CanadarmTutor is that it does not attempt to generate an exhaustive model, but rather uses a scalable algorithm to generate a partial model by discovering patterns in user solutions [10]. Our approach is applied in three steps.

6.1 Recording User Solutions

The first step is to record a set of user solutions for each exercise [10]. In CanadarmTutor, an exercise consists of moving the robotic arm from an initial configuration to a goal configuration. For each attempt, a *sequence of actions* is created in a database. We defined 112 actions that can be recorded including 1) applying a rotation value to one of the seven arm joints 2) selecting a camera, and 3) performing an increase or decrease of the pan/tilt/zoom of a camera. An example of a partial action sequence recorded for a user in CanadarmTutor is <(0, *rotateSP*{2}), (1, *selectCP3*), (2, *panCP2*{4}), (3, *zoomCP2*{2})> which represents decreasing the rotation value of joint SP by two units, selecting camera CP3, increasing the pan of camera CP2 by four units and then its zoom by two units. Furthermore, we annotated sequences with contextual information called "dimensions." Table 1 shows an example of a toy database containing six solutions annotated with five dimensions. In this Table, letters *a*, *b*, *c*, and *d* denote actions. The dimension "*Solution state*" indicates if the learner solution was successful. Values for this dimension are assigned automatically by CanadarmTutor by checking if the arm has entered the goal state. The four other dimensions represent the user profile. Adding these dimensions is not required but it makes the approach tailored to learner profiles. In this example, we show four such dimensions. "Expertise" denotes the expertise level of the learner who performed a sequence. "Skill_1," "Skill_2," and "Skill_3" indicate whether any of these three specific skills were demonstrated by the learner when solving the problem. This example shows five dimensions. However, any kind of contextual information can be encoded as dimensions. In CanadarmTutor, we used 10 skills that are the most important according to our observations at the Canadian Space Agency, and the "solution state" and "expertise level" dimensions to annotate sequences.

6.2 Generating a Partial-Task Model

In the second step, we apply a data mining algorithm to extract a partial problem space from user solutions. To perform this extraction, an appropriate method is needed that considers many factors associated with the specific conditions in which a tutoring system such as CanadarmTutor operates. These factors include the temporal dimension of events, actions with parameters, the user's profile, etc. All these factors suggest that we need a temporal pattern mining technique. According to Han and Kamber [49], there are four main kinds of patterns that

TABLE 2
Some Patterns Extracted from the Data Set of Table 1 with Minsup $= 2$

ID	Dimensions					Sequence of actions	Support
	Solution state	Expertise	Skill_1	Skill_2	Skill_3		
P1	successful	novice	no	yes	*	<(0,a)>	2
P2	*	*	*	yes	yes	<(0,a)>	2
P3	*	*	no	yes	*	<(0,a), (1,b)>	3
P4	successful	expert	yes	yes	yes	<(0,d)>	2

can be discovered in sequential data. These are trends, similar sequences, sequential patterns, and periodic patterns. We chose to mine sequential patterns, as we are interested in finding relationships between occurrences of events in users' solutions. To mine sequential patterns, several efficient algorithms have been proposed. These have been previously applied, for example, to analyze earthquake data [50] and source code in software engineering [51]. While traditional sequential pattern mining algorithms (SPM) have as their only goal to discover sequential patterns that occur frequently in several sequences of a database [50], [51], other algorithms have proposed numerous extensions to the problem of sequential pattern mining such as mining patterns respecting time-constraints [50], mining compact representations of patterns [51], [53], [54], and incremental mining of patterns [51]. For this work, we developed a custom sequential pattern mining algorithm [10] that combines several features from other algorithms such as accepting time constraints [50], processing sequence databases with dimensions [54], [55], mining a compact representation of all patterns [53], [54], and that also adds some original features such as accepting symbols with parameter values [10]. Our algorithm's implementation can be downloaded freely as part of the open-source SPMF data mining software (http://www.philippe-fournier-viger.com/spmf/). The algorithm takes as input a sequence database and efficiently finds all sequential patterns. A sequential pattern is a subsequence that is common to at least *minsup* sequences, where *minsup* is a parameter of the algorithm. Table 2 shows some sequential patterns found from the database shown in Table 1 with $minsup = 2$. Consider pattern P3. This pattern represents doing action b one time unit (immediately) after action a. P3 appears in sequences S1, S3, and S4 of Table 1. It has, thus, a *support* (frequency) of three. Moreover, the annotations for P3 tell us that this pattern was performed by users with varied expertise level who do not possess Skill_1 but possess Skills_2 and that P3 was found in plan(s) that failed, as well as plan(s) that succeeded. Note that actions in a sequential pattern do not need to appear contiguously in a sequence. For example, the pattern <(0, a), (2, d)> would be considered as appearing in sequence <(0, a),(1 ,b), (2, d)> even if there is a some actions between (0, a) and (2, d). The reason for allowing noncontiguous sequential pattern is to eliminate "noisy" (nonfrequent) learners' actions. In our algorithm, we offer parameters to set the minimum and maximum size of the gap between actions in a sequential pattern. In CanadarmTutor, we have observed that setting the minimum gap to 0 and the maximum gap to 2 provided the best results. The benefits of accepting a gap of two is that it eliminates some noisy actions, but at the same time it does not allow larger gap sizes that could make patterns less useful for tracking a learner's actions.

Another important consideration is that when applying sequential pattern mining, there can be many redundant sequential patterns found (patterns included in other patterns having the same support). To eliminate this redundancy, we have adapted our algorithm to mine only *frequent closed sequential patterns* based on the work of Wang et al. [53] and Songram et al. [54]. Closed sequential patterns are patterns that are not contained in another sequential pattern having the same support. Mining closed sequential patterns has the advantage of greatly reducing the number of patterns found, without information loss (the set of closed sequential patterns allows reconstituting the set of all sequential patterns and their frequency) [53]. Finally, note that our algorithm was also extended to handle actions having parameters. This allows representing actions such as rotating a joint of the arm by 20 degrees. The interested reader can refer to [10] for full details about this extension.

6.3 Supporting Tutoring Services with the Partial-Task Model

We then implemented three tutoring services in Canadarm-Tutor that use the partial task models. The basic operation that is used for providing assistance is to recognize a learner's plan. In CanadarmTutor, this is achieved by the plan recognition algorithm presented in Fig. 7. This algorithm named *RecognizePlan* is executed after each student action. It takes as input the sequence of actions performed by the student (*Student_trace)* for the current problem and a set of sequential patterns (*Patterns*). When the plan recognition algorithm is called for the first time, the variable *Patterns* is initialized with the whole set of patterns *Patterns* to note all the patterns that include *Student_trace*. If no pattern is found, the algorithm removes the last action performed by the learner from *Student_trace* and searches again for matching patterns. This is repeated

```
RecognizePlan(Student_trace, Patterns)
    Result := Ø.
    FOR each pattern P of Patterns
    IF Student_trace is included in P
        Result = Result ∪ {P}.
        IF Result = Ø AND length(Student_trace) ≥ 2
            Remove last action of Student_trace.
            Result:= RecognizePlan(Student_trace, Patterns).
        Return Result.
```

Fig. 7. The plan recognition algorithm.

until the set of matching patterns is not empty or the length of *Student_trace* is smaller than 2. In our tests, removing user actions has improved the capability of the plan recognition algorithm to track learner's patterns considerably, as it makes the algorithm more flexible. The next time *RecognizePlan* is called, it will be called with the new *Student_trace* sequence and the set of matching patterns found by the last execution of *RecognizePlan*, or the whole set of patterns if none matched.

After performing preliminary tests with the plan recognition algorithm, we noticed that in general, after more than six actions performed by a learner, it becomes hard for *RecognizePlan* to tell which pattern the learner is following. For that reason, we made improvements to how CanadarmTutor applies the sequential pattern mining algorithm to extract a knowledge base. Originally, it mined frequent patterns from sequences of user actions for a whole problem-solving exercise. We modified our approach to add the notion of "*problem states*." In the context of CanadarmTutor, where an exercise consists of moving a robotic arm to attain a specific arm configuration, the 3D space was divided into 3D cubes, and the problem state at a given moment is defined as the set of 3D cubes containing the arm joints. An exercise is then viewed as going from a problem state P_1 to a problem state P_f. For each attempt at solving the exercise, CanadarmTutor logs 1) the sequence of problem states visited by the learner $A = P_1, P_2, \ldots, P_n$ and 2) the list of actions performed by the learner to go from each problem state to the next visited problem state (P_1 *to* P_2, P_2 *to* $P_3, \ldots,$ P_{n-1} *to* P_n). After many users perform the same exercise, CanadarmTutor extracts sequential patterns from 1) sequences of problem states visited, and from 2) sequences of actions performed for going from a problem state to another. To take advantage of the added notion of problem states, we modified *RecognizePlan* so that at every moment only the patterns performed in the current problem state are considered. To do so, every time the problem-state changes, *RecognizePlan* will be called with the set of patterns associated with the new problem state. Moreover, at a coarser grain level tracking the problem states visited by the learners is also achieved by calling *RecognizePlan*. This allows connecting patterns for different problem states. We describe next the main tutoring services that a tutoring agent can provide based on the plan recognition algorithm.

First, CanadarmTutor can assess the profile of the learner (expertise level, skills, etc.) by looking at the applied patterns. For example, if a learner applies patterns with the value "intermediate" for the dimension "expertise" 80 percent of the time, then CanadarmTutor asserts that the learner expertise level is "intermediate." In the same way, CanadarmTutor can diagnose mastered and missing/buggy skills for users who demonstrated a pattern by looking at the "skills" dimensions of the applied patterns.

The second tutoring service consists in determining the possible actions from the set of patterns and then, if needed, proposing one or more actions to the learner. In CanadarmTutor, this functionality is triggered when the learner selects "What should I do next?" in the interface menu. CanadarmTutor then checks the matching patterns to make a recommendation to the learner. For example, if the learner performed a rotation of the joint SP followed by a rotation of the joint EP and ask "What should I do next?", CanadarmTutor will look for pattern(s) that match with SP, EP to suggest what next action(s) the learner should do.

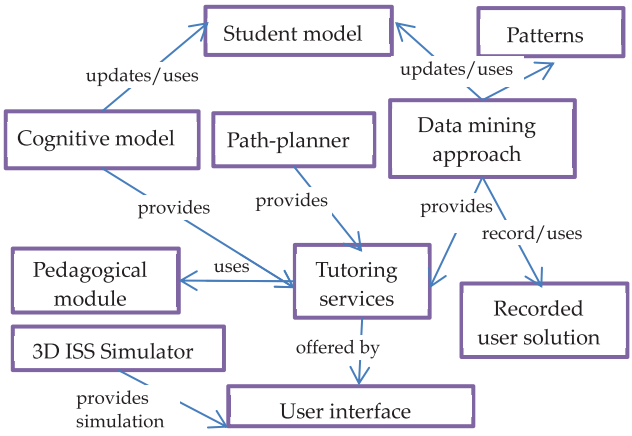

Fig. 8. The multiparadigm version of CanadarmTutor's architecture.

The third tutoring service is to let learners explore patterns by themselves to find out about ways to solve problems. CanadarmTutor provides an interface that lists the patterns and their annotations, and provides sorting and filtering functions (e.g., to display only patterns leading to success by intermediate users). This tutoring service is inspired by the idea of open-learner models used in other ITS [57].

The paradigm of learning partial task models from user solutions has several advantages. Unlike the path-planner (see Section 4), it allows us to provide tutoring services based on real users' arm manipulations (multiple user profiles). Moreover, it allows us to assist learners about how to choose a joint rotation—which was impossible to achieve with the cognitive model (see Section 5). However, an important limitation with the partial task model paradigm is that no help can be offered to learners for unexplored solution paths. Thus, each of the three paradigms that we have separately tested into CanadarmTutor has its own advantages and limitations. Based on this observation, we decided to combine them to create a multiparadigm version of CanadarmTutor.

7 Combining the Three Paradigms

The goal of the multiparadigm model is to combine the paradigms to take advantages of each one's strength.

7.1 Architecture of the Multiparadigm Version of CanadarmTutor

Fig. 8 illustrates the architecture of the multiparadigm version of CanadarmTutor. The basic idea of this new architecture is the following. The cognitive model, path-planner and data mining approach collaborate to support tutoring services. The cognitive model and the data mining approach use the same student model to provide tutoring services and update it. The data mining approach uses recorded user solutions to generate patterns and uses the patterns to support tutoring services. The tutoring services are regulated by a module called "pedagogical module," which determines how tutoring services interact with learners from a pedagogical perspective (see [43] for details

about how this module interacts with users). Finally, the last components of the architecture are the 3D simulator and the user interface.

7.2 The Multiparadigm Tutoring Services

We now describe how the tutoring services of the three paradigms are integrated. Some tutoring services can be offered by more than one paradigm while others can only be offered by a single paradigm. To determine what to do in case of conflicts between paradigms, we defined two principles. *Principle 1: If multiple paradigms can provide the same tutoring service in a given situation, use the first applicable paradigm according to this order: cognitive model, data mining approach and path-planner.* We give priority to the cognitive model over the data mining approach because the cognitive model is very detailed and has been built by hand based on extensive observation of real users. It should, thus, provide more relevant tutoring feedback than the data mining approach, which is approximate and automatic. We give priority to the data mining approach over the path-planner because the data mining approach is based on human solutions while the path-planner is not. For this reason, the data mining approach should generally provide more realistic feedback than the path-planner. *Principle 2: If multiple paradigms can offer complementary tutoring services in a given situation, the information provided by the tutoring services can be aggregated and then presented to the learner.*

We now explain how the multiparadigm version operates. During arm manipulation exercises, CanadarmTutor performs model-tracing with the cognitive model to update the student model. The student model is a list of knowledge units from the cognitive model. Each unit is annotated with a probability that indicates if the knowledge is mastered by the learner or not. Moreover, the student model is also updated when a learner answers questions asked by CanadarmTutor (see Section 5).

When an exercise is completed (fail or success), the solution is added to the sequence database of user solutions for that exercise (a database similar to the one shown in Table 1). The solution is then annotated with the dimension "Solution State" automatically to indicate the success or failure. Moreover, CanadarmTutor uses the skills of the cognitive model to annotate sequences as dimensions (if the mastery level is higher than 0.8 in the student model, the skill is considered mastered). Thereafter, when a minimum of 10 sequences has been recorded for an exercise, the data mining algorithm is applied for extracting a partial task model for the exercise.

When CanadarmTutor detects that a learner follows a pattern during an exercise from the corresponding partial task model, dimensions of the pattern are used for updating the student model. For example, if a learner applies a pattern common to learners possessing "Skill_1," the mastery level of "Skill_1" in the student model will be heightened by a small increment (we use 0.05 in CanadarmTutor). In this way, the partial task models are also used for updating the student model (the student model is shared by the cognitive model and the partial task model approach).

During a learning session, CanadarmTutor uses the student model for generating exercises that progressively involves new knowledge or knowledge that is judged not yet mastered by the learner (this is done as explained in Section 5). Either the exercises that are generated are questions about declarative knowledge of the cognitive model or robotic arm manipulation exercises.

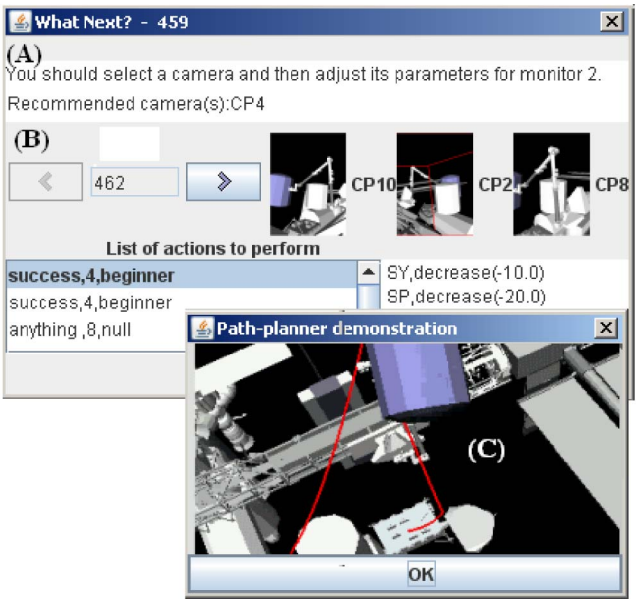

Fig. 9. A hint offered by the multiparadigm approach.

During an arm manipulation exercise, when a learner asks for help about what should be done next, the system generates a solution using the three aforementioned approaches (see Fig. 9). First, the cognitive model gives the general procedure that should be followed for moving the arm such as "You should select a camera and then adjusts its parameter for monitor 2" (see Fig. 9A). This help is generated by performing model-tracing with the cognitive model. Then, in the same window, the patterns of the partial task model that match the current user solution are displayed to the learner (Principle 2). For example, three patterns are presented in Fig. 9B. The learner can view a pattern as an animation by using the arrow buttons. Patterns contain the information about the joint rotations that should be performed for moving the arm. If no pattern matches the current learner solution (Principle 1), a demonstration is generated by the path-planner that demonstrates possible paths (see Fig. 9C).

Furthermore, CanadarmTutor can provide proactive help to learners such as assisting the learners to choose the best cameras thanks to the cognitive model (see Section 5). CanadarmTutor can also let the learner explore patterns from the partial task models (see Section 6) or the cognitive model (see Section 4) to learn about different ways to solve problems or about the general procedure for moving the arm. The learner can also request demonstrations at any time from the path-planner (see Section 4) or the cognitive model (see Section 5).

8 A Pilot Study

We performed an evaluation with 10 users to evaluate the multiparadigm version of CanadarmTutor. The goal of the evaluation was mainly to observe if the new version of CanadarmTutor provides a positive user experience. Moreover, we wanted to observe if the tutoring services

help the learners to learn and if, during an exercise, CanadarmTutor's interventions are relevant to the current solution. Before performing the experiments, we recorded at least 30 solutions for each robotic arm manipulation exercise. That is, to make sure that for each exercise some patterns are extracted by our data mining algorithm.

8.1 Experimental Procedure

The experimental procedure is as follows: We first explained the experimental procedure to each participant. We informed them about the kind of data that will be collected and how it will be stored. Then, we asked each participant to perform 15 procedural exercises of moving the arm from a given position to a goal position. Sequences of exercises were determined by CanadarmTutor based on the student model (as explained in Section 5). Completing the exercises took about 1 hour for each participant. During this session, we allowed participants to use all tutoring services. We set CanadarmTutor to record all participant solutions so that they can be examined after the experiment. During the experiment, we observed each participant and took notes to evaluate 1) if the tutoring services gave relevant help when they were used and 2) whether the learner corrected his mistakes after using the tutoring services or if he was more confused. Thereafter, we performed a 5-minute interview with each participant to get his or her opinion on the same two aspects, and also their general opinion about the tutoring services and how CanadarmTutor could be improved. Finally, after the experiment was completed, we reviewed the data recorded by CanadarmTutor to make sure that nothing important went unnoticed.

8.2 Experimental Results

All participants completed the 15 exercises. Most participants used all tutoring services. We found that participants relied more on the tutoring services for the most difficult exercises, which is what we expected. All participants mentioned that they found the tutoring services very useful and that the tutoring services helped them learn how to manipulate Canadarm2. Our observation was that learners using the tutoring services did not repeat their mistakes after receiving feedback in most cases. Users also agreed that the set of tutoring services would be less interesting if some were removed, which seems to indicate that the multiparadigm version of CanadarmTutor is superior to previous single-paradigm versions of CanadarmTutor. Note that we previously did pilot studies with single-paradigm versions of CanadarmTutor (see [10], [11], [12] for details), and the feedback for the multiparadigm version was considerably more positive. This is what we expected because this version is much more complete in terms of terms of features (tutoring services). Besides, we received comments for improvements and bug fixes. For example, two participants said they would like that CanadarmTutor has more elaborated pedagogical strategies and also the capability of generating more complex tutorial dialogues. This is because we used an early version of the tutoring module, which provided a limited pedagogical adaptation in this experiment. Recently, we have developed a newer version of this module (see [43] for a description of this module).

9 Lessons Learned

We have learned several important lessons from our work on the multiparadigm version of CanadarmTutor. The first lesson is that combining several paradigms for a procedural and ill-defined domain can provide a better solution than using a single paradigm. In CanadarmTutor, it allows providing more tutoring services to learners compared to previous single-paradigm versions of CanadarmTutor because some tutoring services can only be offered by a single paradigm. Based on our experience, we, therefore, encourage ITS researchers to attempt to build multiparadigm ITS in other ill-defined domains.

A second lesson learned is that if two paradigms can be applied to provide the same tutoring service in a given situation, it is important to define a priority order to choose which paradigms to use. In CanadarmTutor, there is a clear priority hierarchy (see Principle 1). When possible (for well-defined parts of the task), the cognitive model is used. For parts of the task that the cognitive model cannot be used (ill-defined parts of the task—joint rotations), the data mining approach is used (as an alternative or to provide complementary help with the cognitive model). Finally, if the cognitive model is not applicable and there is no patterns found by the data mining approach, the path-planner is used as a fallback approach.

A third lesson learned is that paradigms should preferably be integrated together because the sum of all the paradigms can be greater than the sum of each paradigm. For example, in CanadarmTutor, it is beneficial to use skills from the cognitive model to automatically annotate sequences used by the data mining approach. This allows finding patterns by the data mining approach that are tailored to the skills of each user. In our literature review, we have found an instance of a multiparadigm tutor combining a cognitive model and the constraint-based modeling paradigms. However, the paradigms are not integrated (one is used for providing help about domain knowledge and the other is used for supporting the learning process) [56].

A fourth lesson learned is that there are still several open-challenges for building multiparadigm ITS. For instance, a challenge is to make the best use of each paradigm when it is more appropriate from a pedagogical perspective. Another challenge is to build multiparadigm tutors using other paradigms or different techniques for each paradigm.

Finally, a fifth lesson learned is that a multiparadigm model as a whole may be hard to reuse in other domains since it is a specific combination of several paradigms that is tailored for a specific ill-defined domain. In the case of CanadarmTutor, the multiparadigm model developed may only be reusable for robot manipulation tasks. However, the work that we have done on each single paradigm is reusable individually. For example, the cognitive model and the data mining approach could be reused in other domains because their basic structure has been designed to be domain independent as much as possible. The path-planner could also be reused in other tasks even outside of the domain of ITS. For example, it could be used to design an intelligent robot that moves a robotic arm by itself. Our suggestion is that if ITS designers want to be able to reuse

each paradigm of a multiparadigm tutor, they should use a good object-oriented design.

10 Conclusion

In this paper, we have described how we tested three different paradigms to support tutoring services in CanadarmTutor. This has led us to observe that each paradigm has its advantages and limitations for the task of operating Canadarm2. Based on this finding, we have created a multiparadigm version of CanadarmTutor that integrates the three paradigms. The hypothesis underlying this work is that combining several paradigms could help overcome each paradigm's limitations because different approaches may be better suited for different parts of a same ill-defined task.

The multiparadigm version of CanadarmTutor provides more and richer tutoring services compared to previous versions of CanadarmTutor. We have performed a small-scale experiment with users. We have observed that the tutoring services were generally helpful and appreciated by users and that, in general, users did not repeat the same mistakes after receiving feedback. Moreover, users provided more positive feedback on the general user experience compared to our previous studies with single paradigm versions of CanadarmTutor.

References

[1] B.P. Woolf, *Building Intelligent Interactive Tutors: Student Centered Strategies for Revolutionizing E-Learning.* Morgan Kaufmann, 2009.

[2] C. Lynch, K. Ashley, V. Aleven, and N. Pinkwart, "Defining Ill-Defined Domains: A Literature Survey," *Proc. Ill-Defined Domains Workshop,* pp. 1-10, 2006.

[3] H.A. Simon, "Information-Processing Theory of Human Problem-Solving," *Handbook of Learning and Cognitive Processes,* vol. 5, W.K. Estes, ed., John Wiley & Sons, 1978.

[4] V. Aleven, B. McLaren, J. Sewall, and K. Koedinger, "The Cognitive Tutor Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains," *Proc. Eighth Int'l Conf. Intelligence Tutoring Systems,* pp. 61-70, 2006.

[5] K.R. Koedinger, J.R. Anderson, W.H. Hadley, and M.A. Mark, "Intelligent Tutoring Goes to School in the Big City," *Int'l J. Artificial Intelligence in Education,* vol. 8, pp. 30-43, 1995.

[6] A. Mitrovic, M. Mayo, P. Suraweera, and B. Martin, "Constraint-Based Tutors: A Success Story," *Proc. Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology (IEA AIE '01),* pp. 931-940, 2001.

[7] W. Clancey, "Use of MYCIN's Rules for Tutoring," *Rule-Based Expert Systems,* B. Buchanan and E.H. Shortliffe, eds., Addison-Wesley, 1984.

[8] A. Graesser, P. Wiemer-Hastings, K. Wiemer-Hastings, D. Harter, and N. Person, "Using Latent Semantic Analysis to Evaluate the Contribution of Students in Autotutor," *Interactive Learning Environments,* vol. 8, pp. 149-169, 2000.

[9] T. Barnes and J. Stamper, "Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data," *Proc. Ninth Int'l Conf. Intelligent Tutoring Systems,* pp. 373-382, 2008.

[10] P. Fournier-Viger, R. Nkambou, and E. Mephu Nguifo, "Learning Procedural Knowledge from User Solutions to Ill-Defined Tasks in a Simulated Robotic Manipulator," *Handbook of Educational Data Mining,* C. Romero et al., eds., pp. 451-465, CRC Press, 2010.

[11] K. Belghith, F. Kabanza, R. Nkambou, and L. Hartman, "An Intelligent Simulator for Tele-Robotics Training," *IEEE Int'l J. Trans. Learning Technologies,* vol. 5, no. 1, pp. 11-19, Jan.-Mar. 2012.

[12] P. Fournier-Viger, R. Nkambou, and A. Mayers, "Evaluating Spatial Representations and Skills in a Simulator-Based Tutoring System," *IEEE Trans. Learning Technologies,* vol. 1, no. 1, pp. 63-74, Jan.-Mar. 2008.

[13] N. Burgess, "Spatial Memory: How Egocentric and Allocentric Combine," *Trends Cognitive Sciences,* vol. 10, no. 12, pp. 551-557, 2006.

[14] L. Nadel and O. Hardt, "The Spatial Brain," *Neuropsychology,* vol. 18, no. 3, pp. 473-476, 2004.

[15] B. Tversky, "Cognitive Maps, Cognitive Collages, and Spatial Mental Models," *Proc. Int'l Conf. Spatial Informaton Theory (COSIT '93),* pp. 14-24, 1993.

[16] G. Gunzelmann and R.D. Lyon, "Mechanisms for Human Spatial Competence," *Proc. Int'l Conf. Spatial Cognition V: Reasoning, Action, Interaction,* pp. 288-307, 2006.

[17] V. Aleven, N. Pinkwart, K. Ashley, and C. Lynch, "Supporting Self-Explanation of Argument Transcripts: Specific v. Generic Prompts," *Proc. Intelligent Tutoring Systems for Ill-Defined Domains Workshop (ITS '06),* 2006.

[18] C. Lynch, K. Ashley, N. Pinkwart, and V. Aleven, *Proc. Workshop AIED Applications in Ill-Defined Domains (AIED '07),* 2007.

[19] V. Aleven, K. Ashley, C. Lynch, and N. Pinkwart, *Proc. ITS for Ill-Defined Domains Workshop (ITS '08),* 2008.

[20] P. Fournier-Viger, R. Nkambou, and E. Mephu Nguifo, "Building Intelligent Tutoring Systems for Ill-Defined Domains," *Advances in Intelligent Tutoring Systems,* R. Nkambou, R. Mizoguchi, and J. Bourdeau, eds., pp. 81-101, Springer, 2010.

[21] A. Mitrovic and A. Weerasinghe, "Revisiting Ill-Definedness and the Consequences for ITSs," *Proc. 14th Int'l Conf. Artificial Intelligence in Education,* pp. 375-382, 2009.

[22] V. Aleven, "Using Background Knowledge in Case-Based Legal Reasoning: A Computational Model and an Intelligent Learning Environment," *Artificial Intelligence,* vol. 150, pp. 183-237, 2003.

[23] K.D. Ashley, R. Desai, and J.M. Levine, "Teaching Case-Based Argumentation Concepts Using Dialectic Arguments vs. Didactic Explanations," *Proc. Sixth Int'l Conf. Intelligent Tutoring Systems,* pp. 585-595, 2002.

[24] E. Walker, A. Ogan, V. Aleven, and C. Jones, "Two Approaches for Providing Adaptive Support for Discussion in an Ill-Defined Domain," *Proc. ITS for Ill-Defined Domains Workshop (ITS '08),* 2008.

[25] T. Dragon, B.P. Woolf, D. Marshall, and T. Murray, "Coaching within a Domain Independent Inquiry Environment," *Proc. Eighth Int'l Conf. Intelligent Tutoring Systems,* pp. 144-153, 2006.

[26] R. Hodhod and D. Kudenko, "Interactive Narrative and Intelligent Tutoring for Ethics Domain," *Proc. ITS for Ill-Defined Domains Workshop (ITS '08),* 2008.

[27] C.-Y. Chou, T.W. Chan, and C.J. Lin, "Redefining the Learning Companion: The Past, Present, and Future of Educational Agents," *Computers and Education,* vol. 40, pp. 255-269, 2003.

[28] V. Kodaganallur, R. Weitz, and D. Rosenthal, "An Assessment of Constraint-Based Tutors: A Response to Mitrovic and Ohlsson's Critique of "A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms," *Int'l J. Artificial Intelligence in Education,* vol. 16, pp. 291-321, 2006.

[29] S. Moritz and G. Blank, "Generating and Evaluating Object-Oriented Design for Instructors and Novice Students," *Proc. ITS for Ill-Defined Domains Workshop (ITS '08),* pp. 35-45, 2008.

[30] J. O'Keefe and L. Nadel, *The Hippocampus as a Cognitive Map.* Oxford, 1978.

[31] A.D. Ekstrom et al., "Cellular Networks Underlying Human Spatial Navigation," *Nature,* vol. 425, pp. 184-187, 2003.

[32] J.S. Taube, R.U. Muller, and J.B. Ranck, "Head-Direction Cells Recorded from the Postsubiculum in Freely Moving Rats. 1. Description and Quantitative Analysis," *J. Neuroscience,* vol. 10, no. 2, pp. 420-435, 1990.

[33] J.S. Taube, R.U. Muller, and J.B. Ranck, "Head-Direction Cells Recorded from the Postsubiculum in Freely Moving Rats. 2. Effects of Environmental Manipulations," *J. Neuroscience,* vol. 10, no. 2, pp. 436-447, 1990.

[34] A. Terrazas et al., "Self-Motion and the Hippocampal Spatial Metric," *J. Neuroscience,* vol. 25, pp. 8085-8096, 2005.

[35] L. McNaughton et al., "Path Integration and the Neural basis of the Cognitive Map," *Nature Rev. Neuroscience,* vol. 7, pp. 663-678, 2006.

[36] D. Carruth et al., "Symbolic Model of Perception in Dynamic 3D Environments," *Proc. 25th Army Science Conf.,* 2006.

[37] M. Harrison and C.D. Schunn, "ACT-R/S: Look Ma, No 'Cognitive Map!'" *Proc. Fifth Int'l Conf. Cognitive Modeling,* pp. 129-134, 2003.

[38] R.M.J. Byrne and P.N. Johnson-Laird, "Spatial Reasoning," *J. Memory and Language,* vol. 28, pp. 564-575, 1989.

[39] G. Gunzelmann and R.D. Lyon, "Mechanisms of Human Spatial Competence," *Proc. Int'l Conf. Spatial Cognition,* 2006.

[40] D. Redish, A.N. Elga, and D.S. Touretzky, "A Coupled Attractor Model of the Rodent Head Direction System," *Network: Computation in Neural Systems,* vol. 7, no. 4, pp. 671-685, 1996.
[41] J.R. Anderson, *Rules of the Mind: Hillsdale.* Erlbaum, 1993.
[42] A. Mayers, B. Lefebvre, and C. Frasson, "MIACE: A Human Cognitive Architecture," *ACM SIGCUE Outlook,* vol. 27, no. 2, pp. 61-77, 2001.
[43] U. Faghihi, P. Fournier-Viger, and R. Nkambou, "CELTS: A Cognitive Tutoring Agent with Human-Like Learning Capabilities and Emotions," *Intelligent and Adaptive Educational-Learning Systems: Achievements and Trends,* A.P. Ayala, ed., pp. 339-365, Springer, 2013.
[44] V. Aleven, B. McLaren, J. Sewall, and K. Koedinger, "A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors," *Int'l J. Artificial Intelligence in Education,* vol. 16, pp. 291-321, 2006.
[45] G. Sanchez and J.C. Latombe, "A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking," *Proc. Int'l Symp. Robotics Research (ISRR '01),* pp. 403-417, 2003.
[46] M. Likhachev, D. Ferguson, A. Stentz, and S. Thrun, "Anytime Dynamic A*: An Anytime Replanning Algorithm," *Proc. Int'l Conf. Automated Planning and Scheduling,* 2005.
[47] N. Le, F. Loll, and N. Pinkwart, "Operationalizing the Continuum between Well-Defined and Ill-Defined Problems for Educational Technology," *IEEE Trans. Learning Technologies,* vol. 6, no. 3, pp. 258-270, 2013.
[48] D.H. Jonassen, "Instructional Design Models for Well-Structured and Ill-Structured Problem-Solving Learning Outcomes," *Educational Technology Research and Development,* vol. 45, no. 1, pp. 65-94, 1997.
[49] J. Han and M. Kamber, *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2006.
[50] Y. Hirate and H. Yamana, "Generalized Sequential Pattern Mining with Item Intervals," *J. Computers,* vol. 1, no. 3, pp. 51-60, 2006.
[51] D. Yuan, K. Lee, H. Cheng, G. Krishna, Z. Li, X. Ma, Y. Zhou, and J. Han, "CISpan: Comprehensive Incremental Mining Algorithms of Closed Sequential Patterns for Multi-Versional Software Mining," *Proc. Eighth SIAM Int'l Conf. Data Mining,* pp. 84-95, 2008.
[52] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. Int'l Conf. Data Eng.,* pp. 3-14, 1995.
[53] J. Wang, J. Han, and C. Li, "Frequent Closed Sequence Mining without Candidate Maintenance," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 8, pp. 1042-1056, Aug. 2007.
[54] P. Songram, V. Boonjing, and S. Intakosum, "Closed Multi-Dimensional Sequential Pattern Mining," *Proc. Third Int'l Conf. Information Technology: New Generations,* pp. 512-517, 2006.
[55] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal, "Multi-Dimensional Sequential Pattern Mining," *Proc. 10th Int'l Conf. Information and Knowledge Management,* pp. 81-88, 2001.
[56] I. Roll, V. Aleven, and K. Koedinger, "The Invention Lab: Using a Hybrid of Model Tracing and Constraint-Based Modeling to Offer Intelligent Support in Inquiry Environments," *Proc. 10th Int'l Conf. Intelligent Tutoring Systems,* pp. 115-124, 2010.
[57] V. Dimitrova, G.I. McCalla, and S. Bull, "Open Learner Models: Future Research Directions," *Int'l J. Artificial Intelligence in Education,* vol. 17, no. 3, pp. 217-226, 2007.
[58] N. Matsuda, W. Cohen, J. Sewall, G. Lacerda, and K. Koedinger, "Performance with SimStudent: Learning Cognitive Skills from Observation," *Proc. 13th Int'l Conf. Artificial Intelligence in Education,* pp. 467-478, 2007.
[59] P. Suraweera, A. Mitrovic, and B. Martin, "Constraint Authoring System: An Empirical Evaluation," *Proc. 13th Int'l Conf. Artificial Intelligence in Education,* pp. 451-458, 2007.

Philippe Fournier-Viger received the PhD degree in cognitive computer science from the University of Quebec in Montreal in 2010. He is an assistant professor of computer science at the University of Moncton, Canada. His research interests include data mining, artificial intelligence, intelligent tutoring systems, and cognitive modeling. He is the author of the open-source SPMF data mining software (http://goo.gl/xa9VX).

Roger Nkambou received the PhD degree in computer science from the University of Montreal in 1996. He is a professor of computer science at the University of Quebec at Montreal and director of the GDAC (Knowledge Management Research) Laboratory (http://gdac.dinfo.uqam.ca). His research interests include knowledge representation, intelligent tutoring systems, intelligent agents, ontology engineering, student modeling, and affective computing. He serves as a member of the program committees of the most important international conferences in artificial intelligence in education.

Engelbert Mephu Nguifo received the PhD degree. He is a professor of computer science at the Université Blaise-Pascal.

André Mayers received the PhD degree. He is a professor of computer science at the University of Sherbrooke. He founded ASTUS (http://astus.usherbrooke.ca/), a research group for intelligent tutoring systems, which mainly focuses 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. He is also a member of PROSPECTUS (http://prospectus.usherbrooke.ca/), a research group on data mining.

Usef Faghihi received the PhD degree in cognitive computer science from the University of Quebec in Montreal in 2010. He is an assistant-professor of computer science at Sul Ross State University.