

Chapter 5

Development of Knowledge Based Intelligent Tutoring System

*Sunandan Chakraborty, Devshri Roy, Anupam Basu**

Abstract : Intelligent Tutoring Systems (ITS) are computer-based tutors which act as a supplement to human teachers. The major advantage of an ITS is it can provide personalized instructions to students according to their cognitive abilities. The classical model of ITS architecture has three main modules – domain, model, student model and teaching model. In this chapter, we have discussed the recent developments in those areas and tried to address the important issues involved with them. We have developed an ITS. We have used two major data structures (a tree and a directed graph) to represent the domain knowledge within the domain model. The domain model also consists of a repository which is a collection of study and test materials. The contents in the repository are annotated using various informative tags for intelligent retrieval of them for the students. A fuzzy state based student model is used to represent a student's cognitive ability and learning behavior. This is an important task as the adaptation techniques to be followed by the system are dependent to the student's ability. The actual adaptation techniques are planned and executed by the control engine, which is the teaching model of the system. The control engine communicates with the domain and the student model and takes decisions using fuzzy rules. We have evaluated our ITS and found quite encouraging results.

Keywords: Intelligent tutoring system, domain knowledge base, fuzzy based student model, teaching model

1. INTRODUCTION

One of the most important challenges faced by most of the developing and the underdeveloped countries is the spread of education among all. As an example, in India the literacy rate was about 65% and in the state of West Bengal it was 69.22% in 2001 (www.censusindia.net). A plausible reason behind this low literacy rate is attributed to the need of proper schools, proper infrastructure and poor teacher to student ratio (<http://education.nic.in/stats/detail/18.pdf>). The said perspective vindicates the

* {sunandanchakraborty, droy.iit, anupambas}@gmail.com
Indian Institute of Technology, Kharagpur, India

necessity of developing an alternate attractive, affordable and effective teaching platform that will capture the attention of children. In this scenario an Intelligent Tutoring System (ITS) can be quite relevant. An ITS can to a large extent address the issue of unavailability of skilled teachers. Intelligent Tutoring Systems (ITS) are computer-based tutors which act as a supplement to human teachers. The Association for the Advancement of Artificial Intelligence (AAAI) defined ITS as,

An intelligent tutoring system is educational software containing an artificial intelligence component. The software tracks students' work, tailoring feedback and hints along the way. By collecting information on a particular student's performance, the software can make inferences about strengths and weaknesses, and can suggest additional work (AAAI, AI Topics/Intelligent Tutoring Systems).

Ideally, an ITS tries to simulate a human teacher and sometimes it may prove to be more advantageous than its human counterpart. One of the main advantages of ITS is individualized instruction delivery, which means the system will adapt itself to different categories of students. A real classroom is usually heterogeneous where there are different kinds of students, from slow learners to fast learners. It is not possible to provide attention to them individually, thus the teaching may not be beneficial to all students. An ITS can eliminate this problem, because in this virtual learning environment the tutor and the student has a one-to-one relationship. The students can learn in her own pace. Another advantage is that using this system teaching can be accomplished with minimum intervention from the teachers. Therefore, ITS can be really effective in areas where there is dearth of trained teachers.

The functionalities of an ITS can be divided into three major tasks, (1) organization of the domain knowledge, (2) keeping track of the knowledge and performance of the learner, (3) planning the teaching strategies on the basis of the learner's knowledge state. Hence, to carry out these tasks, the ITSs have different modules and interfaces for communication (Wenger, 1987; Freedman, 2000a; Chou et al, 2003). The different modules are:

- Domain model
- Teaching model
- Student model
- Learning environment or user interfaces

Two major issues related to an ITS are “what to teach” and “how to teach” (Murray, 1999; Ong & Ramachandran, 2000). The domain model deals with the “what to teach” part, whereas the teaching and the student model are concerned with the “how to teach” part. The main objective behind an ITS is its intelligence or adaptive teaching methods. Adaptation means that the system moulds itself to cater to different needs of different students. Among the two functions of ITS described earlier, adaptation is a part of the “how-to-teach” part. To decide on *how to teach*, the system needs to judge the student's knowledge state and her preferences. The tool which mediates between the system and the student is the *student model*. Thus, a robust, flexible and comprehensive *student model* is required to build a real adaptive Intelligent Tutoring System. The domain model is also important as it is the representation of the domain knowledge. Good design principles adapted in designing the domain model would help the system in selecting the proper methods for teaching. This would also help the system in the search for alternative teaching plans when a particular method fails to produce success. Finally, the most important part of an ITS is the *teaching model*. This module is the centre of the whole system. It communicates with the other modules and does the entire decision making.

A knowledge base ITS can only be effective if the teaching model plans the teaching strategies properly. A major drawback of any knowledge base ITS is writing rules for generating teaching strategies. It solely depends on the knowledge of the expert. Another problem is that often experts themselves disagree and can write different rules for the same given problem. One other drawback of a knowledge base ITS is building an efficient and complete student model. The adaptation technique which changes dynamically for different kind of students depends on the student model. Again the development of the domain knowledge base is a very tedious job. It can only be developed by a

domain experts. Development of domain knowledge base is very costly in terms of time and effort required.

Most of the existing knowledge base ITSs were restricted to a single domain. So, changing them or reorganizing them to teach different subjects were either impossible or required the intervention of the system developers. Again, attempting to design a domain independent system has got its own problems. A procedure or a strategy to teach Physics may not be successful when used to teach History (Freedman, 2000b). A single teaching method would not work in a multi domain teaching environment. Another disadvantage is that the delivery medium of all the existing systems was also restricted to usually one language. This may prove to be a serious drawback in a cosmopolitan society. Reviewing the state-of-the-art we found some general drawbacks in the existing systems. The drawbacks for each module are summarized below:

Domain Model

- Domain knowledge representations are not generic (Baffes et al, 1996; Khuwaja et al, 1994). For example the ITS Andes (Conati et al, 2002; Gertner & VanLehn, 2000) is a domain dependent and its framework is dependent upon the domain. Different courses from different domains cannot be represented using the same framework. There is a need to develop domain independent representation scheme.
- In many ITS, Learning materials are implicit to the systems. Adding new materials into the system is cumbersome. Moreover, materials incorporated are homogeneous. Different formats like, animation or video files are not supported by many of the systems. Furthermore, documents supported in these systems are mostly monolingual.

Student Model

- Graphical probabilistic models such as Bayesian Networks are often used for student modeling (Conati et al. 2002, Vomlel 2004) which requires considerably expert modeling effort and hence very costly. There is need to give attention to build rapid, cost-effective means of developing these environment (Martin et. al., 2005, Kodaganallur et al. 2005, Aleven et al. 2006).
- Domain dependency was inherent in many of the existing student models. The design of the existing student models were either fully (Conati, 2002, Zhou & Evens, 1999a) or partially domain dependent (Zapata, 2004). Hence, configuring the corresponding ITSs to teach in other domains required major changes in the configuration of the student model.
- Configuration of student models, particularly in authoring system is often necessary. In various ITSs the configuration of student models involved complex operations. The teachers need to have advanced knowledge of programming to do those tasks (Baffes & Mooney, 1996). The modifications also involved the use of some terms, which can be difficult for teachers to interpret or perform (Zapata, 2004, Thompson, 1996).
- In most of the student models, different parameters to represent the student's cognitive ability were not used (Conati, 2002, Zapata, 2004, Baffes & Mooney, 1996). There can be various attributes describing a student's cognitive ability, and their values can give a better approximation of the student's cognitive ability. Thus a monolithic representation of the student as used might not be effective.
- In some other cases the student model is not transparent to the students. A student's interaction with the system is very vital for her performance. The student at any point of time must know the state she is in. But mostly the student is unaware of her status. The outputs of

the student model should be presented to the student in a clear, easily understandable format and should contain the right level of details.

Teaching Model

- Usually the teaching models in ITSs cannot be modified. The teaching strategy defined in the teaching model is fixed and the changes can only be performed by the system developers. But the teaching strategies might require modifications. So there should be provisions for minor modifications in the teaching model.

From the above discussion, it is clear that there are scopes for further improvement in ITSs and its various modules. There is a need to develop ITS which should be able to reduce the drawbacks of the existing systems. We have developed an ITS whose design goals are as follows:

- The domain knowledge representation scheme should be generic so that a course from any domain can be configured in the system.
- The system should support learning materials of different types and formats, so that it can cater to diverse groups of students. The inclusion and modification of the materials should be simple. The materials should be reusable.
- The student model should be able to represent a student's cognitive ability and learning preferences. The model should consider the different parameters through which a student's capabilities can be estimated better. This should also help the system to analyze a student's drawbacks and adapt accordingly. The student model should also be independent and compatible with any domain.
- The student model should be transparent so that students can be aware of their performance.
- The teaching model should try to plan the best teaching method for each student. In case of a failure the module should be able to identify the situation and re-plan as necessary.
- There should be provisions for teachers to change the rules of teaching (teaching strategy) so that the system can be used in different situations.

The chapter is organised as follow. First, we review the related work done in the field of ITS in Section 2. In section 3, we discuss the overall architectures of our ITS. Next, in section 4, we present a detailed description of the various modules of the ITS. Section 5 presents a case study. Section 6 concludes the paper.

2. LITERATURE REVIEW

In 1984, Benjamin Bloom defined the “two-sigma problem,” which states that students who receive one-on-one instruction perform two standard deviations better than students who receive traditional classroom instruction (Bloom, 1984). It is impossible for any institution to provide personal teachers to each and every student. This drawback strongly supported the use of computers, as a replacement to human teachers. Motivated by the above cause, lots research groups started to work in this field and developed various systems with various features.

2.1 Earlier Systems

The first generation of computer assisted education tools were called, **Computer-Aided Instruction (CAI)** systems. One of the early examples of such a tutoring system is the system by Uhr in the year 1969 (Sleeman & Brown, 1982). This system generated problems on arithmetic and questions on

vocabulary. But main problem of this system were it had no user modeling or adaptation technique. However, some contemporary systems, like Suppes (1967), system by Woods and Hartley (1971) (Sleeman & Brown, 1982) could be called adaptive because here the problems were according to the user's performances. But the user model they used was quite primitive in nature. The model was only a parametric summary; the actual knowledge state of the user was not stored. These systems can be termed as the precursor to Intelligent Tutoring System (ITS).

In the meantime, another genre of tutoring system came up. These types of systems were called **Drill and Test**. Here only problems were presented to the students in form of tests. The students on the other hand are provided with the test results. A simple variation of this system was the Adaptive Drill and Test. In this version instead of just presenting the problems, the student's performance and response were collected, tabulated and later used to select problems. Thus at this point of time, it was felt that the student needed to be considered as an important factor, and no longer predetermined rules will work. An adaptation technique was quite necessary to tackle all possible responses from the students.

2.2 Emergence of ITS

In 1982, Sleeman and Brown reviewed the state of the art in computer aided instruction and first coined the term **Intelligent Tutoring Systems (ITS)** to describe these evolving systems and distinguish them from the previous CAI systems. They defined ITS as being computer-based (1) problem-solving monitors, (2) coaches, (3) laboratory instructors, and (4) consultants. (Sleeman & Brown, 1982). And for the first time the use of Artificial Intelligence was seen, which made the systems "intelligent". At this point of time emerging trends in Artificial Intelligence were applied in these systems. With new AI techniques coming up it seemed that the computers were almost capable of "thinking" like humans. This motivated ITS research further. Application of AI in ITS made it possible to achieve the goals more easily. Other reasons which motivated ITS research were:

- Modular and fine grained the curriculum.
- Customized for different student populations.
- Individual presentation and assessment of the content.
- Collection of data which instructors could use to tutor and remediate students

In spite of these there were some typical drawbacks found in many early systems. These drawbacks included:

- Domain dependent. The targeted domains initially were (Sleeman & Brown, 1982)
 - Symbolic Integration [e.g. Calculus Tutor (Kimball, 1982)]
 - Electronic troubleshooting [e.g. SOPHIE (Brown et al, 1981)]
 - Mathematics [e.g. EXCHECK (McDonald, 1981)]
 - Medicine [e.g. GUIDON (Clancey, 1982)]
- System assumed much or too little of student's knowledge.
- Presented documents had the wrong level of details.
- Little interactivity, lead to limitations in student's expressivity.
- Inflexible.

2.3 Recent Systems

Since the emergence of ITS various systems were built using different techniques and with different objectives. In this section, we discuss about some of the recently built ITSSs.

2.3.1 Andes

Andes (Conati et al, 2002; Gertner & VanLehn, 2000) is an ITS which was developed to teach physics for the students in Naval Academy. Bayesian networks were primarily used in Andes for decision-making. The major foci of the system are (1) Select the most suitable strategy for the student (2) Predict Student's actions (3) Perform a long term assessment of the student's domain knowledge.

Andes is a domain dependent ITS. Each problem in the system was broken into some steps and Bayesian network was formed using those steps as nodes. So, the problems were represented in the system as Bayesian networks. The Bayesian network would predict the most probable path for the student during a course. Each student could have different approaches to a problem, the network would be adjusted accordingly (the probabilities would change) and finally for a new problem it would predict the best strategy for the student. There is also a problem-solver in the system. This problem-solver partially or wholly solved a problem to help the students. The Bayesian networks had two parts: static and dynamic. The static part had Rule nodes and Context-rule nodes. The rule node represented general physics rules and had binary values, T and F. The probability $P(\text{Rule}=\text{T})$ was the probability the student could apply the rule properly in any situation. Initially these prior probabilities had values 0.5 but the authors claimed more realistic values could be obtained after a trial run in the Naval academy.

The dynamic part contained the Context-rule node as well as four other nodes: fact, goal, rule-application and strategy-nodes. The fact and the goal nodes were also binary. $P(\text{Fact}=\text{T})$ was the probability that the student knew the fact and $P(\text{Goal}=\text{T})$ was the probability that the student is pursuing the goal. There might be more than one way to reach the goal or the fact nodes and that lead to having so many parents. The conditional probabilities $P(\text{Fact} = \text{T} | \text{parent}_i)$ represented the probability of reaching the fact from parent_i . The strategy nodes were there, where, students had more than one options to choose from. And the rule application nodes represented the children of those strategy nodes. The rule application nodes basically represented the different applications of the strategy nodes. The strategies were mutually exclusive; i.e. the student could choose one and exactly one strategy at a time. Thus if an evidence increased the probability of one of the strategies it would definitely decrease the others. The probability values ($P(\text{Strategy-node} = \{x: x \in \{\text{child1}, \dots, \text{childn}\}\})$) of these nodes would depend upon the number of children the node had. Finally the Rule-application nodes were the connectors between context-rule nodes, Strategy nodes, fact and goal nodes to new derived Fact and goal nodes. In other words, those nodes had a single Context-rule node and strategy node and one or more than one fact and goal nodes (preconditions) as parents, and children of those nodes included some facts and goal nodes (Conclusion). They had binary values and $P(\text{R-A} = \text{T})$ meant the probability of a student applying the parent Context rule to the preconditioned fact and goal nodes to get the derived fact and goal nodes. The probability values would vary with students and thus application of rules, choosing from alternate paths etc would depend upon each student. But how the probabilities were derived was not stated.

2.3.2 VisMod

ViSMod is another ITS which used Bayesian network (Zapata-Rivera et al, 2004). In the system the Bayesian network was divided into three levels. At the top most level the concepts (to be taught) were represented in a hierarchical manner. After that in the second level student's performance and behavior were described. Finally the third level nodes represented some analysis on the student's performance. Only the first level is domain dependent, whereas other two levels would remain same over different domains. Again student can observe only the top two levels of the Bayesian net. The third level is only visible to the teachers. During a course the probabilities in the second and third level of the Bayesian net changed according to the student's performance. Those probabilities made a change in the first level values. i.e. the probability values in the first level were directly dependent on the two lower

levels. After the probabilities were computed the most probable path along the first level was determined and the first node of the most probable path was chosen as the next step in the course.

The Bayesian networks were initially formed by human teachers using textbooks. The prior and conditional probabilities were set according to the level of the topic. That is each topic was classified by three categories, *beginner*, *intermediate* and *expert*. The probabilities will vary according to this categorization. The initial prior probability values were set by conducting a manual session with the students. The Bayesian networks were constructed using concepts from the topic in concern. Suppose a concept has two parents, which means that the parents have some influence on their children. In other words evidence of knowing the parent increased the probability of knowing the children. There was a threshold value, determined by domain experts, which depicted the highest probability a student could reach in knowing the child concept. There was another parameter called the *weight* which defined the degree of influence the parent had on the child. The weights were calculated by subtracting the base value between the parent and the child from the threshold value and dividing it by the number of parents the child had.

The conditional probabilities were calculated from the above information. The conditional probability $P(\text{child-concept} = \textit{knows} \mid \text{parent}_1 = \textit{knows}, \dots, \text{parent}_n = \textit{knows}) = \text{base value} (1/n) + \sum \text{weight}_i$. Each node in the network could have two values, *knows* and *doesnot-knows*. If one of the parent concepts was not known by the student then that weight would be equal to 0. In case two parents had equal values of conditional probability then expert's choice would break the tie. The Bayesian network for a topic was first constructed by making the nodes as concepts and the edges as their dependencies. Then the conditional probabilities along the edges were calculated using the above procedure. That's how the whole network was built.

2.3.3 InterMediActor

InterMediActor (Kavcic et al, 2003) is an ITS which used fuzzy inference mechanism. It had a data structure called *navigation graph*. This graph determined which concept comes after which. When there were multiple choices, decisions were made using fuzzy rules. The fuzzy rules actually connected the student's capability and the nature of the concept in hand to decide whether the concept was suitable for the student or not. The student's information and other related information of the topic and concept were described in the system as fuzzy sets. The fuzzy rule base established a relation between them and helped to make decisions. The rules antecedent consisted of three parameters. Firstly the difficulty level of the topic, it had the values *easy*, *normal* or *difficult*. The values of these variables were dynamic. If in a topic students scored high marks consistently, the degree of difficulty would be decreased and in case the scores were low, difficulty would be increased (this was done using some fuzzy rules). The second parameter in the antecedent was the score in the final test. That variable could take values as *positive*, *negative* or *no mark*. The final parameter was, knowing the prerequisites. Its values could be *not*, *little*, *enough*, *well*, and *very well*. The fuzzy rules mapped those parameters to a consequent which was the recommendation level of the next topic, i.e. to determine whether the concept was suitable or not for the student. The consequent variable *level-of-recommendation* had values like, *learned*, *more recommended*, *recommended*, *less recommended* or *forbidden*. To defuzzify the consequent of the rules Centre of Maximum technique was used. The navigation graph, which represented all the topics and their dependency, was annotated by this crisp value of the level of recommendation. The annotated version of the graph represented the student's level of performance and understanding of the topic. The rules look like,

IF the student has *positive* mark AND the competence is *easy* THEN the concept is *more recommended*.

2.3.4 SQL-Tutor

SQL-Tutor (Wang & Mitrovic, 2002; Mitrovic, 2003) is an ITS, which as the name suggests is to teach SQL. Artificial Neural Network (ANN) was used in SQL-Tutor for decision-making. An agent was present who analyzed the student and selected an appropriate problem from the database. That agent was modeled using an ANN.

This ITS was developed to teach university students SQL. Here the solutions to the problems were represented in the system as constraints. Whenever a student submitted a solution the system calculated the correctness by comparing the number of constraints violated by the student. The next problem to be chosen or any other teaching decision to be taken depended on this information, how many mistakes or violated constraints the student has committed. To make this prediction the system used an artificial neural network (ANN). The ANN was a feed-forward network and had four inputs, one output and a hidden layer. Delta-bar-delta back propagation and linear tanh (LT) transfer function was used and the inputs consisted of different information relating to the student like, (1) Time needed to solve the problem, (2) The level of help provided to the student (3) The complexity of the problem (4) Also, the level of the student in hand.

In the output the ANN tries to predict the number of errors (i.e. the no. of constraints violated) will be committed by the student. This prediction is used to take the next teaching decision (like the next problem to choose from the problem database). However how the weights of the ANN were chosen and how exactly the ANN was trained were not explained clearly. About the performance of the system, the authors claimed that the ANN could predict the correct number of errors with an accuracy of 98.06%.

An added advantage of this system was it provided feedbacks to the student after checking her solution. The feedback might contain hints, partial solution or complete solution as required.

2.3.5 C++ Tutor

C++ Tutor is a rule-based ITS (Baffes & Mooney, 1996). Here the concepts of C++ were explained using some rules. These rules were in form of Horn sentences and were called the *Theory*. The problems were produced to the students in the form of feature vectors. The students were supposed to label the vectors choosing from a set of labels. An algorithm called *NEITHER* took these labeled vectors (student's solution) as input and modified the correct rule base, so that the modified rules implied the student's solution rather than the correct solution. This process is called *Theory-revision*. So now the modified rule base reflected the student's state of understanding, representing the student's correct knowledge as well as the misconceptions. After the theory-revision was complete the system tried to explain the bugs in student's concept by showing examples, which enumerated the areas where the student had gone wrong. This was done automatically by the system, by comparing the modified rules with the correct ones.

2.3.6 CIRCSIM Tutor

CIRCSIM is a dialogue based system which taught some topics on Physiology to medical students (Evens et al, 2001; Khuwaja et al, 1994). Here the problems were presented in the form of dialogues and the interactions with the students were also done in dialogues. In CIRCSIM the student model was divided into four components (1) Performance model (2) Student reply history (3) Student solution record (4) Tutoring history model.

In the performance model the student's performance was stored and analyzed. This assessment was done in four levels, *global*, which dealt with the overall performance of the student. *Procedure-level*, this is concerned with the particular problems the student has solved. *Stage assessment* checked the student's understanding about the different stages of Physiology in the problems. Finally, *local assessment* measured the student's perception in the Physiology variables that have been tutored. There were some rules which defined the priorities of the above components and the action taken. By priorities it is meant that even if the overall performance (component i) was poor for a student but the

reply history (component ii) was strong the system would treat the student as a competent student, i.e. component (i) had a lower priority than component (ii). Going by the rules the system would generate fitting dialogues to interact with the student. Thus the generated dialogues would depend upon the student model and would try to respond as correctly as possible. For example the system would try to give simpler problems (providing various hints if not solved correctly) to a student whose overall model was poor. And all these were done through dialogues. Other decision-making processes were also done using *simple rules*. It is stated that the rules could take consistent decision, considering all the information from the student model, but the exact methodology was not stated.

We have developed an ITS which has a flexible domain model. The domain model has a generic structure and courses from different domains can be built. It contains learning material in different languages. We found that several authors have explored the use of Bayesian belief networks to represent student models (Collins et al., 1996; Conati et al., 2002; Horvitz et al., 1998; VanLehn & Martin, 1997; Reye, 1996; Reye, 1998; Villano, 1992). Although existing configurations of student models can be represented using Bayesian belief networks in a solid and computationally tractable fashion, understanding the status of a Bayesian network can be difficult. It also requires considerably expert modeling effort and hence very costly. There is need to give attention to build rapid, cost-effective means of developing these models. There is a need to develop student model which should be easily understandable by students and teachers. Bull and Pain (1995) found that students seem to understand textually presented models. In order to build a flexible student model, we propose a fuzzy-state transition based mathematical student model, where a student's performance is represented by fuzzy-states and the transitions signify the student's progress. Here, some inputs are fuzzy variables and requires linguistic words to be given as input. The domain expert need not to provide the precise numerical value as a result authoring the student model becomes easier and requires less time and effort. The student model is transparent to the student. The overall architecture of the system is discussed in Section 3.

3. ARCHITECTURE OF THE ITS

We have built an ITS, which provides adaptive learning environment for the students by providing personalized instructions. It also exploits the use of multilingual and multimodal contents, which offers the students with a variety of materials, with different media and alternative modes of explanation. We adapted the classical structure of an ITS with four basic modules (Wenger, 1987; Chou et al, 2003) *domain model, student model, teaching model and student's interface*.

The domain model has a flexible structure and the different courses from different domains can be represented by using the same representation scheme. The *domain model* has two major parts. The first is called the *Domain Organization Model*, it is the knowledge base of the system. The other is the *Repository*, which stores the metadata annotated learning and test materials. *Student model* stores information about the student's performance state in terms of fuzzy variables, her preferences and her learning history. This module provides the basis of the adaptation policy employed by the ITS. The *control engine* or the *teaching model* is the pedagogical agent of the system and it executes the actual teaching process. In logical sense this module lies in the centre of the system and by communicating with the other modules it provides adaptive instructions to the students. It is rule-based, where the rules define the teaching strategy. The rules are authorable that enables the teaching model to be customized. The features of the authoring tool have been extended to support the modification of the teaching model. A domain expert can utilize this facility and apply her skills in reorganizing the teaching method to suit in different environments. *Student's graphical user interface* (GUI) is the gateway of the system for communicating with the student. All the learning materials and test sets are presented to the student through this interface and test results are fed back to the system for assessing the student. The overall architecture of the ITS is shown in Figure 1. We discuss its various modules and their functionalities.

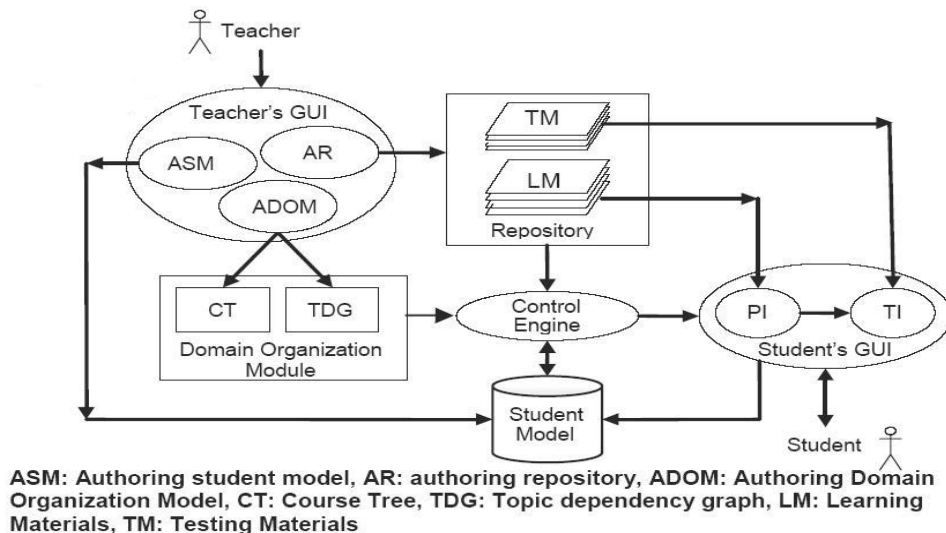


Figure 1. Overall architecture of the ITS

3.1 Domain Model

The domain model is the knowledge base of the system and it organizes the course structure, its various components and the relationship among the components. This model mainly deals with the what-to-teach part of an ITS (Wenger, 1987; Murray, 1999).

The knowledge base of our system has two main modules: *Domain organization module* and *Repository*. *Domain organization module* is concerned with the structure and organization of the course and its topics and relationship between the topics. It is the knowledge base of the system. The second module is the *Repository* which stores learning materials and also the test materials. Each learning material and test material is described by a set of metadata to provide mechanisms for finding and reusing of existing resource in the knowledge base.

3.1.1 Domain Organization Model

Domain organization module is a structural representation of the different courses. Two major data structures are used for this purpose are *Course Tree (CT)* and *Topic Dependency Graph (TDG)*.

Course Tree (CT): CT is a hierarchical structure of a course stored in the system. The root of the tree keeps the name of the course, and is called course node. Subsequently, the different sections/subsections and topics of the course are kept in lower parts of the tree. The leaf nodes of the tree are called topics and are the atomic teachable units. Topic nodes are also associated with different features like *Concepts (sub-topics)*, *prerequisite topics*, *threshold score*, *difficulty*, and *importance*. A sample CT is shown in Figure 2.

Topic Dependency Graph (TDG): The nodes of TDG are constituted of the topics from the corresponding CT, whereas the edges in TDG depict 'prerequisite' relation between the nodes (topics). TDG drawn from the CT in

Figure is shown in Figure 3.

Every course stored in the system will have its own pair of CT and TDG.

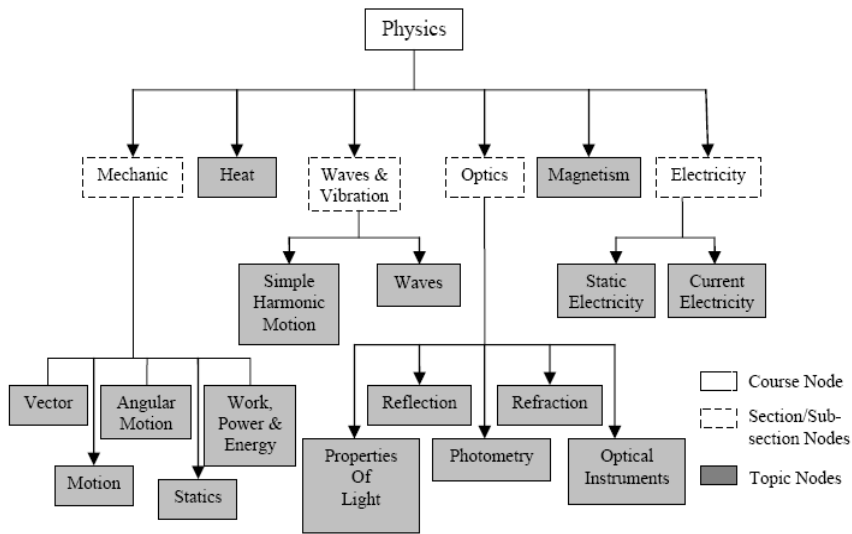


Figure 2. Course Tree for Physics

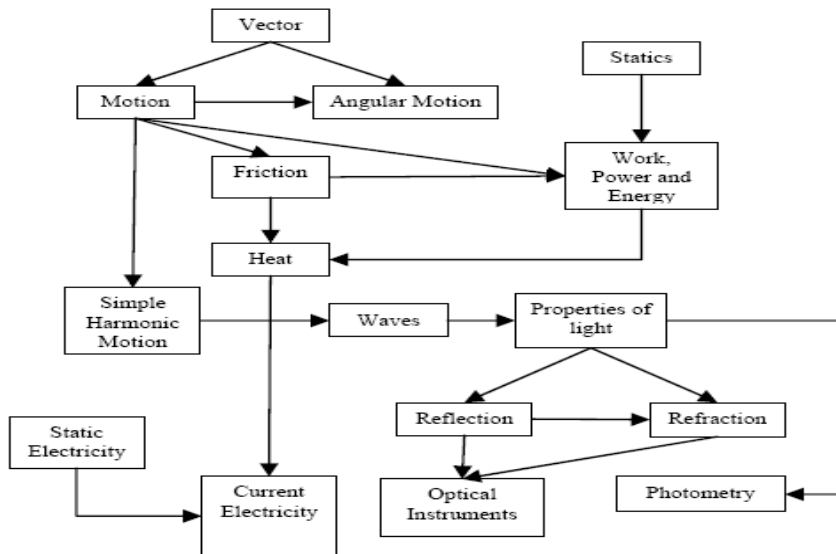


Figure 3. Sample TDG drawn from the CT in Figure 2

3.1.2 Repository

The Repository is a pool of learning and testing materials. For efficient access of materials from the repository the materials are tagged with various features. These features provide description of the documents, which helps the material selection agent of the system in efficient and customized retrieval of the materials for the students. In order to tag the study materials a subset of the standard set of tags specified by IEEE LOM is used. In addition some extra features are added locally to incorporate some system specific information in the document description.

3.2 Student Model

The most important task of an ITS is the application of its “intelligence”. This means applying different methods of teaching to different students, without any intervention from the human teachers. In order to decide upon the method of teaching, the system must know about the nature and preferences of the student. Teaching can be most effective when the learner’s cognitive ability and her behavioral pattern can be properly assessed from her learning outcome. Pointing out a student’s deficiencies and concentrating on those areas is an important phase in any teaching process and can certainly result in a gain in her performance. In this teaching environment there is no other human involvement apart from the student. Hence, a tool or a utility is required within the system which will mediate between the system and the student. This tool should be able to assess the student properly and provide the decision-making agent of the system with all the relevant information, such as, the drawbacks of a student, which is required for adaptive teaching. This tool is called the *student model*. A robust, flexible and comprehensive student model is very vital for an adaptive Intelligent Tutoring System. Ideally a student model should keep the cognitive state of a student, her learning preferences, goals etc. In other words, the student model is an abstract representation of the student. Similar to normal teaching, where the human teacher reacts and changes the teaching method according to the feedback received from the student, the planning agent in an ITS must adapt itself and modify the teaching plan according to the feedback received from the student model. Our goal is to design such a student model which should be able to provide the system with all the required information. This information should be effectively used by the system’s planning module to provide the students with personalized education, meeting the needs of the individual students. Thus, a well designed student model is necessary in meeting the objectives of an ITS.

The mostly used techniques for student modeling are Bayesian networks, Rule-based systems, Fuzzy logic, Dialogue-based systems etc. Andes (Conati et al, 2002) and VisMod (Zapata et al, 2004) are two ITSs which used Bayesian network to model a student. SQL-Tutor (Wang et al, 2002) modeled the student using artificial neural network. Fuzzy logic is another popular method used in student modeling (Ali et al, 2004; Kavcic, 2003; Weon, 2001; Xu, 2002). C++ Tutor (Baffes & Mooney, 1996) adapted a technique called theory refinement, which was also capable of capturing the students’ misconceptions. Some other methods include feature-based modeling (Webb, 1997), machine learning methods, like, expectation maximization (Ferguson et al, 2006), reinforced learning (Beck & Woolf, 2000; Beck, 1997). Apart from the above techniques there are some works which focused on non-cognitive aspects of a student to model their learning styles. D’Mello (2005) used affect sensors to track the emotion of the learner and the system acted according to the students’ state of mind.

We have adopted a fuzzy state based transition model for our student model. The transitions represent student’s change of performance after a learning session. After a student covers a topic a state transition may occur or she might stay in the same state.

3.2.1 Student Profile

Student model should be a good approximation of a student and her learning progress. Therefore, our student model primarily consists of a student profile. In this profile a student is described using various parameters. Presently we are using only two aspects to measure the cognitive state of a student, namely *Comprehension-ability* (C) and *Problem-solving skills* (P). *Comprehension-ability* represents the student’s ability to comprehend or understand a particular concept, whereas *problem-solving skill* indicates the student’s capability to apply the concept which she has learnt particularly in problem solving. These two parameters collectively represent a student’s overall aptitude. The values of C and P are taken to lie in the range [0, 1]. The student profile is a dynamic module, where, these values are updated every time a student appears in a test on a topic. So, at a particular instant the value of <C, P> pair signifies the current status of the student’s cognitive state. The individual values of these parameters will help the system in assessing the student properly and act accordingly. For example, if a

student has a high value in comprehension ability but relatively low score in problem-solving skill, the system will try to provide her with learning materials, which will enhance her problem-solving skills. In the student profile two parameters are used to evaluate a student, but it can be easily extended to accommodate other parameters. The inclusion of some extra parameters will help to capture some other aspects of a student. Evaluation of a student will be stronger as various such parameters will give a better analysis of the student's cognitive state. Furthermore, adaptation will be more effective as the system will have a more accurate insight of the student's weaknesses and strong points.

In addition to *Comprehension-ability* and *Problem-solving skills*, the student profile also keeps records of some other information related to a student, for example, the types of documents preferred by the student. A particular student may have her own preferences, which might help her to learn better. This information kept in the student profile helps in the planning process, where the system will try to select a set of suitable materials to teach a topic better. Apart from this, the student profile also keeps other records, such as, sequence of topics covered and the student's performance in them, all the materials studied during the learning phase, detailed test results etc. The whole student profile is dynamic. After each performance evaluation of the student, the various fields of the student profile are updated from the evaluation results.

3.2.2 Fuzzy-state based State Transition Model

We have adopted a fuzzy-state based transition model for student modeling. In this model, each state represents the current status of the student's progress in a course including her performance. The transition from one state to another signifies the student's improvement or deterioration from her previous performance. Hence, the overall traversal in this model will depict the student's net performance throughout the course. In this system more than one course can be configured. For different course there will be separate transition models. Thus, the performance of a student in a particular course will not affect the decision-making in other courses.

The transitions help the planning agent of the system to take appropriate decisions during the planning process. A transition showing no improvement from the previous performance signifies the failure of the existing mode of teaching. Thus, the agent will try to select an alternative mode of teaching, such as selecting an alternative set of materials, to help the student to learn better. Similarly, improving transitions express the success of the current teaching method, which leads to the retaining of the same. We now illustrate the transition model in detail.

The student model is defined as a 3-tuple transition system, $\langle S, I, \delta \rangle$. Where,

- S is a set of states,
- I is the input set, where $i \in I$ is a real number,
- δ is the transition function.

Definition of States: There are several parameters for pedagogical modeling of a student during a learning process, but in this work we consider two parameters,

- Performance of a student, measured through her ability to comprehend and her problem solving skills.
- Coverage: The topics covered by a student

Consequently, the state of a student is a composite state

$$S = S1 \times S2,$$

Where S1 denotes the set of performance-states and S2 denotes the set of coverage-states.

Therefore, set of states S is the set of composite states, $S = \{S_i\}$,

where each composite state S_i (after the i^{th} input) is a tuple $\langle \text{Performance}_i, \text{Coverage}_i \rangle$

Coverage_i is the list of topics that a student has covered successfully. Successful coverage of a topic implies that the student has scored above the set threshold value in the test conducted.

Performance_i represents the student's performance. We propose a fuzzy state representation, where the performance-state of a student is represented in linguistic terms, represented by fuzzy sets. Performance_i is denoted as a fuzzy term as defined below:

$\text{Performance}_i = \{\text{Excellent, Very Good, Good, Average, Poor}\}$

The advantages of using fuzzy sets are, the use of linguistic terms are natural and easier to understand for the users (both students and teachers) and they can manage the uncertainties involved in these parameters.

Computation of Fuzzy State: For a student k , all the state information is stored in the student profile. At any "Present State" a student has already covered some of the topics successfully and has demonstrated a particular level of performance, which might have been modified through her last test. Computation of Coverage_k for a student k is simple as it is augmentation to a list. The list is augmented when a student clears the test corresponding to a topic with a score greater than the set threshold score. For computation of Performance_k , we are using the two parameters kept in the student profile, *Comprehension-ability* (C_k) and *Problem-solving skills* (P_k). The values of C_k and P_k lie in the range $[0, 1]$. These values are computed every time a student appears in a test on a topic. After each update of the $\langle C_k, P_k \rangle$ pair for a student in her student profile, the fuzzy state is computed as follows (Weon & Kim, 2001):

Assume that, after the student k has covered the j^{th} topic her value of comprehension-ability and problem-solving skills have been C_j^k and P_j^k , respectively.

$$\text{Now, let } v = \frac{(C_j^k + P_j^k)}{2}$$

The membership functions of the fuzzy states are defined as:

$$\begin{aligned} \mu_{\text{Excellent}}(v) &= 1 \text{ if } 0.9 \leq v \leq 1 \\ &= 0 \text{ otherwise} \end{aligned} \quad (1)$$

$$\mu_{\text{VeryGood}}(v) = v$$

$$\begin{aligned} \mu_{\text{Good}}(v) &= 2v \text{ if } 0 \leq v < 0.5 \\ &= (2-2v) \text{ if } 0.5 \leq v \leq 1 \end{aligned}$$

$$\mu_{\text{Average}}(v) = 1-v$$

$$\begin{aligned} \mu_{\text{Poor}}(v) &= 1 \text{ if } 0 \leq v \leq 0.1 \\ &= 0 \text{ otherwise} \end{aligned}$$

The state with the highest membership will be taken as the performance state. The results of the test taken by the students after each topic serve as the input to the state machine. The input value to the state machine is calculated from the test results using the following parameters:

- Correctness of the answer (c)
- Response Time (t)

The correctness value is a binary value for the questions of objective nature. For subjective questions the teachers need to evaluate them manually and feed in the performance score along with the recorded

response time. The score for these questions are scaled down, so that they lie between [0,1]. The response time is the time taken to answer each question. The teacher puts a threshold time (t') for each question (in min.). If the student answers correctly within t' then she gets full credit. Then from t' to $3t'$ the value diminishes according to Eq. 2. Finally, after $3t'$ the credit becomes 0. The score for each question, is calculated using the following formula.

$$\begin{aligned} \alpha &= c \text{ if } t \leq t' \\ &= c \times \left(1 - \left(\frac{t-t'}{3t'-t'} \right)^2 \right) \text{ if } t' < t \leq 3t' \\ &= 0 \text{ if } t > 3t' \end{aligned} \quad (2)$$

where t' is the threshold time set for the question

where c is the correctness of the answer, having value 1 for a correct answer, 0 otherwise, t is the response time, and t' is the threshold time.

Input value surpassing the threshold value of a topic indicates a successful completion of that topic. This leads to the augmentation of this topic in the *Coverage* part of the state. Again, the input value is combined with the previous performance state to compute the next performance state.

3.3 Teaching Model

In any ITS, a teaching agent is required which will control and conduct the teaching processes of the system. This agent is the nucleus of the ITS which communicates with the other modules and plans the teaching strategies to be taken for individual students. It, along with the student model defines the “how-to-teach” task of the ITS (Wenger, 1987; Murray, 1999). Among the various tasks it needs to carry out, one of the most important one is to plan the sequence of the various curriculum and course elements to teach a particular course (Brusilovsky & Vassileva, 2003). Most of the planning is done separately for individual students. Thus, the communication with the student model is essential for any decision-making done by this module.

In our system, the overall task of the teaching model is divided into a couple of smaller assignments or sub-tasks. Depending on the data about the students’ state available in the student model the teaching model engine –

- Traverses the Topic Dependency Graph to select the most appropriate topic sequence separately for each student,
- Selects the most suitable set of contents for the chosen topic (by the topic planner), from the Repository for a particular student.
- Collects and analyzes the feedbacks (test results) from the students and plans how to improve upon the drawbacks found in the feedbacks

The first task is done by the module *Topic Planner*. The second task is done by the *Material Selection Module*. A third module within the teaching model, called *Result Analyzer* assesses the student’s result in the recent test and performs some tasks as well as decides on certain issues. These include updating the student model according to the performance of the student, checking the performance and deciding whether she needs to undergo repetition of the whole topic or some parts of it. This module also checks whether the current teaching plan is successful or needs alterations. To summarize the objective of this module, it can be stated that it deals with the dynamic adaptation of the system. If any decision-making does not prove to be effective this module identifies those situations and produces

alternative plans. It also performs some post-test operations such as updating the databases. It takes the results from the testing interface and analyzes them. Then it updates the student model and signals the control engine (TP and MSM) for the necessary changes.

3.3.1 Topic Planner

Numerous topics constitute a course and studying each and every topic will eventually cover the whole course. These topics can be ordered or sequenced in various ways, where each sequence will represent a different way of covering a course. Each sequence represents a separate teaching plan. Different plans can be more effective in teaching as different students can be offered different teaching plans. In a classroom only a single plan can be followed for teaching and all the students (having different capabilities and objectives) have to follow the same plan. Thus, in a classroom the method of teaching may not be effective for all the students. The objective of the Topic Planner is to search for all the possible plans and work out the most suitable one for each student, using which it is assumed that the student will learn efficiently.

Linearization of Topic Dependency Graph: Often a topic is related to the other and coverage of a topic is not possible without the knowledge of its dependent topic. This relation or the dependency is called *prerequisite-of* relation. If a topic is a prerequisite of another topic, the former should always be taught before the latter. In case in a sequence of topics, a prerequisite topic occurs after its dependent topic, the sequence will be an *inconsistent* sequence. The topic planner searches in the TDG for the prerequisite relations between the various topics and plans consistent sequences. These sequences are followed by the students to study a course. Different students may be offered different sequences.

We define three states for a given topic. The states are *learnt*, *ready-to-learn* and *not learnt*. As the names suggest the state *learnt* means that the topic is taught to the student and he learnt the topic. The state *not learnt* means that the topics is not taught to the student. A topic in *ready-to-learn* state depicts that although this topic is not yet studied by the student but all its prerequisites have been covered by her and this topic now can be studied. The difference between the '*not learnt*' and '*ready-to-learn*' topics are that, although both kinds of topics are not studied by the student, a '*ready-to-learn*' topic is permitted for teaching in the current conditions. After each topic is successfully learnt by a student, the Topic Planner searches for all the *ready-to-learn* topics. Among the searched topics it tries to compute the most suitable one as the next topic to be taught (the selection process is discussed later in this section). This process is repeated after each topic is learnt until the last topic is covered. This process of searching an appropriate and consistent sequence of topics for a student is called the *Linearization of the TDG*.

Figure 4 shows a section of the TDG of the course Optics. Root is a dummy node whose children are the possible starting topic (which has no prerequisites) of the course. The leaves are the final topics of the course and these topics are not prerequisites of any other topics in the course. In this tree every path from the root to any leaf is valid sequence covering all the topics in a course and the course can be taught by traversing from the root to any of the leaf. All the prerequisites of a topic (node) lie in the path between the root and the node's parent. This makes all these paths to be consistent. The task of the Topic Planner is to construct this tree and check which of the path will help the student to learn most effectively. However, the whole tree is not constructed by the Topic Planner in one go. The tree is expanded dynamically and only in one direction, and the other parts are pruned. This saves computation time as well as memory space. We will now discuss about how the tree is expanded and traversed, in order to cover or learn the course.

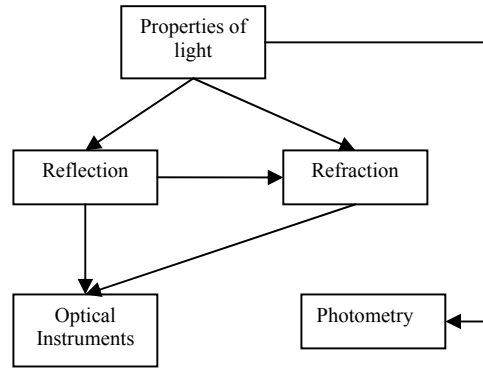


Figure 4. A section of the TDG of course Optics

According to the TDG (Optics) in Figure 4, ‘Properties of Light’ is the only topic without any prerequisites. Therefore, the starting topic will be ‘Properties of Light’. After the successful completion of this topic there are two choices. The student can either study ‘Reflection’ or ‘Photometry’ next. These are the only two topics whose prerequisite is ‘Properties of Light’. Thus the coverage of ‘Properties of Light’ changes the status of those topics to *ready-to-learn*. The expanded structure of the tree is shown in Figure 5. Now the task of this module is to decide which topic between the two will be better for the student to study next.

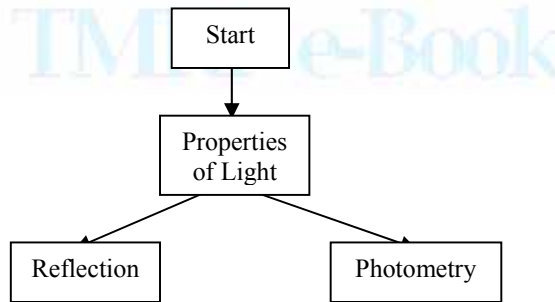


Figure 5. Initial expansion of the tree

In order to take such a decision the Topic Planner compares all possible topics (in this case ‘Reflection’ and ‘Photometry’) with the student’s current learning state, obtained from the student model. Consulting the student model, each topic is assigned a *suitability value*. This value expresses the suitability of a topic for a particular student. Now, the topic with a higher value of this *suitability value* will be chosen as the next topic. Say, in this case ‘Reflection’ had a higher value and was selected.

The suitability value of the *ready-to-learn* topics will be calculated again. Note that the suitability value of ‘Photometry’ might be different from the value calculated earlier, as student’s current learning state is used to calculate this value, and the learning state might change after the coverage of ‘Reflection’. Selection and expansion will continue until all the topics in the course are covered successfully. In Figure 6 a possible expansion of the tree is shown.

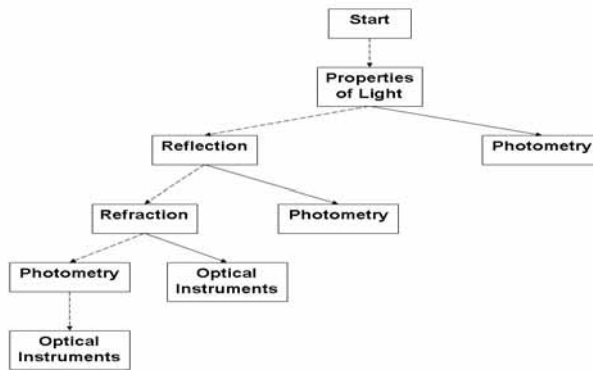


Figure 6. A possible expansion of the tree

Methodology for Linearization: The decision making in this module, like the evaluation of the suitability values of the topics are based on fuzzy rules. There is a rule base consisting of various fuzzy rules. Tasks, like how to select a topic for a student or how to plan the course of teaching for a particular student are defined by these rules. These rules define the teaching strategy to be taken. Suitability value of a topic is the degree of appropriateness between this topic and a student. Therefore, calculation of this parameter involves a comparison between the topic and the student's current state of knowledge and her cognitive ability. Topics are described using various attributes and are kept in the domain model and the description of the student and her cognitive abilities can be extracted from the student model. Some of these attributes are used to establish a relation between the topic and the student and calculate how suitable the topic is for the student. The fuzzy rules define such relations (Kavcic et al, 2003). The antecedent of the rules is constituted by these attributes, whereas the consequent is the *suitability value*. The various features used in the antecedent of the rules, along with their meaning and values are shown in .

Table 1. Each of these terms is a separate fuzzy variable, whose values are defined using specific fuzzy sets (Ross, 1997).

Table1. Antecedents of the Fuzzy Rules

Topic Attributes		
Name	Values	Meaning
Hardness	{Tough, Moderate, Simple}	Hardness of the material
Importance	{VeryHigh, High, Moderate, Low, VeryLow}	How important this topic is in this course
Scope	{Excellent, VeryGood, Good, Average, Poor}	How many opportunities or avenues does this topic open up in the course
Student Attributes		
Performance	{Excellent, VeryGood, Good, Average, Poor}	Overall performance of the student in this course up to this point
Prerequisite Performance	{Excellent, VeryGood, Good, Average, Poor}	Performance in the prerequisite topics of this topic
Interest	{VeryHigh, High, Moderate, Low, VeryLow}	How much interest the student has in this topic

The rules derive the *suitability value*. Fuzzy term ‘Suitability’ has fuzzy values “Very_High”, “High”, “Medium”, “Low” and “Very_Low”. The ‘Suitability’ value is defuzzified using *centroid* defuzzification method (Ross,1997). A typical rule is shown here.

IF Performance is *Excellent* **AND** Pre-equisite_Performance is *Excellent* **AND** Interest is *VeryHigh* **AND** Hardness is *Simple* **AND** Importance is *VeryHigh* **AND** Scope is *Excellent*

The rule-base is editable. Thus, they can be constructed or modified by experienced teachers using the Rule Editor of the authoring system. The rule base contains various such rules. During a decision-making process the inputs are fired with each and every rule in the rule base to get the individual outputs. Then the outputs are combined to compute the final *suitability value*. Individual *suitability values* are obtained for all the topics in *ready-to-learn* state. Finally, the topic with the highest Suitability crisp value from the set of *ready-to-learn* topics is chosen as the next topic. This process is continued till the *ready-to-learn* set is empty. This infers that all the topics in the course have been covered by the student.

3.3.2 Material Selection Module

In the previous section we discussed about how topics were selected and sequenced for individual students. However, actual teaching or studying involves selection and presentation of the study materials. Selection of materials is also a part of the personalization scheme of this ITS. There are various kinds of documents to teach the same topic. Some are relatively difficult than the rest and different documents some have different approaches of explaining the same concepts. The system must interpret the student’s preference and ability to select the proper type of documents for her. This module searches the repository to construct a set of suitable documents to teach each topic for each student. After the *Topic Planner* selects a topic the *Material Selection Module* (MSM) is invoked. This module produces a set of documents which the student follows to learn the current chosen topic. The repository contains documents associated with a set of metadata. The metadata associated with the documents helps in retrieving the learning materials for the chosen topic.

3.3.3 Result Analyzer

The third task of the Control Engine is to analyze feedbacks obtained from the student’s test results and perform some operations accordingly. This task is undertaken by the Result Analyzer (RA). After a session of learning through study materials the students must undergo a test. These tests are a reflection of the student’s understanding in the topic. Using this feedback, the ITS gets an updated knowledge of the student’s performance and uses this knowledge to take various decisions.

Preliminary task of the RA is to provide the results to the student model and update the various fields of the student profile. It supplies the test results to the student model so that it can compute the next state of the student’s performance. Apart from these, RA scrutinizes the test results to obtain some analysis on the student’s performance. This analysis finds out the various faults in the student’ learning and some decisions are taken, so that the student can overcome her flaws. RA also checks whether a change in the teaching plan is necessary for a better outcome from the student. Thus, RA helps the system to adapt dynamically with the changing condition of the student. An earlier plan may not work for a student and she may get stuck at some point and cannot advance from there. Such a condition can be only detected from the feedbacks (test results) from the student. Thus, RA identifies such situations from the test results and works out methodologies to get out from those situations i.e. revise the teaching plan computed earlier.

4. CASE STUDY

The system was used in a real teaching and learning scenario. We organized a workshop for school students to learn basic 'Programming in C' using the ITS, *Shikshak*. Thirty three students from the different schools in the campus of IIT, Kharagpur participated in this workshop. The participants varied from 12 to 16 years in age. No students had any prior exposure to programming. The students studied various topics of C programming. First step was to configure the course in the ITS. Course Tree Editor was used to configure the Course Tree (CT) for this course. The configured CT is shown in Figure 7. As this course was meant for school students, only four basic topics were chosen. The topics were,

- Variables, data type and operators
- Loops
- Branching
- Array

The attributes and their values of these topics are shown in Table 2.

Table 2. Topics and the attributes

Topics	Hardness	Importance	Prerequisite topics	Concepts
Variables,Data types, Operators (V)	0.5	High	None	Variables, Arithmetic and logical operators, data types, printf, scanf
Loops (L)	0.8	High	V	For loops, while loops
Branching (B)	0.6	Moderate	V	If-else, switch-case, continue, break
Array (A)	1.0	Low	L	Linear array, string



Figure 7. Screenshot of the Course Tree Editor showing Course Tree constructed for the experiment

All the topics were furnished with the required details, such as *concepts*, *prerequisites*, *difficulty* etc. Accordingly the system drew the TDG from the prerequisite information. The drawn TDG is shown in Figure 8.

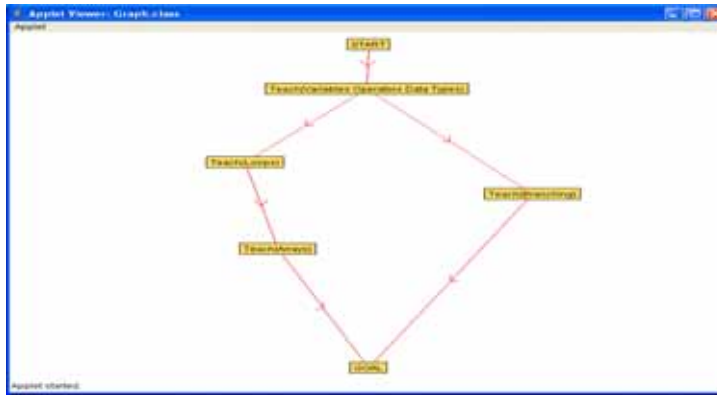


Figure 8. Screenshot of TDG drawn by the system from the CT in Figure 7

The next step involved preparation of the repository. For each topic various materials were created or collected and incorporated into the repository. Care was taken to make sure the materials were heterogeneous in nature. Moreover, files with different presentation styles like *text-intensive*, *animation based* or *presentation type* of documents were also incorporated. Some materials, particularly Macromedia flash based animation based documents were built specially for this purpose. Screenshot for such a document is shown in Figure 8. Variety of documents enriched the repository and helped the *Material Selection Module* to select the appropriate type of materials from the different variety available. This served the basic purpose of the ITS, where unlike a classroom, students had the opportunity of studying from materials which suited them best. This was followed with the annotation of the materials. Annotation was done very carefully, as incorrectly annotated materials would lead to inconsistent retrieval from the repository and affect the system's performance. In order to test the students' knowledge various questions were also frames. The questions varied in terms of difficulty and focus (comprehension-ability or problem-solving skills)

Apart from domain model organization, some other preparations were required in the student model and the control engine. Student model was configured to accommodate the participating students. This initialization of the student model involves furnishing of some initial information about the students. These data were obtained from their school records and fed into the student model. In the control engine some rules were framed, using the Rule Editor. These rules defined the strategies to be taken by the Topic Planner to sequence the course for each student.

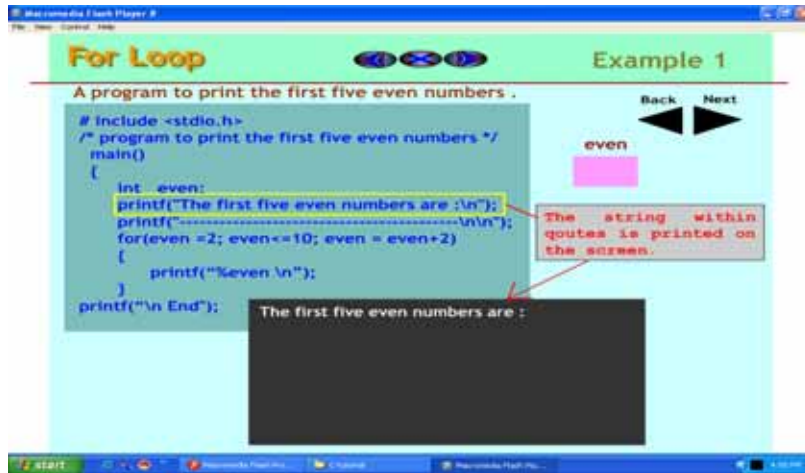


Figure 8. A sample learning material

The Topic Planner computes separate topic sequences for different students. In Figure we show the different sequences can be drawn for this course.

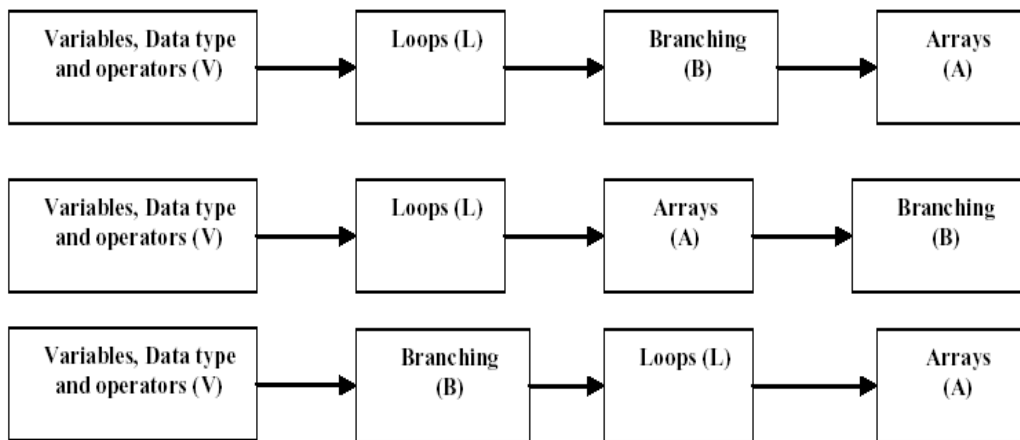


Figure 9. All possible sequences drawn from the TDG (Figure) used in the workshop

We present the percentage of students following each sequence in Figure 10. The “others” column collectively represents all the cases were sequences had to be re-planned for some students.

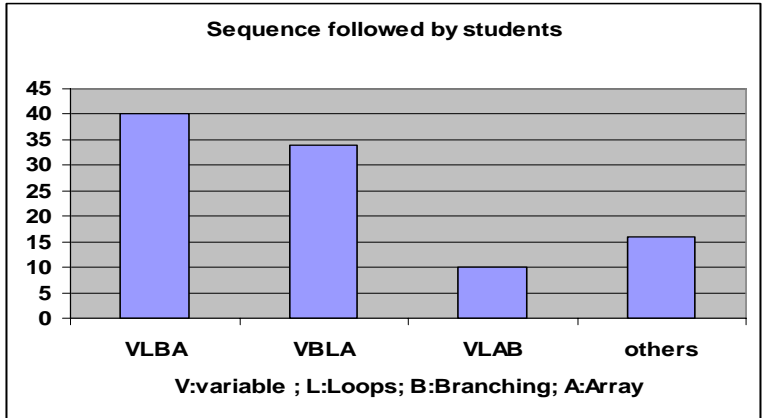


Figure 10. Percentage of each Sequence followed by students

We claimed that an important aspect of an ITS is that the students can learn and progress along a course at their own pace. The duration of the course was 15 days but many fast learning students completed the course before that. However, some of the students could not finish the course in 15 days. Perhaps 15 days were not sufficient and they required some more time. The detailed observations are shown in Figure11. The red (last three) columns signify that those students were unable to complete the course in 15 days.

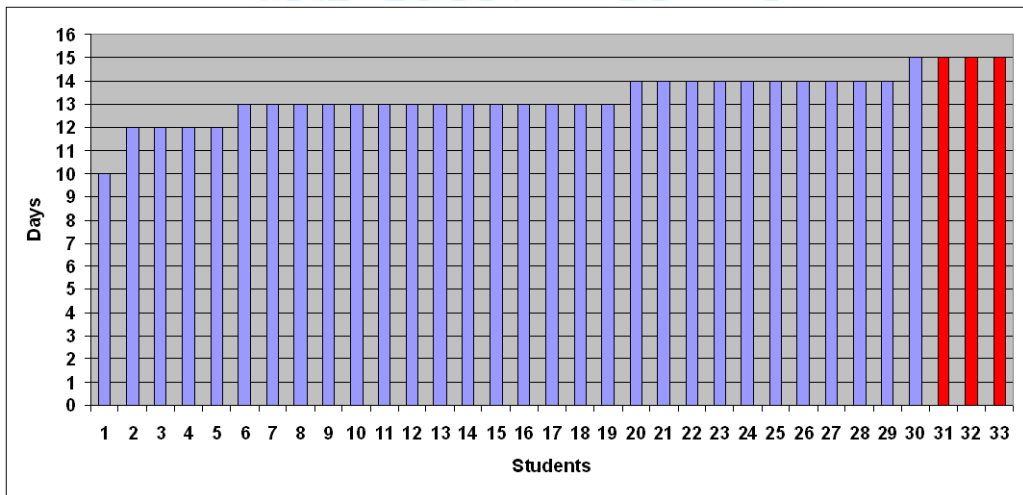


Figure 11. Days taken by each student to complete the course (the last three columns represent the students who attended the course for 15 days but could not finish)

5. CONCLUSION

In this chapter, we have discussed about the development of an Intelligent Tutoring System (ITS). We started the chapter by discussing about ITS and some its general features. In the following section we presented some previous works related to ITSs. Then, we described the complete system with a full

description of its each module. Finally, we presented some results obtained during a field trial of the system.

The main objective of our work was to develop a system that can teach and deliver contents in a customized and adaptive manner. The system considers the cognitive ability of each student and devises teaching plans which would suit them best individually and maximize their performance. In summary, we can claim that the system exhibits moderately good performance. The main contribution of our works are summarized below:

- A generic and domain independent structure to store and represent the courses. Using this representation scheme various courses can be configured in the system. Simple courses for primary education to more complex courses of higher education can be configured using this scheme. The configuration of courses using this scheme is also simple and requires very little experience of computer usage from the teachers.
- A set of tags through which learning materials can be annotated. Most of the tags have been used from the IEEE LOM set of standard metadata for annotating learning materials. However, we have added some tags locally to make the annotation scheme compatible with our system.
- A fuzzy state transition based student model which is an abstract representation of the students using the system. This model also uses two important parameters to evaluate the cognitive ability of a student. They are comprehension ability which tests a student's ability to understand a concept or a topic. The other is problem-solving skills, which checks a student's ability to apply a concept in solving problems. The student model provides the basis of adaptation offered by the system.
- Another important contribution of this work is a fuzzy rule based teaching agent which controls the adaptive teaching process of this system. This agent communicates with the other modules of the system and produces customized teaching plans for the individual students. This module also keeps track of a student's changing requirements along a course and adapts itself accordingly.

However, there are some limitations in the present system. One of the limitation is that there are no facilities to provide hints to students. This might help a student immensely in problem solving and avoid getting stuck into a problem. The present student model is capable of capturing two aspects of a student. These two parameters are not enough to capture a complete picture of a student's cognitive ability. The current system has been evaluated on the basis of the users who came from good urban schools. However, this system is to deploy in rural areas where the system can be quite relevant and might play an important role in improving the existing infrastructure in education. It is quite important to check how the system performs when users from those areas actually use the system. Therefore, a major future work is to deploy the system in various rural educational centers and evaluate its performance.

References

Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. 2006. The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. In M. Ikeda, K. D. Ashley, & T. W. Chan (Eds.), in 8th International Conference on Intelligent Tutoring Systems (ITS 2006), Berlin: Springer Verlag, pp. 61-70.

Ali, M.S., Ghatol, A.A. 2004, A Neuro Fuzzy Inference System For Student Modeling In Web-Based Intelligent Tutoring Systems, International Conference on Cognitive Systems, New Delhi, December 2004.

Baffes, P., and Mooney, R. 1996. Refinement-Based Student Modeling and Automated Bug Library Construction. In *Journal of Artificial Intelligence in Education*, 7, 1 (1996), pp. 75-116,

Beck, J.E. 1997. Modeling the student with reinforcement learning. Machine Learning for User Modeling Workshop at the Sixth International Conference on User Modeling, 1997.

Beck, J.E., Woolf, B.P. 2000, High-Level Student Modeling with Machine Learning, in 5th International Conference on Intelligent Tutoring Systems, G. Gauthier, C. Frasson, K. VanLehn (Eds.): ITS 2000, LNCS 1839, pp. 584–593, Montreal, Canada, 2000.

Bloom, B.S. 1984. The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher* 13 (6):4–16.

Brown, J., Burton, R., deKleer, J. 1981, Pedagogical Natural Language and Knowledge Engineering Techniques in SOPHIE I, 11, and IIIH, Tutoring Systems, Sleeman et al (eds), Academic Press.

Brusilovsky, P., Vassileva, J. 2003. Course sequencing techniques for large-scale web-based education. In *International Journal of Content Engineering and Lifelong Learning*, Vol. 13, Nos. 1/2, 2003.

Bull, S., Pain, H. 1995. "Did I say what I think I said, and do you agree with me?": Inspecting and questioning the student model, *Proceedings of AI-ED'95 - 7th World Conference on Artificial Intelligence in Education*, AACE, pp. 501-508.

Chou, C., Chan, T., & Lin, C. 2003. Redefining the learning companion: the past, present, and future of educational agents, *Computers & Education*, v.40 n.3, pp.255-269, April 2003.

Clancey, W. J. 1982, GUIDON, In Barr and Feigenbaum (editors), *The Handbook of Artificial Intelligence*, chapter Applications-oriented AI research: Education. William Kaufmann, Inc., Los Altos, 1982.

Collins, J. A., Greer, J. E., Huang, S. X. 1996. Adaptive Assessment using Granularity Hierarchies and Bayesian Nets. In Frason, C., Gauthier, G. and Lesgold, A., (eds.), *Proceedings of Intelligent Tutoring Systems ITS'96*. Berlin: Springer, pp. 569-577.

Conati, C., Gertner, A., VanLehn, K., & Druzdzel, M. 2002. On-line student modeling for coached problem solving using Bayesian networks. In *Proc. Sixth International Conference on User Modeling*, Vienna, 1997, pp. 231-242.

D'Mello, S. K., Craig, S. D., Gholson, B., Franklin, S., Picard, R. W., & Graesser, A. C. 2005. Integrating affect sensors in an intelligent tutoring system. In *Affective Interactions: The Computer in the Affective Loop Workshop at International Conference on Intelligent User Interfaces 2005*, San Diego, 2005, pp. 7-13.

Evens, M. W., Brandle, S., Chang, R., Freedman, R., Glass, M., Lee, Y. H., Shim, L. S., Woo, C. W., Zhang, Y., Zhou, Y., Michael, J. A., & Rovick, A. A. 2001. CIRCSIM-Tutor: An intelligent tutoring

system using natural language dialogue, in Proc. *Twelfth Midwest AI and Cognitive Science Conference*, Oxford, 2001, pp. 16-23.

Ferguson, K., Arroyo, I., Mahadevan, S., Woolf, B., & Barto, A. 2006. Improving intelligent tutoring systems: Using expectation maximization to learn student skill levels. In Proc. 8th International Conference on Intelligent Tutoring Systems, Taiwan, June, 2006, *Lecture Notes in Computer Science*, No 4053, pp. 453-462.

Freedman, R. 2000a, What is an Intelligent Tutoring System?, Published in *Intelligence* 11(3): pp. 15–16.

Freedman, R. 2000b. Plan-based dialogue management in a physics tutor. Proceedings of the Sixth Applied Natural Language Processing Conference. Seattle, WA, pp. 52-59.

Gertner, A., & VanLehn, K. 2000. Andes: A coached problem solving environment for physics. In *Proc. 5th International Conference. Intelligent Tutoring Systems*, Berlin, 2000, pp. 133-142.

Horvitz, E., Breese, J. S., Heckerman, D., Hovel, D., Rommelse, K. 1998. The Lumiere Project: Bayesian user modeling for inferring the goals and needs of soft-ware users. Fourteenth Conference on Uncertainty in Artificial Intelligence. San Francisco: Morgan Kaufmann, pp. 256-265.

Kavcic A., Pedraza-Jimenez R., Molina-Bulla H., Valverde-Albacete F.J., Cid-Sueiro J., Navia-Vazquez A. 2003. Student modeling based on fuzzy inference mechanisms, *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, vol.2, no.pp. 379- 383 vol.2, 22-24 Sept. 2003

Khuwaja, Ramzan, A., Evens, M. W., Michael, J. A., & Rovick, A. A. 1994. Architecture of CIRCSIM-Tutor (v.3): A smart cardiovascular physiology tutor. In *Proceedings of the 7th Annual IEEE Computer-Based Medical Systems Symposium, Winston-Salem, NC, 1994*, pp. 158-163. IEEE Computer Society Press

Kimball, R. 1982, A self-improving tutor for symbolic integration. In D. Sleeman & J.S. Brown (Eds.), *Intelligent Tutoring Systems*, New York: Academic Press, pp. 283-308.

Kodaganallur, V., Weitz, R., & Rosenthal, D. 2005. A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms. *International Journal of Artificial Intelligence in Education*, Vol. 15, pp. 117-144.

Martin, B., Koedinger, K., Mitrovic, T., & Mathan, S. 2005. On Using Learning Curves to Evaluate ITS. *Twelfth International Conference on Artificial Intelligence in Education*, pp. 419-426, Amsterdam.

McDonald, J. 1981, The Excheck CAI System, in *University-Level Computer-Assisted instruction at Stanford: 1968-1980*, P. Suppes, ed., Inst. for Math. Studies in the Social Sciences, Stanford Univ., Palo Alto, Calif., 1981.

Mitrovic, A. 2003. An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education*, 13(2-4), 2003, pp. 171-195.

Murray, T. 1999. Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art. *International Journal of Artificial Intelligence in Education* (1999), 10, pp. 98-129.

- Ong, J., & Ramachandran, S. 2000. Intelligent Tutoring Systems: The What and the How. <http://www.learningcircuits.org/2000/feb2000/ong.htm>
- Reye, J. 1996. A Belief Net Backbone for Student Modeling, Proceedings of third International Conference on Intelligent Tutoring Systems, June 12-14, 1996, pp. 596-604.
- Reye, J. 1998. Two phase updating of Student Models based on Dynamic Belief Networks, Proceedings of fourth International Conference on Intelligent Tutoring Systems, Aug 16-19, 1998, pp. 274-283.
- Ross, T.J. 1997. Fuzzy Logic with Engineering Applications. McGraw-Hill, 1997.
- Sleeman, D., & Brown, J. S. 1982. Introduction: Intelligent Tutoring Systems. *Intelligent Tutoring Systems*, D. Sleeman, J. S. Brown, Ed. Academic Press, 1982, pp. 1-11.
- Thompson, J.E. 1996. Student Modeling in an Intelligent Tutoring System. *MS Thesis*.
- VanLehn, K., Martin, J. 1997. Evaluation on an assessment system based on Bayesian student modeling. *International Journal of Artificial Intelligence in Education*, Vol. 8. pp. 179-221.
- Villano M. 1992. Probabilistic Students Models, Bayesian Belief Networks and Knowledge Space Theory, Proceedings of the second International Conference on Intelligent Tutoring Systems, June 10-12, 1992, pp. 491-498.
- Vomlel, J. 2004. Bayesian networks in educational testing. *Int. J. Uncertain. Fuzz. Knowl. Based Syst.* 12(Suppl. 1), 83–100 (2004)
- Wang, T., & Mitrovic, A. 2002. Using neural networks to predict student's performance. In *Proc. of International Conference on Computers in Education, 2002*, pp. 969-973.
- Webb, G. I., B. C. Chiu, and M. Kuzmycz 1997. Comparative Evaluation of Alternative Induction Engines for Feature Based Modelling. *International Journal of Artificial Intelligence in Education* 8. Amsterdam: IOS Press, pages 97-115.
- Wenger, E. 1987. *Artificial Intelligence and Tutoring Systems*. Los Altos, CA: Morgan Kaufmann.
- Weon, S., & Kim J. 2001. Learning achievement evaluation strategy using fuzzy membership function. In *Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference, Reno, NV, 2001, October* pp. 10–13.
- Xu, D., Wang, H., Su, K. 2002. Intelligent Student Profiling with Fuzzy Models, In *Proceedings of the 35th Hawaii International Conference on System Sciences, Hawaii, USA (2002)*, p. 81b.
- Zapata-Rivera, D., Greer, J. 2004. Interacting with Inspectable Bayesian Student Models. *International Journal of Artificial Intelligence in Education*, Vol 14. pg., 127 – 168
- Zhou, Y., Evens, M. W. 1999a. A Practical Student Model in an Intelligent Tutoring System, *ICTAI*.