Inspectable Bayesian Student Modelling Servers in Multi-Agent Tutoring Systems

JUAN-DIEGO ZAPATA-RIVERA, JIM GREER

ARIES Lab, Department of Computer Science, University of Saskatchewan.
Saskatoon, Saskatchewan, Canada.
{Diego.Zapata,Jim.Greer}@usask.ca

User modelling shells and learner modelling servers (LMS) have been proposed in order to provide reusable user/student model information over different domains, common inference mechanisms, and mechanisms to handle consistency of beliefs from different sources. 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 Bayesian 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. Users can use ViSMod, a Bayesian student modelling visualization tool, and SMV, a student modelling database viewer, to visualize and inspect distributed Bayesian student models maintained by SModel server. SModel has been tested in a multi-agent tutoring system for teaching basic java programming.

KEYWORDS: Student Modelling Shells and Servers, Distributed Multi-Agent Tutoring Systems, Inspectable 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 (Key, 1998; Kobsa & Pohl, 1995; Paiva et al.
1995; Paiva & Self, 1995). Kay (1998) 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. (1999) propose a learner modelling server (LMS) that responds to KQML requirements made by pedagogical agents distributed over the system.

Several authors have been investigating the effects of opening the student model to students and/or the teacher (inspectable student models) (Bull & Shurville, 1999; Dimitrova et al., 1999; Greer et al. 1999; Müehlenbrock 1998; Paiva et al., 1995). 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 Bayesian 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. Users can interact with Bayesian student models maintained by SModel using ViSMod -Visualization of Bayesian Student Models (Zapata-Rivera & Greer, 2000) and SMV -Student Modelling Viewer. 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. Student Modelling Shells and Servers

User/student modelling shells offer reusability of user/student models and at the same time provide different tools to manipulate them. According to Kay (1998) user modelling shells should have five essential properties, that is:
• representation for the user model components probably using different types of knowledge;
• support for the use of evidence from several sources including the system;
• support for inference within the model;
• mechanisms for handling uncertainty, noise, and inconsistency;
• and broad applicability across domains.

In order to create a student/user modelling shell, it is necessary to select the most popular demands for a user model and condense them in an application-independent user model called user modelling shell system; similar research can be found in the area of expert shell systems. Several student/user modelling shell systems have been developed. In this section, we review some of the main characteristics of some of the existing user/student modelling servers and shells.

The Generalised User Modelling System, GUMS (Finin, 1989) allows just one type of assumptions at the same time (usually knowledge). It accepts queries and new facts. It also reports a list of inconsistencies to the application system that is using it. It maintains a stereotype hierarchy, which is used to accommodate users according to the level of specialisation without inconsistencies. Since it does not store the result of inferences, it repeats the same inferences when a query is repeated. This can be impractical in real applications. An important contribution of Finin’s work is the creation of four general points to take into account when designing a student/user model:

• Who is being modelled (i.e. degree of specialisation and temporal extent of the student/user model.)
• What is being modelled (i.e. goals, plans, capabilities, attitudes, and knowledge or beliefs.)
• Model acquisition (i.e. implicit and explicit) and maintenance (i.e. how new information is added to an existing model and how inconsistencies are handled.)
• Why is the model there. How the student model will be used (i.e. understanding the user's information, seeking behaviour, providing help and advice, getting input from the user, providing output to the user.)

In UMT – User Modelling Tool (Brajnik & Tasso, 1992) stereotypes are defined by the developer as a set of attribute values pairs. Each stereotype contains an activation condition. Stereotypes can be
organized in arbitrary hierarchies. UMT accepts assertions and classifies them as invariable premises or retractable assumptions. It contains a reason maintenance module to determine all possible models which are composed by all the consistent sets of premises and assumptions in the system. UMT’s inference mechanism is carried out in a forward-chaining fashion. The main characteristic of this system is its mechanism for recording inferential dependencies between assertions that helps to determine the maximum number of consistent assumptions available for a particular user.

The Belief, Goal and Plan Maintenance System, BGP-MS (Kobsa & Pohl, 1995) allows the representation of several types of information at the same time (e.g. knowledge and goals). It supports user’s interviews and inferences about the user by the system. It is independent of the programming language and its inference mechanisms are bi-directional. BGP-MS uses inter-process communication technology to connect the user modelling shell with different applications and users. A drawback of UMT and BGP-MS is that stereotype retraction is not integrated into a truth maintenance scheme. PROTUM (Vergara, 1994) provides a more sophisticated mechanism for stereotype retraction but most of UMT characteristics also apply to this system.

TAGUS (Paiva & Self, 1994) is classified as a user and learner modelling services provider. It has been used in several ITSs and other kinds of interactive learning environments. Some of the services provided by TAGUS are: providing information in form of justifications, updating the content of the user model using several sources of evidence, and simulating processes. It uses first-order formulas to describe different aspects about the user and some meta-operators to classify each aspect of the model (e.g. ‘belief’, and ‘goal’). TAGUS includes an inference mechanism, a truth maintenance module, a diagnostic component that uses a library of misconceptions, and two interfaces. The user or learner modelling module (ULM) on TAGUS considers beliefs, goals, problem solving capabilities, and problem solving strategies. ULM is divided in three levels: belief level, reasoning level, and monitoring level. TAGUS accepts several queries including user simulations and diagnosis of unexpected behaviour. TAGUS uses several viewers to externalise the learner model to the learner in order to motivate reflection about the learner’s problem solving strategies.
THEMIS (Kono et al., 1994) and the student modelling maintenance system, SMMS (Huang et al. 1991) focus on dealing with inconsistency of beliefs. THEMIS handles the most common kinds of inconsistencies in four different categories as follows: inconsistencies due to changes in the student (e.g. learning), slips and careless errors, contradictory beliefs held, and inconsistencies due to system inference from changed ground beliefs. Beliefs are divided into different consistent worlds organized in a concept discrimination tree. ATMS belief revision processes are used within each world to remove any inconsistency. SMMS uses justified beliefs to represent deductive knowledge and a stereotype hierarchy for stereotypical knowledge. In SMMS new evidence is used to update the deductive knowledge base, belief revision is performed to ensure consistency, and finally a stereotype is activated or deactivated from the stereotype hierarchy.

The User Modelling Shell, UM (Kay, 1995) offers a simple architecture that has been tested in two different applications: a coaching system for a text editor application, and a movie advisory system. UM uses attribute-based structures called components to represent users’ preferences, beliefs, and knowledge. Each component has associated an uncertainty value, a source of evidence and a type of observation. UM includes several application-independent and application-dependent tools for constructing, inspecting, and refining user models, such as: user interviewing, knowledge elicitation, model verification by the user, user observation, and basic user model evaluation. UM organises components in a hierarchy of topics called ‘partial models’ that allow an easy inspection by the user. Different levels of granularity (Greer & McCalla, 1989) are provided for the user to view their hierarchy of partial models. Users in UM can modify the content of the user model and decide which information to share. Using the viewer tools on UM it is possible to add explanations and justifications on each of the leaf nodes components in the hierarchy. UM does not override the user model information with new information from the user. New information is considered as additional evidence that can be integrated into the model.

DOPPELGANGER (Orwant, 1991; Kobsa & Orwant, 1995) belongs to a special kind of user models called ‘user model servers’. User model servers offer several user modelling services to several users and applications on a network. DOPPELGANGER does not focus on any particular cognitive model. By the
contrary, it focuses on gathering users data from several applications and extrapolate this information using different learning techniques, such as: beta distributions, linear prediction, Markov models, and unsupervised clustering to form probable communities (stereotypes). Users in DOPPELGANGER can connect to the server and view or modify their models.

Some advantages of centralised user model servers are:

- Information is processed in a powerful centralised server. This reduces the need for complex computational processes on the client machine.
- It is possible to integrate and/or share information (evidence) gathered by different applications about the user. This allows us to understand users’ behaviour while working on different environments.
- It is more convenient to update user modelling information (e.g. stereotypes, or inference rules) on a central server rather than on each client.

It is important to take into account security risks involved in this kind of architecture. This is why DOPPELGANGER uses the server’s authentication mechanisms to control access to its resources. DOPPELGANGER lets the user to decide what information is going to be private or public.

In a more recent paper, Kobsa (2001a) describes additional advantages that client-based user server architectures offer, such as: these architectures makes it possible to avoid redundancy and to handle consistency and coherence of user modelling information more easily; it is possible to apply methods of security, identification, authentication, access control and encryption; and complimentary and disperse information can be integrated more easily. Among the disadvantages he mentions: the dependency on a network connection and the potential of failure in the central server.

Most of the user/student modelling shells and servers need to know about domain knowledge in 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, contradictory beliefs, additional inference mechanisms, and visualization and inspection of learner models make them valuable tools for
the success of any system that cares about the user. When choosing a particular shell it is important to know that more features do not imply better usefulness. Selection of shells should be done based on needs of the system being built (Kobsa & Pohl, 1995).

As a user modelling server, SModel offer many advantages to a group of agents interested in accessing the user information in a distributed environment. Agents interacting with SModel can use their own view of the user and share general information, that is: agents can use inference mechanisms provided by SModel with their own partial user models. In addition, SModel provides tools for humans to interact with the model and take control on their own information. Sections 3 and 4 describe SModel server.

3. Bayesian Student Models

Numerical techniques for uncertainty have been applied to user/student modelling extensively specially in the last 10 years. Quantitative approaches can be used in conjunction to qualitative techniques in order to handle uncertainty. Numerical techniques, such as Bayesian Belief Networks (BBNs), Dempster-Shafer theory of evidence (DST), and fuzzy logic (FL) are some of the most broadly applied (Jameson, 1996).

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 (Russell & Norvig, 1995). The causal information encoded in BBNs facilitates the analysis of action sequences, observations, consequences, and expected utility (Pearl, 1998). 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 (Conati et al., 1997; VanLehn & Martin, 1997; Reye, 1996, 2001; Villano, 1992). Mayo (Mayo, 2001) classifies Bayesian student modelling approaches based on how the structure of the network and prior, conditional probabilities are elicited. Mayo identifies three types of Bayesian student
models: expert-centric, efficiency-centric, and data-centric. Expert-centric models use experts to specify initial prior and conditional probabilities (Mislevy & Gitomer 1996; Conati & VanLehn 1996; Gertner & Venlehn 2000). Resulting networks usually contain a big number of variables. It makes difficult to evaluate the model. Efficiency-centric models restrict the structure of the network in order to maximise efficiency (Reye, 1996; Murray 1999; Collins et al. 1996). There are several risks with this approach, such as: oversimplifying the model and/or introducing incorrect assumptions. Finally, data-centric models use data from previous experiments and/or pre tests to generate the network and its probabilities (Stern et al. 1999; Mayo 2001). Although the resulting network and probabilities can be used efficiently in implementing ITSs, it does not guaranty that the final structure can be easily inspectable and understood by a human.

SModel server uses an adapted version of the belief net backbone structure for student models proposed by Reye (1996). 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 of learning level) holds global nodes that represent general characteristics of the student that affect his/her learning process (e.g. students’ confidence, competitiveness, cooperativeness, helpfulness, eagerness, and others). Teachers who are interested in including social aspects of learning as part of the Bayesian model can use this level to integrate a causal model of selected social aspects. See figure 1 - SModel server architecture.
4. 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.

![Figure 1. SModel server architecture.](image)

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. 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 (see figure 1).

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. Information in the SMDB has been adapted to contain a subset of the learner model standard –PAPI Learner - proposed by the IEEE 1484.2 Learner Model Working Group (PAPI Learner, 2000) This standard includes information such as students’ personal information, preferences, relations, security, performance, and portfolio. The SMDB also stores a list of activities and/or quizzes associated with each node of the Bayesian student model.

---

**Figure 2.** SModel general platform. Several agents interacting with SModel. ViSMOD and SMV are available for human scrutiny of their models.

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 (Cozman, 1998) was adapted to handle queries in a distributed platform. Figure 2 depicts the platform in which SModel server is used. Several agents connected through CORBA can access the SMDB and the BSM using the services provided by SModel. In addition, humans can use ViSMod and SMV to interact the their student models directly. ViSMod and SMV are described in sections 4.2 and 4.3 respectively.

4.1. STUDENT MODELLING SERVICES PROVIDED BY SMODEL

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.

- **getNodes.** 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 getNodes 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.

4.2. VISMOD – VISUALIZING AND INSPECTING BAYESIAN STUDENT MODELS

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 (Villano, 1992). Using BBNs, assessment of students’ knowledge can be carried out effectively.

ViSMod (Zapata-Rivera & Greer, 2000), a visualization tool for distributed Bayesian student models, has been successfully integrated in SModel general platform. 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. Section 5.3. shows some figures in which ViSMod is used to inspect Bayesian student models in a java tutoring system.

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.

4.3. SMV - STUDENT MODELLING DATABASE VIEWER

Students should be in control of their own information. In order to facilitate students’ interaction with the SMDB, we developed a student modelling database viewer. Using SMV, students can define the actors (i.e. persona agents, teachers, peers, external agents, etc.) who can interact with information in the student model (i.e. good or bad relationships). Every actor or entity that wishes to access a student model should establish a disclosure agreement with the owner of the model. This is important to keep track of the kind of relationship the student maintains with each entity. Indeed, students can terminate a particular relationship if they consider that there are reasons that justified it. This agreement defines what an actor can or can not do with information from the student model.

Students can establish whether a variable is considered as private or not. It is also possible for students to grant actors access to particular shareable variables (i.e. security). For example, actors can be authorised to read, update, and/or share the content of student model variables.

Figure 3 shows a screenshot of SMV. Student model variables have been classified in different categories according to the PAPI learner model standard and the requirements of SModel. Categories appear in the upper side of the figure as an independent tabular option in the tabular panel. These categories are as follows: personal information, preferences, assessment, portfolio, security, relationships, activities and quizzes, and current/registered Bayesian student models. The figure shows variables from the personal information category. It includes information about privacy of each variable and a log of changes or history (i.e. who change the variable, and when and how was it changed).
4.4. USING SMODEL WITH DIFFERENT TUTORING SYSTEMS

SModel supports the creation of agent-based tutoring systems and distance education systems. Software agents and humans distributed in different locations can interact with Bayesian student models through CORBA. Student modelling information is available to different agents through the services provided by SModel. Bayesian student models are represented in XML and information from the student database module follows the PAPI learner standard. This facilitates sharing of information with different applications.

In order to create an ITS following the Bayesian backbone structure, it is necessary to define the conceptual level (a domain dependent conceptual structure) and adapt some of the SMDB information, such as: new activities and quizzes. Nodes and structure of the assessment level and global social aspects of learning level are similar across many domains.
It is recommended that teachers or instructional developers define an initial Bayesian model based on the concepts the system aims to teach. This initial conceptual level can be as detailed or as general as needed by the system, and the teacher. As said before, the initial Bayesian model can be created by the teacher based on a prerequisite conceptual structure taken from a textbook or from a tutorial system directly. It is important to take into consideration that students will be interacting with their model, so an intuitive model will help both the student and the system. Students will be able to interact with the model and the system will use students’ feedback to improve the interaction with the student.

By using ViSMod, teachers can keep track of evidence being gathered by different ITSs. Bayesian models will evolve with the interaction of students, teachers and agents from different systems. For example, it is possible that students suggest new arrangements of the model and/or new concepts to be included into the model. Indeed, teachers should keep track of the evolution of Bayesian models. Teachers can use this information to adapt their teaching strategies in order to facilitate learning for their students.

Pre-assessment quizzes can be used to initialise the model. Pedagogical activities can be defined and associated to different nodes of the model using SMV. This can be done in advance. Pedagogical agents and planners can be constructed to manipulate this information in order to facilitate learning. Bayesian student models maintained by SModel server become instrumental in determining the behaviour of different pedagogical agents.

4.5. SECURITY AND PRIVACY ISSUES

Ensuring privacy is an important issue for any system that handles personal information. Different countries have created their own privacy laws. These laws have a huge impact on the design, implementation and usage of user/student modelling servers (Kobsa, 2001b). Users in SModel have full control on their models. It includes access to the methods used to update variables in their models and information about who and when changed a particular variable.
SModel provides different tools (ViSMod and SMV) to ensure user can access their own information. SMV allows users to determine who has access to what information. It is also possible to establish different levels of access based on who intends to use the information and what type of information is going to be used. Figures 4 and 5 show how SMV allows users to manage their relationships and to control access to their information.

Due in part to the academic context in which SModel server was developed, we did not explore more sophisticated mechanisms to ensure security and privacy of the user information. Future work in this area will lead us to implement encryption mechanisms and other security measures needed to successfully deploy this system in other contexts.

4.6. CONSISTENCY

Consistency is an important factor to be considered in user/student modelling servers. Fink (1999) describes several transaction management methods to deal with transactional consistency in the user model.
Because SModel has been implemented Java to run in a CORBA platform, different technologies such as Java Transaction Service (JTS), a java implementation of the CORBA Object Transaction Service (OTS), can be used to guarantee consistency. In addition, students in SModel are represented by information coming from a Bayesian student module (BSM) and from the student modelling database (SMDB). Agents use methods such as processQuery to access information stored in the SMDB. processQuery acts as an ODBC bridge from a database engine and agents interacting with SModel. Most of the commercially available database engines support transaction management techniques. SModel can then transfer transaction control mechanisms to the database engine directly.

Agents can use hypotheticalQuery to test their Bayesian queries without changing the model. In this way SModel could let agents read the Bayesian model and only lock it when final changes are executed.

Another interesting issue is that SModel supports the use of partial Bayesian student models. In this case agents can use general inference mechanisms to update the beliefs of their own partial models of the student. Partial Bayesian models are used by individual agents in their own context. Agents can make use of general information maintained in the model while maintained their own independent view. Therefore, SModel consistency mechanisms can be used in different degree by different agents according to the requirements of the platform in which the agents run.

5. JTS: Java Tutoring System

A java tutoring system (JTS) (Agarwal et al. 2000) 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 and SVM, 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 6 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.

5.1. 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.

CORBA bus

![Diagram of CORBA bus with SModel Server attending to requests from several agents/components in JTS]

**Figure 6.** SModel attending to requests from several agents/components in JTS

### 5.2. 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 7 shows a screenshot of JTS, in which MAJI presents the content of the tutorial and the initial menu to the student.

5.3. 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 aspects of learning level in JTS depicts a tentative set of relationships among confidence, competitiveness, cooperativeness, assertiveness, eagerness, and helplessness. As a tentative causal model, it remains opened for teachers to include the social aspects that are relevant in their own context. 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 of learning 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 8, 9, and 10 show each of the levels used by JTS in its Bayesian student model. These figures were produced using ViSMod.

**Figure 8.** 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 9. 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 10. Social aspects of learning 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.
5.4. IMPLEMENTATION ISSUES IN JTS

During the implementation of JTS, SModel server was subject of refinement. It is important to notice that agents in JTS did not exploit the full range of services offered by SModel. Therefore, it would be interesting to create or to employ existing agents that use all of the SModel services. This will lead us to improve the existing services and offer new ones.

We encountered many technical problems during the creation of JTS. Development and integration of agents in different platforms is not trivial. Technical problems are still an important factor to be considered. Probably future developments do not require as many hours expended solving technical and connectivity problems as we did in JTS, but this remains to be seen.

It is also expected that development of ITSs using reusable components such as SModel server will benefit and encourage the creation of interdisciplinary groups. Teachers, designers, students, psychologists, cognitive and computer scientists can collaborate and integrate their modules smoothly. Development platforms based on modular and reusable components from different research groups allow the creation of useful applications faster and easier. Participation of people from different areas should be accompanied by the application of software engineering principles that assure a successful development and integration of components.

6. Conclusions

Student models are no longer encapsulated components used by the system; open student models allow teachers and students to interact with the model through special interfaces and visualization tools. Student models maintained by the server become learning tools that can be integrated on any learning environment to support reflection.

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 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.

SModel server is an example of reusable components in the development of educational systems. Offering Bayesian inference mechanisms to a group of agents makes it possible to create more interesting and powerful mobile user-adaptive applications.

By using ViSMod and SMV, 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 (Vassileva et al., 1999). 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


