interaction protocols. OPN are a formalism combining coherently *Petri nets (PN)* theory and the *Object-Oriented (OO)* approach. While PN are very suitable to express the dynamic and possibly concurrent and open behavior of a protocol, the OO approach permits the modeling and the structuring of its active (actor) and passive (information) entities. In our case, actors correspond to teacher and students, while information corresponds to domain and scenarios.

The paper is organized as follows: Section 2 presents an overview on the related work. Section 3 introduces the FAST ITS building tool. Section 4 explains how a group interaction scenario is specified and Section 5 presents a simple example of a scenario, discussing in particular what a group interaction is. Finally, in Section 6, we present some conclusions and future works

2 Related Work

Organization modelling is recognized as an essential mechanism for structuring the design of *Multi-Agent Systems (MASs)* and coordinating their executions. Indeed, this approach provides high level concepts, such as groups, roles, protocols or commitments, useful to structure and rule, at a macro level, the coordination of the different agents involved in a MAS. All these reasons have led to an increased development of agent methodologies (GAIA, MOISE, AALADIN, etc.) structured around organizational concepts (see [20] for a survey). In most of these methodologies, protocols and groups are considered as basic building blocks of an organizational oriented approach of MASs. This is the approach we have followed in this paper by structuring our MAS around groups and protocols: while groups constitute an interaction space for agents, protocols define the rules to enter or leave a group and play a role within a group. The concept of organization (also groups, institutions, communities, etc.) within MAS has been discussed in several papers [10], [13], [9], [11], [12], [24], [19], [26], [16], [14], [28].

Regarding agent-based protocols, [7] provides an interesting survey of the different specification formalisms, and concludes that Petri Nets provide good software engineering properties to specify, validate and execute concurrent protocols. Our work is also related to [16] in which the adequacy of the Petri Net with Objects formalism, to describe real world protocols, is shown. Systems focusing on the concept of group have also been used in the context of ITS [22], [14], [27], [21] and [18]. In this paper, we do not address the automatic group formation problem. This issue is treated, for example, in NetClass [21] using the learner model, the author information and a sociometric test (that measures the degree of cohesion among students). In WhiteRabbit [27], the groups are created from the analysis of the user model based on the keywords (about his projects, experience, etc.) and also on the conversations.

Among ITSs that share our goal of simplifying course development, an interesting example is the Cognitive Tutor Authoring Tools (CTAT) project. It assists in the creation and delivery of ITS based on model tracing [17]. The main goal of this project is to provide tools to reduce the amount of artificial intelligence (AI) programming expertise required to implement ITSs. The project authoring tools support the development of two types of tutors: Cognitive Tutors and Example-Tracing tutors. Cognitive tutors contain a cognitive model that simulates the student thinking in order to monitor student activities and to provide pedagogical assistance during problem solving. In contrast, Example-Tracing Tutors do not contain a cognitive model: to develop a tutor of this kind, the author needs to specify a recording of possible student actions and corresponding feedback messages. Although Example-Tracing Tutors do not require IA programming, they are specific to the given set of problems and cannot deal with student actions which are not pre-specified by the author [17], i.e. they lack adaptiveness.

An example of an ITS that uses multi-agent technology is the DOCTA [5] system. It uses intelligent agents for collaborative learning to support collaboration in a learning scenario on gene technology. Agent system consists of two components: a Student Assistant agent (SA-agent) and an Instructional Assistant agent (IA-agent). Both agents observe and detect problems in the collaboration and knowledge-building process among students, but their presentations are different.

Another example is the COLER system [6]that addresses both social and task-oriented aspects of group learning. It helps students collaborate while solving Entity Relationship modeling problems. Unlike previous work, generally emphasizing dialogue analysis or expert models, this work proposes a new approach to support collaboration that identifies learning opportunities based on the differences between problem solutions and tracking levels of participation. This work demonstrates how intelligent agents can produce reasonable collaboration advice in domains for which structured problem solutions exist by using a few basic knowledge sources, and illustrates several methods for knowledge evaluation and reasoning of complex knowledge-based systems.

3 FAST ITS Building Tool

FAST [3] is a domain independent authoring tool to implement multi-agent Intelligence Tutoring Systems. Courses developed using FAST are based on the conceptual model MATHEMA [8]. This model proposes an ITS architecture that consists of three modules (see Figure 1): the *Tutoring Agent Society (TAS)*, the *Student Interface* and the *Instructor Interface*. The student interface provides access to the system and the instructor interface allows the monitoring of the course. The TAS consists of a multi-agent system where each *Tutor Agent (TA)* contains a

complete ITS focused on a sub-domain of the course target domain. Each of the intelligent tutoring agents in the TAS is responsible for one sub-domain. MATHEMA provides a modeling scheme for these sub-domains that is divided into two views: *external view* and *internal view*.

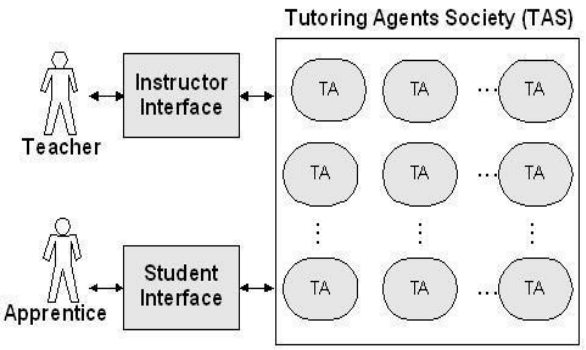

Figure 1: MATHEMA System Architecture.

The external view is a domain knowledge partitioning scheme, based on epistemological assumptions, that guides the author during course development. This partitioning is performed according to two main dimensions: *context* and *depth*. Along the context dimension the domain knowledge is partitioned according to a set of different points of views about its contents. For each particular context, the depth dimension partitionates the domain knowledge according to the methodologies used to deal with its contents. Each pair context/depth is associated with a sub-domain, to be dealt with by one of the TAS agents.

The internal view proposes to organize the knowledge associated with each sub-domain into a set of curricula. Each curriculum is progressively refined according to three levels of detail: *pedagogical units*, *problems* and *interaction support units*. At the *pedagogical unit* level, each curriculum, that describes a possible sequence of sub-domain contents to be presented to the student, is refined into a set of partially ordered pedagogical units, possibly with prerequisites relationships. At the *problems* level, each pedagogical unit is refined into a set of problems, also partially ordered and possibly with prerequisites relationships. Finally, at the *interaction support units* level each problem is associated with a set of interaction units with the student, that support the problem solving activities, such as explanations, examples and exercises.

The domain knowledge of any ITS developed using the FAST tool presents the structure defined by this internal view. This fact allows the construction of group interactions that, although not domain dependent, can explore the domain structure, going beyond the simple communication support between group members and the instructor. This is possible because these group interactions can use the same problem solving activities already defined in the context of the underlying ITS.

A further advantage is that the student model, used in group interaction management, can be defined as an extension of the student model in the underlying ITS. In such a way that the group interaction manager can explore the preferences and previous results obtained by each student in the context of individual learning during her/his interaction with the underlying ITS. Both, domain and student models, are represented using ontologies. These ontologies are briefly described in the next subsections.

3.1 Domain Model

The domain model contains definitions of all the concepts in the internal view of the MATHEMA model. A course is represented as an instance of the domain model and contains all the information provided by the author. This information is of two types: properties and contents. Examples of properties are prerequisite relationships, degree of detail, level of difficulty, etc. Contents is what is presented to the student, typically an interactive page encoded into predefined HTML pages templates.

The ontology described in [15] includes concepts to define prerequisite order graphs, that can be used to define the relationship among pedagogical units or problems and concepts to represent specific types of interaction support units, whose contents are also specified by the author (see Figure 2). These concepts correspond to the elements of the internal view of the MATHEMA model.

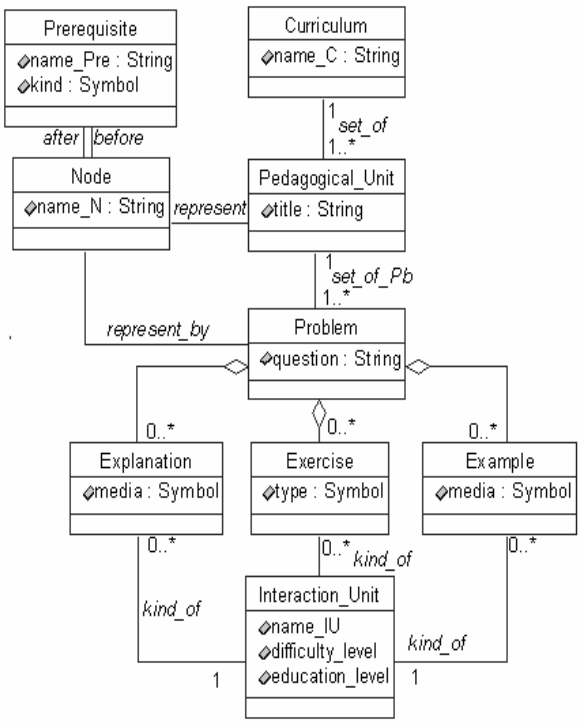

Figure 2: Domain Model.

In particular, the Problem concept (and its sub-concepts) is reused in the definition of the Content Unit concept of the proposed group management method (see Section 4.2).

3.2 Student Model

The student model, proposed in [15], contains definitions of all the concepts necessary to characterize a student and her/his history of interactions with the system. Its contents include static information, such as education level, based on a preliminary test, and preferences; and also dynamic information that consists of descriptions of the student activities during all her/his sessions of interaction with the system. This student model was extended to include the information necessary to group interaction. This also includes static information, such as preferences, and dynamic information, such as the record of the student performance during group activities.

4 Group Management Method

The goal of the proposed group management method is to allow the specification and execution of complex group activities, without burdening the author with the task of specifying how the student and group activity models should be taken into account and updated during the interaction. To support the proposed method, the conceptual model proposed by the FAST tool was extended to include the definition of *group activity*. A group activity involves the *developer* who specifies the scenario library where the group activities are stored; the *author* who chooses and instantiates a suitable scenario to build an actual group activity; and the *instructor* who supervises the group activity, determining the beginning and end of the activity, and verifying student feedback. Each group activity is necessarily based on an underlying ITS built up using the FAST tool. It presents two levels: the *specification level* and the *execution level*, as shown in
Figure 3.

The specification level main concepts are Group and Scenario. A group consists of a set of students. A scenario consists of an operational definition of the group activity. Scenarios are defined by the developer and stored into a scenario library. They are built using predefined activity units that can be reused in different scenarios.

The execution level consists of a multi-agent system that performs a group activity based on an instance of the Scenario concept, as defined in the specification level. To define such an instance, the author chooses the more adequate scenario from the scenario library, provides the contents, and customizes the scenario parameters (e.g., student level requirements, minimal and maximal number of group members, etc). This information is compiled into an OPN able to manage the group activity, in which the tokens are instances of the Group concept. Once the scenario and group instances are defined, the group activity can be made available to the students to be executed under the supervision of the instructor.

The concepts involved in these two levels of the group activity are described in more detail in the next subsections.

4.1 Specification Level

The concepts involved in the specification level (see left side of
Figure 3) include: Group, Role, Scenario, Prerequisites, Activity units (Management and Content units), Prerequisites and Interaction Protocol.

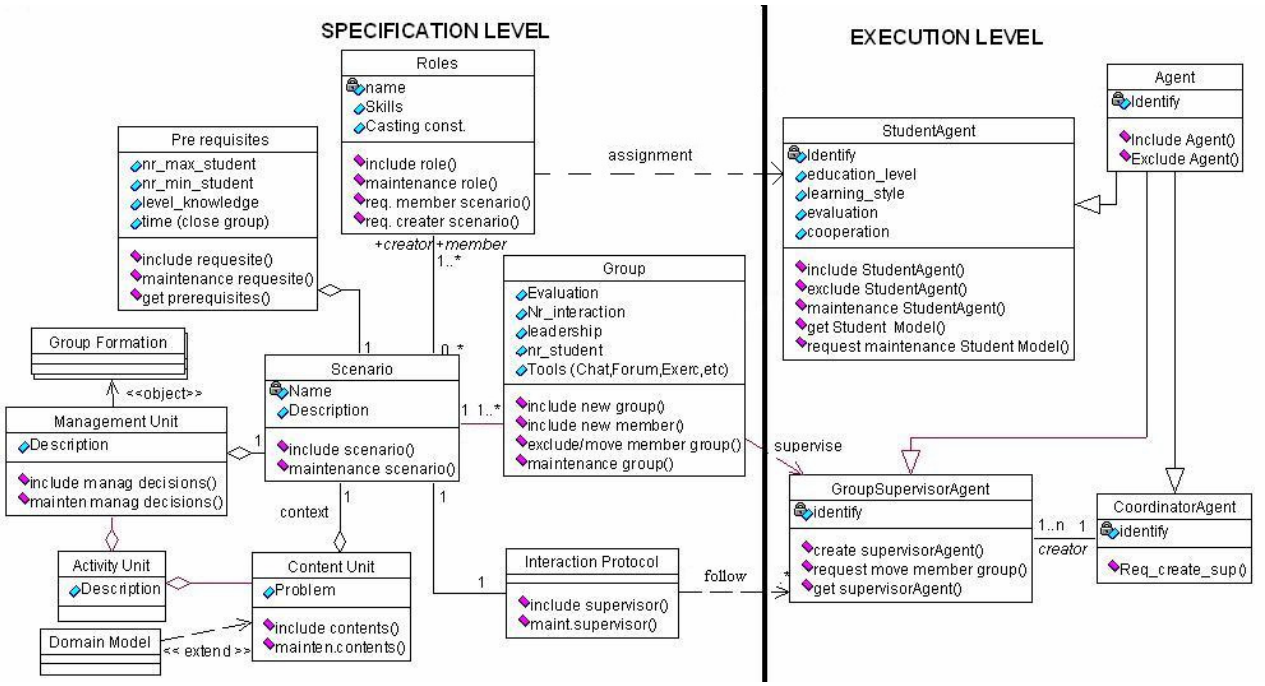

Figure 3: Group Activity Model.

The Group concept joins a set of students already inscribed in the underlying ITS and, optionally an instructor. The members of a group can be assigned to different roles.

The Role concept structures the members of a group into classes according to their participation in a scenario. Each Role is defined by: its name; the required skills (that an agent must meet to be authorized to play that Role); and the casting constraints (such as the maximum number of agents that may play that role, the condition required to play it, etc). Some examples of roles are: Team Leader, and Plain Member.

The Scenario concept consists of an operational definition of the group activity. It includes prerequisites, activity units and an interaction protocol.

The Prerequisite concept defines the initial conditions for a given scenario, e.g., the minimal and maximal number of group members, the situation of these members with respect to the course of the underlying ITS, etc.

The Activity Unit concept defines the different activities that occur in a given scenario. There are two types of activity units:

- Management units: used to define the typical activities of group interactions, e.g., group formation, problem distribution, wait for the first solution, waiting for all solutions, group member instruction, all group members instruction, etc; and
- Contents units: used to define the problem solving tasks associated with a given scenario. These tasks are defined using the Problem specification (that includes Interaction Units) of the domain model of the underlying ITS (see Section 3.1). The content definitions are provided by the author, using the authoring interface of the FAST tool [4] [15] and can be used either in the context of group learning or individual learning.

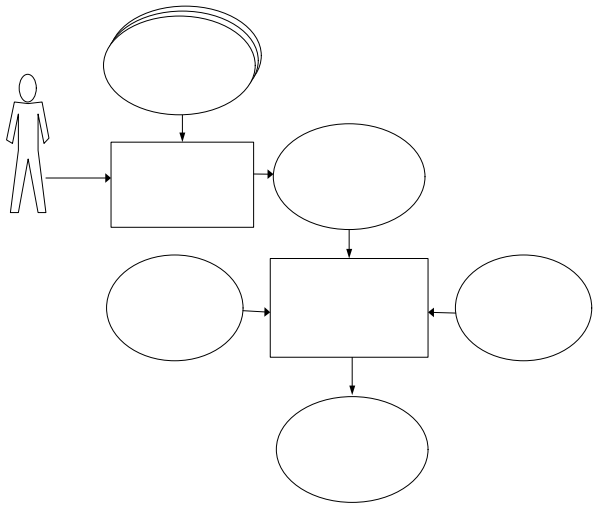

Figure 4: Specification of an Interaction Protocol.

The Interaction Protocol concept contains the operational specification of the group activity, i.e., the order in which the activity units are executed in a given scenario. It is represented by a two level hierarchical OPN. The specification of an instance of an interaction protocol is a two step process (see Figure 4). The first step consists in selecting a scenario from the scenario library and instantiating all its attributes.

The second step compiles this information to produce the OPN that defines the interaction protocol. The compilation process automatically integrates the domain model of the underlying ITS and the use and update of the student models into the conditions of the Petri Net transitions. This integration provides the adaptive character of the interaction protocol.

4.2 Execution Level

The execution level is defined by a multi-agent architecture inspired by Ferber [07] and represented in Figure 5. The concepts involved in the execution level (see right side of
Figure 3) are: Agent, Student Agent, Group Supervisor Agent and Coordinator Agent.

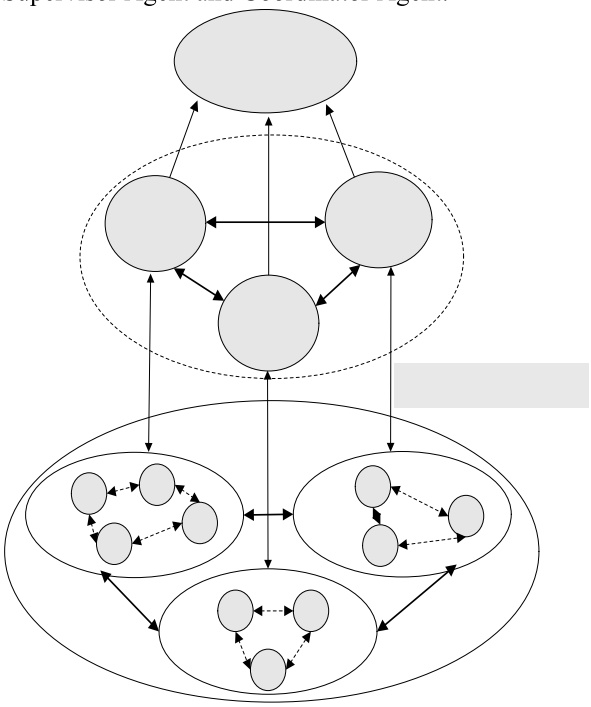

Figure 5: Multi-Agent Architecture.

A Student Agent (SA) represents a student and has a role assigned to it. Each Student Agent stores internally the information of the student model relevant to the group management, e.g., the group activities in which the student has participated, statistics about the role of the student in these groups (leader or not), the number of group communications in which she was involved, etc. It can also consult the student model stored in the TAS agents of the underlying ITS (see Section 3.2).

A Group Supervisor Agent (GSA) supervises a group activity according to the OPN associated with

the interaction protocol defined in the scenario instance. In this OPN, the tokens are instances of the Group concept. This allows the management units, in the interaction protocol, to consult and update group attributes. In a first step, students can be included as group members. Once the student set is available, it can be used to identify the associated student agents, e.g., to allow a broadcast message to be sent or to consult the student models stored in the TAs of the TAS, e.g., to check the performance of the students in a given content unit.

The Coordinator Agent is responsible for the creation and destruction of Group Supervisor Agents at run time, for the permanent storage of all the relevant information about the group activities, and also for monitoring individual learning in order to detect opportunities for group learning activities. The Coordinator Agent also provides an interface for the instructor, through which she/he can monitor the group activity.

5 An Example of a Group Activity

To clarify the notions introduced in the previous section, we present an example of a simple group activity and show how it can be instantiated into a concrete group management process.

The group activity is intended to develop the "divide-and-conquer" strategy in problem solving. It supposes a problem that can be partitioned into a certain number of sub-problems. Each sub-problem may be solved independently and their solutions have to be combined to solve the original problem.

5.1 General Description

According to the specification level of a group activity, defined in Section 4.1, we must define the following concepts: group, roles, scenario, prerequisites, management units, content units and interaction protocol.

Group: the activity needs at least one student per sub-problem.

Roles: the activity includes two roles: sub-problem solver and solution integrator. The solution integrator role should be assigned to one or more students that will be responsible for the integration of the sub-problems solutions. The choice of these students can be done dynamically, e.g., the first to complete a sub-problem solution or the best graded in the underlying ITS. Finally, the sub-problem solver role is assigned to all the students that participate in the activity.

Scenario: it is defined by the following concepts.

Prerequisites: the members of the groups involved in the activity should have the necessary background to solve the problem being considered.

Management units: The following management units are necessary to control the scenario:

- Group formation: assignment of the problem solver roles associated with each sub-problem.
- Sub-problem distribution.
- Monitoring of the sub-problem solutions.
- Coordination of the interaction among group members who incorrectly implemented the interface between their solutions.
- Assignment of the integration group.

Content units: The contents of the scenario, to be provided by the author through the FAST authoring interface (see Section 3), consist of the following problem descriptions:

- A general explanation of the problem and its sub-problems.
- For each sub-problem:
  - a detailed explanation.
  - one or more examples of similar problem solutions .
  - two types of exercises: one that tests the correctness of the sub-problem solution and one that verifies whether the solution correctly implements the expected interface with the other sub-problem solutions.
- An exercise that tests the correctness of the combined sub-problem solutions.

It should be noted that these contents are instances of the problem (and interaction unit) concepts of the domain model ontology described in Section 3.1 and may also be used in the context of an individual interaction with the underlying ITS.

Interaction protocol: The top level of the interaction protocol corresponding to this scenario is represented by the Petri Net shown in Figure 6.

In our context tokens contain an instance of the group concept. For legibility reasons, we have omitted the object dimension of a OPN that is the values of tokens, the Preconditions, Actions and Emission Rules of transitions. The resulting *abstract OPN* just keeps the aspects related the behavioral structure of the protocol. However, from this conventional abstract structure, standard Petri net properties can be proved, such as the presence of *loops* or *cycle* (sequences of transitions that can be infinitely repeated, *deadlocks* (blocking state from which no transition may occur), the (un-)accessibility of a *goal*, *final* or a *home state*, the *boundness* (infinite growing of the number of tokens) or the *lost of tokens* in a hole place.

5.2 Scenario Instance

To instantiate the group activity for the "divide-and-conquer" problem solving strategy, we implement a group activity based on an already existing individual learning ITS for the domain of *Structure of Information*, an undergraduate discipline of the Control and Automation Engineering course at the Santa Catarina Federal University, Brazil [3]. This ITS

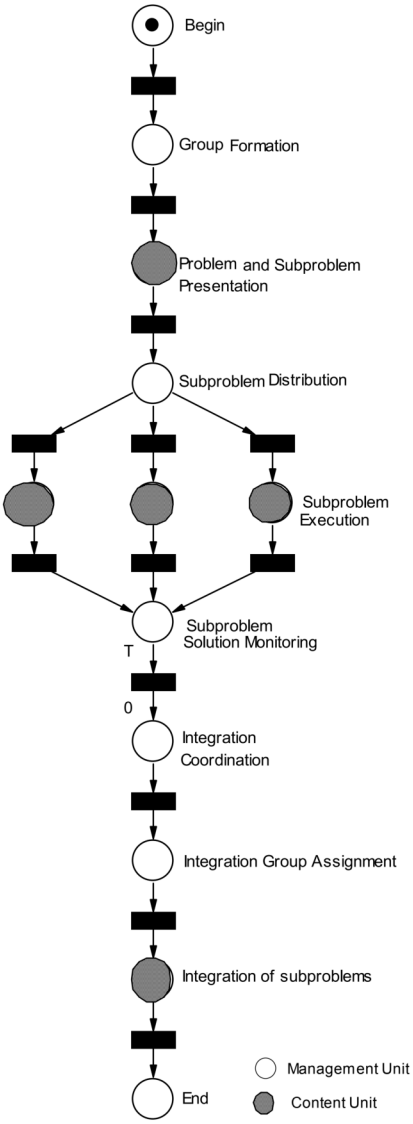

Figure 6: Petri net modelling the interaction protocol of a scenario.

was built using the FAST tool and meets the definitions given in Section 3.

The problem to be solved during the group activity is defined as follows:

- Problem description: given a programming language that supports integer arithmetic operations, how can it be extended to support operations for other types of numbers (rational, float and complex).
- Sub-problems: arithmetic operation packages for each of the three new types of numbers, including appropriate conversion functions.
- Integration: a dispatch function package that integrates all four types of numbers.

The implemented group activity is intended to be developed during a presential course in which the course teacher is the instructor. The instances of the relevant concepts involved in the group activity definition of Section 5.1 are defined as follows.

Roles: The sub-problem solver role is assigned to all students in the class room and the solution integrator role is assigned to the students that are members of the first group to solve the assigned sub-problem successfully.

Prerequisites: the students that participate in the activity must have completed the necessary pedagogical units (basic programming, abstract data types).

Management units: the implemented activity uses a synchronous group formation in which an invitation is sent to all the students in the classroom. The students should answer with the identification of their preferred sub-problem. The system controls the maximum size of each group automatically. The necessary prerequisites are also verified for each student. The Group Formation place in the top level OPN (see Figure 6) is exploded into the bottom level OPN shown in Figure 7. Problem distribution is generated automatically. Monitoring of sub-problem solutions is

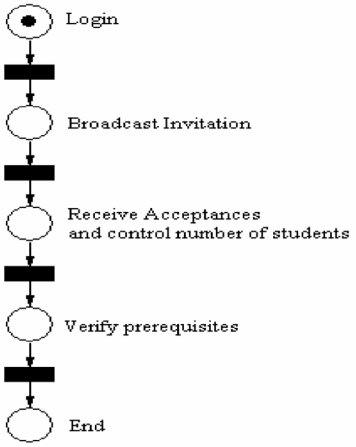

Figure 7: Petri Net Modelling the Group Formation Protocol.

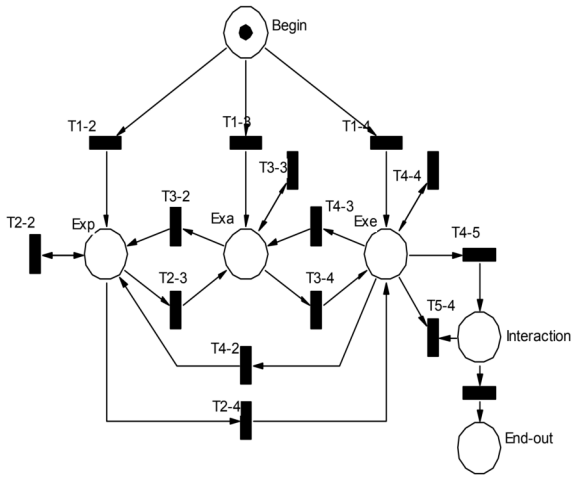

Figure 8: Petri Net Modelling the Problem Solving Protocol.

based on correctness exercises included in the contents units. Interface problem coordination is performed under the instructor responsibility, through a chat tool.
Content units: the bottom level OPN that implement the sub-problem and integration problem units are implemented using the FAST tool. Their general form is shown in Figure 8, where *Exp*, *Exa* and *Exe* are interaction units that present to the students, respectively, explanations, examples and exercises.

6 Conclusion

This paper has presented a method, based on ontologies and Petri nets, to allow the development of group learning in the context of *Intelligent Tutoring Systems (ITSs)*. While ontologies represent domain and student models in a shareable format, Petri nets formally specify group interaction protocols. The method is intended to be used with ITSs that are built using the FAST authoring tool. This allows the method to explore the student and domain models of these ITSs to increase the adaptiveness of the interaction. The method includes a library of group activities scenarios, previously defined by the system developers. To build a group activity instance, the teacher chooses one scenario, customizes its parameters and provides the contents of the activity. This information is compiled into an object Petri net that guides the group activity. The use and update of the group activity and student models is automatically included in the transitions of this Petri net.

Presently, the proposed method only allows the management of intra group activities, coordinating the tasks performed by the students that are member of a group. We intend to extend the method to allow the management of inter group activities, increasing the complexity of possible scenarios. Ongoing work also includes the enhancement of the student model attributes that are relevant to group activities and the development of further group activity scenarios.

As future work we intend to develop a friendlier authoring tool to develop these group activity scenarios.

Acknowledgements

This work was partially supported by CAPES/Cofecub (processes 400/02 and 0212/05-9) and CNPq (Brazilian National Research Council) (process 140005/2004-8).

References

[1] Alpert, S.R., Singley, M.K., Fairweather, P.G, 1999. Deploying intelligent tutors on the web: An architecture and an example. JAIE, 10:183–197.

[2] Brusilovsky, P. 2000, Adaptative hypermedia: From intelligent tutoring systems to web-based education. LNCS, Intelligent Tutoring Systems 2000.

[3] Cardoso, J., Bittencourt, G., Frigo, L.B., Pozzebon, E., Postal. A, 2004. MathTutor: A multi-agent intelligent tutoring systems. In 1st IFIP Conf. on AI Applications and Innovations, WCC´04, pages 22-27.

[4] Cardoso J., Bittencourt,G., Frigo, L.B., Pozzebon,E., 2004. Petri nets for authoring mechanisms. In XV Simpósio Brasileiro de Informática na Educação (SBIE'2004), ISBN 85-7401-161-4, pages 378–387, Manaus, AM, Brazil.

[5] Chen, W. & Wasson, B. 2004. Intelligent Agents Supporting Distributed Collaborative Learning. In: Lin, O., red. Designing Distributed Learning Environments with Intelligent Software Agents. IDEA Publishing Group;

[6] Constantino-González M., Suthers, D., Santos, J.E.G., Coaching Web-based Collaborative Learning based on Problem Solution Differences and Participation, in International Journal of Artificial Intelligence in Education, 2003, vol 13, 263 - 299

[7] Cost R.S., Chen Y., Finin T., Labrou Y.and Peng Y., Modeling Agent Conversations with Colored Petri Nets, In Proc of the Workshop on Specifying and Implementing Conversation Policies, Seattle, May 1999, pp. 59-66.

[8] Costa, E.B., Lopes, M.A., Ferneda, E, 1995. Mathema: A learning environment based on a multiagent architecture. In LNAI - Advances in Artifical Intelligence, volume 991, pages 141–150.

[9] Coutinho, L., Sichman, J.S., Boissier, O, 2005. Modeling Organization in MAS: A Comparison of Models. in: First Workshop on Software Engineering for Agent-oriented Systems, Uberlândia.

[10] Cuesta, P., Gomez, A., Gonzalez, J.C., Rodrıguez, F., A Framework for Evaluation of Agent Oriented Methodologies. The MESMA Approach for AOSE. Proceedings of Fourth Iberoamerican Workshop on Multi-Agent Systems (Iberagents'2002), at IBERAMIA'2002, the VIII Iberoamerican Conf. on Artificial Intelligence, Malaga, Spain.

[11] Dignum, M.V., V´azquez-Salceda, J., Dignum, F.P.M., 2004. OMNI: Introducing social structure, norms and ontologies into agent organizations. In P. Bordini & et al. (Eds.), PROMAS 2004 (pp.183-200). Heidelberg: Springer.

[12] Esteva, M., Padget, J., Sierra, C.,2001. Formalizing a language for institutions and norms. Proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001), pp 106–119.

[13] Ferber, J., Gutknecht, O., Michel,F.,2003. From Agents to Organizations: an Organizational View of Multi-Agent Systems. 4th Int. Workshop on agent-oriented software engineering, IV AOSE 2003, Melbourne, Australia.

[14] Frasson, C., Martin, L., Gouarderes, G., Aimeur, E.,1998. Lanca: A distance learning architecture based on networked cognitive agents. In Lectures Notes in Computer Science. Inteligent Tutoring Systems. Procedings of 4th International Conference, 1:594–603, August 1998.

[15] Frigo, L., Cardoso, J., Bittencourt. G., 2005. Adaptive Interaction in Intelligent Tutoring Systems. In : HT-2005, Intern. Workshop on Combining Intelligent and Adaptive Hypermedia Methods/ Techniques in Web-based Education Systems, Salzburg, Autriche.

[16] Hanachi, C., Sibertin-Blanc, C.,2004. Active Middle-Agents in Multi-Agent Systems. Dans : Autonomous Agents and Multi-Agent Systems, Kluve Academic PublishersNetherlands, V. 8 N. 3, p. 131 - 164, avril 2004

[17] Harrer, A., McLaren, B., Walker, E., Bollen, L., Sewall, J. (2005). Collaboration and Cognitive Tutoring: Integration, Empirical Results, and Future Directions. In C.-K. Looi et al. (Eds.), Proceedings of the 12th International Conference on Artificial Intelligence in Education (pp.266-273). Amsterdam: IOS Press.

[18] Hernandez-Dominguez,A.,1998. An Architecture of Cooperative Learning in a Distance Education context International Conference on Engineering Education, 1998, August 17-20, Rio de Janeiro, Brazil.

[19] Horling, B., Lesser. V.,2004. Quantitative Organizational Models for Large-Scale Agent Systems. Proceedings of the International Workshop on Massively Multi-Agent Systems,Melbourne, LNCS 2935, pp 214-230, December 2004. Kyoto, Japan

[20] Iglesias C., Garijo M., Gonzales J. C., A survey of Agent-Oriented Methodologie", in J.P. Müller, M..P. Singh and A. S. Rao, eds, Proceedings of the Fifth International Worshop on Agent Theorie, Architecture and Languages (ATAL 98), LNAI, vol 1555, Springer-Verlag, Heidelberg, 1999.

[21] Labidi, S., Souza, C.M., Nascimento, E., 2004. NETClass: Cooperative Learner Modeling in Web-Based Environment 6th International Conference on Computer Based Learning in Science CBLIS, Nicosia.

[22] Miao, Y., Pinkwart, N., Hoppe,U.,2006. Conducting situated learning in a collaborative virtual environment. Proceedings of the 5th International Conference on Web Based Education, pp. 7-12. Anaheim, CA: ACTA Press. 2006

[23] Mizoguchi, R., Bourdeau, J.,2000. Using ontological engineering to overcome common AI-ED problems. J. of AI in Education, 11(2):107–121, (2000).

[24] Odell,J., Nodine, M., Levy,R.,2005. A Metamodel for Agents, Roles, and Groups. Agent- Oriented Software Engineering (AOSE) V, James Odell, P. Giorgini, J¨org M¨uller, eds., LNCS, Springer, Berlin.

[25] Sibertin-Blanc, C., 1985. High-level Petri nets with data structures. In EuropeanWorkshop on Application and Theory of Petri Nets, pp 141–170.

[26] Silva, V., Choren, R., Lucena, C.,2004.A UML based approach for modeling and implementing multi-agent systems. In [AAMAS 2004], pages 914.921.

[27] Thibodeau, M., Belander, S., Frasson, C., 2000. White rabbit.matchmaking of user profiles based on discussion analysis using intelligent agents. Procedings of 5th International Conference, ITS 2000, 1:113–122, June 2000. Montreal/Canada.

[28] Zambonelli, F., Jennings, N., Wooldridge, M., 2001. Organisational Abstractions for the Analysis and Design of Multi-Agent Systems. In: Ciancarini P.,Wooldridge, M. (eds.): AgentOriented Software Engineering, LNCS 1957, Springer-Verlag.