ITEAMS - AN INTELLIGENT TEACHING ENVIRONMENT WITH ASSESSMENT
MODULES FOR SELF-STUDY

A Thesis

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science

by

Daniel Dentinger, B.Sc.

Matthias Scheutz, Director

Graduate Program in Computer Science and Engineering
Notre Dame, Indiana
July 2005
ITEAMS - AN INTELLIGENT TEACHING ENVIRONMENT WITH ASSESSMENT MODULES FOR SELF-STUDY

Abstract

by

Daniel Dentinger

In the area of education technology, there are two categories in which instructional software can be placed: Teaching Environments and Intelligent Tutoring Systems. Teaching Environments, such as WebCT, allow instructors to create presentations, lecture notes, and quizzes, but lack certain abilities including instructional styles, lectures tailored to the needs of the students, or automatic assessment of the students. Intelligent Tutoring Systems allow students to progress at their own pace through a single subject matter while receiving individually tailored content for their knowledge level but are either specific to a subject matter or limited to a small number of courses. The main goal of ITEAMS is to integrate the important aspects of Teaching Environments and Intelligent Tutoring Systems to create a unique educational environment that enhances the student’s learning experience while not being restricted to a select group of courses. ITEAMS incorporates key features such as (1) the infrastructure to organize lecture materials, (2) the ability to track and assess a student’s performance, (3) the ability to dynamically select lecture materials based on the student’s inferred knowledge level, (4) the automatic grading of quizzes and assignments, and (5) an interface to external applications through a unique ‘plugin’ system.
This thesis is dedicated to my father for always giving his constant support, concern, and motivational talks.
CONTENTS

FIGURES ................................................................. vi

TABLES ................................................................. viii

ACKNOWLEDGMENTS .................................................. ix

CHAPTER 1: INTRODUCTION ........................................ 1
  1.1 Terminology ..................................................... 2
     1.1.1 Graph Theory .............................................. 2
     1.1.2 Probability Theory Terminology ....................... 3
     1.1.3 Bayesian Belief Network Terminology ................. 5
     1.1.4 Intelligent Tutoring System Terminology .............. 6
     1.1.5 ITEAMS Specific Terminology ......................... 7
  1.2 Background ..................................................... 7
     1.2.1 Teaching Environments ................................. 7
     1.2.2 Intelligent Tutoring Systems ......................... 7
     1.2.3 ITEAMS Design Goals ................................. 7
  1.3 Thesis Overview ............................................... 9

CHAPTER 2: WHAT IS A TEACHING ENVIRONMENT ............... 11
  2.1 Feedback ...................................................... 12
  2.2 Student Modeling ............................................. 12
  2.3 Instructor Modeling ........................................... 12
  2.4 Teaching Environments in the classroom .................... 13
  2.5 Previous Work in Teaching Environments .................... 13
     2.5.1 WebCT .................................................... 14
     2.5.2 BlackBoard .............................................. 14
     2.5.3 eCollege ................................................ 15
  2.6 Evaluation of current systems ............................... 16
     2.6.1 Instructor Models ..................................... 16
     2.6.2 Evaluation Schemes ................................... 17
  2.7 Chapter Summary .............................................. 17
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.8</td>
<td>Chapter Summary</td>
<td>95</td>
</tr>
<tr>
<td>6.1</td>
<td>Overview</td>
<td>97</td>
</tr>
<tr>
<td>6.2</td>
<td>ITEAMS Design Considerations</td>
<td>102</td>
</tr>
<tr>
<td>6.2.1</td>
<td>What can be evaluated in ITEAMS</td>
<td>103</td>
</tr>
<tr>
<td>6.2.2</td>
<td>What is evaluated in ITEAMS</td>
<td>105</td>
</tr>
<tr>
<td>6.3</td>
<td>Singly Connected Networks</td>
<td>108</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Pearl’s Message Passing Algorithm</td>
<td>109</td>
</tr>
<tr>
<td>6.4</td>
<td>Multiply Connected Networks</td>
<td>113</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Available Inference Methods</td>
<td>113</td>
</tr>
<tr>
<td>6.5</td>
<td>Determining Singly Connected versus Multiply Connected</td>
<td>118</td>
</tr>
<tr>
<td>6.5.1</td>
<td>General Undirected Graph</td>
<td>118</td>
</tr>
<tr>
<td>6.5.2</td>
<td>General Directed Graph</td>
<td>119</td>
</tr>
<tr>
<td>6.5.3</td>
<td>Directed Acyclic Graph</td>
<td>119</td>
</tr>
<tr>
<td>6.5.4</td>
<td>ITEAMS Implementation</td>
<td>124</td>
</tr>
<tr>
<td>6.6</td>
<td>Structurally Dynamic Belief Networks</td>
<td>125</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Background</td>
<td>125</td>
</tr>
<tr>
<td>6.6.2</td>
<td>Updating Network Values and Inference</td>
<td>126</td>
</tr>
<tr>
<td>6.6.3</td>
<td>Updating Network Structure</td>
<td>128</td>
</tr>
<tr>
<td>6.6.4</td>
<td>Student’s Performance</td>
<td>131</td>
</tr>
<tr>
<td>6.7</td>
<td>Chapter Summary</td>
<td>133</td>
</tr>
<tr>
<td>7.1</td>
<td>Teaching Environments</td>
<td>135</td>
</tr>
<tr>
<td>7.2</td>
<td>Intelligent Tutoring Systems</td>
<td>135</td>
</tr>
<tr>
<td>7.3</td>
<td>ITEAMS Architecture</td>
<td>136</td>
</tr>
<tr>
<td>7.4</td>
<td>Future Work and Improvements</td>
<td>136</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Plugin Connections</td>
<td>137</td>
</tr>
<tr>
<td>7.4.2</td>
<td>Database Connections</td>
<td>137</td>
</tr>
<tr>
<td>7.4.3</td>
<td>Hint Providing Capabilities</td>
<td>139</td>
</tr>
<tr>
<td>7.4.4</td>
<td>Open Learning Environments</td>
<td>139</td>
</tr>
<tr>
<td>7.4.5</td>
<td>Student Selected Instructor Style</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>BIBLIOGRAPHY</td>
<td>141</td>
</tr>
</tbody>
</table>
# FIGURES

1.1 Example of a DAG that is a Singly Connected Graph .......................... 3
1.2 Example of a Multiply Connected Graph ............................................. 4
4.1 ITEAMS Plugin Interface ....................................................................... 49
4.2 ITEAMS Online Environment ................................................................. 54
4.3 A diagram of the offline environment. This environment allows users work on *teaching module* when a fla file is provided to them through an instructor’s method of choice......................................................... 58
4.4 Different views and modes in ITEAMS .................................................. 63
4.5 Edit mode in ITEAMS ............................................................................ 64
4.6 Edit Toolbar in ITEAMS ........................................................................ 65
4.7 Section Editing Capabilities in ITEAMS ................................................. 66
4.8 Section Editing features in ITEAMS ...................................................... 67
4.9 Quiz Editing Windows in ITEAMS ........................................................ 68
4.10 Assignment Editing Window in ITEAMS .............................................. 69
4.11 Assignment Section Editing Windows in ITEAMS ................................ 70
4.12 Section Viewing Window in ITEAMS .................................................. 72
4.13 Quiz Viewing Window in ITEAMS ....................................................... 73
4.14 Assignment Viewing Window in ITEAMS ............................................. 74
5.1 ITEAMS Dr Scheme Plugin ................................................................. 92
5.2 ITEAMS Dr Scheme Plugin Question Creation ..................................... 93
5.3 ITEAMS Dr Scheme Plugin Data Correlation ....................................... 94
6.1 Example of a Belief Network that contains all the probability tables ...... 98
6.2 Sample multiply connected teaching module ......................... 106
6.3 Example of a Multiply Connected Graph ............................. 114
6.4 Another Multiply Connected Graph ................................. 114
6.5 Example of a Multiply Connected Graph with conditioning .... 115
6.6 Example of a Multiply Connected Graph with Clustering .... 116
6.7 Sample Graph for Depth-firs Search Algorithm .................. 121
6.8 Sample Graph after Depth-firs Search Algorithm ............... 122
6.9 Sample Graph after Depth-firs Search Algorithm ............... 123
# TABLES

3.1 TEMPLATE TABLE FOR EVALUATING AN ITS. .................. 20
3.2 AN ITS FOR USE IN A WHOLE CLASS ....................... 21
3.3 SUMMARY OF THE POLA ITS ............................... 32
3.4 SUMMARY OF THE ANDES ITS .............................. 32
3.5 SUMMARY OF THE ACE ITS ................................. 33
3.6 SUMMARY OF THE CILE ITS ............................... 34
4.1 INITIAL ITEAMS EVALUATION AS AN ITS ................... 41
4.2 ITEAMS EVALUATION AS AN ITS AFTER INSTRUCTOR MODEL DISCUSSION ........................................ 43
4.3 ITEAMS EVALUATION AS AN ITS AFTER MATERIAL MODEL DISCUSSION ........................................... 46
4.4 ITEAMS EVALUATION AS AN ITS AFTER EVALUATION MODEL DISCUSSION ........................................ 47
4.5 ITEAMS EVALUATION AS AN ITS AFTER PLUGIN DISCUSSION . 51
4.6 ITEAMS EVALUATION AS AN ITS ............................. 75
5.1 SAMPLE RANGES FOR A PLUGIN ............................. 79
6.1 TABLES CONTAINING THE PRIOR PROBABILITIES OF THE TWO ROOT NODES IN THE ALARM BELIEF NETWORK .................. 99
6.2 DEFAULT CONDITIONAL PROBABILITY DISTRIBUTION FOR ANY BASIC BELIEF NODE IN ITEAMS .................... 105
ACKNOWLEDGMENTS

I am grateful to my advisor for his patience, guidance, and allowing me to continue my work.
Within the field of education technology, two categories exist in which instructional software can be placed: Teaching Environments and Intelligent Tutoring Systems. Teaching Environments allow instructors to create presentations, lecture notes, and quizzes, but lack certain abilities including instructional styles, lectures tailored to the needs of the students, or robust automatic assessment of the students. Intelligent Tutoring Systems allow students to progress at their own pace through a single subject matter while receiving individually tailored content for their knowledge level, but are either specific to subject matter or limited to a small number of courses. The main goal of ITEAMS is to integrate the important aspects of Teaching Environments and Intelligent Tutoring Systems to create a unique educational environment that enhances the student’s learning experience while not being restricted to a select group of courses. ITEAMS incorporates key features of Teaching Environments and Intelligent Tutoring Systems such as

1. the infrastructure to organize lecture materials
2. the ability to track and assess a student’s performance
3. the automatic grading of quizzes
4. the ability to select lecture materials based on the student’s inferred knowledge level.

In addition to incorporating existing features, ITEAMS provides unique features which are not present in current Teaching Environments or Intelligent Tutoring Systems:
1. the ability to provide quizzes and assignments tailored to the knowledge level of the student
2. an interface to external applications through a unique “plugin” system.

This thesis begins with a discussion on Teaching Environments and Intelligent Tutoring Systems. The reason for this is to evaluate the current systems as well as to discuss what improvements could make them better. An exploration of ITEAMS follows which explains the different technologies and designs that have been used.

1.1 Terminology

The meaning of technical terms is often excluded from because determining the intended interpretation can often be difficult. To help the readers, we introduce basic terminology before beginning a discussion on Intelligent Tutoring Systems, Teaching Environments, and ITEAMS. The terms introduced here are used throughout the text; in subsequent chapters, new terms specific to them will be introduced as needed.

1.1.1 Graph Theory

Directed Acyclic Graph

Recall from graph theory that a simple graph can be defined as a set of vertices or nodes, \( V \), and a set of edges, \( E \), or \( G(V, E) \). \( V \) is a non-empty and distinct set of nodes, where the nodes are either ordered or unordered. \( E \) is an unordered set of distinct pairs of nodes \( e \), \( e = (X, Y) \in E \). A path from \( x \) to \( y \) is defined as an ordered set of edges that connect from node \( x \) to node \( y \). A directed graph is a graph \( G(V, E) \) where each \( e \in E \) is an ordered pair indicating that the edge \( (x, y) \) goes from node \( x \) to \( y \) but not in reverse. A directed acyclic graph (DAG) is defined as a directed graph \( G(V, E) \) where \( E \) is a set of distinct edges where there is no path from node \( x \) to node \( x \). Figure 1.1 shows an example DAG.
Figure 1.1. Example of a DAG that is a Singly Connected Graph

Singly Connected Graph

A singly connected graph is a DAG in which there is at most one path between any two nodes. This type of graph is also referred to as a polytree. Figure 1.1 shows an example of such a graph.

Multiply Connected Graph

A multiply connected graph is a DAG in which there is more than one distinct path between two nodes. Figure 1.2 shows an example of such a graph.

1.1.2 Probability Theory Terminology

**Random Variable** A placeholder for a single value from a specific set of values $x_1, x_2, \cdots, x_n$.

**Event** An instantiation of a variable. In a probability $P(X), X$ is the event.

**Evidence** Specific values which variables take on. In a probability $P(X|Y), X$ is the event
and $Y$ is the evidence.

**Conditional Probability** The conditional probability of variable $X$ given $Y$ (denoted by $P(X|Y)$) conditional probability is given by

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)}$$  \hspace{1cm} (1.1)

**Conditional Independence** Two variables $X$ and $Y$ are conditionally independent given the graph $G$ if $P(G) \neq 0$ and one of the following is true:

1. $P(X|Y \cap G) = P(X|G)$ and $P(X|G) \neq 0, P(Y|G) \neq 0$ \hspace{1cm} (1.2)

2. $P(X|G) = 0$ or $P(Y|G) = 0$ \hspace{1cm} (1.3)
Bayes Theorem  Given two events $X$ and $Y$ such that $P(X) \neq 0$ and $P(Y) \neq 0$ Bayes Theorem states:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$  \hspace{1cm} (1.4)

Probability Distribution  A set of probabilities for a given random variable, often referred to as $P$ and referenced with respect to a graph, $G$ is a probability distribution. It is the set of probabilities for a variable corresponds to that variable’s set of values. The complete set of probability distributions for all random variables in a graph is referred to as the probability distribution of the graph. A graph $G$ with probability distribution $P$ is represented as $(G,P)$.

Joint Probability Distribution  When each random variable in a graph has a specific probability assigned from its probability distribution the set of probabilities is referred to as the Joint Probability Distribution. A Joint Probability Distribution is referenced with respect to a specific graph.

1.1.3 Bayesian Belief Network Terminology

Markov Condition  Given a DAG $G(V,E)$ and its probability distribution $P$, $(G,P)$ satisfies the Markov condition if for each variable $X \in V$ and the set $X$ is conditionally independent of the set of all its non-descendents given the set of all its parents. In other words a DAG meets the Markov condition if for each variable $X \in V$ $X$ is conditionally independent of the set of all other variables minus $X$’s parents.

Bayesian Network  Also known as a Belief Network or a Bayesian Belief Network. Given a graph $G$ and set of probabilities $P$, if $(G,P)$ satisfies the Markov condition, then $(G,P)$ is a Bayesian network. For a graph $G(V,E)$ in $(G,P) V$ represents the set of random variables and the edges in $E$ represent the causal links between variables.
Node  Another term for a random variable in a Bayesian Network.

Belief Node  Same as Node (see above).

Markov Blanket  For a given DAG $G(V, E)$, the Markov blanket for each variable $X \in V$ is define as any set of variables such that $X$ is conditionally independent of them given the Markov blanket.

Conditional Probability Table  The list of all conditional probabilities for a given “node” in a belief network.

1.1.4 Intelligent Tutoring System Terminology

When referring to “student model” or “instructor model”, it should be mentioned that it is not actually possible to model them, but more correctly it is a profile. For the sake of conformity in the field the term model will be used throughout the text.

Student Model  A student model is generally a profile of information about the student. It can range from observable information to unobservable information. Observable information includes answers to questions, performance in different categories, etc. Unobservable information includes (1) possible emotional information and (2) learning paths, or chronological history, taken through material.

Instructor Model  The style in which material is presented to students. This can determine the structure and flow of the material. This model also determines how hints and suggestions are used, presented, and affects a student’s progress or performance on material. Another term for this model is “instructional model.”

Material Model  The content that is presented to the students. This contains both lecture and quiz materials. This model can also determine the structure and flow of material.
1.1.5 ITEAMS Specific Terminology

Session A specific instance of ITEAMS. When a student or instructor runs ITEAMS, it is referred to as a session.

1.2 Background

Current educational software attempts to provide an instructor with a means of improving the learning experience for students. In Teaching Environments and Intelligent Tutoring Systems, this is accomplished through different means.

1.2.1 Teaching Environments

Teaching Environments allow instructors to present the material they choose or design for students in a mostly rigid format. They also provide features which include allowing students to (1) access class materials online, transfer files and (2) interact with their instructor and other students. Some provide discussion boards while others also provide chat capabilities. Very little automatic grading or assessment is available. When it is available, the level at which it is present is rudimentary.

1.2.2 Intelligent Tutoring Systems

Generally, Intelligent Tutoring Systems allow students to focus on specific courses and are specifically written to allow feedback, hints, etc., to students with respect to that subject (e.g. tutoring). An Intelligent Tutoring System provides an independent tool for students to either improve in a subject in which they are struggling or receive extra help in a subject through applications written specifically for the subject [47].

1.2.3 ITEAMS Design Goals

ITEAMS has been designed to incorporate many of the features of both Teaching Environments and Intelligent Tutoring Systems. There are several unique features that make
ITEAMS powerful yet easy to use. These features include

1. an open instructor model
2. multiple assessment levels of a student’s performance and progress
3. unlimited application integration
4. a ubiquitous learning environment

The open instructional model does not limit instructors in the design of courses. Through the use of Teaching Modules, ITEAMS affords the instructor an ability to structure a course in any way desired, ranging from a completely sequential approach to an unstructured approach where the student chooses the next path to take in the lesson. Not only is the instructor not restricted to a format in which to organize a teaching module but multiple levels of difficult can be offered for any single lecture or presentation. Currently there are four levels specified: overview, intermediate, proficient, and complete. The depth and complexity of the material should increase when moving from “overview” towards “complete.”

ITEAMS allows students to move through a teaching module at their own pace and skill level. This is achieved through the use of many different belief networks which evaluate the students as they work through a teaching module. Each belief network is used to evaluate a different aspect of a student’s current session. By using multiple networks, a fine-grained evaluation and assessment of a student is attained. Benefit also include an ability to suggest to the students what they should work on next. This provides a unique flexibility in how a student is evaluated.

This flexibility results from ITEAMS’ unique ability for unlimited application integration through the use of the ITEAMS Plugin API and Specification. Some teaching environments, such as WebCT, eCollege, and BlackBoard, allow integration with other software, but programming language, platform, or location restrictions may severely limit the desired integration. ITEAMS’ openness to a multiplicity of software, with no mandate that it be present on the machine, allows the professor to integrate any piece of software
for the purpose of evaluating a student’s performance either with that software or with information that comes from that software. This openness allows open source software to easily integrate into ITEAMS by any individual or organization.

Since ITEAMS is intended to be flexible, another unique feature is that it acts as a ubiquitous learning environment. ITEAMS can be used either online or offline. ITEAMS is not limited to in-classroom usage. It can be used also in distance learning or even while traveling. When used online, students and instructors access the content of teaching modules through a database which can be located anywhere. Thus, students at Notre Dame, for instance, can work on a teaching module in a computer cluster, save their current work, then move to their dorm room to continue working at their computer connected to the network, and do this right at the place in ITEAMS where they left off. Likewise, instructors can work on material for their course from their office or at home through an Internet connection. Moreover, if an instructor or student does not always have access to an Internet connection, ITEAMS provides the ability to securely work offline through the use of ITEAMS specific data files.

1.3 Thesis Overview

This thesis presents the design of ITEAMS, which stands for “Intelligent Teaching Environment and Assessment Modules for Self-Study.” It identifies the features that are in Teaching Environments and Intelligent Tutoring Systems as well as identifies what is lacking in those systems. In general the content of the thesis falls into two categories:

1. A survey of existing systems and features
2. A description of the ITEAMS’ system features and technologies

The work on this thesis started with a neural network simulator called NNSim written by Dr. Scheutz. This simulator provided the initial graphical components for graphically editing a teaching module structure and easy extendibility for the rest of the application.
to be built. Besides the author and Dr. Scheutz, work on this project came also from Dr. Danny Chen and David Landeck. Dr. Chen presented the author with the solution to the problem of determining when a network contains multiple paths. David Landeck worked on the database storage facility of ITEAMS.

The thesis is organized as follows: CHAPTER 2 focuses on Teaching Environments. It discusses what they are, how they are used in the educational process, and what features they lack. CHAPTER 3 focuses on Intelligent Tutoring Systems. Similar to CHAPTER 2, it discusses what Intelligent Tutoring Systems are, how they are used, and what they lack. CHAPTER 4 provides an overview of ITEAMS. It specifically introduces features which are discussed in depth in later chapters. The first feature is the plugin interface which is presented in CHAPTER 5. The educational evaluation scheme used in ITEAMS is discussed in CHAPTER 6. The thesis concludes with closing remarks.
A Teaching Environment can be described as any piece of software for use by instructors for organizing courses. More specifically, in Teaching Environments instructors present the material they have chosen or designed for students in a mostly rigid format. Very little automatic grading or assessment is available. When automatic grading is available, the level at which it is present is rudimentary, such as automatically grading multiple choice questions where one is marked as the right answer [45].

A typical Teaching Environment normally demonstrates, but is not limited to, some of the following:

- **Instructors can**
  - Provide lecture material
  - Create assignments
  - Create quizzes
  - Manage and track students

- **Students can**
  - View Lectures or Presentations
  - Receive/Submit homework assignments
  - Take quizzes
  - Receive/Submit projects

Some of popular teaching environments that demonstrate features include Angel, Blackboard, eCollege, and WebCT.
2.1 Feedback

The feedback given in a typical Teaching Environment is sparse. If any feedback is incorporated, it may be no more than such things as e-mail notification that students have turned in an assignment or finished a quiz. This can extend to notifying the instructor or a designated individual that an assignment, quiz, or test is being locked/unlocked for a class to begin taking or finishing taking. We are not aware of any Teaching Environment that provided feedback that would allow the student to discover why certain responses or perceptions are in fact incorrect.

2.2 Student Modeling

Very little student modeling is incorporated into Teaching Environments. If there is a student model, the only information that is tracked is answers to questions or exercises that the student gave.

2.3 Instructor Modeling

One can view Teaching Environments as possessing one of two different instructor models:

1. Feature Based Model
2. Unit Based Model

A Feature Based Model is a model that groups course material by features such as quizzes, lectures, etc. Students get new content by going to the feature that contains the new content. A Unit Based Model groups course material by functional units such that a unit can contain lectures, quizzes, etc. A student can then get new content by viewing new units. This model provides students new to a system an easy way to begin their learning experience. [14]
2.4 Teaching Environments in the classroom

Many institutions use Teaching Environment in the classroom. In a study by Gartner Research [17], 73 percent of college campuses use a Teaching Environment. The four most widely used systems were Angel, Blackboard, eCollege, and WebCT. The University of Notre Dame uses WebCT [45] for many of its classes. At Notre Dame instructors have the ability to perform many tasks including:

- post file
- create quizzes
- post grades
- manage students

The functions provided in WebCT appear to be similar to that which is provided by other systems. Many universities use Teaching Environments to supplement their courses or to provide a localized web-based repository of course resources. It is up to the institution to decide how a Teaching Environment should be used. Some systems even provide enough flexibility that they can be used as a course supplement for courses held at a learning institution or for a distance learning course.

2.5 Previous Work in Teaching Environments

In the area of Teaching Environments, there are many different pieces of software available. There are several systems in particular that will be discussed in this text for comparison. These include WebCT, BlackBoard, and eCollege. Each of these provide some similar features but each does so in a slightly different manner. Some of the similar features include:

- Discussion Board
- Chat
- E-mail options
2.5.1 WebCT

WebCT is a Teaching Environment which gives an instructor several options in organizing a course. As stated above, WebCT allows instructors to:

- post file
- create quizzes
- post grades
- manage students

The instructor model used is a feature based approach. It allows instructors to organize a course into features such as Quizzes, Homework, Presentations, etc. This can facilitate students’ access to material through functional groupings of course material.

WebCT allows instructors to work online if they want to see the changes they are making to the content. It also allows instructors to work on material offline but they must then upload the material after they have finished. They are not able to work on course material while not logged into the system. When later logged into the system, it will not automatically upload the material for them.

2.5.2 BlackBoard

BlackBoard is a system that is designed for flexibility and usability:

- “Curriculum-driven content management and content sharing provides instructors with flexibility and control
- Re-architected assessment management system is designed to improve assessment creation workflow and provide flexibility in deploying tests and surveys
- Designed based on client feedback and usability testing, the Gradebook improves instructor productivity
- New functionality allows instructors to electronically manage the collection and organization of assignments via the integrated Gradebook interface” [3]

The instructional model appears to be a unit based model. It allows for instructor specific naming of course content areas. Along with this, it allows course content units to be copied and dropped into other courses.
2.5.3 eCollege

eCollege is a system that is designed for:

- “Flexibility
- Usability
- Supporting the way you teach
- Supporting the way your students learn” [14]

It uses a unit-based approach which breaks the course material into functional groups that an instructor defines. These groups can include quizzes, assignments, lectures, readings, and more. This is “different from other systems which are feature-based. [14]” eCollege claims that other systems essentially break up courses by features such as readings, quizzes, etc. and organize the course that way with little or no flow. Not only does eCollege provide grouping of course content but it also provides the grouping of students for a class. This creates individual areas for group members to collaborate with each other.

One feature offered by eCollege that is very helpful from the instructor’s point of view is that both student and instructor have the same view of the material. If an instructor wants to add or edit the content, only one button needs to be clicked to access the editing screen. Once the editing is complete, the instructor can switch back to the student view by clicking the same button again. Along with the course content editing, eCollege also allows flexibility to change the look and feel of the software through button layouts, fonts, and colors.

The system requires that instructors work online if they want to see the changes they are making to the content. eCollege does have support for Microsoft Office documents. Instructors can work on material in any Microsoft Office format offline but they must then upload the material after they have finished and the system will convert the document to HTML. They are not able to work on course material while not logged into the system and then have the system automatically upload the material for them. If instructors are at
home with no Internet access, they must bring all file with them back to the office for uploading.

2.6 Evaluation of current systems

The systems that have been discussed above provide many features which are useful for both instructors and students. These have been designed for ease of course management while attempting to provide an environment that both experts and novices can use with good benefit. However, even meeting these design requirements, we feel that these systems come up short of the complete educational system. Many professors are capable of guiding and evaluating students in ways beyond what these current systems offer. Let’s first consider instructor models. Then, more importantly, we can look at evaluation schemes.

2.6.1 Instructor Models

The addition of one or more instructor models to Teaching Environments would give teachers a greater ability to affect how a student can learn. This is accomplished by being able to present material in a way that better suits the instructor’s style to which a student could have become accustomed.

Some systems such as eCollege and BlackBoard allow instructors to create course material in functional units but allow only a linear ordering of the material. Moreover, students cannot be forced to follow the linear ordering presented. They can move around in any order they wish. This is a potential problem when an instructor designs a complete course and releases all of the content at one time. Much of learning is like building a structure. The foundation goes in first, then support for the walls and the roof, and so on. The proper sequence must be followed in order to have the desired end result. If students are free to jump around without regard for the laid-out progression of information, it is
highly possible that the students will not really learn the material in the truest sense of the word. The scattershot approach to information may produce an abundance of information, but the pieces may not be integrated and thus not truly understood.

2.6.2 Evaluation Schemes

Current systems provide very little automatic evaluation and grading. What is provided is some level of automatic grading and no evaluation of a student’s skill. While automatic grading is helpful, tracking a student’s work and evaluating the weakness or strength of the individual would be very beneficial. This task is left up to the instructor and may not always happen. This impacts how effectively a student learns.

2.7 Chapter Summary

This chapter has presented the basic features in a Teaching Environment as well as providing a discussion about some current systems in use by educational institutions. Instructors use Teaching Environments as tools to present the material they have chosen or designed for students in a mostly rigid format. They are limited to “easily” gradable questions (e.g. fill in the blank if the answer is simple, mix and match, multiple choice, etc.). No student assessment is present in Teaching Environments which specifies where students are lacking in their studies. These systems also do not allow instructors to organize their material in more than a linear form. Some of the features in Teaching Environments include:

- Instructors can
  - Create quizzes
  - Create assignments
  - Present lecture material
  - Manage and track students

- Students can
  - View Lectures or Presentations
  - Take quizzes
  - Receive/Submit homework assignments
  - Receive/Submit projects
CHAPTER 3

WHAT IS AN INTELLIGENT TUTORING SYSTEM

An Intelligent Tutoring System, also referred to as an ITS, does not have a simple definition that correctly and fully encompasses what it really means. An overly simple definition might state simply: *a piece of software that helps students learn through the use of tools and techniques in artificial intelligence.* A more adequate definition would have to include a breakdown of at least the different types of ITS’s. What follows is an initial attempt at such clarification.

An ITS can be listed under a number of different categories and is not restricted to any one of them. The following are some categories if ITS’s.

**Single Subject Application** An ITS that is designed specifically for one subject matter.

The subject matter can be either very specific like CIRCSIM-Tutor [47] which tutors medical students on the negative feedback system that controls blood pressure or a more general subject matter tutor like ANDES [7, 18], which helps students learn general Newtonian physics, or ACE which is a general math tutoring system [4].

**Multiple Subject Application** An ITS that can provide tutoring and instructional abilities for more than one specific subject matter [34].

**Discussion Providing** An ITS that can provide a coaching dialog. It is able to interact
with the student to provide information that is not in the form of a “hint.” (See the next bullet.)

**Hint Providing** An ITS that can provide a student with specific help on a problem either when the student asks for help or the ITS determines that the student is struggling.

**Explicit Student Model** An ITS that define a student model, more so than the instructional material.

**Explicit Instructor Model** An ITS that define a model defining the instructor, and could include multiple models because different styles/needs/objectives of instructors.

**Explicit Assessment Model** An ITS that define an explicit form of evaluating the performance of a student.

**Explicit Material Model** An ITS that define what the material is and how it relates to other material.

Two of the above are mutually exclusive. An ITS cannot be both ‘Single Subject’ and ‘Multiple Subject’ When discussing specific ITS’s, a table (such as Table 3.1) will be shown to help determine some of the features of a given system.

### 3.1 Uses for Intelligent Tutoring Systems

The application of ITS’s to courses can be varied. An instructor can use them as the main source of information, resources, and assignments, all the way to using them as a tool for boosting individual students’ abilities to match the rest of a class. Depending on the intentions of the instructor, different types of ITS’s can be used. Table 3.1 shows the minimum categories that an ITS should contain to be useful to an instructor who intends it to be the main educational device for instructing an entire class. Depending on the exact needs of an instructor, other categories such as ‘Explicit Instructor Model’ or
TABLE 3.1

THE LIST OF CATEGORIES IN WHICH AN ITS CAN BE PLACED. WHEN AN ITS CAN BE PLACED IN A CATEGORY AN X WILL APPEAR IN THE SECOND COLUMN.

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Subject</td>
</tr>
<tr>
<td>Multiple Subject</td>
</tr>
<tr>
<td>Suggestion Providing</td>
</tr>
<tr>
<td>Hint Providing</td>
</tr>
<tr>
<td>Explicit Student Model</td>
</tr>
<tr>
<td>Explicit Instructor Model</td>
</tr>
<tr>
<td>Explicit Assessment Model</td>
</tr>
<tr>
<td>Explicit Material Model</td>
</tr>
</tbody>
</table>

‘Explicit Material Model’ could be beneficial. If an entire course were designed around a single ITS, then an instructor would likely want an ITS that can handle more than one subject, since a course requires specific knowledge from other subjects. As an example, a course in robotics could require either previous knowledge of Artificial Intelligence or provide the student with material that introduces the appropriate AI-related background information.

Such a model could also be designed with elements that could assist students who may be struggling to master the core content of the material. Though not necessarily, the need for this may be found in models designed for elementary students more so than in those designed for higher-end students. The design may allow the instructor to recommend use of the ITS as a tutoring tool outside of class, to facilitate “catching up” with the rest of the class, thus allowing this student to be introduced to the newer materials in class on a timely basis. This further demonstrates the high degree of flexibility in constructing an ITS, as well as it’s benefits as a teaching tool.
TABLE 3.2

ANY ITS THAT MEETS THE MAJORITY OF THESE REQUIREMENTS WOULD BE ACCEPTABLE FOR USE IN A WHOLE CLASS.

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Subject</td>
</tr>
<tr>
<td>Multiple Subject</td>
</tr>
<tr>
<td>Suggestion Providing</td>
</tr>
<tr>
<td>Hint Providing</td>
</tr>
<tr>
<td>Explicit Student Model</td>
</tr>
<tr>
<td>Explicit Instructor Model</td>
</tr>
<tr>
<td>Explicit Assessment Model</td>
</tr>
<tr>
<td>Explicit Material Model</td>
</tr>
</tbody>
</table>

3.2 Previous Work in the Field

For several decades now researchers have been working on ITS’s and focusing on different aspects of them. The work can be broken into several groups. These groups include

- Evaluation Models
- Feedback of students’ progress and performance, hints, suggestion, etc.
- Student Models
- Lesson Adaptation
- Instructor Modeling

We will briefly outline the following four systems and offer some evaluation of each.

- POLA
- ANDES
- ACE
- CILE

3.2.1 Evaluation Models

Every tutoring system needs some means to evaluate students’ performance. Evaluation mechanisms range from neural networks [26] to Bayesian networks [4]. Others have
implemented a more straight-forward grading approach such as the Marker system [35].

Each of the models has its own strengths. The grade-based evaluation, for example, is simple to use and fast. The neural network evaluation uses an approach that can be used to evaluate students in a manner similar to that of evaluating by an individual professor. A belief network approach allows for more than a simple assessment of a student’s performance. It can determine deficiencies, generate specific “hints,” and help “decide” how a given student should move ahead through subsequent materials.

Besides being able to evaluate a student, belief networks provide the capability to be queried for deficiencies. This allows for the generation of suggestions and hints that can benefit the student. Some research has been done into the generation of hints and suggestions and how it should affect the evaluation of a student [18, 47]. The feedback mechanisms of these systems and others are discussed in the following section. We will first look at the evaluation mechanisms.

Numerical Evaluation

The word numerical says it clearly. This form of evaluation is very straightforward. Every answer has a specific value. When a professor or another part of the system needs to determine the performance of the student, the process is simple: divide the total value by the number of answers given. This form of evaluation and performance assessment is quite simple. This form can be used when nothing more than a grade-based approach is needed.

The principal disadvantage to this style lies is the small amount of helpful information that it offers for a true evaluation. The strictly numerical basis gives us no information as to why an answer was given. Maybe just a lucky guess? A good evaluation requires much more in order to have insight into what a student really understands. The numerical evaluation also offers no basis for the system to generate “hints” and “suggestions.”
Neural Networks

Neural Networks have been used to evaluate the student and plan the content that will be presented to the student. Neural networks have used patterns of relationships between material to implement strategies for content presentation. [26]. The patterns that are used are based on material the student has already studied. The neural networks use the learner’s knowledge to create the content plan but do not provide any insight into why the path was formed. Neural networks work well for the generation of content presentation but lack the functionality to determine where deficiencies are present. They are able to determine for the student what should be worked on next and present this to the student.

Belief Networks

Many different researchers use belief networks for evaluation. The reasons for this include: the causal relationships inherent in the structure and the network’s ability to determine the most probable explanation in a situation. Belief networks have been used to generate hints [18] and to generate content flow and suggestions [7, 24, 36]. Researchers have also looked into using belief networks for alerting users based on the expected utility of the notification [21].

3.2.2 Feedback

Researchers, using mostly evaluation modules, have examined the question of how feedback can be used effectively to enhance not only the student’s knowledge, but also the quality of the student’s learning process. We look at two questions raised by these researchers.

“How does the introduction of a hint or suggestion affect the evaluation of a student?”

Two schools of thought have emerged. The first school believes that a hint or suggestion affects only the value of the information being presented to the student and does not di-
rectly affect the student. This school assumes that it is important only that the student learned the information, not how the student learned. Thus, this group holds that how students learn is less important than the fact they learned [1]. The other school holds that the introduction of hints and suggestions does not affect the value of the information, but does negatively impact the student’s learning process. This group assumes that students given such help are not really learning on their own and might not be expected to produce similar results later when put in a similar situation [18]. Thus, the assumption with this school is that how students learn is no less important than the fact that they are learning.

“How does one introduce hints or suggestions in a manner that is genuinely effective?” Some researchers attempt to resolve this by introducing hidden variables into a belief network for each belief node, then performing queries on the different hidden variables [28]. Conati [7,8] has addressed this question in two different systems through the use of belief networks to achieve the functional effectiveness of suggestions and hints. Conati’s POLA system uses the diagnostic capabilities of belief networks to generate hints but does not use solution steps [8, 9]. Conati’s later system, ANDES, which uses an advanced version of the hint system in POLA, uses specially designed “solution graphs” to generate hints and suggestions. The “solution graphs” are networks that map out all of the different paths through a solution space to solve a problem [7, 18]. ‘Solution graphs’ are created from a graphical description of a problem along with a coded problem definitio to generate a model of a problem solution space. ANDES also includes expert knowledge of abstract plans for solving physics problems to generate different solution paths in a graph for a problem. Through the combination of the expert knowledge and “knowledge about the qualitative and quantitative physics rules necessary to solve complex physics problems [18]” ANDES can generate all the possible paths through the solution spaces.

Patel uses a combination of several belief networks in the CILE system [36] to pro-
vide feedback to students. The feedback is generated dynamically based on the student’s answer history. This history is used to generate hints that help the student decide what to do next, as well as help the student avoid making the same mistakes over and over again. The feedback is based on an “expert model,” which is a correct path through the solution space. If a student provides a correct answer, the system assumes that the student has knowledge of the necessary concepts. When an incorrect response is given by a student, the system walks the student through the “expert model’s” solution path. There is no checking of solution steps when the correct answer is given, allowing students to use different steps than the “expert model.”

3.2.3 Student Model

Within the field of Intelligent Tutoring Systems there has been much research in the area of student modeling. Some researchers use the student model so the history of the student is known and can be taken into account during assessment [47], while others design the student model around the assessment of the student [7]. Most work in student modeling has focused on the history of a student, but some has also looked into the history of the student in conjunction with environmental data [13]. Some studies [47] in this area like to go well beyond computer-related or specifically intelligence- or artificial-intelligence related information, such as the psychological and physical status of the student at the time of the work. But, this is not germane to this thesis.

In their research of what a student model is supposed to be, how to define it, and what role it should play in an ITS, researchers do arrive at differing conclusions. Overall, they all [8, 28, 34, 47] have started with the history of a student as the basis for their model. Some of the differences should be apparent from what follows.

Regarding CILE, [35, 36], Patel’s system includes student answers, student preferences that impact the learning process, and also how specific questions were answered.
Did the student actually answer the question and answer it correctly? Did the student leave the question blank? Did the student answer the question but answer it incorrectly? Each type of response impacts the student evaluation differently.

Zhou and Evens [47] pose another valuable question: “What aspects of the student should we model in a specific intelligent tutoring system?” Note that the CIRCISM-Tutor model is designed to tutor medical students on specific aspects of blood pressure. Thus, the design of this model is influenced by the specific objective of the CIRCISM-Tutor. The design of the system is mostly to instruct or coach the student through use of an input table with multiple inputs to assist in defining the student’s answers, but also to allow free text input. This system uses a combination of multiple table inputs and free text input to provide a simple form of free text output as a form of feedback and interaction. The system is also designed to present a causal learning environment for the student to develop qualitative reasoning. The software can adjust the level of material presented by looking at the overall knowledge of the student in a given area and their performance in general. The student’s performance also guides how the software assists the student by influencing the level of material. Another aspect of the system is that it can work with the student on several instructional goals concurrently, switching between them based on student responses.

Zhou and Evens’ student model consists of four parts: (1) the student’s performance, (2) the student’s replies to all questions, (3) the student’s errors, and (4) the tutor’s dialog. The evaluation of how students are performing is based on an overall measure of their abilities, a measuring of their knowledge regarding the current procedure, a measuring at each stage in the process, and a measure for each specific idea covered. The student’s reply history is stored for each concept presented and contains a classification for each answer the student gave. The classification is used to help determine feedback. The student’s errors are stored as a count as well as descriptions of the errors. The tutor’s
dialog is just that - a dialog of the text presented to the student.

The software must be consistent in how it attempts to assist the students. To this end, evaluation of current performance must be made under the umbrella of the software’s tracking of a student’s overall performance. Thus, several elements or steps feed into the tutoring decisions. At the “higher” end is an accounting of a given student’s pattern of errors and the software’s resultant history of tutoring this student. At the “lower” or immediate end is the actual dialogue developed on the basis of the student’s current performance and questions responses, but developed under the umbrella of the broader pattern of tutoring for this student.

Conati has worked with several different “student model” designs in different tutoring systems. The first student model, designed for the POLA system [8], used AND/OR networks [20,37] to represent the student model. This model consisted of (1) rules of physics used to solve a problem and (2) probabilistic values of correctness for each physics rule when applied to a specific problem. The nodes in the AND/OR network are all boolean; either they exist or they do not exist, and there are two types of nodes: action nodes and fact nodes. The action node represents what action the student took. The fact node represents a fact that the student should use or learn. The AND/OR graphs are transformed into belief networks in this system. The POLA system is used to help students learn physics and requires them to develop solution steps when working on a problem. When a student does develop a viable step in a solution, the action node for that step is updated with a value of 1 indicating that it was used by the student. This new value is then propagated through the belief network where fact nodes are updated. These fact nodes represent what the student may or may not have learned in the process of solving the problem. Conati extended this student model in the ANDES system [7] which also tutors students in physics. This student model uses explicit knowledge about how an expert would solve a problem as well as plan recognition to determine in which areas the student is deficient.
and also what hints would be most relevant to the student. ANDES determines what the student’s plan for solving a problem is to provide more relevant hints. This information in conjunction with expert knowledge is used to determine where the student is deficient.

Mislevy has created a student model that is based on the history of a student, but
the history comes in a different form - unobservable information [28]. The unobservable information relates the student’s history to information seen from the combination of other models in the system. This information characterizes the “aspects of knowledge and skill that are the targets of inference. [28]” The ‘student model’ stores this information as ‘variables,’ which when combined with the ‘variables’ in the other ‘models’ create the belief networks that evaluate the performance of students.

In Nkambou [34] the student model is broken into three components: cognitive, behavioral, and inference engine. This system employs beliefs of the student and of the system, as well as inferred concepts. As the student begins working on a course, the student model itself evolves throughout the learning process. This student model is a "global model.” The system uses the student model to mimic the learning process of the student. As the student learns new information, the student model ‘learns’ new information as well. In [34], Nkambou introduces a material modeling approach called CREAM which is used to update and influence the evolving student model. CREAM consists of four types of networks: domain knowledge, instructional objectives, learning material, and the curriculum knowledge transition network (CKTN). The student model is based on the current capability state, objective state, and resource state. The reliability of the student responses, or the system’s belief that student understands the material and is not guessing, is measured based on two functions: one used to increase reliability and one to decrease the reliability. If the system ‘believes’ that the student is knowledgeable, the reliability increases. The system decreases the reliability if it does not ‘believe’ the student is knowledgeable.
3.2.4 Instructor Model

Most recent research has focused on how to model the student and (not as much) on how to present the material to the student. Patel [24] pointed out that current ITS’s do not model the teacher and emphasized that there needs to be an increased focus on the teacher model in ITS’s.

Conati [6] as well as Self [39] have brought to light the need to pay more attention to the presentation of material and teaching style. Existing systems have attempted to handle multiple styles of teaching but have typically been restricted to a limited set of styles. While Self has previously worked on the theoretical side of intelligent tutoring systems, he prepared an implementation using a knowledge base to model the instructor and student. He accomplishes this through “Computational Mathematics.” For a fuller discussion of “Computational Mathematics” please refer to [40].

Conati creates an instructor model that is a copy of the student’s model including the student’s emotional state and bodily expressions. This is done to provide the tutoring agent with an expectation of what the student might do next. The tutoring agent can then offer hints and suggestions that the model expects to be more relevant than if the model used only factual material history.

Mislevy [28] has proposed a multiple model system to create a model of the instructor. The multiple models that are used to create the instructor model are:

- Task Model
- Evidence Model
- Assembly Model

The information in these models is brought together to create an instructor model. The Assembly model describes how other models are to be incorporated with each other. The Task Model is similar to a material model in that it describes what is presented to a student but it also define how the material can be presented to the student. The Evidence model
determines how the task model’s observable information relates to information in the student model.

3.2.5 Lesson Adaptation

Magoulas et al. [26] present a method for adapting a lesson based on the learner’s knowledge. The material presented in the lesson is generated from pieces of education material by accounting for the nature of the content being taught and the knowledge level of the learner. This is accomplished through a connectionist network that evaluates the learner on past concepts in the lesson. Magoulas et al. is greatly concerned with how the material is organized and presented. The greatest challenge is “... to build an environment in which the learners are motivated ...”. Magoulas et al’s possible solution is adaptive lesson presentation.

To accommodate adaptive lesson presentation, an adaptive knowledge structure is needed. The authors purpose is to make the structure adaptive by breaking the knowledge into categories and placing the categorized knowledge in a connectionist network through a modeling process. Knowledge is broken into three categories: outcome concepts, related concepts, and prerequisite concepts.

The knowledge model is a three-layer structure that consists of goals, concepts, and material. Each goal is associated with concepts in the level below, and each concept is associated with material in the third and final level. The associations between goals, concepts, and material form what is called “concept relation patterns.” The concept relation patterns form different strategies for content generation. The educational material related to concepts is structured into divisions such as text, images, exercises, etc.

Planning the instructional material is a two-part process consisting of (1) planning the content and then (2) planning the delivery. Simply put, this is creating and sequencing the goals and material that make up the educational process.
The evaluation of students comes from answers and measurements. Questions are arranged in three categories: factual generalization, functional interaction, and material comprehension. The measurements used include items such as the number of questions attempted, the number of times the questions were attempted, etc. The actual evaluation of the student occurs in a back-propagation network that evaluates the knowledge level of the student. The network is trained using a variable step size back-propagation algorithm. The network places the student in one of six different categories that range from extremely inefficient to sufficient.

3.3 Summary of Intelligent Tutoring Systems and current systems

The current systems discussed above do indeed make available to instructors various tutoring tools that are certainly powerful and which do provide excellent evaluation of students and their learning deficiencies. These systems tutor students in specific subjects like Mathematics [6], Physics [7], Heart Disease [47], and Marginal Costing [35]. Using evaluation criteria from earlier in this chapter, we proceed now to a critique of each of these systems.

3.3.1 POLA

POLA’s summary is shown in table 3.3.1. POLA is an early version of some of Conati’s later systems. It is a system that provides many useful features but lacks an instructor model and the multiple subject capability. The lack of a material model results from the fact that there is only one subject for which POLA is used.

3.3.2 Andes

ANDES’ summary is shown in table 3.3.2. Andes is an enhanced version of the POLA system, which, as stated, offers only physics tutoring. Thus, it also lacks the material
TABLE 3.3

SUMMARY OF THE POLA ITS

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Subject</td>
</tr>
<tr>
<td>Multiple Subject</td>
</tr>
<tr>
<td>Suggestion Providing</td>
</tr>
<tr>
<td>Hint Providing</td>
</tr>
<tr>
<td>Explicit Student Model</td>
</tr>
<tr>
<td>Explicit Instructor Model</td>
</tr>
<tr>
<td>Explicit Assessment Model</td>
</tr>
<tr>
<td>Explicit Material Model</td>
</tr>
</tbody>
</table>

TABLE 3.4

SUMMARY OF THE ANDES ITS

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Subject</td>
</tr>
<tr>
<td>Multiple Subject</td>
</tr>
<tr>
<td>Suggestion Providing</td>
</tr>
<tr>
<td>Hint Providing</td>
</tr>
<tr>
<td>Explicit Student Model</td>
</tr>
<tr>
<td>Explicit Instructor Model</td>
</tr>
<tr>
<td>Explicit Assessment Model</td>
</tr>
<tr>
<td>Explicit Material Model</td>
</tr>
</tbody>
</table>

model but does offer additional information in the student model. This helps in the assessment of students.

3.3.3 ACE

ACE’s, an Open Learning Environment, summary is shown in table 3.3.3. It allows the students to work at their own pace and thus move around in the learning units at their
TABLE 3.5

SUMMARY OF THE ACE ITS

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Subject ×</td>
</tr>
<tr>
<td>Multiple Subject</td>
</tr>
<tr>
<td>Suggestion Providing ×</td>
</tr>
<tr>
<td>Hint Providing</td>
</tr>
<tr>
<td>Explicit Student Model ×</td>
</tr>
<tr>
<td>Explicit Instructor Model</td>
</tr>
<tr>
<td>Explicit Assessment Model ×</td>
</tr>
<tr>
<td>Explicit Material Model</td>
</tr>
</tbody>
</table>

own choosing. There are no restrictions placed on the student since there is little or no explicit instruction which allows a student to explore a subject in an unguided process. The assumption is that the student can build skills associated with exploration as well as deep understanding of the subject. ACE lacks real-time suggestions tailored to the individual needs of a student.

3.3.4 CILE

CILE’s summary is shown in table 3.3.4. The CILE ITS has several interesting features including:

- **random question generation** that randomly generates questions and provides correct solutions for the questions.
- **dynamic feedback system** that generates feedback messages based on work the student has already done and what the student should do to continue.
- **knowledge base** that contains both conceptual rules and processing information.

The principal drawback of this system is its single-subject design.
TABLE 3.6

SUMMARY OF THE CILE ITS

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Subject</td>
</tr>
<tr>
<td>Multiple Subject</td>
</tr>
<tr>
<td>Suggestion Providing</td>
</tr>
<tr>
<td>Hint Providing</td>
</tr>
<tr>
<td>Explicit Student Model</td>
</tr>
<tr>
<td>Explicit Instructor Model</td>
</tr>
<tr>
<td>Explicit Assessment Model</td>
</tr>
<tr>
<td>Explicit Material Model</td>
</tr>
</tbody>
</table>

Topic Specific

In the systems reviewed and discussed, none offered the capability of handling multiple subjects. Some theoretical work has been done to move in this direction, but, in general, these systems leave it to Teaching Environments to allow for multiple subjects. Intelligent Tutoring Systems simply lack this dimension. This is the case from the beginning of research on them.

3.3.5 Instructor Models

Despite the advanced research done in the area of intelligent tutoring systems, this research still lacks adequate focus on the instructor model. This may result from the lack of systems that handle multiple subjects. Since multiple subjects are not at issue in the research, there is little need for multiple teaching styles.

3.3.6 Static Material

Beyond the lack of multiple subjects and instructor models, the material being tutored in this kind of model does not change. Thus, it remains static. Only a few of the systems
described above work in this fashion.

3.3.7 External Software Interface

For the most part, Intelligent Tutoring Systems are self-contained. They do not allow for accessing external software that might help in the learning process. A system such as ACE, for example, might enhance the learning process by allowing access to a program such as MATLAB. Since ACE is used for tutoring on simple mathematics, it could be extended to work with more advanced mathematics. With an interface to MATLAB, ACE could open to students the possibility of evaluating complex mathematics equations.

3.4 Overall Evaluation

Current intelligent tutoring systems include several good features:

- Advanced Feedback models (both static and dynamic)
- Complex evaluation models
- Efficient and informative student models

In addition to these features, current tutoring systems provide highly specialized topic specific learning environments in which students can work. However, they lack several features, including the following:

- Non-static Material
- Multiple Subjects
- Multiple Instructor models
- External Software Interface

The next chapter presents the design of ITEAMS and how ITEAMS solves these limitations of Intelligent Tutoring Systems.
CHAPTER 4

ITEAMS ARCHITECTURE

After a brief analysis of the strengths and weaknesses of existing Teaching Environments and Intelligent Tutoring Systems, the focus in this chapter will be on how important elements of these systems are integrated into ITEAMS and developed further in the interest of both instructors and students. This is the principal goal of ITEAMS. This chapter will examine the structures for organizing lecture materials. There will be attention given also to the tracking of grades, as well as to the tracking and assessment of students’ performance. We consider the tracking and storing of information from the former systems. We look also at the possibilities for dynamic selection of lecture materials based on a student’s inferred knowledge level. Finally, this chapter examines automatic grading of assignments.

4.1 Overview

As already discussed, Teaching Environments do allow instructors to create presentations, lecture notes and quizzes. But they lack certain valuable features, such as an openness to tailoring instructional styles and lectures to the varied needs of individual students. Further, there is no automatic assessment of student progress. Moving a step further, Intelligent Tutoring Systems do include many of these features, but for the most part they are application- or field-specific [47]. ITEAMS attempts to incorporate features that are lacking in Teaching Environments. Note however that special care has been given to
keeping ITEAMS as generic as possible, so as to render it applicable to any subject area, while still utilizing valuable functionalities of tutoring systems.

Recent research has focused mostly on how to model the student, and somewhat, but considerably less, on how to present material to the student. It is held that current ITS’s do not sufficiently model the teacher and that there needs to be a stronger focus on the teacher model [24]. This does not diminish the significance of the student model, however. Thus, ITEAMS places a strong emphasis on an explicitly defined student model.

ITEAMS, recognizing the need for more attention to the presentation of material and to teaching styles [6, 39], allows instructors to build their own teaching styles into the modules. The instructors draw upon their own experience as educators and expose students to a learning environment that they may sense as ’familiar”, since it is patterned after the teaching styles of their current instructors. Some existing systems have attempted to handle multiple teaching styles, but typically have been restricted to a limited set of styles. There is support in ITEAMS for structuring the lecture material. It is the openness of ITEAMS that allows for variations in instructor styles. Thus, there is no explicit instructional model in this system.

Every learning environment needs some student evaluation schema. Evaluation schemas range from neural networks [26] to Bayesian networks [4]. ITEAMS uses a combination of Bayesian networks to assess the student’s performance and knowledge level.

Several types of belief networks are used in ITEAMS to evaluate and assess, and to help move the student through the material. These belief networks consist of

- singly connected networks
- multiply connected networks
- structurally dynamic networks

and are discussed fully in CHAPTER 6. Learning situations are by no means always the same. Different circumstances call for a different type of network, such as one that determines what material is accessible to the student. At any given point, information
could be required from at least one of each type of network. One significant goal of ITEAMS is to assist in the learning process in real-time. The time required to perform queries and to update the network based on a student’s actions must be minimal and efficient. Therefore it is most important to consider how and why each type of network is used.

ITEAMS consists of several components and sub-components, designed specifically to effect the objectives presented above. An in-depth discussion of each component and sub-component follows. The Student Model is discussed first, followed by the Instructor Model, Material Model, and Evaluation Model. The Application Environment and Graphical User Interface conclude the chapter.

4.2 ITEAMS Components

ITEAMS consists of two major components: (1) a student model and (2) a model of the domain. The student model contains information about the students that is used in their assessments. The model of the domain contains the material for the teaching module and possible instructional paths (i.e., sequences of presentation) through the material.

ITEAMS also contains several sub-components:

- a teaching module interface
- an external interface
- an intelligent assessment component.

The teaching module interface allows instructors to prepare and structure lecture material for each teaching module and allows the student to work with each teaching module. The external interface, also known as the “plugin interface”, allows ITEAMS to connect to other applications or systems through the use of “plugins” (i.e., purpose-written interfaces that handle the data exchange between ITEAMS and other systems). The ability to allow students to use external systems (e.g., a robot simulator, MATLAB, etc) for assignments and to gather data from that system about the student’s performance is unique.
to ITEAMS. Finally, assessing the student’s overall performance and level of knowledge (to the extent that this can be measured with the methods employed in ITEAMS) is accomplished by causal queries of the various belief networks, which represent knowledge about students gathered while they work on teaching modules.

Throughout the chapter ITEAMS will be evaluated based on the criteria for evaluating Teaching Environments and the criteria for evaluating Intelligent Tutoring Systems. After each section the evaluation of ITEAMS will be updated for the current topic.

4.2.1 Student Model

The student model in ITEAMS is comprised of all of the information that is gathered while a student works on a teaching module, also referred to as a student’s history. Currently the history information includes:

- Quiz question Answers
- Assignment exercise Answers
- Completed teaching modules
- Completed sections
- Completed goals
- Active teaching modules
- Active sections
- Active goals

For each of these items, students’ associated performance is recorded. For example, in a quiz question answer the associated performance is the answer’s correctness value and for a section the associated performance is the students’ evaluated performance from one of the belief networks. The evaluation of performance is discussed in full in CHAPTER 6.

Design Considerations

ITEAMS needs to track the learning profile of students. The learning profile is developed from any information that can be used to create a basis for evaluation and comparison.
The information in the profile can be compared with the student’s prior performance, performance with this particular material, versus other material, etc. In the current implementation, ITEAMS models students similar to [47] focusing on the history of the student to build a learning profile.

To build students’ learning profiles, ITEAMS collects data from what students have worked on, as well as what they are working on. This information for the profile comes in the form of student answers and the learning path. The learning path is a trajectory through the material in which a student works on material in a teaching module. The order in which a student answers questions or views material issued by ITEAMS creates an accurate picture of the student, and can be of use to the instructor, in understanding, for example, how the student arrived at a given point in the teaching module.

Currently, the “answers” given by students are the most relevant part of the student module to ITEAMS. ITEAMS tracks two different types of answers: question answers and exercise solutions. The record of a student’s answer has two components: the actual answer and a quantitative value assigned to the answer. The value is assigned by the instructor and defines a degree of correctness for an answer.

In the case of assignment exercises, one of two possible records is stored. When an exercise which is part of an assignment uses a “plugin,” the student’s record consists of any information the plugin gives and a qualitative assessment of the performance. When the exercise is a question, ITEAMS stores the answer and a quantitative value of the answer.

Since students can review any previously viewed material as often as they want, including goal, question, and assignment content, they could receive the same question or exercise more than once when retaking a quiz or assignment. In this case, ITEAMS modifies the values in the conditional probability tables of the belief network nodes that represent the questions and exercises the student has already answered. It thus reflects the
student’s previous performance. This is accomplished through the updating procedures of the belief networks. We are currently evaluating different ways to update the conditional probability table values when questions and exercises are given more than once to the same student. Table 4.2.1 shows the initial evaluation of ITEAMS as an Intelligent Tutoring System.

### Table 4.1

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Subject</td>
</tr>
<tr>
<td>Multiple Subject</td>
</tr>
<tr>
<td>Suggestion Providing</td>
</tr>
<tr>
<td>Hint Providing</td>
</tr>
<tr>
<td>Explicit Student Model</td>
</tr>
<tr>
<td>Explicit Instructor Model</td>
</tr>
<tr>
<td>Explicit Assessment Model</td>
</tr>
<tr>
<td>Explicit Material Model</td>
</tr>
</tbody>
</table>

4.2.2 Instructor Model

There are various possible approaches for modeling the instructor. At one extreme, the model could be restricted to one approach only for presenting and structuring information. At the other extreme, the model might allow for any type of structure. Both approaches, as well as any in between, are valid. The following sections discuss some of the issues regarding these approaches.
Single Style Approach

The ‘Single Style’ approach clearly makes the design of a system easier, since the system has one configuration only to handle, not multiple ones. This can also make the design of content easier to use, since all that is required is to fill in the model with the information.

This leads to some potential problems, however. Foremost is the lack of flexibility afforded the instructor in designing the course. If an instructor is not comfortable with the one teaching style that must be used, it is altogether possible that the end result will be a poorly designed course. The ‘Single Style’ approach can restrict the creativity of an instructor by reducing the options for presenting material in a manner or format which this instructor feels would most benefit the students.

Multiple Style Approach

Between the two extremes is the ‘Multiple Style’ approach. This approach makes the design of a system more complicated than when using a single style since the system has more configuration to handle. However, this does allow for more freedom and creativity for the instructors.

This approach likewise has its problems. Foremost again is the lack of flexibility the instructor has in designing the course. If an instructor is not comfortable with one of the teaching styles provided, then once again the result may be a poorly designed course. And as stated above, this approach may restrict the instructor’s creativity and thus restrict the potential benefit to the students.

Instructor Independent Approach

At the far extreme of possible approaches is the ‘no-style’ approach. This style allows for total freedom and flexibility. The instructor can design the course without restriction. This approach can make the design of content easier for some instructors since there is
no model into which they must force their material.

A major drawback with this approach is its inherent difficult in implementation. This difficult arises from trying to “allow for everything.” Every system will be designed with its own time and space constraints. It is simply not feasible to make a system so wide open that it can effectively allow for anything and everything.

**TABLE 4.2**

THE EVALUATION OF THE ITS FEATURES THAT ITEAMS CURRENTLY POSSESS AFTER THE DISCUSSION OF THE INSTRUCTOR MODEL.

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Subject</td>
</tr>
<tr>
<td>Multiple Subject</td>
</tr>
<tr>
<td>Suggestion Providing</td>
</tr>
<tr>
<td>Hint Providing</td>
</tr>
<tr>
<td>Explicit Student Model</td>
</tr>
<tr>
<td>Explicit Instructor Model</td>
</tr>
<tr>
<td>Explicit Assessment Model</td>
</tr>
<tr>
<td>Explicit Material Model</td>
</tr>
</tbody>
</table>

Even though this approach is more difficul to use, the design of ITEAMS does incorporate this approach as its own model. ITEAMS is intentionally designed to be fl xible and to allow instructors maximum freedom in the design of their courses. ITEAMS does not yet allow for everything, but every effort has been made to develop a system that is fl xible to the maximum degree. Since there is no explicit instructor model, the overall system evaluation of ITEAMS is still the same. See table 4.2.2.

4.2.3 Material Model

As mentioned in section 1.1.6, the “material model” is define thus

**Material Model** The content that is presented to the students. This contains
both lecture and quiz materials. This model can also dictate the structure and flow of material.

ITEAMS’ material model does just that - it defines the content that is presented to students as well as the structure and flow of the material.

The material model allows instructors the freedom to create teaching modules in their own teaching style and in a manner which is logical and straightforward.

Design Considerations

The material model gives instructors the freedom to create teaching modules containing lecture materials, quizzes, and assignments arranged in the instructor’s own specific teaching style. The material model provides content in terms of instructional categories, teaching modules, sections, goals, quizzes, assignments, and associates them with each other.

Instructional categories are instructor-specific objectives or learning categories that are used to help assess a student’s performance. These categories are used by quizzes and assignments to direct how a question or exercise relates to the material and overall performance of a student. An introductory course in programming, for example, could have categories of implementation, programming language knowledge, analytical ability, etc.

The instructor has to define and structure the sequence of lecture material and instructional goals in order for ITEAMS to be able to present the material to the student in a coherent manner. The arrangement is accomplished graphically by “dropping” a section, quiz, or assignment into a teaching module and then adding the necessary instructional material.

Sections are created by adding learning objectives, goals (which, if reached, will accomplish the objectives), quizzes, and assignments. Since ITEAMS attempts to tailor lecture materials to meet students’ learning needs, materials can be added at different lev-
els of difficult. This allows instructors to provide a greater degree of flexibility in the student’s learning process. Each goal represents a lecture or lecture topic, allowing one or more goals to be accomplished in a given session. Each section can have a quiz and an assignment associated with it which are presented to students after they have completed all of the section’s goals. Each goal can also have a quiz and/or assignment associated with any or all of the levels of material.

Currently, quizzes allow one type of question. Questions are in multiple choice format where each answer has some “percentage correct” associated with it and can be specified by the following steps: (1) giving the question text, (2) selecting which instructional category it references, (3) selecting the difficulty of the question, and (4) providing any answers with their associated correctness.

Assignments consist of multiple parts which are separated by exercises. An exercise is very open. It can be a question that students need to answer or it can be a programming problem that requires an external system (e.g., a simulation environment), which is accessed through a plugin. As more types of questions are added to ITEAMS, more will become available for exercises. Exercises using external systems via plugins allow the instructor to specify parameters of the external system that will be used for grading (see the next subsection).

Once one or more sections have been created, an instructor links the sections together to create the flow of information and depict instructional dependencies in the material (e.g., one section should come before another).

Since ITEAMS allows instructors the freedom to create their own course content ITEAMS is able to be used in multiple subjects.
### TABLE 4.3

THE EVALUATION OF THE ITS FEATURES THAT ITEAMS CURRENTLY POSSESS AFTER THE DISCUSSION OF THE MATERIAL MODEL.

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Subject</td>
</tr>
<tr>
<td>Multiple Subject</td>
</tr>
<tr>
<td>Suggestion Providing</td>
</tr>
<tr>
<td>Hint Providing</td>
</tr>
<tr>
<td>Explicit Student Model</td>
</tr>
<tr>
<td>Explicit Instructor Model</td>
</tr>
<tr>
<td>Explicit Assessment Model</td>
</tr>
<tr>
<td>Explicit Material Model</td>
</tr>
</tbody>
</table>

#### 4.2.4 Evaluation Scheme

There are three types of evaluation schemes that ITEAMS could use for its evaluation scheme:

1. Numerical Evaluation
2. Neural Networks
3. Belief Networks

The usefulness, in regards to ITEAMS, of each of these schemes is discussed below.

**Numerical Evaluation**

This form of evaluation is simple and easy to implement. Numerical Evaluation is used when a simple grade-based performance is desired. Since ITEAMS is an Intelligent Tutoring System, ITEAMS needs to have the possibility for generating hints, suggestions, and determine deficiencies in students; thus numerical evaluation is not a viable evaluation scheme.
Neural Networks

Since neural networks provide a mechanism for determining the path a student takes through material, they are a strong possibility for the evaluation scheme in ITEAMS. This, however, might not be the best mechanism to use when creating an educational system that is a combination of both Teaching Environments and Intelligent Tutoring Systems, since evaluation of the student is wanted as well as being able to determine the student’s misconceptions and to generate suggestions.

Belief Networks

Since belief networks provide the most flexibility in evaluation compared to neural networks and numerical evaluation, ITEAMS uses this form of evaluation scheme. Using this form of evaluation, ITEAMS can also generate suggestions. The types and uses of belief networks used in ITEAMS are discussed in full in the chapter “Belief Networks in ITEAMS”.

TABLE 4.4

THE EVALUATION OF THE ITS FEATURES THAT ITEAMS CURRENTLY POSSESS AFTER THE DISCUSSION OF THE EVALUATION MODEL.

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Subject</td>
<td></td>
</tr>
<tr>
<td>Multiple Subject</td>
<td>X</td>
</tr>
<tr>
<td>Suggestion Providing</td>
<td>X</td>
</tr>
<tr>
<td>Hint Providing</td>
<td></td>
</tr>
<tr>
<td>Explicit Student Model</td>
<td>X</td>
</tr>
<tr>
<td>Explicit Instructor Model</td>
<td></td>
</tr>
<tr>
<td>Explicit Assessment Model</td>
<td>X</td>
</tr>
<tr>
<td>Explicit Material Model</td>
<td>X</td>
</tr>
</tbody>
</table>
4.2.5 Assessing Student’s Quiz Performance and Assignment Section Performance

A student’s performance on a quiz is determined by the answers to questions that are given to the student through the course of the quiz. The performance is a numerical evaluation of the correctness values for each question answered. Thus if the student answers ten questions, then the numerical values of the correct answers to the questions will be added and averaged. The numbers are not just totaled; higher difficult questions influence the quiz more than lesser difficult questions. For example, when a question of difficult level two is answered, the numerical value of the answer is added twice to the total value of the other answers, or it counts as if the student answered two level one questions. Each level is treated in a similar manner; for each level beyond the first the value of the answer is added in again. Using this method, a level four question is then worth four level one questions. This is the initial method for incorporating difficult levels into the evaluation scheme. The reason for selecting this method is that it is a static or constant measure for “difficult.” This method is relativly simple and is used because we are focusing our work in other areas of ITS and not in modeling the relationship of difficult levels in questions (and exercises) to student performance in general topics. Our focus is on producing a viable educational system which happens to include the feature of assessing students based on multiple levels of difficult.

Assignment Sections work the same way as quizzes; the values of the exercises are added together and averaged over the number of exercises given. The value of an exercise is adjusted based on difficult in the same way a question’s value is adjusted when calculating the performance of a student.

4.2.6 ITEAMS Plugins

Some Teaching Environments, like eCollege and WebCT, have features that interface with external software, but restrict how the software can interface the environment by
restricting the platform, programming language, or both. This restriction is overcome in 
ITEAMS through the external interface, also referred to as the “plugin interface.”

Through the “plugin interface,” instructors are able to use applications, environments, 
etc., for their course and subsequently to allow students to interact with them from within 
ITEAMS. For each application, a separate “plugin” has to be define and implemented, 
which translates the ITEAMS external interface format into a format recognized by the 
external application (e.g., via the Component Object Model or external function calls in 
MATLAB). This format is used for several reasons:

- system configuration
- information exchange
- automatic evaluation

Figure 4.1. A diagram of the plugin interface. It shows how ITEAMS and external appli-
cation communicate with the plugin interface. The diagram also shows how the applica-
tion specific interface communicates with the ITEAMS plugin interface.

The application “plugin” that must be define for each application works as a bridge be-
between external systems and the ITEAMS “plugin interface.” Figure 4.1 shows functionally where “plugins” fit into the overall ITEAMS plugin interface. It is possible to envision designing one or more “plugins” per application as a severe restriction. By placing some of the design of “plugins” to external sources, such as instructors or application developers, a more robust product can be created. If a single generic “plugin” were developed for ITEAMS that would access any software it would have to do so in a generic fashion. As a result specialized performance evaluation would be lost. By requiring each application to use a “plugin” designed specifically for it, a student’s performance should be more accurately evaluated.

The “plugins” provide several integral functions. They determine the kinds of assignments that are possible based on the parameters they provide, and they also provide the means to grade/evaluate the student’s performance on exercises based on these parameters. In general, they are used to determine the student’s performance on an assignment.

Plugins and their functionality are discussed in full in the next chapter, where all key features will be explained:

- how a plugin interacts with both ITEAMS and the external systems
- how a plugin can be designed
- the Plugin API

Since other Intelligent Tutoring Systems do not provide this feature, nothing is updated in the ’ITS Evaluation Matrix’ for ITEAMS. Table 4.2.6

4.2.7 Application Environment

In the design of any computer application many different design decisions must be made. Most of these revolve around how the system works or what programming language will be used, the intended audience, etc. Many times in the design of a Teaching Environment, one aspect is left out, namely where the system is going to be used. This can mean either at what physical location the software is used, such as home, office or school, or what
TABLE 4.5

THE EVALUATION OF THE ITS FEATURES THAT ITEAMS CURRENTLY POSSESS AFTER THE DISCUSSION OF THE PLUGIN CAPABILITIES.

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Subject</td>
</tr>
<tr>
<td>Multiple Subject</td>
</tr>
<tr>
<td>Suggestion Providing</td>
</tr>
<tr>
<td>Hint Providing</td>
</tr>
<tr>
<td>Explicit Student Model</td>
</tr>
<tr>
<td>Explicit Instructor Model</td>
</tr>
<tr>
<td>Explicit Assessment Model</td>
</tr>
<tr>
<td>Explicit Material Model</td>
</tr>
</tbody>
</table>

type of learning location is used, such as lab, classroom, or independent use. The current Teaching Environments are all web-based applications which can limit a student’s and instructor’s ability to work with them. Teaching Environments do have some advantages, including 1) easy maintenance, 2) common servers and repository of data, and 3) data and environment consistency, etc. Intelligent Tutoring Systems are just the opposite of the Teaching Environment. They are self-contained applications with most or all of the information needed to present to a student packaged with them.

With the differences in the application environments that Teaching Environments and Intelligent Tutoring Systems use being so dramatic, ITEAMS is designed to use both as its environment. This creates a ubiquitous learning environment which allows instructors and students to work on teaching modules when they want and where they want. ITEAMS allows users to work either online or offline and switch between the two without the user’s knowledge. Each environment has its own benefits problems, and application to learning. Each of these environments will now be discussed.
Online Environment

There is a growing number of Teaching Environments which are all web based as well as a growing number of universities that are offering distance learning courses [17]. In order to better meet the educational needs of those institutions, ITEAMS has been designed to work in an online environment. This environment allows both students and instructors to work from anywhere there is an Internet connection.

The ITEAMS online environment uses a database, which is reachable from the user’s computer, to store all the information that is needed for ITEAMS to operate. This information includes both teaching module content as well as student and instructor information. ITEAMS connects to the database and requests both user information and teaching module information. The database host must be provided to ITEAMS through either a configuration file or from within ITEAMS using a dialog box. This enables students and instructors to work on different teaching modules located at any location. Figure 4.2 shows the online environment.

The database is designed for several purposes. The first is to simply store the information of the teaching modules and students’ information. This is realized by storing the teaching module data and student model data. The second purpose is to provide a means of history and revisions for the teaching modules. This has been accomplished by adding extra information to the data stored such as a date and revision numbers. The last purpose is for quick data acquisition and processing when an instructor, for instance, wants to see how a class or student is performing.

The advantages of using a database format are numerous: Some are discussed here. Databases are becoming very common in many environments to provide a central location for information, allowing ITEAMS to be easily integrated into an existing database structure. Currently ITEAMS is designed to be used with the MySQL [31] relational database. And, since standard SQL statements have been strictly followed, migrating to
a different database server does not pose any problems. Databases have also increased in speed, making queries and inserts into a database very fast and more efficient. Since almost all databases are accessible from remote connections over a network, little effort is needed to connect to and communicate with them from virtually anywhere (as long as the database is accessible from an outside network address).

Since the database handles most of the processing of data for storage and retrieval of data, ITEAMS is able to use the database also for a revision system. Instructors can have multiple versions of a teaching module. They can have a ‘release version’ which has been released to the students as well as a ‘current version’ which they are updating and modifying. The database also provides a ‘deleted’ history. In the future, ITEAMS will include functionality so an instructor can retrieve information from older teaching modules that have been “deleted” or removed from the released status. Providing this flexible structural format for instructors gives them flexibility to test new course designs before the class can see the changes. This does create more work for the instructor, but the benefit to the learning process should be greater than the effort put into the design of a good course.

There are several disadvantages to using a database. These include accessibility issues, database portability issues, performance bottlenecks, and also costs. Since a database could be running on a remote machine, that machine could become unreachable. This could happen for any number of reasons ranging from hardware issues to software issues. The hardware issues include problems such as

- Power failures
- Network cable and hardware issues
- Server hardware failure

Any of these problems will lead to the database not being reachable. On the software side potential problems exists as well. These are mainly configuration issues and include

- Network configuration problems
- Missing software drivers
- Hacked systems

Currently ITEAMS uses an unencrypted format to send data to the database. This places clear text on the network. Consider this extreme case: a student is viewing packets on the network and discovers the ability to intercept answers or update one’s own performance. There are currently several possible designs that ITEAMS can use to address this potential problem.

Online Environment Scenario

Figure 4.2. A diagram of the online environment. This environment allows users to connect to a central server to retrieve course data from any location that can connect with the database server.
In this scenario ITEAMS is the instructional tool that was selected to use in a course on Software Engineering. At end of the course the instructor wants to present a teaching module on “Secure Application Development.” This module will instruct students on the principles of designing secure applications throughout the entire software life cycle. The environment in which this instance of ITEAMS is run is an online environment. The instructor and students each have a version of ITEAMS and a separate computer on which it runs. The teaching modules are stored in a database located on a host separate from each student’s computer and the instructor’s computer. This setup is shown in figure 4.2.

To create and complete this module, the instructor first creates the module by starting an instance of ITEAMS. Once ITEAMS has loaded, the instructor provides the location of an ITEAMS database and then creates a new teaching module, inputs several instructional categories, and adds several sections. At this point the instructor stops working and saves the teaching module. The data that has been input by the instructor is then stored in the database. Note that in this example the teaching module has been given the following name: “Secure Application Development.”

At a later time, the instructor decides to complete the teaching module and release it for the students to use. The instructor runs ITEAMS and accesses the same database location as before. Instead of creating a new teaching module, the instructor “opens” the “Secure Application Development” teaching module. ITEAMS requests the teaching module data from the database. Since the database contains the teaching module, the information is transmitted back to ITEAMS where the data is loaded. The instructor then completes the teaching module, saves it to the database, and releases it to the students.

Now that the teaching module has been released to the class, each student can access it from any location. Students who need to work on the new teaching module run ITEAMS and access the location of the course’s database and login to the course. The login process retrieves the students’ most current student model. The students then load “Secure Appli-
cation Development,” after which they work through the sections in the teaching module. As students work through a teaching module, their student model is updated with their history. Once the teaching module is completed, the students’ information, in the form of the student model, is sent to the database. After students have begun working on a teaching module and saved their progress at least once, the instructor views how the students are performing in the “Secure Application Development” teaching module.

Offlin Environment

In order to meet the needs of Intelligent Tutoring Systems which rely on pre-structured information that is always available to the user, ITEAMS also provides an offlin environment. This environment allows students and instructors to work from any location regardless of the presence of an Internet connection.

The offlin environment uses an encrypted file to store teaching module information and student information. The data is stored in an XML format which is easily loaded and generated and provides ITEAMS with a format that is accessible on any operating platform. The XML format is used for processing efficiency. The data in this format can be imported and exported and we are assured that the data is valid. Data is exported in a specific XML structure which models the internal ITEAMS representation of a teaching module. Using this format an exported teaching module can be imported into ITEAMS with its structure and data intact. Other formats, such as binary, are not as easily read in. There is no limit on the length of information input into ITEAMS. This would require that extra information be stored and retrieved thus causing an increase in processing complexity when importing and exporting.

In order for ITEAMS to use the offlin format, the instructor must generate the appropriate file and distribute them to the class or make them available in some form. How the file are distributed is instructor dependent. There are at least two file that must be
generated by the instructor for any individual to work with ITEAMS. These file are the teaching module file and the student record file. Currently the file are automatically created and saved whenever an instructor updates information. When a teaching module is updated, a new teaching module file is updated. The same happens with the student record file. When the instructor updates a student’s profile a new record file is generated. When a student works on a teaching module, only a new student record file is generated. This file contains all the new material that the student worked on during the current session with ITEAMS.

One advantage associated with using this environment is that the student and instructor can be completely separated from the online environment. This can also be a disadvantage since the data must be stored locally on the user’s machine. In the student’s case, the record file must be returned to the instructor for evaluation when he/she finishes the teaching module or when it is requested. The file can be returned by whatever means the instructor requires.

Another advantage of this environment is that it can act as a revision system. By making copies of the teaching module file before saving a new version over it, a revision system can be created that allows previous teaching modules to be archived. This also allows an instructor to have one version of the teaching module that is ‘released’ to the class and also have a version that is the ‘current’ version being revised. The disadvantage is that a file structure can become loaded with revision file and consume more space than is desirable.

Offlin Environment Scenario

In this scenario, as before, ITEAMS is the instructional tool selected for use in a course on Software Engineering. At end of the course the instructor wants to present a teaching module on “Secure Application Development.” This module will instruct students on the
Figure 4.3. A diagram of the offline environment. This environment allows users to work on teaching module when a fla file is provided to them through an instructor’s method of choice.
principles of designing secure applications throughout the entire software life cycle. The environment setup in which this instance of ITEAMS is run is an offline environment. The instructor and students each have a version of ITEAMS and a separate computer on which it runs. Unlike the online scenario, there is no central course database for everyone to use. The instructor has a small classroom of computers which are not connected to a network. Floppy drives are the means of teaching module distribution. The teaching modules are stored in encrypted file and transferred via floppy diskette. This setup is shown in figure 4.3.

To create and complete this module, the instructor again creates the module by starting an instance of ITEAMS. Once ITEAMS has loaded, the instructor performs several tasks: 1) creates a new teaching module, 2) inputs several instructional categories, and 3) adds several sections. After the instructor completes these tasks, he stops working and saves the teaching module. The data that has been input thus far by the instructor is then stored in an encrypted format onto the computer’s hard drive. Note that in this example the teaching module file has been given the following name: “Secure-Application-Development.iteams.”

Several days later, the instructor decides to complete the teaching module and distribute it to the students. Instead of creating a new teaching module, the instructor, this time, opens the “Secure Application Development” teaching module by loading the file “Secure-Application-Development.iteams”. The instructor then completes the teaching module, saves it to “Secure-Application-Development.iteams,” copies the file to a floppy disk and finally distributes it to the computers in the classroom.

Now that the teaching module has been distributed to the class, each student can access the teaching module from any computer in the classroom. Students who need to work on the new teaching module run ITEAMS and login to the course. The login process attempts to load the students’ student model from disk. If the student has not worked
at that computer and does not have a copy of the student model, a new student model is created. The students then load the “Secure Application Development” file and work through the sections in the teaching module. As students work through the teaching module, their student model is updated with their history. Once the “Secure Application Development” teaching module is completed, each student’s information, in the form of the student model, is stored in a new student model file. The instructor can monitor the class’s progress or a single student’s progress by loading each student model file after the file have been collected.

Combination Environment

One unique feature of ITEAMS is its ability to allow students and instructors to work offlin and then to work online again, updating the database for them with or without the users being aware. This can be thought of as the ‘combination’ environment. It combines the user’s offlin work seamlessly with their online work. This type of environment is meant for individuals who travel or have a portable computer on which ITEAMS is used. When individuals, either students or instructors, work with ITEAMS, a database might not be available. In this case the offlin is being used. The user has access to teaching modules through a fla file stored on their computer. When the user goes to a different location which provides access to the database any changes made during previous offlin sessions are updated in the database. This has been designed to be of use to both students and instructors.

This environment is still in the initial testing phase. The full benefit of this environment has not yet been seen. More testing is needed before it is fully included in the current version of ITEAMS.
Combination Environment Scenario

In this example, as before, ITEAMS is the instructional tool that was selected for use in a course on Software Engineering. At end of the course the instructor wants to present a teaching module on “Secure Application Development.” This module will instruct students on the principles of designing secure applications throughout the entire software life cycle. The environment setup in which this instance of ITEAMS is run is a combination environment. The instructor and students each have a version of ITEAMS and a separate computer on which it runs. The instructor and several students have portable computers on which ITEAMS runs. The locations in which ITEAMS is run, might not have a connection to the database. In which case, the fla file is used to retrieve and store information.

To create and complete this module, the instructor again creates the module by starting an instance of ITEAMS. When the instructor begins working on the teaching module, a database is accessible. Once ITEAMS has loaded, the instructor performs several tasks: 1) create a new teaching module, 2) input several instructional categories, and 3) adds several sections. After the instructor has accomplished these tasks he stops working and saves the teaching module. The data that has been input thus far by the instructor is then stored in the database and also stored in an encrypted format onto the computer’s hard drive. Note that in this example the teaching module file has been given the following name; “Secure-Application-Development.iteams.”

Several days later, the instructor decides to complete the teaching module. Now, there the database for the course un available due the lack of a network connection. Instead of creating a new teaching module, the instructor, this time, selects to “opens” the “Secure Application Development” teaching module which is loaded from the file “Secure-Application-Development.iteams.” The instructor then completes the teaching module, saves to “Secure-Application-Development.iteams.” The instructor knows that
there was no connection available at the time the saved occurred and when at home starts ITEAMS again. Once loaded, the “Secure-Application-Development.iteams” teaching module is loaded and the changes are sent to the database. At this point the teaching module is released to the students.

Later, in class, the updated “Secure-Application-Development.iteams” fil is given to several students who do not have access to the database. Now that the teaching module has been distributed to the class, each student can access the teaching module from any computer. Students who need to work on the new teaching module run ITEAMS and login to the course. The students then select to load the “Secure Application Development” teaching module. If there is no connection to the database, ITEAMS loads the fla file. Once loaded the students work through the sections in the teaching module. As students work through the teaching module, their student model is updated with their history. Once the “Secure Application Development” teaching module is completed, the students’ information, in the form of the student model, is stored in the database or a new student model file. If a student does not have a database connection, they bring the student model fil back to the instructor. Otherwise, the instructor retrieves students information from the database.

4.2.8 Graphical User Interface

ITEAMS has two different user modes for working with teaching modules. The first is the Administrator/Instructor Mode and the other is the Student mode. The Administrator/Instructor mode is an enhanced version of the student mode. Both modes are discussed below.

Administrator/Instructor Mode

The Administrator/Instructor mode, which will be referred to simply as the Instructor mode, allows for the creation of teaching modules through a graphical user interface.
Figure 4.4. Several different modes and views are shown. The frontmost window is the Section Interface. In the back is the Teaching Module Viewer (left) and Assignment Viewer (right).
This interface allows instructors to “drop in” sections and quizzes when working on a teaching module.

A simple graphical user interface has been designed for the teaching module, which allows the instructor to design the material in terms of instructional categories, teaching modules, sections, and goals and associate them with each other.

Figure 4.4 shows a collection of views that are common in ITEAMS. These are discussed more fully next.

![ITEAMS Interface](image)

Figure 4.5. The Edit mode in ITEAMS is the mode in which instructors create and modify teaching modules.
The Instructor mode provides the means to create and modify teaching modules. This is done through the “Teaching Module Viewer.” This viewer is the basic window in ITEAMS; all teaching module information is accessed through this screen. In the Instructor mode, this screen provides an editing space which is shown in Figure 4.5. This mode looks identical to the main Teaching Module Viewer screen but with the addition of an editing toolbar (figure 4.6). Functionally the two modes are different even though they look similar. The editing toolbar includes access to instructors’ editing capabilities:

1. create a teaching module
2. add and edit instructional categories
3. add a section
4. add a quiz

Figure 4.6. The Edit toolbar in ITEAMS contains quick access to editing capabilities.

Once sections or quizzes have been added to the teaching module, they can be clicked on to open their own editing windows. The section editing window (figure 4.7) allows instructors to create learning objectives, add instructional goals, add quizzes, add assignments, and organize the material in whatever order is desired. Once a goal has been added to a section, it can be modified by adding material. Each goal can have material of varying difficult levels associated with it. Once the goals are created, they can be arranged in any order desired. Figures 4.7 and 4.8 show some of the different section and goal editing windows in ITEAMS. Quiz editing takes place in its own windows shown in Figure 4.9. There are two different screens shown: one is the main window for creating quizzes, shown on the top; the other is for creating a question. When an instructor
This window provides instructors with the functionalities of creating and editing the learning content which comprises a teaching module.

is working on a question, he/she can select to which instructional categories it relates. The question must relate to at least one instructional category. Because of this, if no instructional category is specified it is implied that all instructional categories relate to the question. This is a short cut to selecting all the instructional categories.

Assignment editing is performed in another set of screens (Figures 4.10 and 4.11) which are specific to assignments. These screens allow instructors to organize the assignments into sections which have exercises at the end. When creating an exercise, the instructor has the choice of what type of exercise to create. Figure 4.11 shows the choice screen for exercises. Any available plugins are shown in this list. When a plugin is selected as the exercise, ITEAMS starts the plugin in its administrator mode, which allows an exercise to be configured and saved. This is discussed more fully in the next chapter.

It should be noted that exercises must also relate to at least one instructional category. When no category is selected, ITEAMS assumes it relates to all categories.

Once an instructor has worked on a teaching module and created at least one section with one goal, it is possible to view what a student views by selecting the Section Viewer. The Section
Figure 4.8. Instructors provide lecture material for the instructional goals through the Material editor.
Figure 4.9. The Quiz Editing windows contain the tools to create and organize quizzes within I TEAMS.
Figure 4.10. The Assignment Editing window contains the tools to create and organize assignment within ITEAMS.
Figure 4.11. The Assignment Section Editing windows contain the tools to create and organize assignment sections and exercises within ITEAMS.
Viewer is described in the Student Mode section.

Student Mode

The student mode provides students as well as instructors with a means of interacting with the sections within a teaching module. The mode is started by selecting 'View Sections' from the list of available modes. Figure 4.12 shows the section viewer screen which gives users the functionality to navigate through a section. While the viewer is open, users can view a different section by selecting it in the “Teaching Module” viewer. A section can be viewed only if it has been unlocked by ITEAMS. Sections are unlocked by ITEAMS once a student has successfully completed the prerequisites. The prerequisites for a section are the sections that are linked to it as parent sections. When navigating a section, the user can select different goals to view by selecting the goal from the left hand navigation tree.

Besides the Section Viewer, students have access to the Quiz Viewer and Assignment Viewer. These viewers perform what their names imply. The Quiz Viewer, figur 4.13, is what ITEAMS uses for giving quizzes to the students, and the Assignment Viewer, figur 4.14, is used to present assignments to students. These viewers are updated by the belief networks associated with the content they are presenting. The Quiz Viewer has the next question chosen by a ‘quiz belief network’. The Assignment Viewer has exercises chosen by the belief network that represents the current assignment section.

4.3 Chapter Summary

This chapter has presented the design of ITEAMS and how it incorporates the features of both Teaching Environments and Intelligent Tutoring Systems. The chapter presented the reader with ITEAMS’ Student Model, Instructor Model, Material Model, and Assessment Scheme. It has also introduced the plugin capability that allows any existing software to interface with ITEAMS. The application environment was also introduced followed by
Figure 4.12. The Section Viewing window contains the tools to walkthrough sections and goals in ITEAMS.
Figure 4.13. The Quiz Viewing window contains the tools to take quizzes in ITEAMS.
"Spam: The Phenomenon is a detailed analysis of spam: products, scams, viruses, obfuscation methods, etc. Failed, and doomed-to-fail, methods of blocking spam are described. A general solution is proposed that does not: invade privacy, perform wide censorship or blacklisting, or involve payment and cooperation with corporations (beyond the transport and storage of data)." Hinnick.

Figure 4.14. The Assignment Viewing window contains the tools to take assignments in ITEAMS.
the graphical user interface.

To recap the features, recall the features in table 4.3:

TABLE 4.6

THE EVALUATION OF THE ITS FEATURES THAT ITEAMS CURRENTLY POSSESS.

<table>
<thead>
<tr>
<th>ITS Category Matrix</th>
<th>Single Subject</th>
<th>Multiple Subject</th>
<th>Suggestion Providing</th>
<th>Hint Providing</th>
<th>Explicit Student Model</th>
<th>Explicit Instructor Model</th>
<th>Explicit Assessment Model</th>
<th>Explicit Material Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

ITEAMS is designed to be used in any course and allow the instructors to input any information they need to present to their students. ITEAMS can generate suggestions for the students. These suggestions let the students know that they need to go back and review material before moving on because their performance is not high enough to continue. There is an explicit student model which is used to track the student’s history in a teaching module. It allows the instructor to see how the student is performing and can re-evaluate the student on the fly by loading the student’s information. There is no explicit instructor model in ITEAMS. This allows instructors to model the material any way desired and present the material to students in a way that is most comfortable to them. ITEAMS provides an assessment module which uses several different types of belief networks. The belief networks used are

- Singly Connected Networks
- Multiply Connected Networks
• Structurally Dynamic Belief networks.

A material model is provided to give the instructor a way to structure the material. This model records how the material is to be presented to the student.
CHAPTER 5

ITEAMS PLUGIN

The external interface allows instructors to use applications, environments, etc., for their courses and subsequently allows students to interact with them from within ITEAMS. For each application, a separate “plugin” has to be define and implemented. This translates the ITEAMS external interface format into a format recognized by the external application (e.g., via the Component Object Model or external function calls in MATLAB) if it supports information exchange. A plugin is a piece of software that connects ITEAMS and external software, providing a means of communication and data exchange for the purpose of student learning and evaluation.

5.1 Overview

The plugins provide several integral functions. They determine the kinds of assignment exercises that are possible based on given parameters, and they also make it possible to grade/evaluate the student’s performance on exercises based on these given parameters. In general, plugins are used to help determine the student’s performance on an assignment.

There are several ways in which a plugin works: 1) a plugin can work by starting an external application on the same machine on which ITEAMS is running and connect to that software via a TCP connection; 2) a plugin can work by connecting to an already running application via a TCP connection; 3) a plugin can work by contacting a software registry which then automatically connects software with the “plugin interface”; and 4)
A plugin can itself function as the external software. In this case, there is not actually any piece of software with which the plugin communicates. The third option above is discussed more fully in Chapter 7.

Once a plugin has been started and a connection has been made between ITEAMS and the external software, the plugin sends configuration information to the external software for a specific exercise. Once this information is sent, the plugin is notified by the external software that the exercise can begin. The plugin then allows the student to work on the exercise.

If a plugin is designed in a specific manner, it has the potential of being used for quizzes as well. An example of such would be a plugin for Dr. Scheme [41]. Dr. Scheme is a programming environment for the Scheme programming language. It provides an interactive, graphical environment as well as a command line interface to the Scheme interpreter. One capability of this plugin is that it can generate scheme expressions and evaluate them. Another capability is that the Dr. Scheme plugin has automatic question generation. The automatic generation of scheme expressions allows for automatic generation of questions of several different forms:

1. 'Evaluate this expression.'
2. 'What does this expression evaluate to?'
3. 'True or False. This expression will/will not evaluate correctly.'
4. 'Write the scheme code to do:'

In order for a plugin to assess the progress of students, the instructor must provide a metric for the solution. This is done by specifying ranges in the data that the plugin reads from the output generated by an external application. For example, an agent-based programming exercise in an artificial intelligence course asks the student to design an architecture using a particular design paradigm for a virtual agent that makes it search for a target location while avoiding obstacles. The evaluation could then be based on the agent’s performance: how fast the agent reaches the target location and how many
collisions it had along the way. The instructor would only have to specify the ranges and percentages for both dimensions and the corresponding percentage of the score that is awarded if the respective criteria are met (see Table 5.1).

### TABLE 5.1

SAMPLE RANGES FOR A PLUGIN

<table>
<thead>
<tr>
<th>Collisions</th>
<th>Time-to-Target</th>
<th>Correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \leq 1$</td>
<td>$x \leq 100$</td>
<td>1.0</td>
</tr>
<tr>
<td>$1 &lt; x \leq 5$</td>
<td>$x \leq 100$</td>
<td>0.75</td>
</tr>
<tr>
<td>$1 &lt; x \leq 5$</td>
<td>$100 &lt; x \leq 500$</td>
<td>0.5</td>
</tr>
<tr>
<td>$5 &lt; x$</td>
<td>$500 &lt; x \leq 500$</td>
<td>0.0</td>
</tr>
</tbody>
</table>

The only requirement for an external application is that it be able to produce an output that contains information that can be evaluated by a plugin. Plugins are designed to adhere to certain properties. They must accept *learning parameters* from ITEAMS, which are partially generated by ITEAMS and partially specified by the instructor; must start any external environment; must prepare it for the student’s use; and must use the parameters specified to assess a student’s performance. *Learning parameters* specify 1) how *exercise* answers relate to the correctness values (Table 5.1), 2) how other *exercises* relate to this *exercise*, and 3) which *instructional categories* relate to the *exercise*. The instructor specifies the correctness values and the relation to *instructional categories*. ITEAMS specifies the relationship between *exercises*.

The design of plugin is utilized by any individuals who need to use a specific application with ITEAMS. This can be instructors who want to use ITEAMS in their courses with applications that do not currently have a plugin. It could also be that instructors might require a plugin with a different set of capabilities than currently available for an application and they themselves design the new plugin. The designer could be from the company
whose software is being used. Frankly, the plugin could also be designed by anyone. There are no restrictions on who can design a plugin, as long as the plugin design criteria are met.

5.2 Purpose

Some Teaching Environments offer a means for external applications to be interfaced, but Intelligent Tutoring Systems lack this ability. In Teaching Environments the ability to assess students is limited, allowing mainly multiple choice questions and allowing for instructor-define material. Intelligent Tutoring Systems provide more robust evaluation, allowing for short answer evaluation or other textual information. The content they offer is, however, not always definable by an instructor. ITEAMS tries to close the gap between these systems and the plugin interface by allowing instructor-define material, a robust evaluation scheme that is definable by the plugin designer, and an easily definable interface for quick development and deployment.

As mentioned in the chapter discussing Intelligent Tutoring Systems, a system such as ACE might well benefit from the ITEAMS plugin or a plugin in general by allowing access to a program such as MATLAB. Since ACE is used for tutoring on simple mathematics, it could be extended for more advanced math. ACE could give students the ability to evaluate complex math equations through an interface with MATLAB. One way for ITEAMS to close the gap between Teaching Environments and Intelligent Tutoring Systems is to include the feature to interface with external systems specifically for assisting in the evaluation of a student’s performance in assignments. Since ITEAMS is being designed so that it can be used as a tutoring tool or a teaching tool used in or out of class, automatic evaluation of assignments is a desirable feature.

Several issues have been addressed in the design of the interface. The first is that the plugin interface must be very general and enable an interface with any software that an
instructor wants to use in a class. Flexibility is yet another issue. The plugin interface needs to be very flexible and not restrictive so that its own design constraints do not reduce the usability and effectiveness of the plugin. Similarly, communication must be possible between ITEAMS and the plugin, as well as between the plugin and external software. This last issue was the simplest to solve. It involves only reviewing connection methods and determining which one best meets the needs of ITEAMS.

5.2.1 A Need For Generality

The plugin interface should allow ITEAMS to configure the external software and connect to the software for communication purposes and for repeated uses. Any existing or new system should be able to interact with ITEAMS through the plugin interface. The plugin interface allows systems to have a plugin connection method built into the software. Otherwise an application wrapper can be used. The plugin connection method allows ITEAMS to connect to the software directly via TCP/IP. In this way ITEAMS directly manipulates the external software. The application wrapper, on the other hand, interfaces with the external software and ITEAMS connects to the wrapper.

Further, a plugin can be designed as a self-contained piece of software allowing all the functionality and features that an instructor needs. One such example would be a plugin which offers natural language processing. The plugin would not need any external software; the instructor could provide a sentence and ask the student to specify the parts of the sentence. The processor would be able to provide the answer automatically. If designed with this in mind, the plugin might even be able to generate sentences tailored specifically to the individual student. It could use the student’s name, already in ITEAMS, and if ITEAMS tracked other personal information, the plugin could use this to generate unique sentences specifically relevant to this student, rather than generic sentences of little or no interest to the student.
5.2.2  A Need For Flexibility

Along with a design that is clearly general in nature, ITEAMS must also be highly flexible within the plugin system. Some degree of flexibility has already been demonstrated in the two different methods by which software can be accessed: directly or through an application wrapper. There are other methods as well for building flexibility into the plugin system. These include the use of TCP/IP as the communication medium, the Java API for creating plugins and plugin user interfaces as well as the use of an XML data format. Each of these design features makes for yet another degree of flexibility and usability of the system.

TCP/IP is used because most languages provide a way to make a socket connection. TCP is an easy and reliable form of communication over sockets. Languages from C to Prolog to Perl all use some form of socket connection and communication. By using TCP/IP, the application being interfaced does not need to be running on the same machine.

Since each plugin must be designed to use a graphical interface that allows exercises to be configured by an instructor, as well as offer a view with which the student can interact, ITEAMS lets users develop these in any language. They must be able to be run or be started by ITEAMS and interact via a TCP connection. In order to make a developer’s job easier, a Java API was created that provides most of the low-level functionality such as communication and a standard GUI that needs only to be populated with specific details.

In order to standardize the communication of data between ITEAMS and the plugin as well as between the plugin and the external software, an XML data format was created that allows for easy parsing of data. This format uses a hierarchical structure for the information that needs to be sent from one application to another. The format uses this hierarchical structure for:

- application configuration
- exercise configuration
- exercise creation
• completed exercise information
• application exiting

The application configuration allows ITEAMS to provide external software with configuration information if necessary. This information can be anything from loading a file setting permissions, and setting preferences. Once an application is configured it is ready to be used in an exercise. The application accepts information from the plugin for the exercise to start, and once the exercise has completed any processing, which is up to the software or designed into the plugin, any information that is needed for evaluation is sent back to the plugin.

Plugins are generally used for assignment exercises. Since an exercise could require instructor-provided file or information in order to work correctly, the plugin system allows for storing this information and for transmitting this information to the software through the XML format. The other information that needs to be sent to the plugin from ITEAMS is the list of instructional categories. This is sent so that the instructor will be able to specify how the exercise relates to the rest of the material in the teaching module. This information is automatically provided by ITEAMS when a plugin is run or started.

Before an exercise can be configure and worked by a student, an instructor must first create the exercise and set up the correct parameters for evaluating the student. The parameters for the exercise are stored in the XML format and come in several varieties. The first is the exercise text which tells the student what to do or what to answer. The second is possible answers and their associated correctness value. Another parameter is whether or not the student’s answer requires processing by the application. These parameters allow the instructor to design exercises for which the student provides an answer and the answer is transmitted to the application. Once the answer is sent to the application, the application uses it in whatever manner and returns some value back to the plugin. The returned value is then checked for in the data ranges to determine a correctness value.
These data ranges are specified by the instructor and allow for correlation of data that is returned from the external software.

Consider, for example, a plugin for a robotic simulator. The simulator provides a test bed for designing robotic agents. Some information that a simulator might collect on an agent includes number of collisions, average speed, distance traveled, average distance from objects, etc. From this information other data can be found such as average distance between collisions (distance over collisions) and average time between collisions ((distance / speed) / collisions). The plugin can be designed to generate this information, or the information can be generated by the application. If all of this information is sent back to the plugin after an exercise has finished being processed by the external software, then these and other data can be related to the data ranges. This allows an instructor to set not only ranges of correctness for single data values, but also for groups of data. Thus in the robotic example, the instructor might ask the student to design a wall-following agent and to evaluate the agent’s performance with given sets of data ranges for number of collisions and average distance away from objects.

Once an exercise has been completed by a student and evaluated by the plugin, the student’s answers and answer values must be sent back to ITEAMS for storage in the student model and also for updating the belief networks that evaluate the assignments and quizzes. This information is sent back in the XML format.

When an exercise has been completed and the plugin is designed for single tasks, the application may need to be exited. In this event, the plugin sends a termination code or value which tells the application to exit.

5.3 Protocol Design Considerations

When designing the plugin interface, a number of different methods of connection were considered. In the end TCP/IP was chosen as the implemented method. Others were
considered. There follows a brief discussion of some of the rejected connection methods.

5.3.1 Remote Procedure Calls

Remote Procedure Calls can be a very efficient way of communicating in a distributed computing environment. Since plugins are not required to interact with applications on the same host, a distributed computing environment exists. Not all languages provide the ability to make remote procedure calls. Because of this ITEAM does not allow remote procedure calls as the communication method.

5.3.2 Remote Method Invocation

Remote Method Invocation provides a way for applications to interact with each other in a distributed computing environment which ITEAM can have when using plugins. In designing for generality and flexibility, the plugin system allows any programming language to be used for designing plugins, application wrappers, and application interfaces. Generally, remote method invocation is a language-specific task. Thus everything must be written in the same language, which does not conform to the design goals here.

5.3.3 UDP

Since there is a need to communicate in a distributed environment and possibly in different programming languages, a protocol is needed that is universal. UDP could be used for the transmission of data between plugins and the applications because it is everywhere. It is a format that is easy to use and one of the most common forms of communication on the Internet. However, UDP is currently simply unreliable. There is no guarantee that the packets will arrive at the destination address, and, if they do arrive, that they will even be in order. This aspect of UDP eliminates it as a possible choice.
5.4 Why TCP/IP

ITEAMS needs a protocol that is everywhere, easy to use, and reliable. TCP/IP fits that definition perfectly. It has proven to be a reliable form of data transmission as well as being easy to use. This is especially the case since most languages provide the functionality and packet forming for the programmer. Any language that can access sockets from the operating system can use TCP.

TCP is not tied to any operating system. It is platform independent. At the same time it is not tied to any language and can be used across languages. The flexibility of TCP works perfectly with this flexibility of the ITEAMS plugin interface.

The reliability of TCP is another plus for its use with the plugin. The packets that are sent will arrive at the destination, possibly out of order, and possibly after being retransmitted, but the packets will arrive.

5.5 Designing for Single Tasks

Plugins can be designed to be used for either multiple tasks or for single tasks, such as numerical evaluation. In this current context single tasks will be the focus. The next section will discuss multiple tasks.

Using a very simple example, consider a Dr. Scheme plugin designed for the task of evaluating arithmetic expressions. The example may be overly simple, but it does help to show the design methodology. In this example the plugin is designed in a ‘Task Specific Methodology.’ This methodology has plugins designed for one and only one task. In this example, the plugin will only take numerical expressions in scheme and evaluate them within Dr. Scheme. The plugin allows the instructor to input the question text, the correct answer, data ranges, and associated correctness values for the student’s answer.

The design of a plugin in the ‘Task Specific Methodology’ is straightforward. Only the minimum amount of design is required for the plugin to be functional. An instructor
or plugin designer needs to specify the task for the plugin, then define what information the instructor must specify when creating the specific instance of the exercise. After this has been defined, the format of the answer by the student should be determined. If it is multiple choice, then the designer needs to provide the ability to enter possible answer choices. If it is fill-in-the-blank, then a form of evaluation needs to specified. Once the general format of an exercise has been designed and implemented, the plugin can be used.

Since the plugin is being designed for one task only, it is easier to design than designing for multiple tasks. Multiple tasks require more design features and more design time. The single task methodology provides a quick way to get a plugin up and functional so that students can begin using the external software. The full functionality of the external software is generally not utilized in this design style because the plugin is designed for a single task which can be very simple.

Since only a single task exists for the plugin, the ability to combine tasks and correlate data is lacking. However, this may not always be the case. Some single tasks, ones that are complex, might require multiple pieces of data to be returned from the application, and that data can be correlated better than if only one piece of data is to be returned. On the other hand, the small amount of data being sent back to the plugin after an exercise could conceivably produce incorrect results. This issue requires further study and resolution.

5.6 Designing for Multiple Tasks

Plugins can also be designed for multiple tasks. This means that the plugin can take full advantage of all of the features of an external application or just some of them. It also means that the plugin can take advantage of all of the information that is passed back from the application. Earlier the ‘Task Specifi Methodology’ was introduced for single tasks. When designing for multiple tasks, the designers use what is referred to as the ‘Application Specifi Methodology.’ This methodology uses a simple concept: the plugin
is designed to take advantage of multiple features of the external application.

Modifying the Dr. Scheme example from above, the plugin can be designed using the ‘Application Specifi Methodology.’ The design of the plugin for multiple tasks is also straightforward. An instructor or plugin developer needs to specify what tasks the plugin will support. This requires knowledge of the external software so that an instructor can design an exercise that enhances the learning process of a student. Once the tasks are defined a process is used which is similar to the process for designing for a single task.

That is, the designer must provide the following information for each task:

- define what information needs to be specific by the instructor when creating the specific instance of the exercise
- the format of the answer that the student provides
- what information needs to be transmitted back to the plugin for evaluation

The only additional information required for multiple tasks, information not required for the design of single task plugins, is in the information that is transmitted back to the plugin. This information allows an instructor the freedom and flexibility to select what information applies to each type of task. When creating an exercise for the plugin, the instructor selects what information is needed to evaluate the student and obtain a correctness or performance value from the data ranges. In the example of the Dr. Scheme plugin, the plugin has the ability to evaluate any scheme expression or program. The plugin could allow the student to supply procedures and other definition needed to code to execute correctly.

Since the plugin is being designed for multiple tasks, more time and thought are required for its design and development. Overall, the time required to design and develop an application-specific plugin is greater than for designing a single-task plugin. However, when complete, more tasks are available to the instructor. This design methodology may be better suited for a course that uses one application very often.
This design methodology can facilitate a more robust evaluation, by reason of the larger set of evaluation criteria, than when designing for single tasks. In general, multiple tasks should offer a larger set of parameters or data to be sent back to the plugin for evaluation purposes. It is this higher volume of returned data that supports the assumption of a more thorough evaluation.

Even though, for the stated reasons, the evaluation is more robust, there is increasing difficulty in selecting a relevant subset of criteria as well as relevant values for the data ranges. As the number of evaluation criteria combinations increases, the task of choosing a relevant subset grows exponentially. An instructor can simplify evaluation by focusing only on just a few of the criteria, but one must first decide which few are relevant. If there are only two possible criteria with which to evaluate a student, then there are only two possible choices for constructing the data ranges. When there are four criteria, the number of choices goes to eight possible choices. This is one drawback to using the application-specific methodology. Moreover, choosing a relevant set of evaluation criteria grows more difficult as the total number of evaluation criteria grows.

As the number of criteria for data correlation grows, the overall influence of each can decrease. As many different evaluation criteria are chosen, the overall impact of an individual will decrease with each additionally selected criterion. The reason for this is that each of the selected criteria influence the student’s performance equally. If there are twenty different evaluation criteria and an instructor selects ten of them to represent the combinations in the data ranges, each individual criterion has less of an impact on the student’s overall performance than if there were fewer criteria in the combination. If, for example, the instructor had chosen only five criteria, each would then have twice as much influence on the student’s performance than in the example where ten criteria were chosen.
5.7 The Generic Plugin API

The general ITEAMS Plugin API is now presented. In this section, the general features of the API will be discussed. We will also describe how these features can be used. This last part is accomplished through the design of a simple plugin. This plugin will be for the Dr. Scheme programming environment. Two different plugins will be presented. One will be designed using the 'Task Specific Methodology' and the other using the 'Application Specific Methodology.' Both of these will be designed using only provided layouts, screen configurations and plugin features in the API. There are no extended features in these plugins. All screen shots show what a general plugin looks like after the plugin specific data has been added.

In the design of each plugin, many features will function in the same manner. As a result there will not be a distinction made as to how the plugins are designed. Only one will be discussed; the other is implied. Throughout this section, the person involved with the design of the plugin will be referred to as the designer.

The Plugin API provides a generic administration tool to create an exercise, a generic student tool which allows the student to work on an exercise, and an easy way to extend and modify these provided tools. The administration tool breaks the exercise creation into three different groups, namely, general information, specific information, and data correlation. The student tool is a simple screen which presents the exercise text and gives the student a place to enter or select an answer. The easiest way to demonstrate what this API offers is to work through an example.

If this API is used, all of the communication is already handled. There is no need for a designer to rewrite code that generates the XML for storage and communication. The designer also does not need to be concerned about making the connection to ITEAMS or external applications. The only part the designer must write is the code for the application that accepts the connection from the plugin and processes the information sent to the
application, then write the code that will encode and send the necessary information back to the plugin.

5.7.1 Example Plugin using API

The Java I-TEAMS Plugin API provides a designer with several basic features which can be utilized and expanded for creation of a plugin that will meet his/her needs. The API utilizes an administrative screen which needs to be populated by the designer. When used to create an exercise, this screen provides the instructor with access to all the parameters of the plugin. It allows for the creation of data ranges, selection of evaluation criteria, selection of instructional categories, and provision of exercise specific data.

The API uses several screens which allow an instructor to configure a specific instance of an exercise. These screens break up the creation of an exercise into three functional screens:

1. Provide general exercise information
2. Provide specific exercise information
3. Provide data range correlation for evaluation

The general exercise information includes the evaluation criteria, extra exercise specific information, and the connection method. The evaluation criteria must be provided by the designer of the plugin. Once these have been determined, the instructor can select the data ranges. The designer then needs only to add the evaluation criteria to the plugin and determine the types of ranges that are acceptable. In the scheme examples the application-specific plugin sets the following evaluation criteria:

- Returned Value
- Running time

The task-specific plugin uses only the 'Returned Value.' The 'Returned Value’ can be in either numeric or string format. It allows the plugin to check the value returned by Dr. Scheme, which is any output that evaluated code generates, against the values in the data
Figure 5.1. The main screen in the Dr. Scheme Plugin. This screen shows where general information is set.

range. The connection method of the plugin is selected in this screen as well. The two choices are local host or any host. If local host is specified then ITEAMS only tries to connect to the ITEAMS plugin port on the local host. When any host is selected, a remote host can be designated. If none is designated, then the plugin will ask the student for the location of the application. When one is designated, the plugin will only try to connect to that host on the ITEAMS plugin port.

The screen where these functions are determined in the Dr. Scheme program are shown in figure 5.1. This shows the selection of the returned value as the evaluation criterion and two data ranges. There is no extra information that an instructor provides. Thus, part of the window is blank. The second screen shows where specific exercise data is selected. This consists of the answer type, the answers to be provided to the student if any will be presented, and the creation of the exercise text. In the example plugins, this is the screen where the instructor has the opportunity to specify the answer type that the
student provides from any of the following:

- short answer or code
- single scheme expression
- multiple choice

The ‘short answer or code’ allows students to write code and have it evaluated or to provide as an answer the output of a given scheme expression. The single scheme expression allows the student to provide a single expression which will be evaluated and checked against a correct expression. The multiple choice answer is just that: multiple answers from which to choose. The question type can be selected from the list shown in figure 5.2. The question type corresponds to the type of answer selected. In this screen the instructor specifies what instructional categories pertain to the exercise. Recall that if none are selected, it is the same as if they are all selected, since the material has to relate to at least one of the categories. The screen used to create the exercise question is similar to
that used to create a quiz question. The only difference is that different answer types are available to the exercise.

The final screen provides the plugin with the ability to correlate the evaluation criteria selected in the first screen. This gives the instructor the means with which to specify how a student performs on an exercise based on the information that comes back from an external application. Figure 5.3 shows this screen.

After all of the information has been entered by an instructor and the exercise is configured it is sent to ITEAMS. When a student is working on an assignment and encounters a plugin exercise, the exercise is loaded into the standard exercise viewer (figure 4.14). The students’ view of the plugin exercise is no different than a regular exercise. Internally, though, ITEAMS configures the exercise to communicate with the correct plugin. ITEAMS then routes all student actions to the plugin while waiting for the plugin to provide the evaluation of the student.
The ‘task-specific’ example plugin requires that Dr Scheme be on the local machine. The ‘application-specific’ plugin connects to a specific host on the ITEAMS plugin port. This host can be the local host or a remote host.

5.8 Chapter Summary

This chapter has offered an overview of the design of the ITEAMS Plugin Interface and a description of the Java ITEAMS Plugin API. The plugin interface provides ITEAMS with a standardized way of communicating with external software for the purpose of enhancing a student’s learning experience. The plugin interface creates a platform for automatically grading students’ exercises when working on assignments. The plugin interface allows communication in an XML format through a TCP/IP connection. This XML format provides ITEAMS and the plugin with the following features:

- application configuration
- exercise configuration
- exercise creation
- completed exercise information
- application exiting

A plugin can either allow an interface to an external software or be a stand-alone application that does not need to interface any external software. The information presented to the student through a plugin has two components: what the exercise is that the student must answer and the type of answer that is provided. The information presented to the student is presented as text. The exercise answers can be in any format the instructor prefers, as long as that format is allowed by the plugin.

There are two basic designs for plugins. The first and simplest method is to design the plugin for a single task, which is to be performed in the external system. This method may lack the ability to correlate data correctly, but this method can be designed more easily and more quickly. The second method is to design the plugin to use the full capabilities of
the external software. This method is more complicated and requires more time to design. This method also offers better data correlation, but, as more data is grouped together, the less each piece actually influence the student’s evaluation.
Belief Networks consist of a set of nodes which represent propositional variables. These nodes are contained in a directed acyclic graph. The links or edges between nodes represent a probabilistic causal relationship between nodes. In the absence of a link between nodes, two nodes could be conditionally independent if the values of their parents are known. The set of nodes and edges describes the topology of the network, but that is not all that is needed to fully describe a belief network. In order to fully describe a belief network two more items must be known:

1. the prior probabilities of each state in the root nodes
2. the conditional probabilities of each state of non root nodes given all possible values of their parents.

Once all of this is known a belief network represents the joint probability distribution of all nodes. [38, 43]

A good example of a belief network is given by Russell and Norvig [38]. This example shows a belief network that represents an Alarm, fig 6.1. The nodes in this network are binary which means two things: first that the value \( \neg P(A) = 1 - P(A) \), and second that each node has only two values: **True** or **False**. The network shows that the alarm node has two parents and that they are the root nodes for the network. The full list of prior probability for the roots are shown in Table 6.1.
Figure 6.1. A belief network that contains all the conditional probability tables. The letters $B$, $E$, $A$, $J$, and $M$ represent the names of the nodes that begin with that letter. The nodes are boolean which means that the value $\neg P(B) = 1 - P(B)$.

A node consists of a conditional probability table, a list of parents, and a list of nodes. The values in the conditional probability table can be used to determine the likelihood that the node will hold that value. Belief networks are most beneficial when these values are changed, such as when a node is instantiated. The change in value propagates through the network from the node whose value changed. Since belief networks are directed acyclic graphs, there is no worry that when an update to a node occurs that the propagation of changes in the network will not stop because the change is being propagated down through children. That is, since there are no cycles in the network, the propagation of values will stop when it reaches the leaf nodes after going through the rest of the network. There is a possibility that the update of nodes will never end if the network has multiple connections. The reasons for this are discussed later in the chapter.

Now that the structure and content of belief networks has been discussed, it is important now to look at some of the formulas used with belief networks. There are several
TABLE 6.1

TABLES CONTAINING THE PRIOR PROBABILITIES OF THE TWO ROOT
NODES IN THE ALARM BELIEF NETWORK.

<table>
<thead>
<tr>
<th>Burglary Node</th>
<th>Earthquake Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(Burglary)</td>
<td>P(Earthquake)</td>
</tr>
<tr>
<td>0.999</td>
<td>0.998</td>
</tr>
<tr>
<td>¬P(Burglary)</td>
<td>¬P(Earthquake)</td>
</tr>
<tr>
<td>0.001</td>
<td>0.002</td>
</tr>
</tbody>
</table>

formulas that are key to understanding belief networks. The first of these is the general
formula for calculating a joint probability in the network. Recall that the joint probabil-
ity distribution is the set of probabilities for graph $G(V,E)$ which represent the specifi-
probability given a specific value for each node contained in $V$. This is normally written
with the following notation: $P(x_1,x_2,\ldots,x_n)$ where the $x_i$’s are the specific values that
 correspond to the equivalent $X_i \in V$. The chain rule can be applied to calculate any joint
probability given a set of values. That formula is

$$P(x_1,x_2,\ldots,x_n) = \prod_{i=1}^{n} P(x_i|\text{Parents}(X_i))$$

(6.1)

The values for $P(x_i|\text{Parents}(X_i))$ are stored in the conditional probability tables of each
node. Take for example the conditional probability table for the Alarm node in figure 6.1. The parents of the node are Burglary and Earthquake, both of which have
two states to which they can be set. One conditional probability for the Alarm is if the
alarm goes off and there is no burglary and there is an earthquake which is stated as
$P(\text{Alarm}|\neg\text{Burglary} \land \text{Earthquake}) = 0.29$. Now consider if someone wanted to know
the probability of John calling and Mary not calling when the alarm goes off and there is
no burglary and there is an earthquake. This joint probability is calculated as follows:

\[ P(J \land \neg M \land A \land \neg B \land E) = \]

\[ P(J|A)P(\neg M|A)P(A|\neg B \land E)P(\neg B)P(E) = \]

\[ .90 \times .01 \times .29 \times .999 \times .002 = 0.00000521478 \]

Many researchers have worked with optimizing inference algorithms and belief networks. Inference algorithms are what perform updates and queries on a belief network. In a general belief network it has been shown that inference is NP-Hard [11]. Part of the motivation behind belief networks is to use the conditional independence specified by a network topology to avoid the combinatorial explosion of calculating probabilities in a joint probability distribution. Algorithms have been created to take advantage of these topological properties. In general there are two topologies that are considered: Singly connected topologies and Multiply connected topologies. These two topologies are discussed further below.

Besides singly connected and multiply connected networks, there is another type of belief network that is of interest: dynamic belief networks. These networks can be broken into two groups. The first group is Temporal Networks. These networks model change over time. They allow inferences to be made about the past, present, and future. The other type of dynamic belief network is Dynamic Networks. These networks are used to model dynamic systems and provide a discrete version of the system that is being modeled which can be evaluated over time. In both types, time is a factor. Temporal networks are models that keep state over time by modifying the network structure. Dynamic networks model change over time.

Both regular belief networks and dynamic belief networks have been used in an many different fields. Regular belief networks have been used in quite a few areas including poker games [25], searching [23], Intelligent Tutoring Systems [1, 4, 13, 18, 22, 24, 34, 39,
general decision making [42], prediction and diagnosis [2,21,43], and robotics [46], to name just a few.

Much effort has gone into trying to improve the efficiency of performing updates and queries in a belief network. It has been shown to be NP-Hard for arbitrary networks [11]. Because of this, algorithms for performing these tasks have been specialized to work very efficiently on certain types of networks. There are two types of algorithms which are used: Exact algorithms and Approximate algorithms. These algorithms are topologically restrictive; that is the algorithms have been designed to perform inferences that take advantage of network properties. This is done so that there is no computational explosion which makes general inference NP-Hard [11]. Exact algorithms are able to calculate the exact value of the nodes in a network when a node is updated. Approximate algorithms are not necessarily able to calculate the exact value of a node when an update occurs. Some of the algorithms provide convergence to the exact value, but in large networks the values will never converge; they will only get close. Smaller networks, generally used for trivial information and test cases, do provide convergence but lack significance [27,30].

For singly connected networks there are many different algorithms [5], but the algorithm that most researchers use or extend to meet their needs is an algorithm that Pearl developed in the late 1980’s [37]. All of the algorithms for this type of network are Exact algorithms.

Multiply connected networks cause more problems because the same algorithms that are used for singly connected networks will not work. This is because the networks contain multiple paths between two nodes which causes a loop in the inference algorithm which would never end the computation. This has been shown in Pearl’s work [37] among others [5,20,27,30]. In short, the singly connected algorithms are recursive and walk through the network in both directions from each node updating the nodes as they are
introduced. Only when a node is updated will it start walking through the network, thus when there are multiple paths one node will be reached twice for the original update. When a node is reached the second time, the algorithms all act as if the second update is a completely different update. The cycle then continues from the beginning and will continue to restart forever. Approximation algorithms take different approaches to solving this continual update process through techniques such as clustering, conditioning, and stochastic simulation. Each of these techniques are discussed later in this chapter.

Now that belief networks have been presented and some background information discussed, ITEAMS’ use of belief networks can now be discussed. What can be evaluated in ITEAMS is presented first. This is followed by what is evaluated. The algorithms used to perform updates are then presented. Included in this is a discussion of an algorithm which ITEAMS uses in certain cases to determine which algorithm to use on a network.

6.2 ITEAMS Design Considerations

ITEAMS is designed to take advantage of information contained in belief networks. ITEAMS uses a combination of Bayesian networks to assess the student’s performance and knowledge level. It uses three types of belief networks to evaluate, assess, and progress the student through the material. The types of belief networks used are

- singly connected networks
- multiply connected networks
- structurally dynamic networks.

Each type of network is required for different situations and at any given moment information will be obtained from at least one of each type of network. One goal of ITEAMS is to assist in the learning process of a student; hence, the time required to perform queries and update the network based on students’ actions must be minimal and efficient. In order for each type of network to meet these requirements most effectively, it is important to
consider how and why each type of network is used. Before this is done, one needs to consider what is evaluated in ITEAMS.

6.2.1 What can be evaluated in ITEAMS

ITEAMS has the potential to evaluate just about every aspect of a user. The term user refers to both students and instructors. More important is the student, but there is the potential to evaluate the instructor as well. The different information that can be evaluated includes

- Student Performance
  - Overall Performance
  - Categorized Performance.
- Teaching Modules
- Categories
- Sections
- Quizzes
- Assignments
- User Performance

Each of these can be evaluated to enhance the learning experience of a student.

Student Performance can be broken down into two different items. The first is overall performance. This ties directly into teaching modules. The overall performance is how a student is performing on all of the sections, quizzes, and assignments in a given teaching module. The second is categorized performance. Individual categories or groups of categories can be evaluated to give an instructor a better idea of where a student is having trouble. As an example, if students are performing ‘well’ overall, they might be struggling on one specific category.

ITEAMS can evaluate most of the instructional objects it uses. On the highest level is the teaching module. This can be evaluated to give an overall assessment of a student. It can also be used to determine which sections a student should review or work on next.
In order for ITEAMS to give an instructor the most useful information it can about a student, instructional categories can be evaluated. An instructor can gain a better understanding of a student’s strengths and weaknesses through the evaluation of instructional categories. By evaluating categories, ITEAMS can also direct the student to review different sections. Similar to a teaching module, evaluating categories can be used to determine what needs to be reviewed. They differ in that a section that the categories recommend for review might be completely different from the one that the teaching module recommends. This is because the section recommended by the teaching module might not be directly related to the category(s) in which the student is struggling.

Sections can also be evaluated in ITEAMS. The benefit of this is that ITEAMS can determine what part or parts of the section need reviewing if necessary. The other reason ITEAMS would evaluate sections is that, if a student is doing very well, ITEAMS can have the student skip a goal or a quiz. This could be done if the student is performing at a very high level.

Quizzes are another option for evaluation. The evaluation of quizzes can help to determine how many questions a student should be given. It also can determine which question is to be presented next to the student.

Assignments are very similar to quizzes. Since assignments are broken into sections that have their own exercises, each section can be evaluated separately. This means that assignments can also determine if an assignment section is necessary for a student, which is similar to how sections can be evaluated.

User Performance is different from the rest. ITEAMS has the ability to monitor the user’s efficiency at using ITEAMS. In theory, ITEAMS can provide context specific help dialog to a user who is having trouble using the software. Whether the user is a student or an instructor, the actions in the software can be monitored.
6.2.2 What is evaluated in ITEAMS

Given the scope of the project, only certain items could be evaluated. Out of the list above, the only one not evaluated in some form is ‘User Performance.’ The others are evaluated based on their specific needs. Some of the items evaluated require more than one network. A discussion of each network used follows. First to be discussed is teaching modules, then categories, followed by sections. The discussion ends with quizzes and assignments.

Teaching Modules

ITEAMS can use either a singly connected or a multiply connected belief network depending on the structure of the network. The nodes in the network consist of the sections and non associated quizzes that are graphically depicted in a teaching module. The teaching module in figure 6.2 shows an example of a teaching module. It also shows the network structure that is used for the evaluation. In this example the network has one root node, which is a pre-test. The rest of the network consists of the sections shown.

<table>
<thead>
<tr>
<th>Data Range</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 = Performance</td>
<td>100%</td>
</tr>
<tr>
<td>99.5 ≥ Performance ≤ 89.5</td>
<td>89.5%</td>
</tr>
<tr>
<td>89.0 ≥ Performance ≤ 79.5</td>
<td>79.5%</td>
</tr>
<tr>
<td>79.0 ≥ Performance ≤ 73.5</td>
<td>73.5%</td>
</tr>
<tr>
<td>73.0 ≥ Performance ≤ 69.5</td>
<td>69.5%</td>
</tr>
</tbody>
</table>

Since the network consists of belief nodes that represent either a quiz or a section, there needs to be some way to define the values of the network. In this case, there is a
grading scheme used for each node. That is, each node’s conditional probability distribution consists of the five values shown in Table 6.2.2. The values initially take on the minimum value of the data range and will slowly change as the nodes are instantiated. When a node is instantiated through a student completing a quiz or a section, the student performance in that item is used as the data value. That is, if a section is finished and the student’s performance in the section is at 86%, then the value of the node is 86% and the value that is set is the third data range, $89.0 \geq \text{Performance} \leq 79.5$.

Given that ITEAMS could use a multiply connected network, and multiply connected belief network algorithms are approximate algorithms, it would not be good to use only a multiply connected network to model the teaching modules. ITEAMS uses an algorithm
discussed below to determine which network to create based on the teaching module structure.

Categories

Categories are used to evaluate the performance of a student in a teaching module. Each category has its own network which is used to determine the student’s level of performance. The network that is used for categories is the Structurally Dynamic Belief Network. This network grows over time to accommodate the introduction of new nodes over time. This type of network is discussed more fully later in this chapter. The category network starts off with one node which represents the instructional category. As a student works on quizzes and assignments, the questions and exercises presented to the student are added to the network.

Sections

The network for a section is created as a multiply connected network. The network is created by combining the goals, quizzes, and assignments contained in a section into a network. The network is automatically generated by ITEAMS. ITEAMS uses a simple algorithm for generating this belief network. The algorithm generates one network layer for each goal in a section. The top layer is the first goal and the bottom layer is the last goal. A layer of the network is generated by a node for each difficult level that exists for a goal. Since there are only four difficult levels in ITEAMS, there can be at most four nodes in each network layer. The links in the network layer are determined by placing a link from one difficult level to the one higher. Thus difficult level one is linked to level two, level two to level three. Causal links are also set between difficult levels in the next layer. The first difficult level in layer one is connected to the first difficult level in layer two. The first difficult level in layer one is also connected to the second difficult level in layer two. This pattern continues until all the links are set. These links are set since the
student is only able to move up one difficult level between goals. This is done so that the student does not jump too far ahead. If the section has a quiz and/or goal associated with it, then the last goal comes before the nodes that represent the quiz and assignment.

Quizzes and Assignments

ITEAMS uses an approach similar to the Mini-Bucket Heuristic Search [23] to select the next question for both quizzes and assignments. For a quiz the question nodes are sorted into buckets based on percentage correct. There are four buckets in ITEAMS which are equivalent to the top four data ranges in table 6.2.2. Once the questions have been sorted, the first non-empty bucket with the highest data range has a question randomly selected as the next question. If the first data range is empty and second data range has four questions in it, then one of those four questions will be randomly selected as the next question. The same process is used for Assignments. For each assignment section, the next exercise is selected by sorting the nodes into buckets and randomly selecting an exercise from the highest populated data range bucket. After a question or exercise has been given to the student it is removed from the list of available questions or exercises. This keeps the same question from being asked over and over again during the same ITEAMS session.

6.3 Singly Connected Networks

Singly connected networks are the simplest type of belief network to use. The networks are DAGs which have at most one path between any two nodes and can have multiple root nodes in the tree. These networks are trees, and when there are multiple root nodes present they are also referred to as polytrees. For simplicity the term ‘singly connected’ will be used to refer to these networks.

There are many different types of updating and inference algorithms that are available for singly connected networks [5]. The algorithm ITEAMS uses is a variation [32] of
Pearl’s Message Passing Algorithm [37].

6.3.1 Pearl’s Message Passing Algorithm

The algorithm works by passing messages throughout the network as the means of updating variables. It is a recursive algorithm. There are several items that need to be define before the algorithm is presented. Given a DAG $G = (V, E)$ and let $a$ be a set of values for a subset of nodes $A \subseteq V$, define the following for each node $N$:

**$\lambda$ messages** For each child node $Y$ of node $N$, for all values of $n$ the $\lambda$ message $\lambda_Y(v)$ is defined as:

$$
\lambda_Y(v) \equiv \sum_y \left( \sum_{W \in \text{Parents}(Y)} P(y|n, W) \prod_{w \in W} \pi_Y(w) \right) \lambda(y)
$$

**$\lambda$ values**

- If $N \in A$ and $N$’s value is $\hat{n}$
  $$
  \lambda(\hat{n}) \equiv 1 \\
  \lambda(n) \equiv 0 \text{ for } n \neq \hat{n}
  $$
- If $N \notin A$ and $N$ is a leaf, then all values of $n$
  $$
  \lambda(n) \equiv 1
  $$
- If $N \notin A$ and $N$ is not a leaf, then all values of $n$
  $$
  \lambda(n) \equiv \prod_{U \in \text{Children}_N} \lambda_U(n)
  $$

**$\pi$ messages** If $Z$ is a parent of $N$, then for all values $z$,

$$
\pi_X(z) \equiv \pi(z) \prod_{U \in \text{Children}_Z-N} \lambda_U(z)
$$

**$\pi$ values**

- If $N \in A$ and $N$’s value is $\hat{n}$
  $$
  \pi(\hat{n}) \equiv 1 \\
  \pi(n) \equiv 0 \text{ for } n \neq \hat{n}
  $$
- If $N \notin A$ and $N$ is a root, then all values of $n$
  $$
  \pi(n) \equiv P(n)
  $$
- If $N \notin A$ and $N$ is not a root, then all values of $n$ and all values of all parents $Z$
  $$
  \pi(n) \equiv \sum_{z \in Z} \left( P(n|Z) \prod_{z \in Z} \pi_N(z) \right)
  $$
From these definitions it is possible to calculate $P(x|a)$ as

$$P(x|a) = \delta \lambda(x) \pi(x)$$

where $\delta$ is a normalizing constant. This works because of the definition above. To show that this equation works, recall that to calculate a belief in a Bayesian network one needs to apply the formula

$$P(x|E) = P(X|N_X,D_X)$$

where $N$ are the parents of $X$ which are evidence and $D$ are the children of $X$ which are evidence to obtain $P(X|E)$. Using Bayes Rule we can obtain the following formula:

$$P(x|E) = P(N_X|X)P(D_X|X)P(X)$$

This can be reduced further to

$$P(x|E) = \frac{P(D_X,N_X|x)P(x)}{P(D_X,N_X)} = \frac{P(D_X|x)P(x|N_X)P(N_X)P(x)}{P(x)P(D_X,N_X)} = \delta P(D_X|x)P(x|N_X)$$

$P(D_X|x)$ is the product of the probabilities of all the children of $X$ which can be stated as

$$P(D_X|x) = \prod_{U \in \text{Children}(X)} \sum_{y} \left( \sum_{W \in \text{Parents}(Y)} P(y|n,W) \prod_{w \in W} \pi(y,w) \right) \lambda(y)$$

From the definition above $\lambda(x)$ is calculated with

$$\lambda(x) = \prod_{U \in \text{Children}(X)} \sum_{y} \left( \sum_{W \in \text{Parents}(Y)} P(y|n,W) \prod_{w \in W} \pi(y,w) \right) \lambda(y)$$

thus

$$P(D_X|x) \approx \lambda(x)$$
The same reductions can be made to show that

\[ P(x|N_X) \approx \pi(x) \]

for a complete proof of the equations see either reference [32, 37]

The algorithm used to update the network is now presented, in the algorithm \( \lambda_Y(X) = \text{lambda}(Y)(X) \) and \( \pi_X(z) = \text{pi}(X)(z) \):

\[
\text{update}(\text{Network } G, \text{ Evidence Variables } E, \\
\text{Evidence Values } e, \text{ Variable } V, \text{ value } v) \{
\begin{align*}
&\text{Add } V \text{ to } E; \\
&\text{Add } v \text{ to } e; \\
&\text{for (each } v \neq v \text{ in } V) \{
&\quad \lambda(v) = 0; \\
&\quad \pi(v) = 0; \\
&\quad P(v|e) = 0;
&\}
&\text{for (each parent } Z \text{ of } V \text{ such that } Z \text{ is not in } E) \\
&\quad \text{send lambda message } (V, Z); \\
&\text{for (each child } C \text{ of } V) \\
&\quad \text{send pi message } (V, C);
&\}
&/ / Y sends a lambda message to X \\
&\text{send lambda message } (\text{Child } Y, \text{ Parent } X, \\
\text{Evidence values } e, \text{Evidence Variables } E) \{
&\text{for (each value } x \text{ of } X \{
&\quad \text{calculate lambda}(Y)(x); \\
&\quad \text{calculate lambda}(x); \quad / / \text{compute } X's \lambda \text{ values}
&\}
\end{align*}
\]
c a l c u l a t e P ( x | e );

} normalize P ( x | e );
for ( each parent Z of X where Z not in E )
    send_lambda_message ( X, Z );
for ( each child C of X where C != Y )
    send_pi_message ( X, C );
}

// Z sends X a pi message
send_pi_message ( Parent Z, Child X ) {
    for ( each value z of Z )
        calculate pi ( X ) ( z ); // send X a π message
    if ( X is not in E ) {
        for ( each value of x ) {
            calculate pi ( x );
            calculate P ( x | e );
        }
        normalize P ( x | a );
        for ( each child Y of X )
            send_pi_message ( X, Y );
    }
    // do not send λ messages to X’s
    // other parents if X and all of X’s children
    // are uninstantiated
    if ( not lambda ( x ) == 1 for all values of x )
In the send_pi_message function of the algorithm, the comments that say not to send λ messages to X’s other parents if X and its children are uninstantiated. This is because X separates its parents from each other. If no children of X are instantiated, then from the algorithm all of X’s λ values are all still 1 [32].

The algorithm works by passing π messages and λ messages through the network. π messages are sent to children of a node which update all of the children when a node’s values changes. λ messages are sent to the parents of a node whose values have changed. To use the algorithm, the network does need to be initialized. The λ values, λ messages, π values, and π messages are initialized to 1. The root nodes π values are initialized to the roots values, π(r) = P(r) where r is value of a root. The probabilities of the roots need to be initialized; they are set to P(r|e) = P(r).

6.4 Multiply Connected Networks

A multiply connected network, figur 6.3, is a network where there is more than one distinct path between two nodes. This means that one variable can influence another through more than one causal link. Consider an example taken from [38]. It shows a multiply connected network where the presence of clouds influence whether the grass is wet through both the rain node and the sprinkler node, figur 6.4.

6.4.1 Available Inference Methods

There are numerous implementations for each type of approach for approximate inference. These algorithms are not presented here since it is out the scope of the paper. Only
the algorithm used by ITEAMS is presented. The types of algorithms are discussed with their advantages and disadvantages.

Conditioning

Conditioning is the first approach that can be used. This method finds what nodes are causing the loops and separates that node into two or more nodes. The number of nodes
created is dependant on how many children that node has; if there are three nodes, three copies are made. Assume for now that only one node is causing the loop; in figure 6.3 the causing node is 'A.' If that node has two possible values, then two nodes are created and the topology of the network changes to look like figure 6.5. Then for each value of node 'A' a copy is created and the two nodes that represent 'A' are given that value. Once these nodes have a value, a singly connected algorithm is used on each copy of the network; in [37] a message passing algorithm is used. Once the algorithm has finished the probabilities are averaged to create a ‘more accurate’ value [37].

This approach provides an approximate value since it averages the values obtained in several instances of the network. The algorithm does run in a defined amount of time, which is a constant factor multiplied by the time required for the message passing algorithm. This makes this method very practical since the time it requires can be calculated.

Conditioning does have several disadvantages which have caused it to be excluded from ITEAMS. First is that a copy of the network is required for each value of the loop node - node 'A' from the example. If the network is large and there are many values for
'A’, the storage required can grow quite large. This is the case when there are multiple loop causing nodes. The number of copies of the network will grow exponentially with the number of values in the loop causing nodes. Since a copy is required for each value of a loop node, a complete permutation of the loop causing nodes’ values is required. Because of this the space requirements are just too great for ITEAMS since the networks need to be efficient in both time and space.

Clustering

![Diagram of a Multiply Connected Graph with Clustering](image)

Figure 6.6. Example of a Multiply Connected Graph with Clustering

Clustering of nodes is another approach that can be used. Here instead of the loop-causing nodes being separated and copies made, the nodes that are in the multiple paths are clustered into one node. Figure 6.6 shows the network from figure 6.3 after it has been clustered. A super node is created that represents all the nodes in the paths that were collapsed. Here the propagation of beliefs works the same as in ordinary singly connected networks except that the beliefs have to be aggregated to the individual nodes that were collapsed. This increases the complexity of the message passing algorithm some but does
not effect the time too much [37]. Before the nodes can be clustered, they have to be selected as clustering nodes. This task is very time consuming and usually requires hand selecting the nodes.

This type of algorithm works well when the networks are predefined. It provides a fast algorithm that performs almost as well as a singly connected algorithm. The slowdown occurs when the values have to be re-aggregated to the clustered nodes. The math required to do this does not require much extra computation [32].

The problems with this algorithm stem from ITEAMS’ need to automatically generate networks from data provided by instructors. In particular the quiz and assignment networks are automatically generated. These networks will not be created in a format that has the most efficient topology and has the same causal relationships. A good deal of optimization would be required for this method to even begin to be possible. Even if the topologies do allow clustering, algorithms that automatically cluster [10] produce a structureless topology that is difficult to compute and explain probabilities [37].

Stochastic Simulation

Stochastic Simulation is a method that computes probabilities based on the frequency of events that occur in a series of simulation runs. The probabilities are represented as “frequencies” in a sample of truth values. Simulation is also an excellent candidate for parallelizing. The algorithm that ITEAMS uses for stochastic simulation is a form of logic sampling. This algorithm was introduced by Henrion [19] and has been used by many researchers [6, 27, 37]. The algorithm is straightforward and simple even though it is so powerful.

1. Use a random number generator to produce a sample value for each node
2. Proceed through the network following the arrows from the root nodes; at each node calculate the probability of the node with its value using the Bayesian rules.
3. Repeat steps 2 and 3 until all nodes have been calculated
4. Estimate the prior probability of any event by the truth fraction of its logic sample, which is the fraction of times in which the value was selected in step 1.

5. Estimate the posterior probability for any event based on any evidence as the fraction of scenarios in which the event occurs out of the samples in which the value is true. [19]

6.5 Determining Singly Connected versus Multiply Connected

A concern with the dynamic generation of belief networks in ITEAMS lies in determining when the structure being generated is multiply connected. The structure of the network determines what type of update and inference algorithms ITEAMS can use. If the network is a polytree, then a simple, efficient message passing algorithm can be used. These simple and efficient algorithms cannot be used with multiply connected networks because the update of evidence to the network will never complete. With this limitation, stochastic simulation is used to perform updates and inferences on multiply connected networks. Being able to determine when a multiply connected network is present allows ITEAMS to switch algorithms.

There are several approaches that can be employed to solve the problem. The first approach assumes the network is a general undirected graph. The second approach assumes the network is a general directed graph. The third approach assumes the network is a directed acyclic graph (DAG).

6.5.1 General Undirected Graph

The general undirected graph is the simplest algorithm. By checking for a property of trees in the algorithm, there is a fast and efficient method for determining if a cycle exists in a network. A tree with $n$ nodes has at most $n - 1$ edges, so by counting the number of edges in the network, it is possible to determine if the network has a cycle. If the number of edges is less than $n - 1$, then performing depth-first search on the network will determine if a cycle exists. The depth-first search will find a cycle if a node is reached...
twice. The algorithm for depth-first search is presented below with an example. If the number of edges is equal to \( n - 1 \), depth-first search is again applied. Finally, if there are more than \( n - 1 \) edges there is a cycle.

This algorithm runs in \( O(n) \) time. Counting the edges requires only counting all incoming edges in each node and requires only \( n \) time. Running depth-first search also only requires \( O(n) \) time. These make the total running time \( O(n) \).

### 6.5.2 General Directed Graph

When the algorithm assumes that the network is a general directed graph, the problem becomes very difficult to solve. The 3SAT problem states that, given a set \( X \) of boolean values and collection of three values \( Y \) from that set, there is a truth assignment that satisfy any collection of values \( Y \) [16]. It is possible to transform the 3SAT problem into the disjoint path problem on general directed graphs. The disjoint path problem on general directed graphs can be transformed into finding multiple paths in a general directed network. This problem has been shown to be NP-Complete, and because of this ITEAMS does not make the assumption of a general directed graph [15].

### 6.5.3 Directed Acyclic Graph

Now the algorithm which Dr. Chen worked on is discussed. Assuming that the graph is a DAG, then there are two ways of approaching the problem. The first approach uses depth-first search while the second approach uses topological sort. Depending on the restraints of the algorithm, either approach can be acceptable for the algorithm. Since non-cycles are implied for these graphs, the existence of a multiple connection is to be determined. A multiple connection is defined as two distinct node paths from one node, \( x \), to another node \( y \). This means that the paths contain different nodes between \( x \) and \( y \).
Depth-First Search

Cormen et al [12] define Depth-First Search as a walk through a graph that always edges from the most recently visited node as long as there are edges still going out of it. In general the algorithm works by marking nodes as they are visited and giving them a start time. While a node still has outgoing edges those edges are taken to new nodes. When a node has no more outgoing edges, it is given a finish time and marked as finished. Below is the algorithm for Depth-First Search:

\[
\text{DFS}(G) \\
\quad \text{for each node } v \text{ in } V[G] \\
\quad \quad \text{do } \{ \\
\quad \quad \quad v.\text{status} = \text{unseen} \\
\quad \quad \} \\
\quad \text{time} = 0 \\
\quad \text{for each node } v \text{ in } V[G] \\
\quad \quad \text{do if } v.\text{status} == \text{unseen} \\
\quad \quad \quad \text{then VISIT}(v) \\
\]

\[
\text{VISIT}(v) \\
\quad v.\text{status} = \text{visited} \\
\quad \text{time} += 1 \\
\quad v.\text{start\_time} = \text{time} \\
\quad \text{for each } u \text{ in } v.\text{neighbors} \\
\quad \quad \text{do if } u.\text{status} == \text{unseen} \\
\quad \quad \quad \text{then VISIT}(u) \\
\quad v.\text{status} = \text{finished} \\
\quad \text{time} += 1
\]
Consider running DFS on the graph in Figure 6.7. Assuming that the nodes are ordered in the graph as they are labeled, then applying DFS would produce the results in Figure 6.8. In Figure 6.8 the numbers below the label indicate the starting time and finishing time of a node. Thus “Node 6” has a start time of 6 and a finishing time of 7.

Figure 6.7. Sample Graph for Depth-first Search Algorithm

By applying a DFS to a graph and starting at each root node, also known as a ‘zero in’ degree node, a cycle can be found. If a node is reached more than once it does not necessarily mean that a multiple connection has been found. Only when a node is reached from the same root node will a multiple connection exist.

Starting at each root node and traversing through a graph with \(d\) roots, \(n\) nodes, and \(m\) edges gives a total run time of \(O(d(m+n))\).
Topological Sort

The other approach is to use topological sort on the nodes. Cormen et al [12] define topological sort on a DAG $G(V,E)$ as

"...a linear ordering of all its vertices such that if $G$ contains an edge $(u,v)$, then $u$ appears before $v$ in the ordering."

In order for topological sort to work, the graph must be acyclic. If the graph is not acyclic then no linear ordering is possible. Below is an algorithm that performs topological sort:

TOPOLOGICAL–SORT $(G)$

\[
\text{DFS}(G) \quad \text{as each vertex finishes} \quad \text{insert onto the front of a linked list} \quad \text{return the linked list}
\]

Consider the graph in Figure 6.8. That graph adheres to all the properties of a DAG, thus it is a DAG. Because of this, topological sort can be applied. The resulting linked list

Figure 6.8. Sample Graph after Depth-first Search Algorithm
Figure 6.9. Sample Graph after Depth-first Search Algorithm

for the DAG in Figure 6.8 is shown in Figure 6.9.

To use topological sort for the purpose of determining multiple connections, the sort is performed first. Once the sort is complete, the algorithm simultaneously starts at each root node and walks through the topological sort. At each node, a bucket sort of the reachable root nodes is performed. For each parent, each reachable root node of the parent is inserted into the appropriate bucket; if there is already an element in a root’s parent, then there exists a multiple connection. The algorithm using topological sort is presented below.

\[
\text{FindMultiplePath()} \\
\text{Perform Topological sort} \\
\text{Simultaneously from each root} \\
\text{walk through the sorted list} \\
\text{at each node bucket sort parents’ list of roots.} \\
\text{if a root is already in a node’s bucket list} \\
\text{return 'multiply connected'} \\
\text{return 'singly connected'}
\]

Using this algorithm, the running time is similar to the running time of the depth-first search algorithm. The topological sort requires \(O(m+n)\) time to complete. The bucket sort requires \(O(d)\) time to complete. The total running time of the bucket sort version is still \(O(d(m+n))\). There is an optimization of the algorithm. If the number of root nodes
is on the order of the \( \log_2 n \), where \( n \) is the number of total nodes in the graph, then the bucket sort can be replaced with a bit representation of the root nodes. This allows atomic operations for determining the multiple connections and reducing the overall cost of the algorithm. The bit operation used is \( \text{and} \). The comparison of the root nodes is performed by \( \text{and} \)ing all the parents’ reachable root nodes and checking the return value. If the \( \text{and} \) of two parents’ root node bits does not return a zero, then there is a multiple connection.

The topological sort requires \( O(m + n) \) time to complete. The bucket sort requires \( O(d) \) time to complete. The total running time of the bucket sort version is still \( O(d(m + n)) \). The improvement that the bit operations gives can be quite large, since the total running time of this version is only \( O(m + n) \).

6.5.4 ITEAMS Implementation

Since ITEAMS’ algorithms for inference are based on Bayesian Networks, which requires that the networks are directed acyclic graphs, the topological sort version of the algorithm is what ITEAMS uses to determine if a cycle exists. In general, ITEAMS uses the bit operation version of this algorithm. This is because the number of root nodes in ITEAMS should be \( O(\log n) \) or less if an instructor properly designs the teaching modules.

ITEAMS performs some simple bookkeeping tasks throughout its operation that affect this algorithm. When a student loads a teaching module, the roots of the network are determined as part of the loading process. From the previous chapter it was shown that ITEAMS does not allow a teaching module to be modified while a student is working on it. That is, the student will not see any changes until the next time she/he loads the teaching module. Because of this, ITEAMS is able to perform some bookkeeping tasks at load time such as finding and ordering of the root nodes. This information is already stored in the teaching module.

Since the ordering has taken place, this makes the bit representation simple. In the
current version of ITEAMS, as long as there are less than or equal to 64 root nodes the algorithm will still work. Also because of how ITEAMS loads the teaching module, the sorted list of nodes is also generated. This leaves just the walk through of the list to perform. This happens immediately after the teaching module is loaded by the student. The reason for this being performed at this point is that if the student is able to start working immediately on a section and goal, and the student has already worked on the teaching module, then there is a chance that the student is almost done with a section or a non-associated quiz, which means that the teaching module network will be updated very soon after loading completes.

6.6 Structurally Dynamic Belief Networks

ITEAMS uses a new type of dynamic belief network that the author has not found anywhere else: Structurally Dynamic Belief Networks or SDBNs. It appears that there has not been a need for this type of belief network before now. In general there are two types of dynamic belief networks currently being used: Temporal Networks and Dynamic Networks. Neither of these accomplish what ITEAMS needs in a dynamic network. In order to accomplish what is needed for ITEAMS, a new type of dynamic belief network is used.

6.6.1 Background

There are two different types of dynamic belief networks and neither provides the necessary feature of changing the structure of the belief network as time elapses. The temporal network is the closest to providing this feature but it is still lacking. The ‘Dynamic Network’ really does not provide the feature of structural change that ITEAMS requires either.
Temporal Networks

A ‘Temporal Network’ \[33, 44\] refers to a dynamic belief network which models change over time. The change is modeled by providing causal links in time from one node to itself in time. These networks provide a way to model changes over time but limit the size of the network due to computational restrictions. These networks allow for some change but do not allow for unrestricted structural change in the belief networks. The changes they do allow are for the insertion of copies of the original network.

Dynamic Networks

These networks \[29\] are used to model dynamic systems. These networks use a Hidden Markov model which is an extension of the Markov Condition defined earlier. This model takes hidden information into account. It tries to model the hidden information in terms of a single node that is added to the structure of the network. Dynamic networks here refer to a model of a dynamic system and discretizes it so that the system can be evaluated using Bayesian techniques. The network allows change in time to influence the network’s causal links. Depending on the goals of the network, both exact inference and approximate inference algorithms have been devised. In general, approximate algorithms are used. The uses for these networks are generally filtering applications such as Kalman Filtering \[29\].

Neither of these network types provides the structure updating ability that ITEAMS needs. These either allow for modeling dynamic systems and change the causal links of the network or they grow the network size but are limited in size. SDBNs provide the ability to change causal links and grow the network.

6.6.2 Updating Network Values and Inference

The SDBNs use algorithms which have already been developed for non-dynamic belief networks. The algorithms that have been chosen for each network topology are the ones
already used by ITEAMS.

Singly Connected Network

Pearl’s message passing algorithm is used to update and query the network. Once a node has been added, there is no difference in how the queries and updates occur in a regular non-dynamic network. This can be shown by looking at the lambda and pi values and lambda and pi messages. The definition of these values are given in several sections above. To show that this equation works, recall that to calculate a belief in a Bayesian network, one needs to apply the formula

\[
P(x|E) = P(X|N_X, D_X)
\]

where \( N \) are the parents of \( X \) which are evidence and \( D \) are the children of \( X \) which are evidence to obtain \( P(X|E) \). Using Bayes Rule we can obtain the following formula:

\[
P(x|E) = P(N_X|X)P(D_X|X)P(X)
\]

This can be reduced further to

\[
P(x|E) = \frac{P(D_X,N_X|x)P(x)}{P(D_X,N_X)} = \frac{P(D_X|x)P(x|N_X)P(N_X)P(x)}{P(x)P(D_X,N_X)} = \delta P(D_X|x)P(x|N_X)
\]

\( P(D_X|x) \) is the product of the probabilities of all the children of \( X \) which can be stated as

\[
P(D_X|x) = \prod_{U \in \text{Children}(X)} \sum_y \left( \sum_{W \in \text{Parents}(Y)} P(y|n,W) \prod_{w \in W} \pi_Y(w) \right) \lambda(y)
\]

From the definition above \( \lambda(x) \) is calculated with

\[
\lambda(x) = \prod_{U \in \text{Children}(X)} \sum_y \left( \sum_{W \in \text{Parents}(Y)} P(y|n,W) \prod_{w \in W} \pi_Y(w) \right) \lambda(y)
\]
Thus

\[ P(D_X | x) \approx \lambda(x) \]

The same reductions can be made to show that

\[ P(x | N_X) \approx \pi(x) \]

Multiply Connected Network

Multiply connected SDBNs use stochastic simulation for the updating and querying. Stochastic simulation just reintroduces variables through the logic sampling. This implies that the network is essentially reinitialized each time a node is introduced. But since the logic sampling calculates the average probability given all the probabilities that contain the instantiated variable, the probability should still converge towards the exact value if the network would converge. Even though the network does not contain a large sampling of probabilities that contain a sampling of all values, some algorithms have used biased sampling of values [19, 20, 30]. This is what SDBNs are doing: taking a biased sample of the values.

6.6.3 Updating Network Structure

ITEAMS needs to be able to update the structure of a network on the fly and have the values propagate correctly. Since the network being grown could be either singly or multiply connected, both network types need to be addressed.

Singly Connected Network

When a singly connected network is grown, the addition of a node is straightforward but requires some extra processing. To add a node to the network, some information is needed. What type of node is being added: root, non-leaf, or leaf? How does the link
structure change? If a non-leaf node is added, then there will possibly be a 'loop' created if the links are not reconfigured. The new node can either remove some existing links and add its own or it can just add its own links. If the node is a root node, then only new causal links need to be added. If the node is a leaf node, then only new causal links need to be added. The case of the non-leaf node requires more effort. If the node is being inserted between two nodes, than in order to keep the network singly connected, the link between two nodes that are being separated needs to be removed. This is accomplished by removing the link, and in the original child node removing any probabilities in the CPT that refer to the previous parent.

After the link structure has been updated, the CPT values of the nodes must be updated. To update the CPT values there needs to be some relationship defined already that dictates how a parent influence a child node. This information is needed to correctly specify the values in the CPT. In ITEAMS the new CPT values are calculated by taking an average value of a parent's values and summing all parents’ averages together. This value is then divided by the number of parents. It creates a normalized value that represents the relationship between the nodes. This is done because all parents and values of the parents have an equal probability of being selected. This is a naive approach which could be improved to incorporate a bias based on the current state of the network. But further work needs to be done to determine what the proper relationship is between the parent nodes and child nodes.

When all of the links have been updated, the network needs to be reinitialized for the new nodes and causal relationships. The algorithm that performs this is presented below:

```plaintext
initializeNetwork (NewNode N) {
    for (each value n of N)
        lambda(n) = 1
    for (each parent Z of N);
```
for (each value \( z \) of \( Z \))
\[
\lambda(X)(z) = 1;
\]
for (each child \( C \) of \( N \))
\[
\text{for (each value } n \text{ of } N) \\
\pi(C)(n) = 1;
\]
if \( N \) is a root {
\[
\text{for (each value } n \text{ of } N) \\
\text{P}(n|e) = \text{P}(n); \quad \text{// where } e \text{ is the list of evidence values} \\
\pi(n) = \text{P}(n);
\]
}
for (each child \( C \) of \( N \))
\[
\text{send_pi_message}(C,N,\text{false});
\]
for (each parent \( Y \) of \( N \))
\[
\text{send_lambda_message}(N,Y,\text{false});
\]

The two helper functions send_pi_message and send_lambda_message are modified slightly from the original singly connected network algorithm presented earlier. The difference is that if the boolean value passed in is false, the message stops at that node. Thus in the SDBN only the immediate parents and children are notified of the change.

Multiply Connected Network

Multiply connected networks provide a simpler method of adding nodes to the network. Since loops already exist in the network, adding nodes can be less strict than adding nodes in the singly connected network. The links can be added or deleted in the same fashion as in the singly connected network. The messages are sent only one level up from the added
These networks are used to evaluate the student’s performance in a teaching module on a category basis. The instructional categories are evaluated by examining the questions and exercises that pertain to them. Using these networks, it is possible to evaluate the student in each category, determine when a student needs help, and at what level of difficulty a student should be working.

Evaluating Instructional Categories

SDBNs are used to evaluate the instructional categories. This is done to reduce the complexity of these networks as well as increase relevance of the belief node influence. A multiply connected network could have been used to evaluate the categories, but this was ruled out for several reasons. First is that questions and exercises make up the data for evaluating a category. Including every one of them could create huge networks which would take a while to update and query. The second reason is that it would not be possible to correctly model the influence in such a network. The reason for this is that the questions and exercises that are presented to students are randomly chosen based on the student’s current performance. In order to model the causal dependencies, the DAG property would be broken for the belief network since every node would need a link to every other node. This would create a fully connected network. It would also not be desirable to create a predefine structure, since the causal relationships would not be correct.

These networks are updated when a new question or exercise is presented to the student. When the question is presented, a new belief node is created which represents the question and added to one of the category networks. The new node is added to the net-
work just before the leaf node of the network. When a question pertains to more than one category, a new belief node is created and is added for each category network. When a student answers the question or exercise and the value of the node is determined, then the network is updated. The way these networks are updated when a node is added creates a singly connected network.

The structure of the network is a singly connected network since the new nodes are added just above the leaf nodes in the networks. The leaf node is a node that represents the instructional category. Since belief network update algorithms have the most effect on the values when a parent’s value changes, ITEAMS adds the new node just above the instructional category node.

Determining When a Student Needs Help

These networks play a critical role in determining when a student needs help. By looking at the performance of a student in each category, it is possible to determine where a student is lacking. If any deficiencies are found, ITEAMS then looks at which Goals in the Sections already completed by the student pertain to the deficiencies. The Goals are ranked according to how many of the deficiency categories they relate to. A Goal that relates to two categories is ranked higher than a Goal that only relates to one of the deficiency categories. Right now there is no distinction made between how deficient a student is in a category. In the current implementation of ITEAMS, this information is only collected. It is not yet presented to the student nor is it enforced. That is the student can keep working on new material even though they are deficient in certain categories.

Determining What level of Material to Present to a Student

These networks also play a part in what level of material a student is working. By evaluating the performance of a student in each category, it is possible to determine what difficult level in a Goal should be presented. Since each goal can have a different ob-
jective relating to it, and objectives relate to the categories, it is possible to determine the level of difficult for a given goal. This is a straightforward evaluation. ITEAMS starts the level of difficult at proficient and depending on the performance of the student on the relevant categories, ITEAMS will either increase or decrease the level of difficult.

6.7 Chapter Summary

The chapter began with a discussion of belief networks including how they work and in what way they have been used before. It then presented the different types of belief networks used in ITEAMS and how they are used.

- Teaching Modules use either singly-connected networks or multiply connected networks depending on the outcome of the path detection algorithm. This network is used to evaluate the overall performance of the student.
- Instructional Categories use SDBNs to evaluate the student on a category basis. These values can be used to approximate the overall performance of the student.
- Sections use multiply connected networks which are constructed based on the goals, quizzes, and assignments that are included in it.
- Quizzes and Assignments use automatically generated networks which are multiply connected. From these networks the questions that are presented to the student are chosen.

For non dynamic networks, ITEAMS uses two different inference algorithms: Pearl’s Message Passing Algorithm for singly connected networks and Logic Sampling for multiply connected networks.

An algorithm that will efficiently determine if a network contains multiple paths was also presented. This algorithm uses topological search to sort the nodes of a network and then walks down the network checking which root nodes are reachable from the current node. If a node can already reach a root node that one of its parents can reach, than there is a loop.

Structurally Dynamic Belief Networks were also presented. These networks provide ITEAMS with the ability to grow a belief network as it is being used. These networks can
work with either singly connected topologies or multiply connected topologies. These networks are used to evaluate student performance, evaluate performance in instructional categories, determine when a student needs help, and determine at which level a student should be working.
CHAPTER 7

DISCUSSION

This thesis has presented an analysis of Teaching Environments and Intelligent Tutoring Systems. Teaching Environment systems were evaluated by determining and examining features common among available systems. Intelligent Tutoring Systems were evaluated on the basis of the standard features used within current systems and on the basis of what researchers have pointed to as lacking. The thesis then presents a description of ITEAMS, the ITEAMS “Plugin interface”, and ITEAMS assessment with belief networks.

7.1 Teaching Environments

Teaching Environments give instructors the ability to present the material they have chosen or designed for their students. This is done, however, through formats that are mostly rigid and restrictive. Current formats offer little support for automatic grading and are limited to multiple-choice style questions. There is no student assessment which specifies where students are lacking in their studies. Moreover, these systems do not allow instructors to organize their material in any manner other than a linear form. They do make it possible to interact with external software, but in a restricted manner.

7.2 Intelligent Tutoring Systems

Intelligent Tutoring Systems fall into a number of widely varying categories, determined by many different factors:
These systems lack several significant features. Generally they lack instructor models, as well as a dynamic content for utilizing the models. In addition, they do not offer a robust means of interacting with external software.

7.3 ITEAMS Architecture

ITEAMS, by design, is intended to offer features that are lacking in both Teaching Environments and Intelligent Tutoring Systems. ITEAMS provides a multiple subject-matter environment. This affords instructors the flexibility to design teaching modules with which the designing instructors themselves are comfortable. A teaching module is comprised of instructional categories, sections, goals, quizzes, and assignments. Each of these is a part of the instructor’s process for laying out the instructional material. Teaching modules contain sections, quizzes, and assignments. Sections contain goals, quizzes, and assignments. In order to give instructors an even greater degree of flexibility, ITEAMS allows external software to be integrated with ITEAMS through a robust ‘plugin’ interface. The evaluation of students is based on three different types of belief networks: singly connected, multiply connected, and structurally dynamic belief networks.

7.4 Future Work and Improvements

At this point, no claims are made that ITEAMS is a complete and perfected system. As is the case with almost any software project, there will always be improvements to be
made and new design choices to consider. Several issues and potential improvements are discussed here. These can enhance the overall usefulness and flexibility of ITEAMS. The topics selected for discussion here are not intended to represent a complete list of potential improvements and avenues of research.

7.4.1 Plugin Connections

In order to simplify the students' job when working with a plugin, it would be good to not have them specify a location of an external application. For this to happen a central ‘plugin application registry’ would need to be created. This ‘registry’ would contain connection pools for each application with which ITEAMS can connect. Instructors would need to populate the registry with the locations of different applications so instances of ITEAMS can be given a connection. The location of the server would be provided by an instructor where the application location is provided. When a student begins working on an exercise, the plugin communicates with the ‘registry’ to find the location of a running application. When a location is found, the plugin then connects to the external application, and the student is completely removed from the process.

There are many possible features that can be built into a service such as this. Instead of just providing a pool of connections that are distributed to ITEAMS instances as needed, the ‘registry’ could also be designed to start the applications requested by ITEAMS and then send the new connection to ITEAMS as needed. One could then design some dynamic load balancing in so that no one machine or small group of machines are running all the requested applications. In a large class, (fifty or more students), the benefit of the load balancing might have the most noticeable benefits.

7.4.2 Database Connections

As mentioned in chapter 4 there is an issue with the database connection. The information being sent to and from the database is in a clear text format. This allows anyone to see
the information before it reaches its destination. Students are able to get answers as well as possibly update their performance values. There are currently several possibilities in ITEAMS to fix this. One possibility is to encrypt only the data that is stored in the database before it is sent over the network. Another approach is to create a database wrapper that sits on the same host as the database and have it accept only encrypted information for the database.

The first choice is simple and encrypts the data before it is sent into the database. This would mean that all information in the database is encrypted. ITEAMS would have to encrypt only data portions of the SQL commands before sending the command. This leaves the structure of the command open to viewing. This allows anyone to see how data is stored, make connections to the database, and perform queries and inserts on their own. This is possible even if the database is password protected. The reason for this is that the password encrypted by the login command would still be sent over the network as clear text. Considering these problems makes this option less appealing than other approaches.

The other approach would be to create a database server interface that accepts a connection from an instance of ITEAMS and takes the encrypted information and decrypts it before placing it in the database. This server would need to accept as many connections from instances of ITEAMS as the database can accept. It would also need to handle all the encryption and decryption of data that ITEAMS would require. All information would be sent in an encrypted format from ITEAMS to the server where it gets decrypted and processed. The information would most likely be an encrypted SQL command that the server would give to the database. Any response from the database would be encrypted by the server and sent back to ITEAMS. The problems with this approach still include a possibility that someone could see the information when it is being transmitted on the local host since a TCP connection is generally used to connect to the database. One alternative would be to have the server interface the database from its command line tool and
pipe the information to it that way.

7.4.3 Hint Providing Capabilities

ITEAMS has the potential to provide hints to students based on their performance. The potential lies in the belief networks that are already used for evaluation. Assignment exercises are where hints would be most useful since quizzes are used to assess a student’s mastery of the subject material. Assignments on the other hand can be seen as assessing a student’s mastery of subject material or as a tool to help the student master the subject material. If used in this light, hints would be helpful for the student if they are struggling in their performance. This functionality needs to be explored more thoroughly and tested.

7.4.4 Open Learning Environments

One possible approach to take with ITEAMS is develop it into an Open Learning Environment [4]. ITEAMS already has the functionality for being an Open Learning Environment since a student can already progress at their own pace through a teaching module exploring sections in the order they choose. This is of course dependant on ITEAMS determining that the student has accomplished all or enough of the prerequisites for a section. The benefit of modifying ITEAMS in this manner would require a lot more research and comparison of classroom usage to the current design usage results.

There are a couple of outcomes from exploring the Open Learning Environment approach. One such outcome from looking into an Open Learning Environment implementation of ITEAMS is that there are certain instances, courses, or modules where it would be beneficial to allow ITEAMS to switch between an ’open’ design and the current structured design. In that case, an instructor could specify whether a teaching module is ’open.’ The other interesting outcome is that ITEAMS splits into two distinct applications with one being the structured design and the other being the ’open’ design.
7.4.5 Student Selected Instructor Style

An interesting avenue for ITEAMS to go down is to allow a student to choose what type of instructional model they want to use. This way the student learns in a manner that is best for them to learn. The instructor would enter all the information as they do now, but the student is able to switch from the style the instructor provided to a different style.
BIBLIOGRAPHY


K. P. Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning*. Doctorate, UC Berkeley (July 2002).


P. Software, Dr. scheme. World Wide Web.


