

# SMODEL Server: Student Modelling in Distributed Multi-Agent Tutoring Systems

Juan-Diego Zapata-Rivera, Jim E. Greer

*ARIES Lab, Department of Computer Science, University of Saskatchewan*

**Abstract.** User modelling shells and learner modelling servers (LMS) have been proposed in order to provide a reusable user/student model over different domains. Open and inspectable student models have been investigated by several authors as a means to promote student's reflection, knowledge awareness, collaborative assessment, self-assessment, arrange groups of students, interactive diagnosis, and the use of students' models by the teacher. This paper presents SModel, a student modelling server used in distributed multi-agent environments. SModel server includes a student model database and a Bayesian student modelling component. SModel provides several services to a group of agents in a CORBA platform. In addition, SModel uses *ViSMod* to visualize and inspect distributed Bayesian student models. SModel has been tested in a multi-agent tutoring system for teaching basic java programming.

**Keywords:** Student Modelling Shells, Student Modelling Servers, Distributed Multi-Agent Systems, Open Student Models, Bayesian Learner Models.

## 1. Introduction

During the last few years, *multi-agent systems* (MAS) have been widely used as the target platform for developing many different kinds of computer-based systems. This trend can be attributed in part to remarkable advances in areas such as: hardware, communication technologies, AI, distributed systems, and human-computer interaction. Several kinds of Learning Environments (LE) and especially Intelligent Tutoring Systems (ITSs) have been influenced by this trend. Thus we can now find many examples of learning environments enhanced by agents (e.g. agents in collaboration, pedagogical agents, agents as matchmakers in help systems, personal agents, searching agents, etc.).

MAS impose new demands and offer new possibilities for student modelling design. The need for interaction among various specialised agents makes it necessary to provide support for special kinds of services in a distributed student modelling environment, such as:

- *Reusability.* Agents should be able to use the student model information and common inference mechanisms for their own benefit.
- *Scrutability.* Agents and humans should be able to interact with the student model in order to refine it and learn from it.
- *Sharing student model information.* Different agents create a different image (view) of the student based on their own experience and context. Agents should share specialised views of the student in order to provide better help and adaptive interaction.
- *Offering support for partial student modelling (student model fragmentation).* Agents should be able use inference mechanisms applied to their own partial view of the student. Agents must also integrate several views of the student in order to get a

comprehensive picture of the student (i.e. integration of SM fragments maintained by different agents).

Several authors have been dealing with these issues. *User modelling shells* have been proposed as a means to provide reusable user/student models over different domains. These shells often include tools to allow the programmer an easy interaction with the model [7, 8, 11, 12]. Kay [7] identifies three different actors to interact with the *model (the user, the machine, and the programmer)* and describes high level and fined-grained tools to support scrutability of user/student models. Machado et al. [9] propose a learner modelling server (LMS) that responds to KQML requirements made by pedagogical agents distributed over the system.

Most of the user/student modelling shells and servers need to know about domain knowledge to some degree in order to provide an accurate representation of the learner in a particular area (e.g. domain knowledge and beliefs). This situation makes shells and servers domain and application dependent. Once the domain knowledge has been included into the system (as part of the shell or as a parameter), UM shells' and/or UM servers' general capabilities to handle stereotypes, additional inference mechanisms, and visualization and inspection of learner models make them valuable tools for the success of any system that cares for the user.

Several authors have been investigating the effects of opening the student model to students and/or the teacher (inspectable student models) [2, 5, 6, 10, 11]. User modelling shells and/or servers provide support for *inspectable student modelling*. Visualization and inspection of student models can be done by implementing the appropriate set of interfaces to interact with the learner models maintained by the shell and/or server. Visualization and inspection can be seen as another service provided by the shell/server. Inspectability in student modelling shells/servers is an interesting field to investigate student's reflection, knowledge awareness, collaborative assessment, self-assessment, group formation, and interactive diagnosis.

This paper presents SModel, a student modelling server used in distributed multi-agent environments. SModel server includes a student model database and a Bayesian student modelling component. SModel provides several services to a group of agents in a CORBA platform. In addition, SModel uses *ViSMod* [20] to visualize and inspect distributed Bayesian student models. This allows students and teachers to interact with the model graphically. SModel has been tested in a multi-agent tutoring system for teaching basic java programming. Several agents including collaborative, instructional, and personal agents use SModel's services as part of their normal execution.

## **2. Bayesian Student Models**

Bayesian Belief Networks (BBNs) have become accepted and used widely to model uncertain reasoning situations and cause - effect relationships. Using prior and conditional probabilities attached to each node, it is possible to propagate changes in probability values on receipt of evidence [16]. The causal information encoded in BBNs facilitates the analysis of action sequences, observations, consequences, and expected utility [13]. BBNs offer a powerful technique to model students' knowledge by representing causal relationships among concepts and guaranteeing consistency of beliefs when new evidence (knowledge) is included in the model.

Several authors in different areas have explored the use of Bayesian belief networks to represent student models [3, 17, 14, 15, 19]. SModel server uses an adapted version of the belief net backbone structure for student models proposed by Reye [14, 15]. The belief net backbone structure can be used to reduce the computational complexity through local

propagation of beliefs and it offers a standard methodology to create Bayesian student models.

Bayesian student models in SModel can include information about knowledge, self-assessment and social aspects of the student in a three-level structure. The *first level (conceptual level)* covers a prerequisite structure of concepts. The *second level (assessment level)* consists of a set of topic clusters directly related to each of the nodes from the *conceptual level*. Finally, the *third level (social aspects level)* holds global nodes that represent general characteristics of the student that affect his/her learning process (e.g. confidence, competitiveness, cooperativeness, helpfulness, and eagerness). See Figure 1 - SModel server architecture.

### 3. SModel: A Student Modelling Server

SModel offers several services that involve a student model database (*SMDB*) and a Bayesian student modelling component (*BSM*). Agents in the system can interact with these two modules by invoking any of the services offered by SModel. SModel works as an independent component in a CORBA distributed systems environment. Students in SModel are represented by both a group of records in the SMDB representing personal information and/or preferences, and an overlay of a Bayesian student model provided by the agent or previously defined.

Bayesian student models for BSM are created following the three-layer Bayesian backbone structure explained above. Initial prior and conditional probability values are acquired by running pre-tests and mapping their results to cognitive stereotypes for which prior and conditional probability values have been already defined. Calibration of Bayesian student models is done visually by using ViSMod [20] (See section 5). Agents interacting with SModel can make use of any of the levels of a predefined Bayesian student model or from any other partial Bayesian student model being used by them.

SModel maintains a list of current tables registered in the SMDB. This list includes information about fields, data types and a short description of each table. A list of node names for registered Bayesian student models is also available through the SMDB. Agents can process any kind of standard database query on these tables. These tables include such things as students' personal information and learning preferences (i.e. learning style, preferred learning tools, etc.) The SMDB also stores a list of activities and/or quizzes associated with each node of the Bayesian student model.

SModel was implemented in java using CORBA to communicate with several agents and JDBC to provide a database connection. JavaBayes, a java-based package for Bayesian belief Networks created by Fabio Cozman [4] was adapted to handle queries in a distributed platform. Figure 1 depicts the architecture employed in SModel.

#### 3.1 Student Modelling Services

Agents interacting with SModel are entitled to use any of the following services:

- *RegisterBSM*. Agents who want to make public any Bayesian student model should register it to the server. The registration process requires an XML-based description of the Bayesian student model that contains information about nodes (i.e. name, type, description), network structure (i.e. parents and children), and prior probabilities. Registered Bayesian student models are automatically added to the SMDB for further inspection.
- *getNode*s. Agents can inspect current probability values for specific nodes or segments of a Bayesian SM. Agents can query previously defined Bayesian student models or

their own model (e.g. partial Bayesian student models). Queries can include nodes located on any of the three levels of the Bayesian backbone structure. This service requires the agent to specify the learner, and node(s) of interest. A particular cognitive stereotype and/or a new Bayesian partial model can be also passed as optional parameters to this service.

- *updateNodes*. Agents can include new evidence for specific nodes of a Bayesian student model. This information is automatically propagated throughout the model. After propagation, new probability values for any node in the model can be obtained by invoking the *getNode*s service. New evidence can be added to the current Bayesian student model by using stereotypes (a predefined set of probability values) or specific probability values for particular nodes in the model (a list of nodes and probability values).
- *processQuery*. Agents can query any of the tables maintained in the student model database (SMDB).
- *hypotheticalQuery*. Agents can create what-if scenarios. That is, after including a piece of hypothetical evidence and propagate it throughout the BSM, agents can retrieve new probability values for any node in the Bayesian student model. Agents can use this information to test several paths of action before taking a final decision. Hypothetical queries do not change the internal probability values in the model.

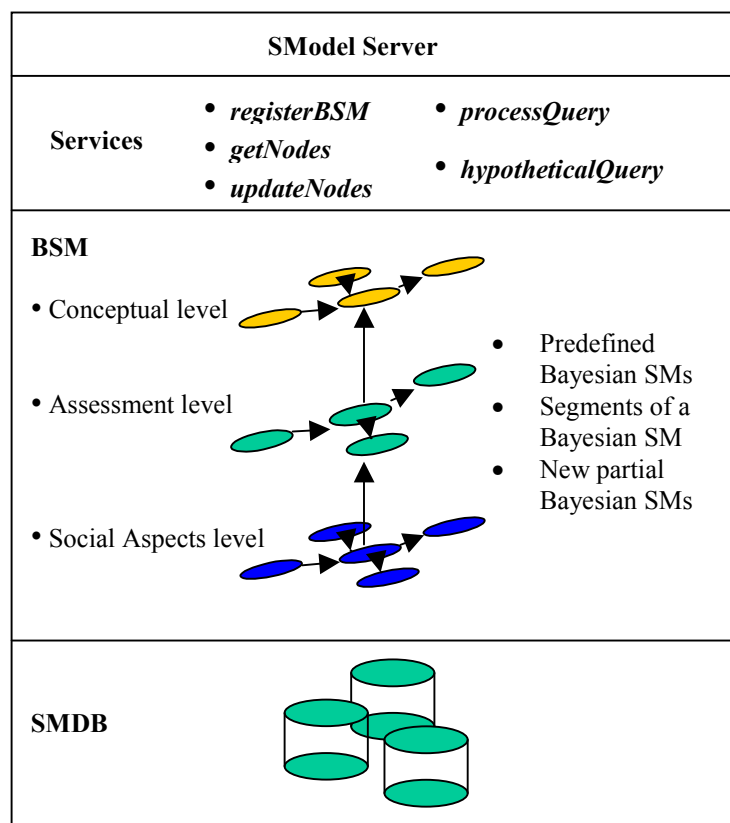


Figure 1. SModel server architecture.

#### 4. JTS: Java Tutoring System

A java tutoring system (JTS) [1] has been implemented as a distributed multi-agent environment using the services provided by SModel. When a student logs into JTS, a *personal agent* (MAJI - a Microsoft-based agent) is assigned to help him/her to interact with the system. Students' interactions in the system are monitored and directed to the appropriate agent throughout the CORBA bus. In order to reach their goals, several agents interact by

sending messages to each other. *SModel server* offers several services that allow the agents in the system to inspect and retrieve students' information from the student model database or from the Bayesian student model. For example, a *pedagogical agent* uses information about a particular student's knowledge level and navigation history to recommend pages, quizzes and activities. The *collaborative agent* suggests collaborative activities and partners with whom to collaborate. Using *ViSMod*, students can interact with the Bayesian student model, analyse what information is being gathered and inferred by each of the agents, and update it in case of disagreement. Figure 2 shows how these components/agents are connected using a CORBA bus.

In the JTS implementation, agents and components developed in different programming languages (i.e. java, visual basic and C++) are connected in a distributed environment using CORBA. In this distributed environment it is possible to replicate agents/components in order to distribute the overall load to different components in the system.

#### 4.1 Student Model Database (SMDB) in JTS

Students' personal information and preferences are stored in the SMDB. Agents can access this information by using *processQuery*, one of the services provided by SModel. Current information handled by SModel in the SMDB for JTS includes:

- students' personal information (ID, password, name, email, age, gender, and cognitive stereotype),
- students' collaborative preferences (tutor or tutee, synchronous or asynchronous collaborative tools, cooperative or collaborative, collaboration state, anonymity, human or artificial partners, and group role),
- description of previously registered Bayesian student models (nodes, structure, and description),
- activities (type of activity, and concepts associated with each activity),
- quizzes (type of quiz, and concepts associated to each quiz),
- and students' current knowledge level on each of the nodes from the conceptual level of the BSM.

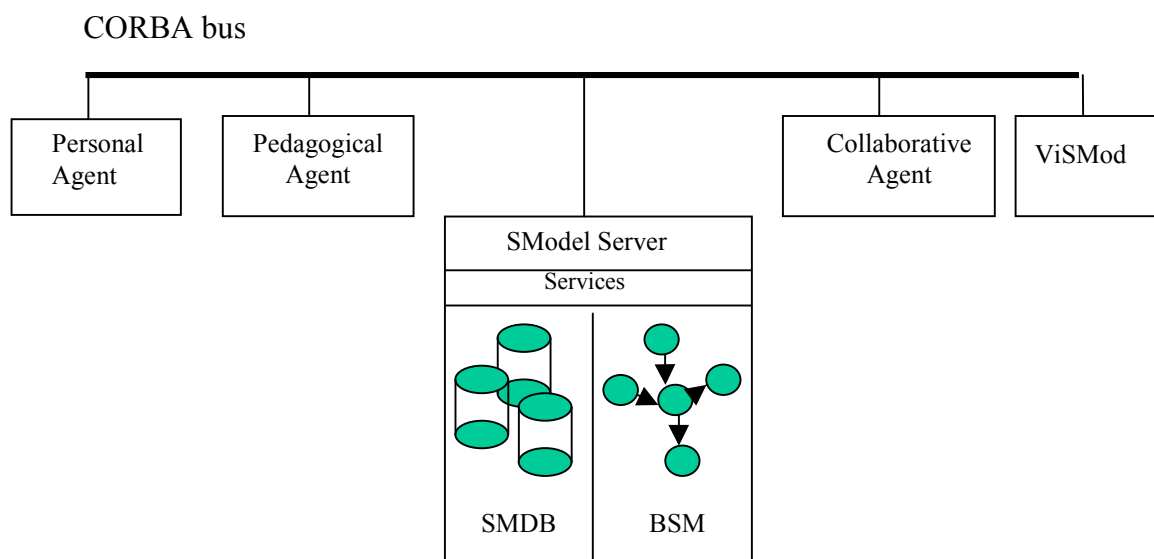


Figure 2. SModel attending to requests from several agents/components in JTS

## 4.2 Interacting with several agents

Agents in JTS need to interact with different pieces of the student model. JTS involves one personal agent per learner. In addition the system has one pedagogical agent, one collaborative agent, and a visualization component (but these can be easily replicated in this distributed multi-agent system to scale up the system).

*MAJI*, the personal agent for a particular learner, initialises the student model (*conceptual level of the BSM*) by determining an initial stereotype based on the results of a pre-assessment test. MAJI also helps the student to update his/her personal information and preferences in the student modelling database (SMDB).

The *pedagogical agent (instructional planner)* updates the student model (*conceptual level of the BSM*), by monitoring the student's behaviour (navigation, quizzes, and activities) throughout the tutorial. It can also change the student's stereotype or include evidence for specific concepts of the model. The pedagogical agent uses the student's current state of knowledge to suggest links, activities, and quizzes.

Finally, the *collaborative agent* suggests collaborative activities and/or possible collaborative partners based on the third level (*social aspects*) and the first level (*conceptual level*) of the Bayesian student model. The collaborative agent also uses information from the SMDB, such as: students' stereotype, activities, quizzes and collaborative preferences. Figure 3 shows a screenshot of JTS, in which MAJI presents the content of the tutorial and the initial menu to the student.

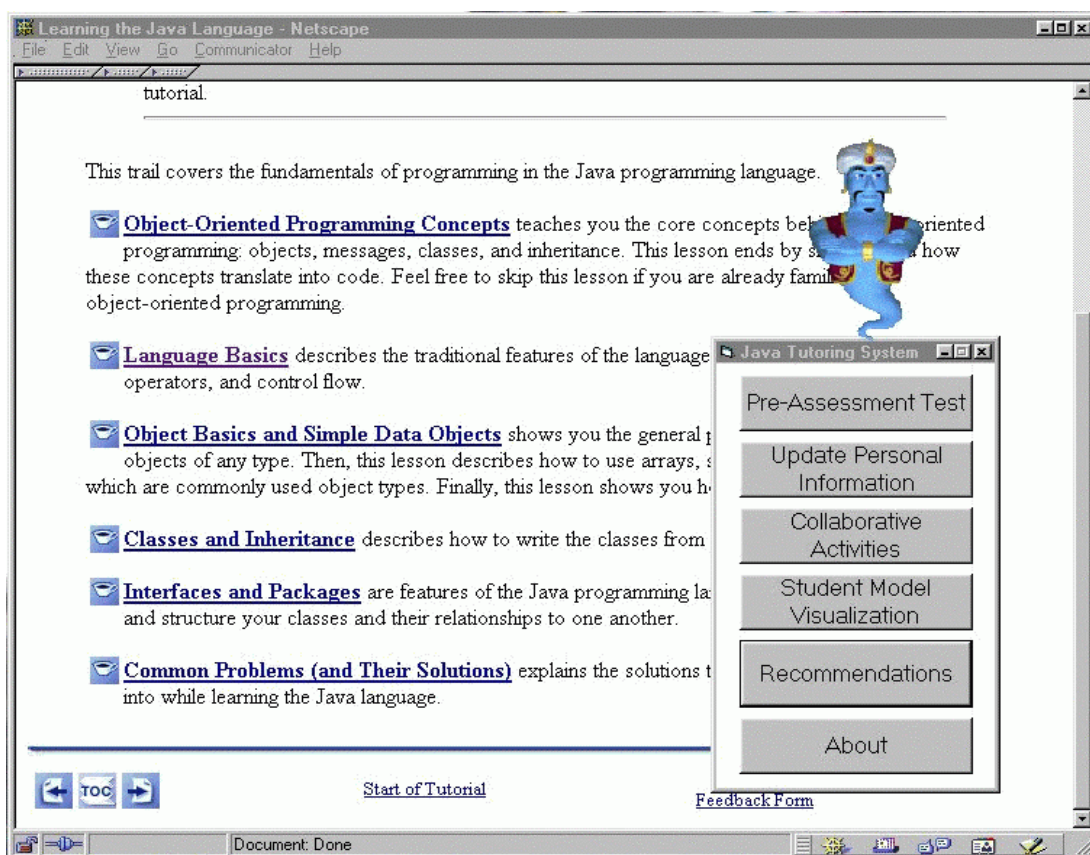


Figure 3. A screenshot of JTS.

## 4.3 Bayesian Student Models used in JTS

JTS uses a prerequisite concept structure of 46 nodes for teaching basic java programming. Figure 3 shows a fragment of the *conceptual level* used by JTS. The second level (*assessment*

level) in JTS is comprised of the nodes *student-claims-to-know(concept)*, *student-demonstrates-to-know(concept)*, and *student-expresses-interest(concept)*. The *social level* in JTS depicts a tentative set of relationships among confidence, competitiveness, cooperativeness, assertiveness, eagerness, and helpfulness. Propagation of probabilities from one level to another is done through the node connected to the backbone structure. In this case the *Social\_Learning* node (*social aspects level*) connects to *S\_Assessment(concept)* node (*assessment level*), and this one to all of the nodes from the first level (*conceptual level*). JTS uses a different overlay of conditional and prior probabilities for each cognitive stereotype in the system (beginner, intermediate, and advanced). Figures 4, 5, and 6 show each of the levels used by JTS in its Bayesian student model. These figures were produced using ViSMod.

## 5. Visualizing Bayesian Student Models (ViSMod)

One of the main advantages of Bayesian belief networks is that they provide an inspectable cause and effect structure among their nodes and direct specification of probabilities in the model [19]. Using BBNs, assessment of students' knowledge can be carried out effectively.

*ViSMod* [20], a visualization tool for distributed Bayesian student models, has been successfully integrated into SModel in JTS. ViSMod opens not only the internal representation of the student's knowledge, but also the mechanisms to update it to the human (teacher or learner) who wishes to know more about the system's knowledge of the learner.

Using ViSMod, learners and teachers can visualize the student model using various visualization techniques (e.g. colour, size, proximity (closeness), link thickness, and animation) to represent the influence of one node on another or the likelihood of a node being known.

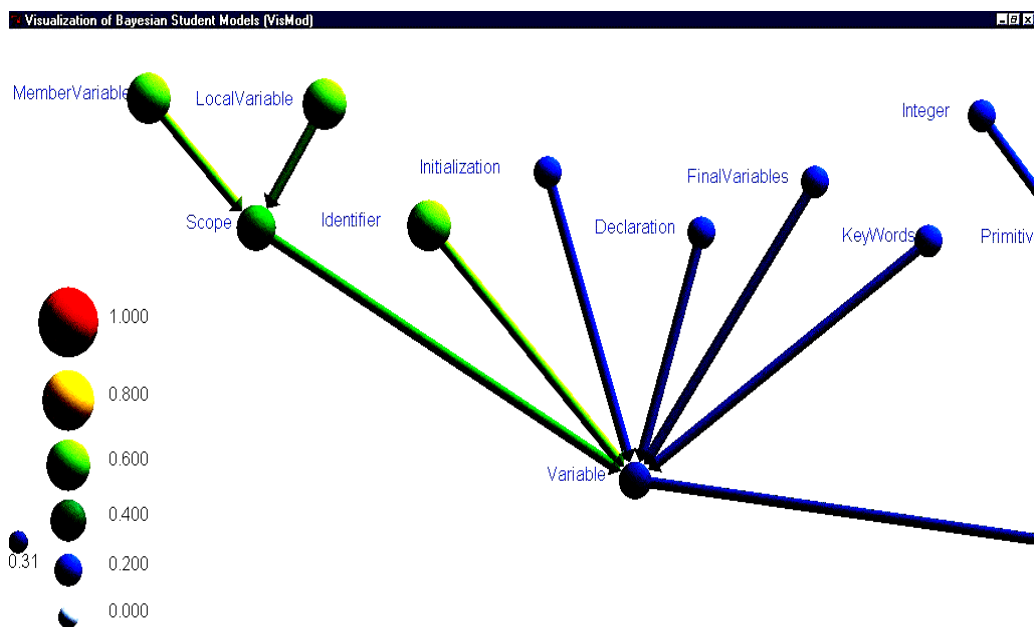


Figure 4. A fragment of the *conceptual level* used by JTS. In this example size and colour are used to show marginal probability values representing the student's knowledge on a particular concept (e.g. 0.31 represents the probability of the student knowing *FinalVariables* in java).

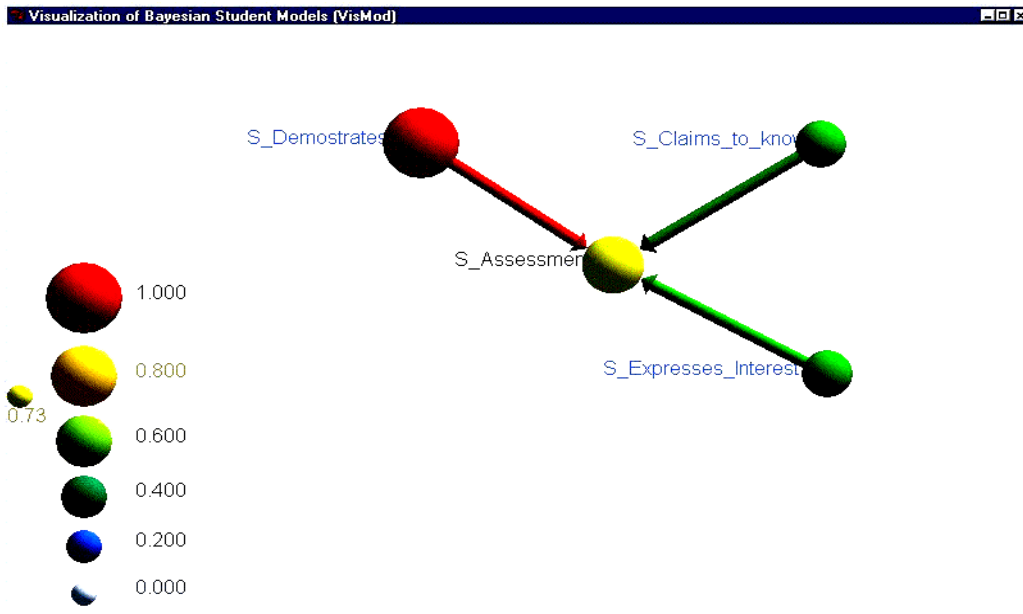


Figure 5. *Assessment level* used by JTS. 0.73 represents marginal probability of the student's assessment ( $S\_Assessment$ ). This value is propagated to a particular node of the conceptual level. The  $S\_Assessment$  node shows how students' opinions and claims are taken into account to determine his/her knowledge.

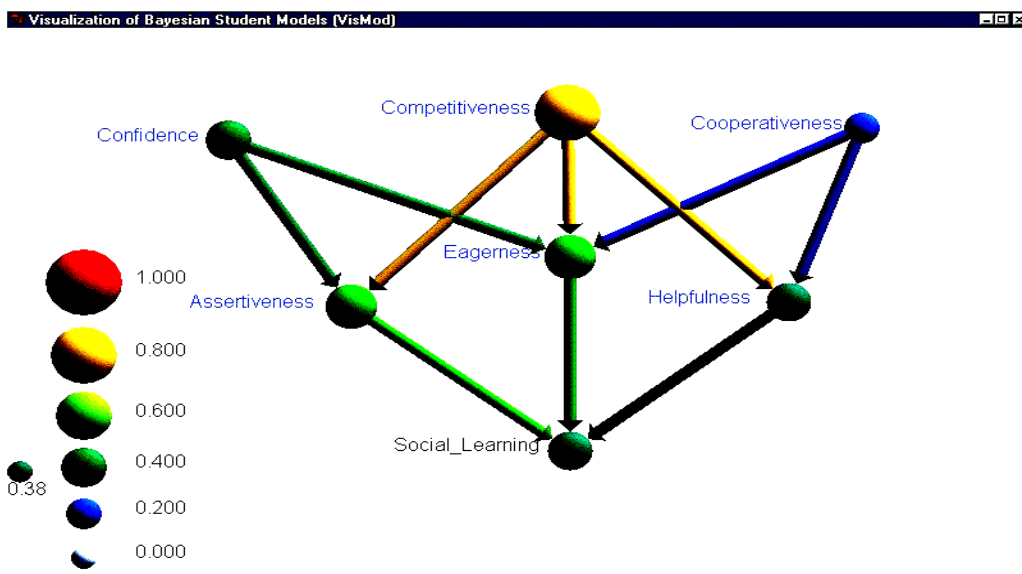


Figure 6. *Social level* used by JTS. 0.38 represents the marginal probability value of  $Social\_Learning$ . It will be propagated to the nodes in the conceptual level.

Some of the benefits that ViSMoD provides to students and teachers are:

- ViSMoD provides a graphical representation of the student model that makes it easier for students to understand Bayesian student models.
- ViSMoD supports multiple views of the student model that makes it possible to inspect, modify and create interesting representations of the learning process.
- By allowing inspection of student models and the creation of what-if scenarios, ViSMoD aims to support students' reflection, knowledge awareness, and refining of student models.

- Finally, ViSMod allows visualization of distributed Bayesian student models with different levels of granularity using several sources of evidence.

## 6. Conclusions and Future work

SModel offers the possibility to integrate a Bayesian student modelling (BSM) component, a student model database (SMDB), and visualization and inspection of Bayesian student models (ViSMod) in a distributed environment. Systems/agents that need to know about the student can use any of the available SModel services through CORBA. Because of its multi-agent, distributed architecture, SModel Server can readily be replicated for large-scale applications and can be used across a wide range of student modelling applications. Although SModel provides a list of nodes with information of the current Bayesian student models, we recognize the need for a common ontology that supports communication among agents.

SModel can be used in the creation of different agent-based tutoring systems. It would be necessary, however, to use a different *conceptual level* (a domain dependent conceptual structure) and adapt some of the SMDB information, such as: new activities and quizzes.

SModel services support the use of partial models and integration of different pieces of evidence. Agents are able to use inference services with either a general Bayesian model or a partial model provided by a single agent. Agents are able to integrate information about several aspects of the student and use the resulting structure (after new evidence has been propagated) as a valuable source of information to reach their goals.

Not only agents can within JTS use the services offered by SModel, but also external agents can be added easily into the system. A multi-agent platform makes relatively easy the cumbersome task of adding, replicating and removing agents in the tutoring system.

By using ViSMod, JTS aims to support students' reflection, knowledge awareness, and refining of student models. SModel was successfully integrated into JTS. Although JTS has not been widely used, it has been tested within the ARIES lab. These tests were sufficient to assure us of the robustness of SModel for this domain. We hope to make JTS available to be used by first year students in computer science courses at the University of Saskatchewan.

SModel will be integrated into I-Help [18]. This will allow us to experiment with different matchmaking algorithms, some of them using Bayesian inference as opposed to simple linear models. In addition, SModel capabilities to support partial models can be used by the personal agents in I-Help to reason about learners in different contexts.

## Acknowledgements

We wish to recognize both the Natural Science and Engineering Research Council of Canada and COLCIENCIAS - Colombia for financial support of this research. We also thank Professor Gord McCalla, Prachi Agarwal, Jeff Solheim, and Mike Winter who worked on the design and implementation of JTS.

## References

- [1] Agarwal, P., Solheim, J., Winter, M., Zapata, J.D. (2000) JTS: A Java Tutoring System. *Aries Lab Technical Report*, University of saskatchewan.
- [2] Bull, S. & Shurville, S. (1999) Cooperative Writer Modelling: Facilitating Reader-Based Writing with Scrawl, in *Proceedings of the workshop 'Open, Interactive, and other Overt Approaches to Learner Modelling' at AIED'99*.
- [3] Conati, C., Gertner, A.S., VanLehn, K., Druzdzel, M.J. (1997) Online student modeling for coached problem solving using Bayesian networks. In Jameson, A, Paris, C., and Tasso, C., (eds), *Proceedings of the sixth International Conference on User Modeling*. New York: Springer-Verlag. 231-242.
- [4] Cozman, F.(1998) JavaBayes Bayesian Networks in Java. Available on-line: <http://www.cs.cmu.edu/~javabayes/>

- [5] Dimitrova, V., Self, J., Brna, P. (1999) STyLE-OLM – an interactive diagnosis tool in a terminology learning environment. In *Proceedings of the workshop 'Open, Interactive, and other Overt Approaches to Learner Modelling' at AIED'99*.
- [6] Greer, J., Zapata, J. D., Ong-Scutchings, C., Cooke, J. E. (1999) Visualization of Bayesian Learner Models. In *Proceedings of the workshop 'Open, Interactive, and other Overt Approaches to Learner Modelling' at AIED'99*. pp. 6-10.
- [7] Kay, J. (1998) A Scrutable User Modelling Shell for User-Adapted Interaction. Ph.D. Thesis, Basser Department of Computer Science, University of Sydney, Sydney, Australia.
- [8] Kobsa, A., & Pohl, W. (1995) The User Modelling Shell System BGP-MS. *User Modelling and User Adapted Interaction*. Kluwer, vol. 4, 2, pp. 59-106.
- [9] Machado, I., Martins, A., & Paiva, A. (1999) One for All and All in One. A learner modelling server in a multi-agent platform. In Kay, J. (eds), *proceedings of the seventh international conference UM99*. Springer-Verlag Wien. New York. pp. 211-221.
- [10] Muehlenbrock, M. et al. (1998) A framework system for intelligent support in open distributed learning environments. *International Journal of Artificial Intelligence in Education*. IJAIED'98. 9, pp. 256-274.
- [11] Paiva, A., Self, J. & R., Hartley (1995) Externalising learner models, *Proceedings of World Conference on Artificial Intelligence in Education*, Washington DC, pp. 509-516.
- [12] Paiva, A., & Self, J. (1995) TAGUS – A User and Learner Modeling Workbench. *User Modelling and User Adapted Interaction*. Kluwer, vol. 4, 3, pp. 197-228.
- [13] Pearl, J. (1988) Probabilistic reasoning in intelligent systems: networks of plausible inference. San Mateo, CA: Morgan Kaufmann.
- [14] Reye, J. (1999) Student Modelling based on Belief Networks. *International Journal of Artificial Intelligence in Education*, vol. 11.
- [15] Reye, J. A (1996) Belief Net Backbone for student modelling. In Frasson, C., Gauthier, C. and Lesgold, A. (eds.) *Intelligent Tutoring Systems. ITS'96, Montreal, Canada*. Berlin:Springer-Verlag, pp. 596-604.
- [16] Russell, S., and Norving, P. (1995) *Artificial Intelligence: A modern Approach*. Prentice Hall, New Jersey.
- [17] VanLehn, K. & Martin, J. (1997) Evaluation on an assessment system based on Bayesian student modeling. *International Journal of Artificial Intelligence and Education*, Vol.8. 179-221.
- [18] Vassileva, J., Greer J., McCalla, G., Deters, R., Zapata, D., Mudgal, C., Grant, S. (1999) A Multi-Agent Approach to the Design of Peer-Help Environments, in *Proceedings of Artificial Intelligence in Education AIED'99, Le Mans, France*, July, 1999, pp. 38-45.
- [19] Villano, M. Probabilistic Student Models: Bayesian Belief Networks and Knowledge Space Theory. In Frasson, C., Gauthier, C. and McCalla, G. (eds) *Intelligent Tutoring Systems ITS'92, Montreal, Canada*, Berlin: Springer-Verlag, pp. 491-498.
- [20] Zapata-Rivera, J.D. & Greer, J. (2000) Inspecting and Visualizing Distributed Bayesian Student Models. In Gauthier, G., Frasson, C. and VanLehn, K. (eds) *Intelligent Tutoring Systems ITS 2000, Montreal, Canada*. , Berlin: Springer-Verlag, pp. 544-553.