Chapter 2

Literature Survey

The design of Intelligent Tutoring System utilizes techniques covering a wide spectrum of disciplines that includes artificial intelligence, domain knowledge, educational psychology, and psychometric. In this context, the literature survey in this chapter focuses on student modeling, pedagogy, and assessment. Table 2.1 summarizes the main points for discussion in the subsequent sections.

Table 2.1: Issue and points of discussion in the literature review of Intelligent Tutoring System.

<table>
<thead>
<tr>
<th>Student modeling</th>
<th>Domain-specific information</th>
<th>Domain-independent information</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Stereotypes</td>
<td>• Constraint-based</td>
</tr>
<tr>
<td></td>
<td>• Overlay</td>
<td>• Model tracing</td>
</tr>
<tr>
<td></td>
<td>• Extended overlay</td>
<td></td>
</tr>
<tr>
<td>Pedagogy</td>
<td>Learning theories</td>
<td>• Behaviourism</td>
</tr>
<tr>
<td></td>
<td>• Cognitivism</td>
<td>• Cognitivism</td>
</tr>
<tr>
<td></td>
<td>• Constructivism</td>
<td>• Constructivism</td>
</tr>
<tr>
<td>Assessment</td>
<td>Action selection</td>
<td>• Alternative</td>
</tr>
<tr>
<td></td>
<td>• Diagnostic</td>
<td>• Decision-theoretic</td>
</tr>
<tr>
<td></td>
<td>• Alternative</td>
<td></td>
</tr>
</tbody>
</table>

Student modeling is defined as the representation of the computer system belief about the learner and therefore is an abstract representation of the learner (Holt et al., 1994). The
research is on student modeling traits of the student such as student’s mastery of the domain and his domain specific behavior. In student modeling, the learner’s answer to prompted item is used to determine his knowledge mastery. Since student’s intermediate solution is usually not visible to tutor, various non-deterministic modeling technologies such as Dampster-Shaffer certainty factor (Shortliffe, 1976), fuzzy logic (Matsubara & Nagamachi, 1996; Tang et al., 2000), and Bayesian network (Reye, 1998; Conati & VenLehn, 2000; Far & Hashimoto, 2000; Gertner & VenLehn, 2000; Millán et al., 2000; Mayo, 2001; Murray, 2005) are required.

A fine grain model can detect specific student’s knowledge mastery with less uncertainty, whereas a coarse grain model provides quick inferences but possesses more uncertainties in student knowledge mastery.

Considering the student modeling in the light of instructional design, learning theory is useful. There are three main learning theories- behaviourism, cognitivism and constructivism. Behaviourism and cognitivism theories are compatible with the notion of modeling the student but constructivism is entirely at odds with student modeling.

Behaviourism theory is the oldest theory, where the behavior of the learner is compared to machine that produces responses when exposed to stimulus. By a process known as conditioning, a particular response is obtained for a specific stimulus. In this process, the stimulus is presented repeatedly and the correct response is reinforced while the incorrect response is punished.

Cognitivism theory postulates that learning is the “acquisition or reorganization of the cognitive structures through which humans store and process information” (Good & Brophy, 1990). This theory explains cognitivism as the formation and reformation of mental representations of the domain knowledge. The knowledge structures are called schema and
the memory is divided into sensory, short term and long term. The “effects” such as “organizational effect” and “meaningful effect” which increase the efficiency of learning is also presented in this theory.

In constructivism theory, learners are said to construct their own reality based on experience. Hence, it is a subjective theory, whereas behaviourism and cognitivism are objective theories. In constructivism theory, new knowledge is formed, rather transferred and every student is expected to construct his or her own unique reality.

2.1 Student Model

Student modeling is the most important part of an Intelligent Tutoring System since the student has the central role in the teaching–learning process. A student model represents what the student knows, and a pedagogical model presents instructional strategies to teach domain knowledge. Student model stores details about the student’s current problem-solving state and long-term knowledge progress, essential for adapting the material to the student’s characteristics (attributes).

Student models can be classified by a number of factors and are constructed from the primitives provided by the system. The information held in a student model is generally categorized as domain-specific and domain-independent information.

2.1.1 Domain-Specific Information

Domain-specific information represents the student’s current state and level of knowledge relating to a particular concept. Stereotype models, Overlay models and Extended Overlay models are few approaches that use the domain-specific information.
Stereotypes are the simplest student model. They are simple abstract classifications of students into groups that are fixed, either permanently or initially. Overlay model represents the learner's knowledge as a subset of domain knowledge. Extended overlay model represents two sets of learner knowledge. One set is similar to the overlay model, and the other set is a set of misconceptions of the learner. This set falls beyond the set of expert knowledge.

**Stereotyping**

Stereotyping is the simplest form of student modeling with students being assigned to a specific predefined stereotype according to his/her responses. There are two types of stereotypes: *fixed* (Winter & McCalla, 1999) and *default* (Kay, 2000a).

**Fixed Stereotype**

The “Fixed” stereotype approach to student modeling captures the student's responses and then assigns the student a stereotype based on the responses. A basic method of fixed stereotyping is to assign a level to a student depending on their performance.

In WPS-Tutor, (Wheeler & Regian, 1999), problems are divided into levels which are the main description of the student in the student model. Each level is only slightly more difficult than the previous level. The level increases when the student solves two or more problems without help on the current level. Winter and McCalla (1999) carried out fixed stereotyping in the domain of software engineering ethics, in which they identified five different stereotypes. In this domain, personality types rather than mastery were modeled.

Milne et al. (1996) also describe fixed stereotyping in their system ATULA. ATULA is a system for teaching mathematical network theory. The aim is to use stereotypes as a guide for teaching actions of the system. When a new student uses the system, the probability of
the student being in each stereotype is calculated after they complete the same initial questionnaires. The student is then assigned to the most probable stereotype.

Fixed stereotyping makes the broad assumption that all students within a stereotype possess the same domain knowledge and display the same problem solving behavior. The system may move a student from one stereotype to the other, but the stereotypes themselves do not change. Although the approach is not useful for complex analysis, it is a realistic student modeling technique for open domains where knowledge cannot be decomposed into atomic units.

In general, fixed stereotyping is a very coarse-grained representation of a student, and the approach is not useful for more complex analysis. It is also questionable whether fixed stereotyping is even valid because it may be the case that the stereotypes vary from one group of students to another. However, for open domains where the knowledge cannot be easily decomposed into atomic units, this approach may be the only realistic student-modeling alternative.

**Default Stereotype**

In this type of stereotyping, the settings of each initial stereotype should be considered "default" values only. When the students start using the system, they are stereotyped, and as the evidence about the student is arrived at through observations, the initial stereotypes are gradually replaced by more individualized setting. This approach is used along with an overlay model, as it provides initial values for the overlay.

This approach is used in a number of systems surveyed by Kay (2000a). StyLE-OLM (Dimitrova et al., 1999) is a learning environment for scientific terminology that uses this approach. The system engages in a natural language dialogue with the student and using its
rules, draws conclusions about the student's knowledge based on the student's responses. However, these default inferences are open to scrutiny by the student. By the process of dialogue, the student is able to fine tune the details.

MANIC (Stern et al., 1999) uses only one default stereotype which initializes its student models to a default called the "population student model" that is derived from empirical analysis. This is considered to be an extreme case of default stereotyping.

**Overlay Student Model**

Overlays represent the student's domain knowledge as a subset of the domain expert's knowledge. In other words, it is a set of masteries over the items of the domain. The model simply estimates the mastery of each element in the domain that an expert would be expected to know. The only precondition that needs to be considered for applying this approach is that it should be possible to break down the domain knowledge into generic items such as rules, concepts, facts, etc.

An example of an overlay model for a domain that is decomposed into ten skills or items is depicted in Figure 2.1. The mastery of each item ranges from (0) to (1). An expert of the domain is expected to master each item at (1). Typically, the overlay model of the student is initially assigned (0) mastery for each item. The mastery changes dynamically according to the student's behavior. In Intelligent Tutoring Systems that have combined stereotypes with overlays, the overlay model would be initialized to the default stereotype.
There are at least two different interpretations of the measure of mastery in the literature. In some systems, mastery is considered as a binary variable that takes the value mastered or not-mastered, and the measure is based on the system's belief (Pearl, 1988) that the item is in the mastered state. This could be termed as probabilistic interpretation, and is representative of Intelligent Tutoring Systems that use Bayesian probability for student modeling. In other systems, the measure is to be interpreted as an actual state of the student mastery. So, for example, if the mastery of an item is \(0.5\), it means that the system believes that the student has only partially mastered the skill and needs more practice, but not a complete novice at the skill. This is representative of the more traditional style of the overlay model. In practice there should be little difference in behavior between systems with differing interpretations, as both will act to maximize the measure.

The overlay model can also be specified at any level of granularity, and rules can be defined to compute the overlay. This approach was adopted by OLAE (VanLehn & Martin, 1997). In many systems, the measure of mastery is a simple function of the frequency with which the item has been used correctly or incorrectly (e.g. Clancey, 1983, 1987; Kay, 2000b), or some function of the frequencies of different observations of student behavior.
e.g. Bloom et al. (1997). “Bounded” representations where the measure is uncertain but between a lower and upper limit have also been proposed (Elsom-Cook, 1990).

Intelligent Tutoring System researchers are recently using Bayesian networks to obtain more accurate probabilistic overlay models (Reye, 1998; Millán et al., 2000; Mayo, 2001). The probabilistic overlay is a set of uncertain, probabilistic variables representing the student's mastery of a domain. The overlay model is updated in a probabilistically sound way from observation which is the main advantage of this approach.

The posterior probability of knowledge mastery of a specific topic is computed using Bayes' theorem (Equation 2.1) from the prior probability ($P(A)$) and observations ($B$) made about the student.

$$ P(A | B) = \frac{P(B | A)P(A)}{P(B)} $$  

where $P(B | A)$ is the probability of a student producing the observed response given his knowledge mastery.

Mayo (2001) has classified Bayesian network student models into three different groups, according to the technique by which they were constructed. These classifications are expert centric, efficiency-centric, and data-centric models.

![Fig 2.2: A classification of student model using Bayesian network.](image)
(a) Expert-Centric Models

A domain-expert directly or indirectly specifies the structure and also the probabilities of the network. ANDES (Gertner & VanLehn 2000; Gertner et al., 1998; Conati et al., 1997), HYDRIVE (Mislevy & Gitomer, 1996), DT-Tutor (Murray & VanLehn, 2000; Murray, 2005), and ADELE (Ganesan et al., 2000), are examples of tutors with expert centric models where large Bayesian networks with structures mostly engineered from complex domain analysis are used.

A major hurdle for these systems is to define conditional probabilities for variables representing unobserved, internal student states. ANDES’ solution is to use “coarse-grained” conditional probability definitions such as noisy-OR and noisy-AND. In practice, restricting conditional probabilities to noisy-ORs and noisy-ANDs significantly reduce the number of required probabilities and make the modeling of unobserved variables much simpler because only the structure and node type (noisy-OR or noisy-AND) need to be specified.

The conditional probabilities are defined subjectively in HYDRIVE, and it assumes equivalent classes of states or scenarios as a student works through a problem. Unlike cognitive tutors who attempt to wholly model the expert, the goals behind the design of HYDRIVE’s student model was only to capture the factors important to discriminating between proficient and less proficient students.

ADELE (Agent for distance learning environments) (Ganeshan et al., 2000) uses causal knowledge to dynamically generate a diagnostic process that is consistent with the best practice to medical diagnosis. This model is an animated pedagogical agent. ADELE uses a combination of hints and other interactions based on multiple-choice questions and guides the student through a reasoning process that exposes him to the underlying knowledge.
DT-Tutor is a generalized domain-independent architecture for student modeling and Pedagogical Action Selection (PAS). Like ANDES and HYDRIVE, it models the student’s knowledge, but it goes much further and attempts to model other hidden states such as the student’s morale, independence, and focus of attention.

Expert centric Bayesian network are not structurally restricted in anyway to match the domains as closely as possible. They have a high proportion of variables representing unobserved internal states of student. A major hurdle is the elicitation of conditional probabilities for these variables in the absence of data.

(b) Efficiency-Centric Models

Efficiency-centric models restrict the structure of the network in order to maximize efficiency. The model is partially specified or restricted in some way, and domain knowledge is “fitted” to the model. The restrictions are generally chosen to maximize some aspect of efficiency, such as specification size (e.g. Millán et al., 2000) and efficiency of evaluation (e.g. Collins et. al., 1996; Reye, 1998).

Reye’s model is a generalization of the student model used in the ACT Programming Languages Tutor (Corbett & Anderson, 1992; Corbett & Bhatnagar, 1997), and a similar approach was used in the student model of SQL-Tutor (Mayo & Mitrovic, 2000). The idea is to model the student’s mastery of a knowledge item over time. The tutor’s current belief that the student has mastered the item ($M_t$) depends on its previous belief ($M_{t+1}$), the outcome of the student’s last attempt at the item ($O_{t+1}$), and the pedagogical response of the tutor to the last attempt ($A_{t+1}$). Using dynamic Bayesian networks, not only can the tutor’s current beliefs be determined, but also its future beliefs at time $t+1$ or beyond, although this is likely to be much more uncertain. This model is depicted in Figure 2.3 for a single knowledge item.
There are several risks with this approach, such as oversimplifying the model and/or introducing incorrect assumptions. Another constraint is the mastery of a knowledge item probabilistically independent of the mastery of any other items that are pre- and co-requisites. For example, if the knowledge items are “high level” such as concepts or topics, then, the mastery of some items would be expected to be dependent on the mastery of items that are pre- and co-requisites. This is a basic assumption of many systems and the rationale behind many approaches to course sequencing, e.g. Brusilovsky (2000). Alternatively, the knowledge items could be “low level” such as constraints (Ohlsson, 1994; Mitrovic & Ohlsson, 1999; Mayo, 2001).

Millán et al. (2000) proposed an architecture that is both expert- and efficiency-centric. Fixing the directionality of the arcs between groups of variables and limiting the states with specific semantics are the ideas of this architecture.

**(c) Data-Centric Models**

In this category, the structure and conditional probabilities of the network are learned primarily from the data collected from the real-world evaluations of the tutor. It does not
model unobserved student states, such as their domain mastery but models relationships between observed variables to predict student performance.

MANIC (Stern et. al., 1999; Beck & Woolf, 2000) have adopted the data-centric approach. There are a number of benefits with this approach. Firstly, because the model is induced from actual data, its predictive performance can easily be evaluated by testing the network on data that was not used to train. Secondly, data-centric models can be expected to be much smaller than the typical expert centric model because the latter represents both observed and hidden variables, while the former models only observable variables.

The data centric model is applied by Mayo (2001) to the development of optimal pedagogical action selection strategies for CAPIT (Capitalization And Punctuation Intelligent Tutor). CAPIT is a constraint-based tutor that teaches the basic rules of English punctuation and capitalization. Mayo concentrates on modeling the relationships between observed variables to predict student performances. The two decision-theoretic pedagogical action selection strategies developed using this methodology are problem selection and error message selection.

There are some limitations with the data centric models. The networks in CAPIT have no explicit models of the student's internal cognitive representation of the domain. It is not scalable to larger and different domains.

**Extended Overlay Student Model**

The overlay student model has two important limitations. First is that it does not acknowledge that students may have beliefs that are not part of the expert's knowledge base. Second is the unrealistic simplification that suboptimal behavior is caused only by the insufficient mastery of individual skill? An extension to the overlay model (see Figure 2.4)
explicitly represents faulty knowledge (or bugs) that the student may have (Holt et al., 1994).

Extensions to include bugs can take various forms (Wenger, 1987). Enumerative modeling is a method where students' misconceptions and errors are recorded. The system, later selects the correct and incorrect model elements that account for the student's solution. The advantage of this method is that likelihood of errors can be derived from domain experts.

A student model could be constructed from the primitives of a language for any domain that covers both correct and incorrect knowledge. The resolution granularity of the primitives is adequate to represent all knowledge and misconceptions by appropriate combinations of primitives. This method could be further classified into generative and reconstructive bug generation.

(a) Enumerative Modeling

In this modeling, a protocol is used to analyze the possible bugs and misconceptions that students can have. Along with the expert's knowledge the bugs also become part of the domain model.
An early Intelligent Tutoring System DEBUGGY (Burton, 1982) is an enumerative extended overlay model of the student. It is used to identify student's knowledge and bugs from the test results. An example of an incorrect solution to typical subtraction problem is depicted in Figure 2.5.

\[
\begin{array}{c}
307 \\
- 135 \\
\hline
232
\end{array}
\]

Fig 2.5: A student's incorrect solution to a subtraction problem.

The bug-rule explaining the buggy answer \((0 - n = n)\), is a modification to the subtraction procedure stating that when the top digit of a column is 0, then the answer for the column is the bottom digit. In the above case, the bug results in the digit 3 from 315 being copied directly to the second column of the solution.

Two recent extended overlay models are described by Webb et al. (1997) and Virvou and Tsiriga (2000). Webb et al. (1997) describe Feature-Based Modeling (FBM) application to the prediction of student problem-solving actions again in the subtraction domain. The goal of this modeling is to use a machine learning algorithm to associate context and actions. The set of features describing the current state of the problem is the context. The actions range from the general to the specific. This model asserts that cognitive assumptions are not necessary, and that machine learning techniques are required to predict errors from problem states. FBM-IS and C4.5 (Quinlan, 1993) are the two machine learning algorithms. FBM-IS and C4.5 increase accuracy, statically significantly to 93% and 92% respectively. In this approach, the composite bugs cannot be predicted by the system which restricts its expressiveness.

Another approach, an Intelligent Tutoring System for algebraic powers (i.e. addition, multiplication etc. of integer powers) is Easy Math (Virvou & Tsiriga, 2000). Empirical
study was conducted before constructing Easy Math. Two hundred and forty students were given a test which covered the entire domain and a library of common mistakes was determined.

LMS (Sleeman & Smith, 1981; Sleeman et al., 1989; Sleeman, 1982) from the domain of algebra, and PROUST (Soloway & Johnson, 1981; Bonar & Soloway, 1985), an Intelligent Tutoring System for teaching Pascal are the other systems with enumerative extended overlay models.

A threefold problem is met with enumerative approach. (i) The computational tractability of searching the space of bug compositions is a severe problem. It may be completely intractable to match composite bugs in a domain more complex than subtraction. (ii) It has been shown that bugs vary greatly from school to school and even class to class (Holt et al., 1994) even within a narrow domain. This limits the generality of any bug library. (iii) As a result of this variability, the bug libraries are expensive to elicit in terms of time. Intelligent Tutoring System community considered machine learning techniques for building bug libraries.

(b) Generative Modeling

In this approach, the Intelligent Tutoring System uses a cognitive model to explain the student’s erroneous behavior. It is assumed that the system will be able to deduce the underlying misconceptions leading to the bug from the cognitive model; hence no bug library is required.

Matz (1982) made an early exploration of generative modeling. Matz’s approach consists of two components: base rules and extrapolation techniques. Base rules are the knowledge obtained from examples or read directly from a textbook. Extrapolation
techniques are rules for applying base rules to an unfamiliar situation. Errors can be explained as the result of failed extrapolation. The majority of buggy student behavior can be explained if the general forms of the extrapolation errors in a domain can be determined.

When analyzing errors made by children learning algebra, Matz found a number of domain specific extrapolation errors capable of explaining large number of errors. For example, one extrapolation error is to decompose the topmost operator of an algebraic expression over its parts. This is sometimes correct e.g.:

\[ A (B+C) = AB + BC \]

But when inappropriately applied, it becomes incorrect, e.g.:

\[ \sqrt{A + B} = \sqrt{A} + \sqrt{B} \]

To cope with new situation, algebra rules can also be falsely revised. For example,

\[ (x - 3) (x - 4) = 0 \]

To obtain the solution \( x = 3 \) or \( x = 4 \). From this the student learns the correct rule.

\[ (X - A) (X - B) = 0 \]

is \( X = \text{Solve}[X - A = 0] \) or \( X = \text{Solve}[X - B = 0] \), when faced with a new, but slightly similar problem of the form

\[ (X - A) (X - B) = K \]

the student incorrectly revises the solution, rule to \( X = \text{Solve}[X - A = K] \) or \( X = \text{Solve}[X - B = K] \). Conceptual changes can lead to extrapolation errors.

REPAIR theory was constructed by VanLehn’s (1982, 1990) after an extensive study of children solving arithmetic problems, where many types of bugs were found. The theory focuses on consistent bugs. This theory also explains that the bugs are not always totally predictable and consistent. Bug Migration is a phenomenon where students shift back and forth between different bugs in the same situation, because when the students arrive at an
impasse during problem solving, they can select any incorrect step in an attempt to "repair" the procedure. This theory was implemented in a system SIERRA (VanLehn, 1990). Different repairs lead to different bugs.

(c) Reconstructive Modeling

This is an approach to reconstruct erroneous reasoning. The assumptions are stricter and it is assumed that the domain is composed of a set of operators. A procedure is composed of sequence of operators and bug forms when the student learns an incorrect sequence of operators. Inducing the operator sequence that best fits a student's or group of student's observed errors is the aim of machine learning algorithm.

An example of an Intelligent Tutoring System with a reconstructive student model is ACM (Automated Cognitive Modeler) (Langley et al; 1990). The tutor is implemented in the domain of elementary subtraction. The system performs a depth-first search of the space of operator sequences, to reconstruct a student's buggy solution. As a result, the observation is that student's buggy solutions are often shorter than the length of correct solution; the search depth is limited to the length of the correct solution plus one.

INSTRUCT (Mitrovic et al; 1996) is another reconstructive tutor. The inferences made about the student from very small amounts of information are one of the problems with ACM and other earlier reconstructive tutors. INSTRUCT solves the problem by tracking the student's problem solving actions so that the additional information can be used to diagnose the student's knowledge effectively. Traces are sequences of operators observed. This also takes into account that students, once they learn a particular trace, can "chunk" the operators into a single macro-operator.
(d) Combinations

SPENGELS (Bos & Plassche, 1994) is an Intelligent Tutoring System combining both enumerative and generative techniques. It teaches the correct conjugation and spelling of English verb forms. Two stages of expert algorithm exist. The first stage is to determine the correct suffix of the verb, a decision tree is used. The second stage is to combine the verb and the suffix using the set of morphological and spelling rules. A considerable amount of bug is represented and it contains explicit buggy rules. It is also generative in the sense that the system can combine buggy rules to find the sequence that explains the student’s incorrect answer and it can traverse the decision tree of the expert spelling algorithm itself in order to find the point where incorrect decision tree was followed.

When existing approaches are generalized to achieve good domain coverage, they tend to produce many more implausible than plausible bugs, because of the huge search spaces involved. The reports of Sleman et al. (1989) further weakened the extended overlay mode. It showed that a simple re-teaching strategy in the domain of algebra was just as effective as bug remediation.

2.1.2 Domain Independent Information

Domain independent information may include learning goals, cognitive aptitudes, motivational states, preferences, learning styles, and factual and historical data.

Two approaches which have different stances on domain-independent information modeling are Constraint-Based Modeling (CBM) and Model Tracing (MT). The main distinction between the two is that model-tracing tutors represent both the procedural and declarative knowledge possessed by the student, whereas constraint-based tutors represent only the declarative knowledge.
**Constraint-Based Modeling**

Constraint-Based Modeling (CBM) is a student modeling method that describes only pedagogically informative states, rather than following the procedure the students used to arrive at their answers. The idea arose from Ohlsson (1994) who suggested that students, in fact, rapidly switch between several different strategies when problem solving, even after instruction. Some of the strategies are correct while others are incorrect. Ohlsson defines this as the *radical strategy variability* phenomenon, and if it turns out to be the normal case, then it invalidates approaches that assume that the student follows only a single path to a problem’s solution.

Another curious phenomenon about students is their ability to find and correct errors before/after making them (Mitrović & Ohlsson, 1999). He asserts that this is because the declarative knowledge learned has not been internalized in the procedural knowledge, and so the number of decisions to be made while performing that the procedure is sufficiently large which leads to mistakes.

In a constraint-based tutor, the domain model is represented by a set of constraints, where each constraint represents a pedagogically significant state. That is, if a constraint is relevant to the student’s answer, this is an example of a principle that is to be taught to the student. If the constraint is violated, the student does not know this concept and requires remedial action. A key test of whether or not a constraint represents a single pedagogically significant state (i.e. that all problems/solutions that fall into this state are pedagogically equivalent) is whether or not a single piece of feedback can be delivered for all problems that violate this constraint.

Once the domain model has been so defined, feedback actions can be associated directly with the constraint. SQL-Tutor (Mitrović 1998) is an example of an Intelligent Literature Survey 33
Tutoring System that uses CBM. The domain model consists of over 500 constraints. A simple overlay student model is used, which records the number of times each constraint has been satisfied or violated. The domain model is limited purely to describing how to critique a student solution. SQL-Tutor, for example, uses a very simplistic student model that does not include any form of curriculum. CBM is also not concerned with how problems are produced or selected, and provides detail of only one type of feedback: declarative messages that are attached to each constraint.

The main advantage of CBM is that it is highly tractable. Little computational effort is required for constraint matching. A corrected version of an incorrect student answer could be generated despite the student and ideal solutions being fundamentally different (Martin & Mitrovic, 2000). Much human intervention would still be required, because the feedback for each constraint would need to be input, the inducer is unlikely to induce all desired constraints; and some of the induced constraints may be fortuitous or incorrect generalizations. Furthermore, an algorithm exists for “merging” constraints into a unified structure called a RETE network that can increase the efficiency of constraint matching even more (Mitrovic, 1998).

Model Tracing

In model-tracing, the tutoring system maintains a model of problem solving that is compared against the student’s actions. Feedback during problem-solving is given based on current state of the model and the rules that represent student cognition and action. They are referred to as model-tracing tutors because they contain an expert model which is used to trace the student’s responses to ensure that the student’s responses are part of an acceptable solution path. The creation of such tutors, particularly the expert models that underlie them requires much knowledge outside the domain being tutored.
The exemplary model tracing tutors are the family of tutors called cognitive tutors that are based on the ACT-R theory developed by Anderson (Anderson et al., 1996; Anderson & Lebiere, 1998; Anderson, 1993). ANDES is an Intelligent Tutoring System in classical physics developed by researchers at the Learning Research and Development Center (LRDC) at the University of Pittsburgh and at the United States Naval Academy (USNA). ANDES allow students to solve physics problems in an environment that provides visualization, immediate feedback, procedural and conceptual help. It is a cognition theory that assumes that all knowledge is either declarative or procedural. Procedural knowledge is represented as a production set. The cognitive tutors represent an expert's knowledge as a large production set called the ideal student model. It uses much more sophisticated Bayesian techniques than knowledge tracing for reasoning about the student's mastery and behavior.

ANDES' Bayesian networks encode two kinds of knowledge: (i) domain-general knowledge, encompassing general concepts and procedures that define proficiency in Newtonian physics, and (ii) task-specific knowledge, encompassing knowledge related to a student performance on a specific problem or example. The evolution of a student's domain-general knowledge as the student progresses through ANDES' curriculum is modeled by a dynamic belief network (Russell and Norvig, 2003), a belief network designed to model the evolution of a system over time. The part of the belief network encoding domain-general knowledge is built when the ANDES knowledge base is defined, and its structure is maintained across problems and examples. At any time, the probabilities in this network represent either a priori assessment of a student's knowledge or the assessment after the last performed exercise. The part of the network encoding task-specific knowledge is automatically generated from the solution graph of each problem for example when the student begins to work on it, and it is initialized with priors coming from the
domain-general nodes. When the student has finished the current exercise, the task-specific part of the network is discarded, and the updated domain-general probabilities are saved to be "rolled-up" as priors for the successive exercise.

Model Tracing tutoring systems have been fielded in a variety of domains like College-level physics (Gertner & VanLehn, 2000; Shelby et al., 2001), High school algebra (Koedinger et al., 1997; Heffernan & Koedinger, 2002; Heffernan, 2001), Geometry (Anderson et al., 1985; Wertheimer, 1990), and Computer programming (Corbett & Anderson 1993; Corbett & Bhatnagar, 1997).

2.2 Learning Theories

Three learning theories, namely Behaviourism, Cognitivism, and Constructivism, are of importance in computer based learning (Hergenhahn & Olson, 1997). These three theories are described briefly in the following sections.

2.2.1 Behaviorism

The behaviorism (Skinner, 1953; Walker, 1996) theory is based on observable changes in behavior. Behaviorism focuses on a new behavioral pattern being repeated until it becomes automatic. It views the learner as a blank slate, and the instructor must provide the experience. A cue or stimulus from the environment is presented and the learner reacts to the stimulus with some type of response. Consequences that reinforce the desired behavior are arranged to follow the desired behavior. The new behavioral pattern is repeated until it becomes automatic. The behavior of the learner signifies that learning has occurred.

It views the mind as a "black box" in the sense that response to stimulus can be observed quantitatively, totally ignoring the possibility of thought processes
occurring in the mind. Some key players in the development of the behaviorist
theory are Pavlov, Watson, Thorndike and Skinner (Sternberg, 2001).

The weakness of this theory is that the learners may find themselves in a
situation where the stimulus for the correct response does not occur, therefore, the
learner cannot respond. Strength is that the learner is focused on a clear goal and can
respond automatically to the cues of that goal.

2.2.2 Cognitivism

Constructivism (Greeno et al., 1996; Mayer, 1996) focuses on preparing the learner
for problem solving in ambiguous situations. Cognitive theorists recognize that much
learning involves associations established through contiguity and repetition.

Cognitive information processing is based on the thought process behind the behavior.
The changes in behavior are observed, but only as an indicator to what is going on in the
learner's head. Cognitive information processing is used when the learner plays an active
role in seeking ways to understand and process information that he or she receives and
relates it to what is already known and stored within memory. The learner is viewed as
having a more proactive role in his/her own learning with this theory. The major
contribution to cognitivism is by Jean Piaget (Sternberg, 2001), who developed the
major aspects of his theory as early as 1920.

The weakness is that the learner learns a way to accomplish a task, but it may
not be the best way, or suited to the learner or the situation. For example, logging
onto the internet on one computer may not be the same as logging in on another

Literature Survey 37
computer. The major strength is that the goal is to train learners to do a task the same way to enable consistency.

2.2.3 Constructivism

Constructivism (Eggen & Kauchak, 2001) is based on the premise that learners construct their own perspective of the world, based on individual experiences and internal knowledge structure. Learning is based on how the individual interprets and creates the meaning of his/her experiences (Brooks, 1990).

The features of constructivism outlined above have founded a wide variety of learning environments (Savery & Duffy, 1995), and the problem-based learning environment is almost the ideal and most popular area that implements the constructivism theory (Savery & Duffy, 1995).

Simulation technologies have always been associated with constructivism. This relationship was perhaps due to the fact that technology provided students with almost unlimited access to information that they needed in order to do research and test their ideas.

The weakness of this theory is that in a situation where conformity is essential, divergent thinking and action may cause problems, and the strength is that because the learner is able to interpret multiple realities, the learner is better able to deal with real life situations.

2.3 Action Selection

Each decision taken by the tutor for remediation is based on the current information in the student model. Generally, there are three main approaches for using Bayesian network-based student model in action selection namely: alternate strategies, diagnostic strategies.
and decision theoretic pedagogical strategies. The three classes and the systems that fall into them are given in Table 3.2.

<table>
<thead>
<tr>
<th>Alternate</th>
<th>Diagnostic</th>
<th>Decision-Theoretic</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISP Tutor</td>
<td>Millán et al., 2000</td>
<td>DT-Tutor</td>
</tr>
<tr>
<td>ANDES</td>
<td>Collins et al., 1996</td>
<td>CAPIT</td>
</tr>
<tr>
<td>ADELE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQL-Tutor</td>
<td>Milne et al., 1996</td>
<td></td>
</tr>
</tbody>
</table>

2.3.1 Alternate Strategies

Alternate strategies take the posterior probabilities of the Bayesian network and use them as the input to some heuristic decision rule. To illustrate, the LISP Tutor uses a simple heuristic to decide whether or not to advance the student to the next section; if the probability of the student’s mastery exceeds some threshold, the student advances to the next section otherwise not (Anderson et al., 1996).

ANDES selects hints for the student based on the solution path that the student is following to solve the current problem (Gertner et al., 1998). The student’s solution path is not certain always. It could be on paths, A, B or C with posterior probabilities \( P(A) > P(B) \) and \( P(A) > P(C) \). The system uses the heuristic of assuring that the most probable solution path is the student’s solution path (e.g. A, assuming \( P(A) > P(B) \) and \( P(A) > P(C) \)). However, this is a sub-optimal heuristic as demonstrated by a simple counter-example. Suppose, the optimal hint for solution path A as defined by the learning theory is \( H_a \), but the optimal hint for both paths B and C is \( H_b \). Then if it is the case that \( P(B) + P(C) > P(A) \), hint \( H_b \) will be optimal, but the rule will incorrectly select hint \( H_a \).
ANDES also have heuristic decision procedures disconnected entirely from the student model. For example, a simple matching heuristic is used to generate feedback on incorrect equation entries (Gertner et al., 1998).

As ANDES, ATULA (Milne et al., 1996) also uses the same "most probable explanation" strategy. It calculates a probability that the student is in each of the six possible different stereotypes.

Another system using heuristic decision procedures is ADELE (Ganeshan et al., 2000). ADELE has a Bayesian network model of the domain knowledge, and it uses a heuristic based on focus-of-attention to select the node in the network about which to provide a hint. Decision-theoretic processes were considered but abandoned because they were considered too inefficient.

SQL-Tutor uses a heuristic for problem selection (Mayo & Mitrovic, 2000). The main rationale for this was that, like ADELE, the computation required for exact decision-theoretic computation (which would have involved more than 500 constraints) made direct application of decision theory intractable.

Ad-hoc decision procedures seem prevalent because there is the obvious convenience, simplicity and efficiency and there is the perception that precision in intelligent tutoring is not required. In the words of Katz et al. (1992), ".....in low risk decision-making situations such as tutoring..... Imprecise student modeling is adequate".

2.3.2 Diagnostic Strategies

Millán et al. (2000) describe an approach for optimizing test question selection. The approach expands on a strategy suggested by Collins et al. (1996). The Bayesian network has nodes representing questions, concepts, topics, and one node (say, A) for the overall
proficiency of the student. Answers to questions are observed and probability distributions over the other nodes are inferred via Bayesian updating.

Millán et al. describe a procedure for question selection and the criterion is informativeness. When \( P(A = \text{Proficient}) \) or \( P(A = \text{Not-Proficient}) \) exceeds some maximum threshold \( t \), then the system stops selecting questions because the probability of mastery or back of mastery is sufficiently precise.

Utility is defined as informativeness; this approach cannot be used for pedagogical action selection. It is useful only for the diagnostic testing of the student’s state of knowledge, and furthermore, it makes the implicit assumption that the student’s knowledge does not change for the duration of the test so remedial actions such as instruction cannot be performed until afterwards. Therefore, outside testing process, the approach has limited applicability.

2.3.3 Decision-Diagnostic Strategies

Decision-Diagnostic Strategies take outcome probabilities computed by a Bayesian network and multiply them by their associated utilities outside the network to compute expected utilities for decision-theoretic action selection. These include an Intelligent Tutoring System for English capitalization and punctuation (CAPIT, Mayo, 2001) and various other user modeling applications (e.g. Horvitz et al., 1999). An advantage of computing expected utility outside the network is potentially faster inference due to a smaller network (no decision or utility nodes with associated arcs) and the flexibility to consider only subsets of actions or outcomes in the expected calculations.

Some student modeling applications use decision network or equivalent representations to directly compute the decision with maximum expected utility. DT Tutor (Murray & Van...
Lehn, 2000; Murray, 2005) and Conati's (2002) representation for an educational game use dynamic decision network architectures to select actions for helping a user with a task. Jameson et al. (2001) use a decision network to decide whether to present instructions individually or several at a time.

One benefit of decision-theoretic representations is support for value of information computations to guide user queries and other information-seeking behaviors. Applications that utilize value of information include those of Horvitz and colleagues (Horvitz et al., 1998).

2.4 Assessment

The assessment stage is meant for assessing the overall progress of the learners' domain competence. The focus of assessment should be on the acquisition of skills in the application of facts in different contextual and non-contextual scenarios and emphasis should be more on the cognitive skills (Stiggins, 1997).

Four useful principles that any assessment system or approach must address in learning settings, according to the BEAR assessment principles are,

i. Assessments should be based on a developmental perspective of student learning.

ii. Assessments in e-learning should be clearly aligned with the goals of instruction.

iii. Assessments must produce valid and reliable evidence of what students know and can do.

iv. Assessment of data should provide information that is useful to tutors and students to improve learning outcomes.
Two basic processes involved in assessment are measurement and evaluation. Measurement is collecting all information by the tutor and evaluation is the process of making decisions based on the measurement.

The measurement methods are traditional and alternative. The traditional methods are multiple choice questions, multiple response questions, and graphical hotspot questions; fill in blanks, text/numerical input questions and matching questions. This testing doesn’t support sufficiently wide and comprehensive knowledge assessment. Those tests don’t allow to assess student’s knowledge structure, i.e., how he/she has understood the relations between concepts or how new concepts are connected with the previously mastered concepts.

Alternative assessments rely on process. It provides methods for holistic learning. Students’ knowledge levels are more accurately estimated, especially for high and low achieving students. This is a way to see the students’ cognitive structure i.e., their knowledge structure. It is difficult to implement due to the limitations of human computer interaction. Examples of Intelligent Tutoring System using alternative assessment are SimQuest (Veermans & Joolingen, 1998) and Ecolab (Luckin, 1998). The SimQuest system is capable of analyzing the learner’s performance based on the information generated, and drawing conclusions about the developing domain of the learner and the quality of learner’s discovery behavior. Ecolab uses collaborative assistant to investigate how software can offer help and support to an individual learner.

OLAE (VanLehn & Martin 1997) which integrates with ANDES physics tutor (Gertner & VanLehn, 2000) is a system designed specifically for assessment. OLAIE is a graphical tool that simplifies the task of assessing the student’s overall performance. The probability
that the student has mastered the group is automatically computed by OLAE. This probability is defined as the product of the individual probability that the student has mastered all (say, \( n \)) the rules. If a student mastered \( n-1 \) rules with a high probability but has a very low probability of mastery of the \( n \)th constraint will have very low overall probability of mastery.

2.5 Human Computer Interaction

Intelligent Tutoring Systems attempt to build a model of the student, and use that model in conjunction with knowledge about the domain of instruction and instructional strategies to modify the order of presentation of material, selection of hints and corrections, and style of interaction with the student.

ELM-ART (Brusilovsky, 1996) is a web-based Intelligent Tutoring System designed to teach an introductory LISP course. It provides learners with visual cues (icons, fonts, colors) that show the type and the educational state of each link. ELM-ART has pioneered the idea of an adaptive electronic textbook and introduced the traffic light metaphor for adaptive navigation support in educational hypermedia.

A study of ELM-ART has demonstrated that casual users stay longer within a system when adaptive navigation support is provided. It also provided evidence that direct guidance works best for users with little previous knowledge while adaptive annotation is most helpful for users with some reasonable subject knowledge. InterBook system (Brusilovsky & Pesin, 1998), a direct descendant of ELM-ART provided the first authoring platform for Web-based adaptive hypermedia.

InterBook has refined the ideas of the adaptive electronic textbook and the traffic light metaphor for adaptive navigation support in educational hypermedia (see Figure 3.6).
Propagated by ELM-ART and InterBook, this metaphor has later been used in numerous adaptive educational hypermedia systems, including AST (Specht et al., 1997), KBS-HyperBook (Henze & Nejdl, 2001), and SIGUE (Carmona et al., 2002). A study of InterBook has shown that adaptive navigation support encourages non-sequential navigation and helps users who follow the system's guidance to achieve a better level of knowledge.

ELM-ART and InterBook have also explored a relatively new adaptive navigation support technology known as link generation. It creates new, non-authored links on a page. InterBook was among the first systems to have implemented adaptive link generation. It has also demonstrated that link generation can be naturally used in combination with link sorting and annotation.

The Knowledge See II system (Brusilovsky & Chavan, 2003) coupled with AnnotatED social navigation system explored some simple forms of social navigation based on group user modeling and the idea of “footprints” (Wexelblat & Mayes, 1997). It uses the simplest
implicit feedback: for each tutorial page it counts how many times it was accessed by a group of users. This amount of traffic is visualized as a color density that students observe during navigation.

2.6 Approach Used in This Research

BiTutor can be used to teach Computer Science courses to the students. The knowledge in these courses consists of rules, procedures, methodologies and problem solving skills. It is difficult to reply on either a set of cognitive processes or predefined operators. In this research, to develop the student model, overlay model is applied. The student model consists of a set of Bayesian networks for each course. Compared to other non-deterministic techniques, Bayesian network enables the domain knowledge structure (curriculum) to be represented in a hierarchical way. The main difficulty in using Bayesian network is in specifying the network structure and assigning its parameters (Conditional Probability Tables).

In the case of BiTutor, expert-centric approach is used to initialize the student model when the expert (teacher) is constructing the network. In expert-centric approach, there is no initial information about the student's knowledge mastery and so default stereotype approach is used. When the data is available, the Bayesian network can be constructed by applying learning Bayesian network techniques on the data. So, when the data is available, data-centric approach is used.

The Cognitivism theory of learning is followed by the pedagogy adopted by BiTutor. Decision-theoretic approach is used in implementation of pedagogy in BiTutor. The values associated with student's learning can be coded in the utility functions. Elicitation of values is separated from probability assignments associated with the student's knowledge mastery.
The decision-theoretic approach is normative because any utility function can be removed or amended without affecting the procedures. This makes it easy to maintain the tutor.

For assessment, multiple choice questions are used. As the question is selected based on the knowledge of the student, using Item response theory, the feedback should also be chosen based on the student's mastery level on the topic. Whenever the student's response is wrong, suitable feedback is posted to the student.

BiTutor differs from CAPIT and DT-Tutor in the following aspects.

i. The item-response theory is applied to improve the accuracy of diagnosis and item selection in BiTutor. Item selection in CAPIT and DT-Tutor are not based on item-response theory.

ii. The expected utility value of an action in BiTutor depends on the gain of information and student's behavior. In CAPIT, the expected utility of each pedagogical action (possible next problem) is calculated when combined with estimates of the probabilities of each outcome (e.g., no error, 1 error, 2 errors, etc.). In DT-Tutor, the utility is based on multiple objectives such as knowledge’s states, focus on attention, independence and morale, action relevance and dialog coherence.

iii. BiTutor can be used to generate a tutoring strategy up to $n$ future actions or till the readiness to terminate the tutoring criteria is met. No strategy generation is mentioned in CAPIT and DT-Tutor.

Finally, BiTutor is developed as a web-based application supporting all range of Internet users. Only authorized staff is allowed to construct a course based on the constraints provided by the BiTutor. Student community can use this BiTutor for getting trained in any of their courses (computer science) of interest which are already constructed by staff.