Planning Agents in a Multi-agents Intelligent Tutoring System

Roger Nkambou[1] and Froduald Kabanza[2]

[1]Département d' Informatique
Université du Québec à Montréal, Montréal (QC) Canada
nkambou.roger@uqam.ca
[2]School of Computer Science
University of Windsor, Windsor (ON) Canada
kabanza@cs.uwindsor.ca

Abstract. Most recent architectures of intelligent tutoring system (ITS) focussed on the tutor or the curriculum components, but with little attention being paid to planning. Therefore, some works identified two main planning process in ITS: content planning and delivery planning. In this paper, we propose a new ITS architecture that involves sophisticated planning agent at four different levels of the ITS processing. The first planning level (course planning) and the second one (Lesson planning) are derived from the traditional content planning. The third and fourth planning levels come from a decomposition of the tutor module into two different components, one specialized in the tutorial actions planning and the other tailored for the generation of multimedia presentations. After a brief presentation of the new architecture, the paper mainly focussed on the first level planning using SIMPLAN. This planning oriented achitecture takes advantages of previous ITS architecture and provides a uniform view of ITS components that can facilitate collaboration between them.

1. Introduction

Over the last ten years, intensive research has been conducted on Intelligent Tutoring Systems (ITS), several of them focusing on architectural issues. The first proposed ITS architectures and included four main components (Burn and Caps, 1988): a curriculum module, a student model, a tutor (pedagogic model), and an interface between the student and the system. This basic architecture has since been extended by many researchers, including Bass, 1998; Titter and Blessing 1998; Frasson et al. 1996, Nkambou, Gauthier and Lefebvre, 1997; Both the architectures designed by Ritter and Blessing and Frasson et al. are focused on the tutor. Bass architecture mainly improves the interface. Nkambou et al. mostly focus on the curriculum component.

One problem with those architecture is the heterogeneity of the components. This makes it difficult to design the communication between components because one has

L. Monostori, J. Váncza, and M. Ali (Eds.): IEA/AIE 2001, LNAI 2070, pp. 921-930, 2001.

to deal with different data representations in different components. This problem is addressed in our new architecture by observing that the curriculum, the planner and the tutor are all concerned in a manner or another with the planning problem, that is, the problem of deciding in advance actions that will be executed in expected future situations in order to achieve a particular goal. Hence, each component can be considered basically as a planning process, although each process operate at its own level of abstraction.

Another problem with the above ITS architecture is that planning tasks are currently done in an ad-hoc and scruffy way. The underlying planning algorithm is not bolstered by a clear model upon which one could rely to prove the correctness or optimality of generated plans. Furthermore, currently used planning approaches in ITS do not reason about temporal constraints associated with the learning process or timing constraints in the synchronization of different medias used to present a course. Rather, such constraints are dealt with heuristically outside the planning processes. Note that these limitations not only apply to the above architecture, but also to most existing ITS architectures.

A great deal of these planning limitations can be overcome by using well-known planning algorithms in the area of artificial intelligence (e.g., see Dean et al. 1995; Kabanza, Barbeau and St-Denis, 1997). These algorithms rely on well defined semantics that allow to prove the correctness and optimality of generated plans. On the other hand, AI planning research has been addressing various issues that are relevant to planning in ITS such as handling temporal constraints and reasoning about uncertain information. Some of this research has been applied to the automatic generation of multimedia presentations (André and Rist, 1996). As part of the tutor may consist in producing presentation, this last application is very relevant to tutor planning.

It is surprising that AI planning techniques have been so under-exploited in ITS, despite the acknowledged importance of planning in ITS. Our new architecture fills in this gap. It is not designed as a mere plug-in of AI planning algorithms into an existing ITS system. In fact, it also involves innovative ideas on the coordination between different planning processes.

Our new ITS architecture hinges on a multi-layer planning system in which several planning agents run concurrently, collaborating in order to prepare lessons and teach them to the learners. Since planning agents play an important role in our new architecture, it helps to pursue immediately with some general background on planning agents in artificial intelligence. Then, we will outline the new architecture by focusing only on its main components, leaving aside the details on how each component actually behave. Finally, we will describe one of the components with more details. Due to space limitations, the other components will not be detailed in this paper.

2. Planning Agents

We are interested in the design and implementation of agents that are processes capable of accomplishing autonomously various tasks on behalf of users or other

processes, sometimes with various intelligent competencies, such as the ability to plan courses at different level of abstraction, or to learn from previous student and ITS interactions, and to plan multimedia presentation from raw media materials.

Planning agents will play three main roles in our new ITS architecture. Firstly, the ITS will be able to accomplish tasks or deal with events for which it has not been explicitly programmed, by planning on-line decision rules that coordinate more basic ITS processes so that they react correctly. Secondly, since planning agents yield decision rules that are constructively proven correct and optimal, the ITS reliability will become increased.

Basic algorithms for planning agents can be defined using explicit exploration of tree of possible ITS and student actions, the dynamic programming of Markov decision processes (e.g., see (Dean et al. 1995)) or control theoretical techniques (e.g., see (Kabanza, Barbeau and St-Denis, 1997)). All these approaches explore a search space to compute decision rules. The search space may, however, be differently structured, depending on formalisms used to model actions, goals, and environments, and search techniques. Since different formalisms and search techniques yield incomparable performances or expressiveness, this justifies research on hybrid approaches.

SIMPLAN (Simulation Based Planner) is a recently developed planning system that combines model checking and Markov decision theoretic techniques (Kabanza, 1999). It will be used as the basic planning system in our architecture. Although we opted for this particular planning system in order to fix a concrete development framework, the architecture of our ITS is open so that the planning algorithm underlying a particular planning agent may be replaced without jeopardizing the correct functioning of the system as a whole.

3. An ITS Architecture Based on Collaborative Planning Agents

The new architecture (figure 1) is articulated around four planning agents, each of them is in charge of a particular aspect of the ITS. Different planning agents may be implemented by the same planning algorithm, but with different input knowledge.

By “planning process”, we mean any process that decide what action to do and when in order to achieve a particular goal. A set of decision rules for accomplishing a given goal is also called plan in the sense that decisions are computed by anticipating future interactions between the processes composing the system and its environment. In the case of the curriculum, the task is to plan individualized course. A basic action is to achieve a particular instructional objective; the goal is to mastery a set of concepts at a given levels. The individualized course planner agent *(IC-Planner agent*) responsibility is to generate a course graph (plan) that fit to the current student group and according to a given goal that is expressed as set of capabilities to be acquire by the student (learning requirement). The Lesson planner agent (*LE-Planner*) based on the course plan, dynamically create the next concept the system has to focus on and recommends general context of its achievement in the current session. A basic action is to include an introduction statement, or a reminder statement; the goal is to have a lesson plan that can permit to mastery a specific

concept at a given level. As the learning actions are based on student abilities, the TA-Planner agent plans the sequence of basics activities for the student and the way those activities will be presented to the student. A basic action is "choosing a learning strategy" to sustain the activity delivery; the goal is to have the sequences of resources that will be activated in order to realize lessons goals.

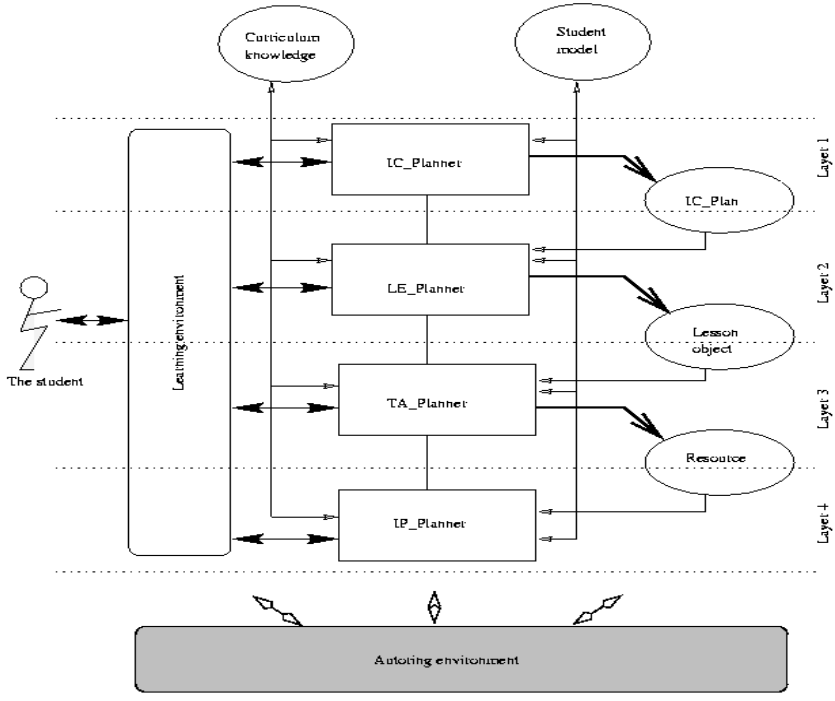

Fig. 1. The new architecture

The IP-Planner is in charge of the presentation of the activity by planning the sequence of basic actions the student deals with. Any planning agent can communicate with the student through a learning interface.

In the next sections, we focus only on the planning strategy in IC-Planner.

4. Using SIMPLAN as the Foundation of Planning Processes in ITS

For our initial implementation, we have decided to use the SIMPLAN planning system (Kabanza, 1999). Reactive plans such as those produced by SIMPLAN are finite state machine. They specify the actions to be executed by an agent depending on the current agent's state and current environment situation. This contrast with sequential plans often produced by other planning systems in which produced plans

are just sequences of actions without any additional pre-condition or context under which actions in the plan are applicable. In fact, sequential plans are appropriate for predictable environments in which a planned sequence of actions is always expected to provide the anticipated outcome. On the other hand, reactive plans such as those produced by SIMPLAN are suitable for unpredictable environments in which the agent has no control on events generated by the environment. Hence the agent cannot anticipate the outcome of its own actions and this is why a conditional plan that map agent actions to sets of expected situations is necessary. This is the case in ITS planning because the IC-Planning agent for example cannot anticipate choices made by the student when following a course.

The IC-Planning agent relies on *Curriculum Knowledge Transition Network (CKTN)* for representing planning knowledge (Nkambou, Frasson and Gauthier, 1998). However, although SIMPLAN accepts different planning knowledge representations, none of them coincides with CKTN. The closest one is the *Action Description Language (ADL)* frequently encountered in the artificial intelligence planning field. So we begin with a brief reminder of CKTN description. Then we explain how this knowledge is translated into an ADL representation acceptable for SIMPLAN. This should demonstrate that SIMPLAN effectively fits the new architecture, at least as far as IC-Planning is concerned.

4.1. IC-Planning: The Actual Process

The Curriculum and its Transition Network. By using CREAM modeling approach, curriculum knowledge are represented as a network of domain knowledge, of instructional objectives and of learning materials. Thoses networks contains special links from instructional objectives to domain knowledge, which denote the contribution of an instructional objectives to the acquisition of a domain knowledge, or the fact that a domain knowledge is prerequisite to an instructional objective. Learning materials are attached to instructional objective as resources that support its achievement. Instructional objective with the attached learning materials constitute a special node called “transition”. The resulting network is called CKTN (Curriculum-knowledge-transition network). An example of CKTN is shown in figure 2. The CKTN is central in the old planning process in ITS. The core of the system reasoning during planning process is based on this network.

The Target Group Knowledge. We define a target group (TG) as a group of students' state of knowledge of various capabilities which may be part of several curriculums. For instance, the knowledge of a novice nurse on the handling of the Baxter pump will not be the same as that of an advanced nurse. Therefore, a course on this topic should not include the same transitions for the former as for the latter. The advanced nurse would waste her time learning things that she already knows. Thus, these two groups of nurses constitute two different student target groups and the planner should build a course well-suited to each one.

The Training Requirement. Training requirement is expressed either as a set of objectives that the course should reach, or as a set of capabilities to be acquired by the student. It is also possible to mix these two approaches.

The old IC_Planning Process. The generation is performed by going through the CKTN. But before that, the TG state of knowledge is assigned to the capability nodes and the links which constitute the CKTN. The resulting graph is called a dynamic CKTN (DynCKTN) and we call this operation the marking of CKTN. More precisely, it consists firstly in attributing to each prerequisite link a value in the list: {*acquired, partially acquired, not acquired*} indicating whether the minimum acquisition level on this link has been reached according to the target group state of knowledge; and secondly, to each domain knowledge (capability) a value in {*possessed, partially possessed, not possessed*} representing the acquisition level of the target public and calculated from the levels assigned to the links.

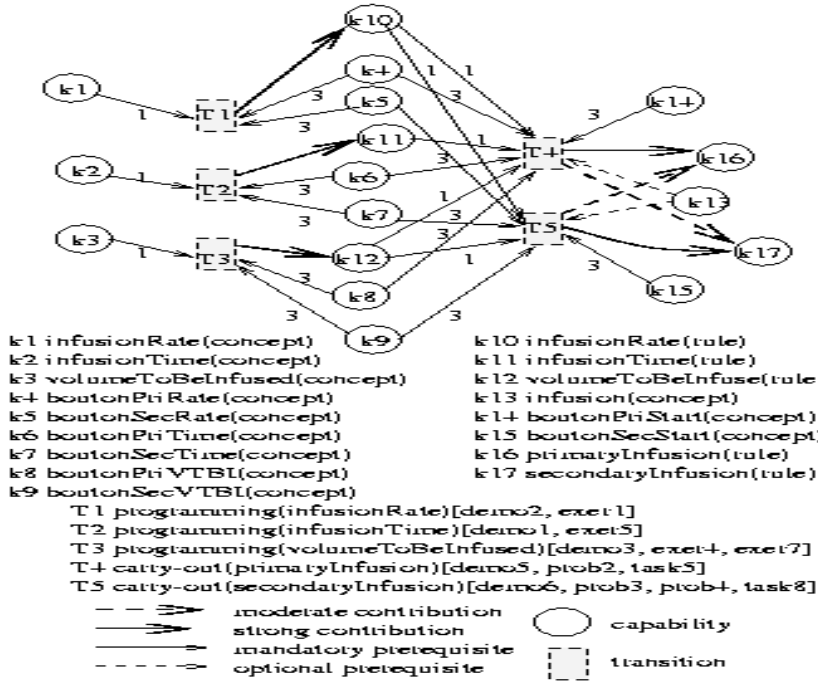

Fig. 2. Part of CKTN on the Baxter Pump

After the marking process, the generation algorithm traverses the resulting DynCKTN to determine which transition have to be included in the course in order to permit the acquisition of the knowledge specified in the requirement, a backward chaining traversal of the sub-graph rooted at the capability in order to choose the transition judged as necessary for the acquisition of that capability. We first evaluate the immediate transition that contribute to the capability and then, since some transitions possess mandatory prerequisite knowledge which in turn has contributing transition, we have to trace the sub-graph back until we reach a transition without any pre-requisite or a capability already mastered by the student (as specified in the training requirement or seen by the marking).

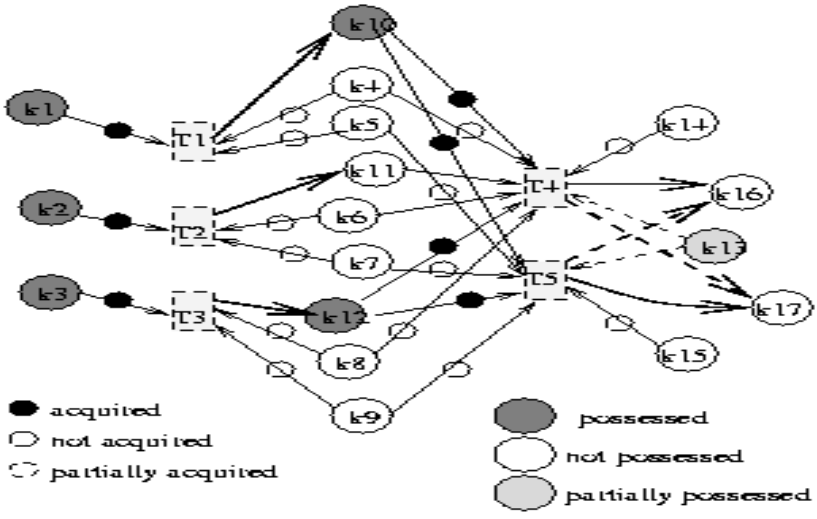

Fig. 3. DynCKTN from the CKTN in figure 2

The choice of relevant transition is carried out by applying heuristic rules introduced into the system and which consider several parameters: the links between capabilities and transitions (prerequisite or contribution), the knowledge in the training requirement and also the DynCKTN. As the output of this process, a course graph is generated. Figure 3 shows the DynCKTN derived from the CKTN in figure 2 after the marking process. If the teaching goal is to make student acquire the concept k16 at a strong level, then one possible generated course graph will includes transitions T1, T3, T4 and T5.

4.2. From CKTN to ADL

ADL specifies an agent's action in terms of *precondition*, *a set of removed facts*, and *a set of added facts*. Roughly, the *precondition* is a logical formula expressing facts that must be true in a state for the action to be meaningful in that state. In other words, if the precondition is true, then we are in a state where the agent *can* accomplish that action. Whether the agent *will* indeed accomplish the action or not, this will depend on whether or not the agent's plan specifies that action for that state. In fact, in every state, there may be several actions that the agent can execute. However, an agent can execute only one action at a time, so it must decide upon which action to execute. This is why it needs a plan that tells which action to execute in the current state. The *set of removed facts* specifies facts that are true in the current state become false after the action is executed, while the *set of added facts* specifies facts that are not true in the current state but becomes true after the execution of the action. Both the set of removed and added facts form the *effects* or *postconditions* of the action. Hence,

given a description of a current state in term of facts true and an ADL description of an agent's action, a planning system can predict the state that would result from the execution of the action by removing facts in the set of removed facts and then adding facts in the set of added facts.

The translation of CKTN into ADL is quite easy. Input capabilities with their preconditions become represented by preconditions, transitions are replaced actions, while output capabilities are specified as added and removed facts. Figure 4 shows part of the ADL translation of the CKTN in Figure 3. This translation was made automatically, by a translation algorithm.

Because of space limitations, we cannot give a full description of the SIMPLAN's implementation of ADL. Accordingly, we do not expect the reader to fully understand the above example without a previous reading of the SIMPLAN's manual (Kabanza, 1999). Nonetheless, we provide a rough explanation of the syntax in the figure which allows a coarse understanding. Roughly, SIMPLAN uses a Lisp-like notation. Each action schema is described by an *adl-add-op* expression, with the keywords *:pre*, *:add* and *:del* for, respectively, the precondition, added facts and deleted facts. In addition we have a *:name* keyword that uniquely identifies each action schema. For instance, the first *add-adl-op* expression describes the transition *T1* in the CKTN of Figure 3. The expression *:var-gens* is roughly a declaration of the variables involved in description of the action together with some information on how the variable will get instantiated. The *:form* keyword describes a logical formula that must hold for the precondition to be true, for a literal to be deleted (in the *:add* component) or to be deleted (in the *:delete* component).

```
(add-adl-op
  :name '(T1)
  :pre (adl-pre :var-gens '((?x1) (k1 ?x1) (?x4) (k4 ?x4)
                            (?x5) (k5 ?x5))
  :form  '(and (>= ?x1 1) (>= ?x4 3) (>= ?x5 3)))
  :add (adl-add (adl-cond :form '(not (exists (?x) (k10 ?x)
                                  (>= ?x 10)))
  :lit '(k10 10)))
  :del (adl-del  (adl-cond :var-gens '((?x) (k10 ?x))
  :form '(< ?x 10)
  :lit '(k10 ?x))))
```

Fig. 4. Part of the ADL corresponding to the CKTN in figure 2.

4.2. SIMPLAN Planning with CKTN

Once we have a translation of CKTN into ADL it becomes easy to generate plans. The training goal is easily expressed as a *goal* for SIMPLAN while the student's profile corresponds to *the initial state* for SIMPLAN. Figure 5 shows a trace generated by SIMPLAN from a goal and initial state corresponding respectively to the target knowledge and student's profile highlighted in the CKTN of Figure 2, and with the ADL actions (part of this is shown at Figure 4) which are translated from the CKTN of Figure 2. The plan is at the end of the trace.

More explanations on the representation of plans by SIMPLAN are needed in order to better understand the plan in Figure 5. A plan is an automaton with states identified by numbers (ID) and labeled with sets of propositions true. To each state corresponds the action to be executed by the agent and a set of expected resulting states. More specifically, because of nondeterminism resulting from uncontrollable events, it may be the case the outcome of an action is uncertain. In this case, we have a set of successor states, which means that when the action is effectively executed, we cannot predict beforehand the outcome, but the resulting state is expected to be one among the set of successors. For instance, the first two lines of the reactive plan in Figure 5 specify that state with ID 0 has the attached set of proposition true. Whenever these conditions hold, the planning agent must perform transition T3 and the expected resulting state is the one with ID 1. In this particular example, all actions are deterministic so that we always have only one successor state. We can see that, the state with ID 5 contains the goal (K16, 5) that is why the planner stops.

SimPlan 1.0
Init state: ((K15 5) (K14 5) (K9 4) (K8 4) (K7 4) (K6 4) (K5 3) (K4 3)
(K3 2) (K2 1) (K1 2))
Goal: (EVENTUALLY (K16 5))
Reactive Plan:
[ID 0 state ((k15 5) (k14 5) (k9 4) (k8 4) (k7 4) (k6 4) (k5 3) (k4 3)
(k3 2) (k2 1) (k1 2)) ACTION ((**T3**) **1**)]
[ID 1 state ((k15 5) (k14 5) (k12 7) (k9 4) (k8 4) (k7 4) (k6 4) (k5 3)
(k4 3) (k3 2) (k2 1) (k1 2)) ACTION ((**T2**) **2**)]
[ID 2 state ((k15 5) (k14 5) (k12 7) (k11 10) (k9 4) (k8 4) (k7 4)
(k6 4) (k5 3) (k4 3) (k3 2) (k2 1) (k1 2)) ACTION ((**T1**) **3**)]
[ID 3 state ((k15 5) (k14 5) (k12 7) (k11 10) (k10 10) (k9 4) (k8 4)
(k7 4) (k6 4) (k5 3) (k4 3) (k3 2) (k2 1)
(k1 2)) ACTION ((**T5**) **4**)]
[ID 4 state ((k17 10) (k16 5) (k15 5) (k14 5) (k12 7) (k11 10) (k10 10)
(k9 4) (k8 4) (k7 4) (k6 4) (k5 3) (k4 3) (k3 2) (k2 1)
(k1 2)) ACTION ((**T5**) **5**)]
[ID 5 state ((k17 10) ***(k16 5)*** (k15 5) (k14 5) (k12 7) (k11 10) (k10 10)
(k9 4) (k8 4) (k7 4) (k6 4) (k5 3) (k4 3) (k3 2) (k2 1)
(k1 2)) ACTION ((w 1) 5)]

Fig. 5. SIMPLAN generated plan from ADL of figure 4

Conclusion

We have presented a multi-agent ITS architecture that includes four planning agents. In this architecture, planning agents collaborate in order to make the learning process more adapted to the student. The planning process is supported by SIMPLAN, a recently developed planning system that combines model checking and Markov decision theoretic techniques. Reactive plans produced by SIMPLAN specifies the actions to be executed by an agent depending on the current agent's state and current environment situation (student knowledge, learning goals …). By considering

teaching/learning process in an ITS as a multi-layer planning task, we lead to a uniform view of ITS processes from one layer to another. What change is the semantic of the planning process. Thus communication between ITS component is made more easy due to the homogeneity of the components. Also, generic planning agent view of ITS components make it easy to re-use the same technology in many application domain. This represent a major contribution in component-based ITS which is the current focus of several ITS researches (Ritter, Brusilovski and Medvedeva, 1998; Devedzic, Radovic, and Jerinic, 1998). Our future works will concentrate on the collaboration between planning agents and more experimentation in real word applications.

References

André, E. and Rist, T. Coping with temporal constraints in multimedia presentation planning.In *Proc. of 13th National Conference on Artificial Intelligence*, pp 142-147. AAAI Press, 1996.

Bass, E.J. Towards an Intelligent Tutoring System for Situation Awarwness Training in Complex, Dynamic Environments. In proceedings of the 4th International Conference on ITS., pp..26-35. Springer-Verlag, Berlin, 1998.

Burn and Caps. *Intelligent Tutoring Systems*. Lawrence Erlbaum Ass., Hillsdale, NJ., 1988.

Dean, T., Kaelbling L.P., Kerman, J. and Nicholson, A. Planning under time constraints in stochastic domains. *Artificial Intelligence*, 76(1--2):35-74, 1995. Elsevier.

Devedzic, V., Radovic, D. and Jerinic, L. On the notion of components for intelligent tutoring systems. In: Proceedings of the 4th International Conference on ITS., pp..504-513. Springer-Verlag, Berlin, 1998.

Frasson, C., Mengelle, T., Aïmeur, E. and Gouardères, G. An Actor-Based Architecture for Intelligent Tutoring Systems. In proceedings of the 3rd International Conference on ITS., pp 57-65. Springer-Verlag, Berlin, 1996.

Kabanza, F. SimPlan's Manual. 1999. The manual is part of the system which can be downloade from the site http://www.dmi.usherb.ca/~kabanza/simplan.

Kabanza, F., Barbeau, M. and St-Denis, R. Planning control rules for reactive agents. *Artificial Intelligence*, 5(1):67--113, 1997. Elsevier.

Klausmeier, H.J. Conceptualizing. In: B.F. Jones and L. Idol (Eds). Dimensions of thinking and cognitive instruction, pp. 93-138. NJ:LEA, 1990.

Nkambou, R., Gauthier, G. et Lefebvre, B.. Un modèle d'architecture de STI pour l'enseignement à grande échelle. *Dans Environnement interactif d'apprentissage avec ordinateur*, pp. 236-249. HERMES, Paris, 1997.

Nkambou, R., Frasson, C. and Gauthier, G. "A New approach to course authoring for Intelligent Tutoring System: The authoring environment". Computer in Education: an international journal, Vol. 31, No. 1, pp. 105-130, Elsevier Science, London, 1998.

Ritter, S., Brusilovsky, P. and Medvedeva, O. Creating more versatile intelligent. Learning environments with a component-based architecture. In: Proceedings of the 4th International Conference on ITS., pp. 554-563. Springer-Verlag, Berlin, 1998.