

A C++ STANDARD TEMPLATE LIBRARY  
INTELLIGENT TUTORING SYSTEM  
WITH BAYESIAN AND FUZZY LOGIC STUDENT MODEL

CHRISTINE LEE SIEW KEN

THESIS SUBMITTED IN FULFILLMENT OF THE  
REQUIREMENTS  
FOR THE DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND  
INFORMATION TECHNOLOGY  
UNIVERSITY OF MALAYA  
KUALA LUMPUR

JUNE 2006

## Abstract

Earlier work on Intelligent Tutoring Systems (ITSs) for programming focused more on teaching programming syntax than its application. The main tutoring approach is to present a problem specification for the student to solve, followed by intelligent analysis of the solution with various feedback. It is also observed that existing ITSs suffer from static domain knowledge and are restricted to the tutoring session. Therefore, this research proposes the development of a web-based ITS for both curriculum planners and implementer-tutors to teach students the application of the C++ Standard Template Library (STL) to problem solving.

From experience, it is discovered that students find the C++ STL difficult due to their weaknesses in understanding various object-oriented concepts. This ITS overcomes the learning and teaching challenges by modelling the program specification based on prerequisite concepts. Bayesian Theorem is applied to model the student's knowledge and direct the tutoring intelligently. Bayesian probability reasoning is a well-known Artificial Intelligence technique for uncertainties management. The development of the C++ STL ITS applies practices from the eXtreme Programming methodology and J2EE technologies. The 3-tier architecture ITS constitutes three main components – Student Modelling Module, Tutoring Module and Users Administration Module providing the authoring of the domain knowledge dynamically. Hence, tutors can then fully participate in the design of the curriculum and tutoring sessions as well as in the implementation of the tutorials for their students for effective teaching and learning.

Both summative and formative evaluations were conducted on the C++ STL ITS. The evaluation results revealed that the Bayesian Theorem has the capability of modelling the student's prerequisite and directing the student during the tutorial session. The Fuzzy Stereotyping of Students Expert System works well in categorizing the students according to four stereotypes – novice, beginner, intermediate and advanced.

Short term future enhancements include extending the tutorial questions, domain knowledge, accommodating more feedback on the programming syntax, and incorporating the fuzzy expert system into the C++ STL ITS. Three areas of research proposed for long term are application of alternative knowledge acquisition techniques, integration of learning styles into the student model, and representation of domain knowledge using ontologies.

## Acknowledgement

I express my most sincere gratitude to

<i>GOD</i>	...	constant grace, strength, wisdom and blessing
<i>My supervisor</i> <i>Assoc. Prof. M Sapiyan Baba</i>	...	invaluable help and guidance
<i>My husband</i> <i>Seng Chor</i>	...	loving care, help and patience
<i>My baby boy</i> <i>Jordan</i>	...	a source of strength and motivation
<i>Dad Peter and Mum Agnes</i>	...	a source of inspiration
<i>My sister Eva</i>	...	love, concern and moral support
<i>My brothers Felix and Alex</i>	...	love and moral support
<i>My brothers and sisters</i> <i>in Christ</i>	...	prayer support and encouragement
<i>Guru Max Lam</i>	...	advice and kind assistance
<i>Colleagues at</i> <i>KBU International College</i>	...	advice and kind assistance
<i>BSc (Hons) Software Engineering</i> <i>and</i> <i>BEng (Hons) Electronics &amp;</i> <i>Computing Students</i>	...	patience and co-operation
<i>Friends</i>	...	comments and proofreading

And all other friends, relatives and well-wishes who have contributed in one way or another in making this Research a reality.

# Table of Contents

Abstract .....	ii
Acknowledgement.....	iii
Table of Contents .....	iv
List of Figures .....	vii
List of Tables.....	ix
Chapter 1 Introduction .....	1
1.1 Motivation : Why C++ STL ITS on the Web? .....	4
1.2 Aims and Objectives.....	6
1.3 Project overview .....	7
1.4 Contribution.....	8
1.5 Structure of the Thesis .....	10
Chapter 2 Intelligent Tutoring Systems with Focus on Student Modelling .....	12
2.1 Components of an ITS .....	14
2.1.1 Expert Model.....	15
2.1.2 Student Model.....	17
2.1.3 Tutoring Model .....	19
2.1.4 Student Interface .....	19
2.2 Student Modelling Techniques .....	20
2.2.1 Overlay Model (Carbonell, 1970; Carr & Goldstein, 1977).....	22
2.2.2 Differential Student Model.....	23
2.2.3 Perturbation Model or Buggy Student Model.....	23
2.2.4 Cognitive Model.....	25
2.2.5 Constraint-based Modelling (CBM) .....	27
2.2.6 Machine Learning (ML) Techniques.....	28
2.2.7 Stereotype-based methods.....	29
2.2.8 Combination of Methods .....	30
2.3 Machine Learning Techniques in ITS .....	31
2.3.1 Case-based Reasoning .....	32
2.3.2 Theory Refinement .....	33
2.3.3 Fuzzy Logic, Neural Network or Hybrid.....	34
2.3.4 Bayesian Network or Probabilistic Model.....	36
2.4 Neural Networks vs Bayesian Networks .....	39
2.5 Summary.....	43
Chapter 3 ITS for Computer Programming .....	48
3.1 Prior Work – Overview .....	48
3.1.1 LispTutor (Reiser, Anderson & Farell, 1985).....	48
3.1.2 PROUST (Johnson, 1986).....	50
3.1.3 BRIDGE (Bonar and Cunningham, 1988).....	51
3.1.4 ASSERT (Baffes and Mooney, 1996).....	52
3.1.5 ELM-ART (Weber & Specht, 1997) .....	54
3.1.6 Virtual Campus PROLOG Tutor (Peylo et al, 2000).....	56
3.1.7 Tutor on C++ Programming (Kumar, 2002).....	57

3.1.8	<i>JITS - Java Intelligent Tutoring System (Sykes, 2003)</i>	58
3.1.9	<i>Pseudocode Tutor (Chad Lane &amp; VanLehn, 2003)</i>	59
3.1.10	<i>BITS – Bayesian Intelligent Tutoring System (Butz et al, 2004)</i>	60
3.2	ITS for Programming – Comparative Analysis	64
3.2.1	<i>Domain Knowledge</i>	64
3.2.2	<i>Level of Programming</i>	65
3.2.3	<i>Tutoring Goals</i>	66
3.2.4	<i>Student Modelling Techniques</i>	68
3.3	Summary	68
Chapter 4	C++ STL and Bayesian-Fuzzy Student Modelling	73
4.1	The Domain Knowledge – C++ STL	74
4.2	Difficulties in Learning and Teaching C++ STL	78
4.2.1	<i>Learning the C++ STL</i>	78
4.2.2	<i>Teaching the C++ STL - Curriculum Planners vs Implementer-tutors</i>	81
4.3	Dynamic Domain Knowledge Modelling	84
4.4	Student Modelling with Transparency	85
4.5	Uncertainties Management	87
4.5.1	<i>Bayesian Reasoning</i>	89
4.5.2	<i>Bayesian-based Predictive Initial Student Model</i>	91
4.5.3	<i>Tracking Student's Progress during Tutorial Sessions</i>	96
4.5.4	<i>Updating Student Model</i>	102
4.5.5	<i>Evaluating and Categorising Student's Behaviour</i>	106
4.5.6	<i>Acquired Knowledge</i>	111
4.6	Summary of Problems and Solutions	111
Chapter 5	STL Tutor Architecture and Development	113
5.1	Overall Architecture	113
5.2	Student Modelling Module	116
5.3	Domain Knowledge Module	116
5.3.1	<i>Topic and Sub-Topic Repository</i>	117
5.3.2	<i>Program Specifications Repository</i>	118
5.3.3	<i>Tutorials and Sub-Tutorials Repository</i>	119
5.4	Teaching Strategies Module	121
5.4.1	<i>Pre-Test Module</i>	121
5.4.2	<i>Tutorial Module</i>	125
5.4.3	<i>Post-Test Module</i>	125
5.5	Users Administration Module	126
5.6	The eXtreme Programming Methodology	127
5.7	The 3-Tier System Architecture	128
5.7.1	<i>The Client-tier Layer</i>	134
5.7.2	<i>The Middle-tier Layer</i>	134
5.7.3	<i>The Data Source-tier Layer</i>	137
5.8	XML Syntax Parser and Student Model Update Algorithm	137
5.8.1	<i>XML Syntax Parser</i>	138
5.8.2	<i>Student Model Update Algorithm</i>	141
5.9	Fuzzy Expert System	145
5.10	Development Tools	150

Chapter 6	C++ STL ITS System Evaluation – Methodologies and Results.....	151
6.1	Evaluation Methodologies and Evaluation Requirements.....	152
6.2	Evaluation of the C++ STL ITS Architecture .....	155
6.2.1	<i>Domain Knowledge Module</i> .....	155
6.2.2	<i>Teaching Strategies Module</i> .....	159
6.2.3	<i>Student Modelling Module</i> .....	160
6.2.4	<i>Graphical User Interface Module</i> .....	164
6.3	Pre-Test Results and Analyses.....	166
6.4	Tutorial Sessions Results and Analyses .....	169
6.5	Post-Test Results and Analyses .....	173
6.6	Evaluation of the Fuzzy Stereotyping of Students Expert System .....	176
6.7	Related Work .....	179
Chapter 7	Conclusions and Future Work.....	182
7.1	Conclusions .....	182
7.2	Future Work.....	183
7.2.1	<i>Short Term</i> .....	183
7.2.2	<i>Long Term</i> .....	185
References	.....	190
Appendix	.....	205
Appendix A	– List of Conditional Probabilities .....	206
Appendix B	– List of Topics and Sub-Topics.....	208
Appendix C	– Vision.....	210
Appendix D	– User Stories .....	211
Appendix E	– Acceptance Tests .....	216
Appendix F	– UML Design Diagrams .....	222
Appendix G	– Session Beans .....	226
Appendix H	– Data Tables Descriptions.....	251
Appendix I	.....	261
	XML Syntax Parser Code.....	261
	Student Model Update Code.....	269
Appendix J	– Fuzzy Rules Table .....	299
Appendix K	– build.xml.....	300
Appendix L	– List of C++ STL vector Topics and Sub-Topics.....	307

## List of Figures

Figure 2.1	Routing within a Computer Aided Instruction System	13
Figure 2.2	Typical Basic Architecture of an ITS	14
Figure 2.3	An illustration of the overlay model	22
Figure 2.4	An illustration of the differential model	23
Figure 2.5	An illustration of the perturbation model or buggy student model	24
Figure 2.6	A Neural Network Topology	40
Figure 3.1	A partial Directed Acyclic Graph implemented in the BITS	61
Figure 3.2	Routing of Student's Understanding	63
Figure 4.1	Prerequisite Iteration with its sub-skills	93
Figure 4.2	A Sample Layout of a Problem Specification during the Tutoring	97
Figure 4.3	Directing Student after Pre-Test	98
Figure 4.4	Various Paths during Tutoring Session	101
Figure 4.5	Gradual Change in Conditional Probability	105
Figure 4.6	The Four Tasks in the Fuzzy Expert System	107
Figure 5.1	Components of C++ STL ITS	114
Figure 5.2	System Architecture of the ITS	115
Figure 5.3	A Screen Shot of an Interface during the Pre-Test Evaluation Session	123
Figure 5.4	Partial Screen Shot of the View from 'Question Contents'	124
Figure 5.5	Cycles in XP	127
Figure 5.6	3-Tier System Architecture of C++ STL ITS	133
Figure 5.7	XML-based format describing the answer to populate a vector from keyboard	139

Figure 5.8 Phases in the XML Syntax Parser	140
Figure 5.9 Flowchart of Student Model Update	141
Figure 5.10 Update Pseudocode for Number of Attempts = 0	143
Figure 5.11 Update Pseudocode for Number of Attempts = 1	144
Figure 5.12 Fuzzy Sets of Conditional Probabilities	146
Figure 5.13 Fuzzy Sets of Time	147
Figure 5.14 Fuzzy Sets of Number of Attempts	147
Figure 5.15 Fuzzy Sets of Number of Hints	147
Figure 6.1 Understandability of Pre-Test Questions	158
Figure 6.2 Understandability of Post-Test Questions	158
Figure 6.3 Tutorial Framework and Sub-Tutorial Questions	164
Figure 6.4 Student's Evaluation of the Interface for the Pre-Test Module	165
Figure 6.5 Student's Evaluation of the 'Question Content' Functionality	165
Figure 6.6 Prerequisites for STL vector Question 1	171
Figure 6.7 Tutorial Students Post-Test Results	174
Figure 6.8 Non-Tutorial Students Post-Test Results	174
Figure 6.9 Rule Viewer	177
Figure 6.10(a) Three-dimensional Plot for Understanding-Time-Conditional Probabilities Relationship	178
Figure 6.10(b) Three-dimensional Plot for Understanding-Attempt-Conditional Probabilities Relationship	178
Figure 6.10 (c) Three-dimensional Plot for Understanding-Hint-Conditional Probabilities Relationship	179



## List of Tables

Table 2.1	Similarities and Differences in Neural Networks and Bayesian Networks	43
Table 2.2	Summary of Student Modelling Techniques	44
Table 2.3	Area of Application of Student Modelling Techniques	47
Table 3.1	Multi-layered Overlay Model in the ELM-ART II	55
Table 3.2	Tutoring Goals of ITSs to teach programming	66
Table 3.3	Techniques employed in current ITSs	69
Table 4.1	STL Components	76
Table 4.2	Examples of Student's Pre-Test Performance	95
Table 4.3	Four Levels of Hints	99
Table 4.4	(a) How the Student Model is Updated when $P(U/C) < 0.6$	103
Table 4.4	(b) How the Student Model is Updated when $0.6 \leq P(U/C) \leq 0.8$	103
Table 4.4	(c) How the Student Model is Updated when $P(U/C) > 0.8$	104
Table 4.5	Summary of Identified STL Problems and Proposed Solutions	112
Table 5.1	Partial List of Topics and Sub-Topics	117
Table 5.2	An Example of a Program Specification with Prerequisites and Acquired Skills	119
Table 5.3	An Example of Sub-Tutorial Problems	120
Table 5.4	Overview of Implementation of Web-based Intelligent Tutoring Systems	130
Table 5.5	Special Characters	139
Table 5.6	Example of Conditional Probabilities Achieved by a Student	142
Table 5.7	Linguistic variables, values, range and membership function	149
Table 6.1	Classification Table of Evaluation Methods	153

Table 6.2	Sample – Pre-Test performance of various students	161
Table 6.3	Overall students’ Pre-Test performance for each topic	163
Table 6.4	Common Programming Mistakes Example 1	167
Table 6.5	Common Programming Mistakes Example 2	167
Table 6.6	Common Programming Mistakes Example 3	168
Table 6.7	Example – Variance of Percentage in Selection of Answers	169
Table 6.8	Survey – Questions on Tutorial Session in C++ STL ITS	172
Table 6.9	Survey – General Questions on Learning Experience with C++ STL ITS	172
Table 6.10	Class Average for Post-Test Results	173
Table 6.11	Partial Post-Test Results	175
Table 6.12	Numerical Range for Stereotype	176

## Chapter 1 Introduction

In the twilight years of the 19<sup>th</sup> Century, Chinese Hakka immigrants, especially the Basel Christians, arrived in the Bornean Malaysian State of Sabah (formerly North Borneo), together with their families, under the immigration schemes promoted by the British North Borneo Chartered Company at the time. *Like all Chinese, Hakkas place great emphasis on education. When the initial batches of Hakka immigrants arrived, informal schooling was generally the rule, often in the home or neighbourhood. Most fathers had had a few years of education in China, so they taught their children, recycling Chinese calendars to use as writing pads. The children later continued their lessons in the homes of literate elders who tutored several neighbourhood children in return for gifts of rice or, occasionally a chicken.* (Zhang, 2002).

The Chinese Hakka immigrants had a strong desire to preserve and sustain their cultural identity. Thus, they established their own traditional *sishus* (mini private schools), which led to the birth of the first Chinese school in Sabah in 1886 in Kudat, the first capital of Sabah (Wong, 2004).

Let us now turn the clock back to a learning scenario in 1920 in Kudat, situated at the northern tip of Borneo. Eighty-five years ago, under the nipa-palm roof of a traditional *sishu*, ten young boys sat on the wooden floor, cross-legged and barefooted, (their wooden clogs neatly placed beside the door), listening attentively to their tutor explaining some Arithmetical sums on a slate board. Each of the pupils had a piece of wooden-framed slate and slate pencils for writing.

Amongst this small group was an eight-year old boy who was an Arithmetic wizard. Fifty-one years later, he became my late paternal grandfather. He could understand and catch on faster than the rest of the Class. Seeing that the young boy had also the patience to guide the others, the tutor gave him a role as a co-tutor for the slow pupils who were a few years older than him. He co-tutored the weak pupils, one at a time, using a method which could well be termed as *individualized one-to-one tutoring*. It was a typical example of *pupils helping pupils* to achieve learning. Thus, it goes on to show that this underlying concept was being practised a long time ago with small-sized classes.

As decades passed, the student population in Sabah, as in all places throughout Malaysia, increased as both males and females pursued education to earn a better living. *Sishus* in Sabah were upgraded to formal schools, and teachers had to be recruited from Mainland China. The class size grew bigger and bigger, often reaching about 50 students per class, which had certainly affected teaching as well as learning techniques. Teachers would use chalk to write on the blackboard while teaching. Students would use lead pencils or pens to write in exercise books. In such a scenario, it is humanly impossible for teachers to give their students individual attention. Teachers who have an instructional style of teaching would be able to do a good job in handling large classes. In colleges where there may be about 100 students in a class, teachers would turn lecturers instead.

Then, in the late 50's and early 60's, College-trained teachers prepared lesson notes in advance and used teaching aids to enhance the art of teaching and make the learning process more interesting and effective. Even in Teachers' Training Colleges at that point in

time, trainees who were better in certain subjects like Mathematics, would be assigned co-tutor roles to help their fellow College mates learn in small tutorial groups.

As I gathered through ‘oral history’, from the time of my ancestors in the late 19<sup>th</sup> Century to my grandparents in the early 20<sup>th</sup> Century and to my parents in the mid-20<sup>th</sup> Century, and from my own experience as a secondary student in the 80’s, rote learning was the most widely practised form of learning. Apparently, this trend was attributed to a number of factors, namely, fear of the teachers, fear of failing and subsequent class retention, fear of corporal punishment, for example caning, wishing to attain perfect test scores and be in the Top Ten in Class.

Rote learning still seems to be the order of the day when it comes to learning just to pass the tests or examinations. Students memorize and cram as much information and facts as possible into their heads, in *parrot fashion*, often without proper understanding and later regurgitate for the tests or examinations. At the end of the day, such students find that they have practically learned nothing that they can truly apply and use in real life.

From the 60’s onwards up till the present time, it has become a necessity for students to attend Tuition classes after school, two, three or more times a week. Some children are sent for Tuition class for nearly all subjects including Art and Craft, not forgetting music or dance lessons or martial arts, to name a few. Initially, only the academically weak students attended Tuition classes, but as competition got more intense among the students not only in their class or school but with other schools, the *bright sparks* also jumped aboard the

band-waggon. As a result of this, Tuition classes expanded and became quite similar to a normal classroom situation, which obviously defeats the purpose of such tutoring classes.

With the advent of the Internet and the availability of a vast quantity of information into the education scene in the early 90's, the computer has taken on new, unprecedented dimensions never before experienced in history. In time to come, rote learning could gradually be overtaken and phased out by the use of computers. The computer has become an invaluable teaching and learning tool, fostering quality teaching and quality learning and above all, helping to achieve a common goal, that is teaching and learning with understanding.

It is believed that any difficulties or problems relating to understanding and learning challenging subjects such as Programming, can best be solved using Intelligent Tutoring Systems (ITSs). Empirical studies have proven that one-to-one tutoring is the most effective mode of teaching and learning, and this individualized tutoring is uniquely offered by ITS (Bloom, 1984).

### 1.1 Motivation : Why C++ STL ITS on the Web?

Intelligent Tutoring Systems (ITSs) are computer-based instructional systems with models of instructional content that specify what to teach, and teaching strategies that specify how to teach (Wenger, 1987, Ohlsson, 1987). Most traditional educational software focus on the teaching material which is usually presented in a sequential manner to the user. However, different users have various needs and knowledge. The main purpose of the ITS

is to suit the users and its goal is to communicate its embedded knowledge effectively. Therefore, the system is to be user adaptable and flexible. This is achieved through Artificial Intelligence (AI) techniques.

One major contribution of the research and development of ITS is in teaching computer programming. The use of ITSs to support students in learning various programming languages from the 1980s to the current millennium, has matured in the areas of choice of programming languages, tutoring goals and application of AI techniques.

The level of programming taught in the current ITSs such as ASSERT (Baffes & Mooney, 1996), C++ Tutor (Kumar, 2002), JITS (Sykes & Franek, 2003) and BITS (Butz et al, 2006) covers elementary topics that are typically found in an introductory course to Computer Programming. Indeed, there is a need for tutoring materials that target a higher programming level. These include application of the programming language to create data structures and solve more complex problems which leads to the motivation to choose the C++ Standard Template Library (STL) as the domain knowledge for this ITS.

There is no one right way to learn and teach C++ and its associated design and programming techniques. The aims and background of each student differ (Stroustrup, 1999). As ITS seeks to mimic the human tutor as it imparts knowledge to the student, it is highly suitable for guiding students who have different levels of prerequisites with respect to the application of the C++ STL. The student model in an ITS has the capability of recording the student's information and keeping track of the student's action as they

progress in their learning. With this information, teaching and learning can proceed in a variety of ways based on the needs and interests of the students.

Combining web technologies and ITS provides globalization and at the same time individualization of teaching and learning. Globalization in this context refers to providing the similar course materials and tools to different locations in the world. This allows sharing and reusability of teaching and learning materials to save resources. In addition to this, institutions that franchise their courses can enhance the collaboration with their partners. On the contrary, individualization uniquely identifies a student providing an environment similar to face-to-face tutoring. The student actually learns at his/her own pace within their own workspace in a global environment.

## 1.2 Aims and Objectives

The aim of this research is to develop a web-based C++ Standard Template Library Intelligent Tutoring System using AI techniques. The objective of this system is to provide personalized problem solving support to students in the learning of the C++ STL.

The objectives of this research are :

- To review current student modelling techniques.
- To review current techniques employed in existing ITSs for programming.
- To identify a suitable student modelling technique for the C++ STL ITS.
- To develop the C++ STL ITS based on the proposed technique.
- To evaluate the proposed student modelling technique for effectiveness.



### 1.3 Project overview

The C++ STL ITS includes the following functionalities:

- Authoring of pre-test, tutorial sessions and post-test
- Pre-assessment module to test student's prerequisite knowledge
- Problem solving support with various teaching strategies
- Post-assessment module to examine student's understanding after the tutorial
- Reporting of pre-test, tutorial and post-test performance results
- Administration of users – students and tutors

The system architecture of the C++ STL ITS consists of four main modules :

- i) Graphical User Interface Module
- ii) Student Modelling Module
- iii) Teaching Strategies Module
- iv) Domain Knowledge Module

The Graphical User Interface (GUI) Module handles the interaction between the user (tutor or student) and the system. It accepts input from the user and directs the information to the other modules for processing. The Student Module contains the dynamic model of a student which stores the personal details and knowledge of the student. Input obtained from the GUI Module through the Teaching Strategies Module and Domain Knowledge Module is used to update the student model. The responsibilities of the Teaching Strategies Module include authoring and maintenance of pre-test, tutorials, alternative teaching strategies and post-test, stored in the Domain Knowledge Module.

The student's knowledge of the domain is evaluated using the overlay model and represented using the Bayesian theorem. Tracking information measured from the problem solving session is then directed to the fuzzy expert system to categorize the students.

The system was tested and evaluated by students enrolled in the BSc (Hons) Software Engineering and BEng (Hons) Electronics and Computing courses at KBU International College. The evaluation results showed that the Bayesian theorem is able to model the student's prerequisite knowledge effectively to guide them during the tutorial session.

## 1.4 Contribution

This research makes a contribution to the fields of 1) teaching and learning the C++ Standard Template Library (STL) as well as 2) web-based Intelligent Tutoring Systems architectures.

This ITS contributes to the teaching of C++ STL through intensive authoring tools provided by the system. This includes the authoring of the pre-test, tutorial sessions and post-test. The system is a complete tool to aid curriculum planner and implementer tutor in their teaching. One unique feature is that different computer programming languages can be represented in the system.

From experience, it is observed that students who are weak in their elementary programming, struggled in learning the C++ STL. Subsequently, the domain knowledge of this ITS is modelled in a 2-level hierarchical structure representing the prerequisites and the corresponding STL topics. The understanding of the student's prerequisites is determined

through a pre-test and represented using the Bayesian Theorem. Then, the conditional probabilities of understanding obtained are utilized to guide them during the tutorial session. The Bayesian Theorem provides a transparent student model to help students reflect on their ongoing progress.

The web development of this project applied industry practices in eXtreme Programming methodology and Java 2 Platform, Enterprise Edition (J2EE) to enhance the value and contribution of this research. These practices bring the C++ STL ITS closer to the industry level and promote its adoption in the educational sector. The 3-tier system architecture forms a strong foundation for future development of Intelligent Tutoring Systems.

## 1.5 Structure of the Thesis

Chapter 2 describes the general architecture of ITS which typically includes a Student Module, Domain Knowledge Module, Pedagogical Module and Graphical User Interface Module. It focuses on the techniques employed in student modelling and compares the various techniques.

ITS for programming is the main theme in Chapter 3. Ten ITSs for programming were selected for review, ranging from different eras, domain knowledge and techniques. Four aspects of the ITSs were compared: domain knowledge, level of programming, tutoring goals and student modelling techniques.

Chapter 4 introduces the domain knowledge of the ITS which is the C++ STL. It highlights the difficulties in learning and teaching C++ STL, discusses two issues with current ITSs – dynamic domain knowledge modelling and student modelling with transparency. This chapter places emphasis on uncertainties management and subsequently proposes the application of the Bayesian theorem for student modelling.

Chapter 5 presents the overall architecture of the C++ STL ITS which consists of four main modules: Student Modelling Module, Domain Knowledge Module, Teaching Strategies Module and Users Administration Module. The next section briefly describes the eXtreme Programming methodology. This is followed by the specification of the C++ STL ITS 3-tier system architecture which comprises of client-tier layer, middle-tier layer and data source-tier layer. Two key algorithms are illustrated in this chapter: the XML syntax parser

and student update model. Lastly, this chapter includes the implementation details of the fuzzy expert system using the MATLAB Fuzzy Logic Toolbox.

The C++ STL ITS system evaluation is furnished in Chapter 6. Evaluation methodologies and evaluation requirements are examined. The evaluation covers the C++ STL ITS architecture. Results from pre-test, tutorial, post-test and fuzzy stereotyping of students expert system are analyzed in detail. Next, the application of the Bayesian theorem and fuzzy logic are evaluated. This chapter ends with comparison of existing web-based ITSs.

Finally, chapter 7 concludes the research and proposes future work on the C++ STL ITS.

.

## **Chapter 2   Intelligent Tutoring Systems with Focus on Student Modelling**

Intelligent Tutoring System (ITS) is an advanced training software that mimics a human tutor by adapting its instructional approach to each individual student. ITS was developed to overcome the deficiencies of traditional Computer Aided Instruction (CAI) systems. These traditional systems were developed to provide students with instruction material in a particular topic after which they were tested. The typical flow of structures in CAI is depicted in Figure 2.1 (Smith, 1998).

Students are directed in the course of study by answering a series of questions. If the student answers correctly the next level of instruction is entered and subsequently, new problems are selected and presented. If a student answers the questions incorrectly the instructional material is presented again, perhaps in a slightly different format. If, after the material has been repeated, the student again answers incorrectly then remedial instruction is invoked. Remediation usually involves review of the earlier material and requires some attempt to identify and rectify the source of errors.

Figure 2.1 reveals that there are some places where flexibility can be introduced to improve the model. One area is to present different problems to solve before invoking the remedial lessons. These could be sub-problems of the original problem or similar problems. Another improvement is to provide a choice for remediation to allow more attempts of the same problem.

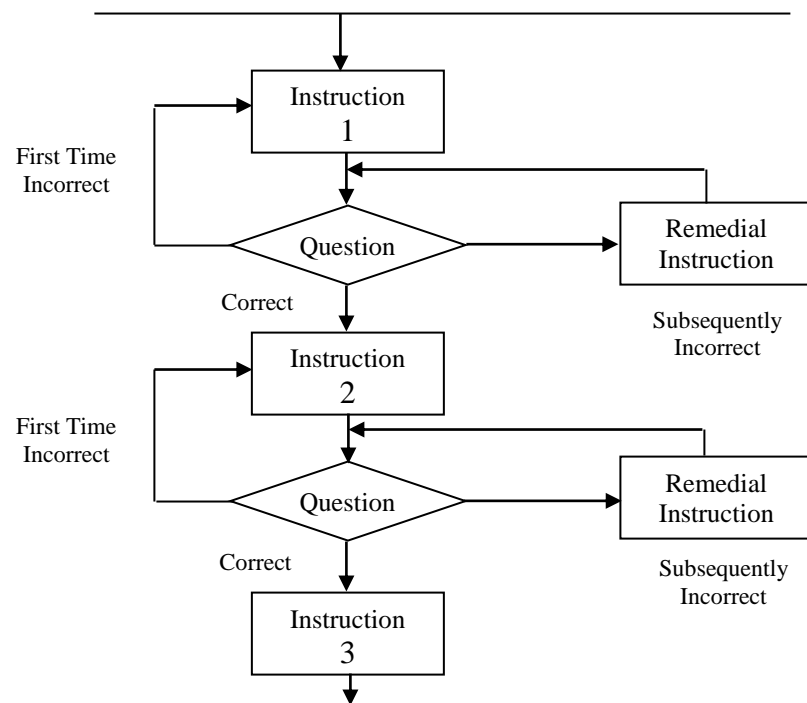


Figure 2.1 Routing within a Computer Aided Instruction System

(Smith, 1998)

Such systems seem to imitate intelligence by being able to adapt to student misconceptions. However, this is the result of the tutor anticipating all the possible errors that the student can make. The possibilities are hard-coded into the system. If the tutor anticipated an incorrect interpretation, then the system will not be able to provide a remedial instruction to the student. These systems are not capable of dynamically generating response to a particular situation like a human tutor.

In ITS, students interact with the system by answering simple questions about themselves and their understanding of the domain knowledge. The information is then used to find out the student's knowledge. This is achieved via an expert system. Then, the ITS will display the appropriate interface for the current student.

## 2.1 Components of an ITS

The basic architecture of an ITS consists of the following components (refer to Figure 2.2 below) :

- i) the domain knowledge, containing the structure of the domain and the educational content (Domain / Expert Model).
- ii) the user modelling component, which records information concerning the user (Student Model).
- iii) the pedagogical model, which encompasses knowledge regarding various pedagogical decisions (Teaching / Tutoring Model).
- iv) the user interface (Student Interface).

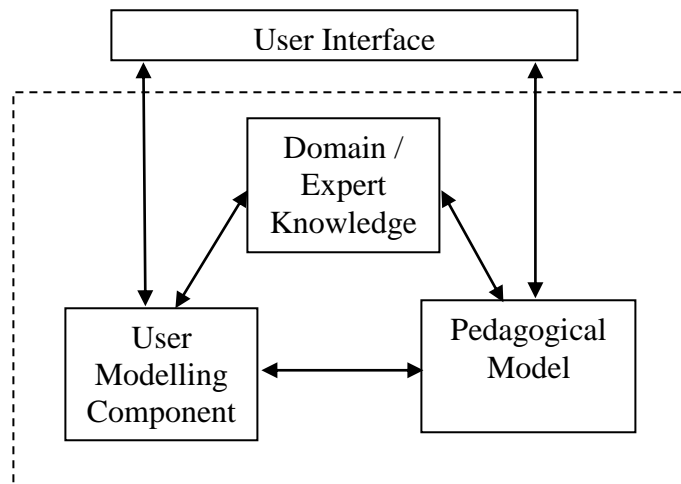


Figure 2.2 Typical Basic Architecture of an ITS

Shute and Psotka (1996) quoted that an early outline of ITS requirements was presented by Hartley and Sleeman (1973). They argued that ITS must possess : (a) knowledge of the domain (expert model), (b) knowledge of the learner (student model), and (c) knowledge of



teaching strategies (tutor model). Interestingly, this simple structure has not changed in more than 30 years (Shute & Psotka, 1996; Psotka, Massey, and Mutter, 1988; and Sleeman & Brown, 1982).

### *2.1.1 Expert Model*

The Expert Model provides the ITS with a representation of the domain knowledge. The representation should depend on the domain to be taught. It is considered the heart of an ITS as it heavily influences many other parts of the ITS. If the representation of the knowledge is poor, the whole ITS is deemed unsuccessful.

Experts are commonly classified as (Anderson, 1988) :

#### i) Black Box

In this model, the computer assesses the performance of the student without the need of “human intelligence”. The dialogue between the student and ITS is very simple as the computer responds with a simple yes/no to student’s answer. A clear disadvantage is, that the computer does not provide detailed explanation why the answer is incorrect (if it is).

#### ii) Glass Box

The goal in this model is to make the reasoning of the expert visible for the student. The problem solving capability is actually transparent to the student. One problem is that student may not understand the reasoning provided. The computer should be able to provide an alternative reasoning or reformulate the reasoning. It is important

that the expert system is well-structured to provide better explanation and instruction.

iii) Issue-based

Issue-based tutoring is a modified model of the Black Box. In this model, the programmer attaches instructions to specific issues observable in the behaviour of both the expert and the student. If the student fails to meet the prescribed behaviour criterion, the computer will give the feedback immediately. Feedback includes an explanation of the rule. The dialogue can be just simple feedback on the correct action or very complex by providing detailed reasoning behind the behavioural rule.

iv) Cognitive Modelling

Cognitive models simulate human problem solving which is how human uses knowledge. The model embodies 3 types of knowledge – declarative, procedural and heuristic. Declarative knowledge includes facts or concepts required to solve the problems. Procedural knowledge relates to step by step procedures on how a task is performed by the expert. Heuristic knowledge is the actions and rules of expert's experience in relation to problem solving.

A number of strategies are used to represent and organize knowledge. Some of these are :

- If-then-rules
  - Often called production rules.
  - Form : **IF** *condition(s)* **THEN** *conclusion(s)*

- If-then rules with uncertainty measures
  - Uncertainty measures can be used in rule based systems to indicate the level of confidence associated with a production rule.
- Semantic networks representation
  - Represents knowledge as a set of nodes connected by labelled arcs.
- Frame based representation
  - A frame is a collection of attributes and values and can be regarded as an extension of a semantic network.

### 2.1.2 *Student Model*

The Student Model represents the student's level of knowledge. The computer only has an impression of what the student knows. It influences how the knowledge is presented to the student and can be used to predict, describe, and explain student behaviour.

The Student Model may consist of four types of items :

- i) personal data (e.g name, email, age)
- ii) interaction parameters  
(information recorded from the interaction with the system)
- iii) knowledge of the concepts  
(include items like multimedia type preferences)
- iv) student characteristics  
(which can either be directly obtainable or inferable)

According to Wenger (1987), student models have three tasks.

- i) They must gather data from and about the learner. This data can be explicit - asking the student to solve specific problems or implicit - tracking the students navigation and other interactions and comparing them to information about similar learner responses.
- ii) They must use that data to create a representation of the student's knowledge and learning process. This often takes the form of "buggy" models that represent the student's knowledge in terms of deviations from an expert's knowledge. The system then uses this model to predict what type of response the student will make in subsequent situations, compares that prediction to the students' actual response, and uses that information to refine the model of the student.
- iii) The student model must account for the data by performing some type of diagnosis, both of the state of the student's knowledge and in terms of selecting optimal pedagogical strategies for presenting subsequent domain information to the student. One of the biggest challenges is to account for "noisy" data, the fact that students do not always respond consistently, particularly when their knowledge is fragile and they are uncertain about the correct responses.

### *2.1.3 Tutoring Model*

The Tutoring Model is also known as the Pedagogical Model or Instructional Model. It captures the expertise of a model tutor and uses it to determine when and how to teach and communicate with the student. Generally speaking, it represents the teaching process.

According to Smith (1998), the function of this model is three fold. Firstly, it must control the presentation, ordering and selection of material most appropriate for the student. Secondly, it must be able to answer questions from the student and thirdly, it must determine which type of help should be given to the students. The first two functions are commonly associated with the Curriculum Module where as the third function is represented by the Instruction Module.

To be effective, the ITS must meet the ever changing needs of the student. From the information in the Student Model, the ITS diagnoses the student's weaknesses and adapts the instruction accordingly. As the knowledge or skills of the students increases, ITS will ideally conform to the evolving level of the student's understanding from novice to an expert.

### *2.1.4 Student Interface*

The Student Interface is linked with the other three models to allow the student to explore the ITS learning environment. In general, it processes the flow of communication in and out the ITS.

There are two classical approaches to the dialogue: natural or command language and graphical interfaces with windows, icons, buttons, and so on. The ITS should be as self-explanatory as possible so that the student is able to interact with the ITS without any guidance from a human tutor. Command language is well understood but the language used may not be self-explanatory. It can be vague as it has still some restrictions. Graphical interface can be distracting as it may draw the student's attention to the interface itself (graphics, icons or various elements) rather than the knowledge.

The selected interface should be able to motivate the student to observe and test hypotheses as they interact with the ITS. The goal of this model is to ensure successful knowledge communication.

## 2.2 Student Modelling Techniques

Without an understanding of the knowledge and needs of a student, tutors cannot effectively achieve their purpose in teaching. Tutors are not teaching a domain but they are teaching students. Similarly, ITSs are for students. Hence, it is important that the system models the students and records information about the students as accurately as possible. This is necessary in order to facilitate individualized learning. A lot of research has focused on the student modelling techniques. This section is a summary of some of the work.

The aim of the student model is to guide the tutor in deciding the best approach or teaching strategies for a specific student. In making the decisions, several factors must be considered. These include the student's current knowledge and behavioural characteristics, the goals of the training or learning objectives, interaction with the students and so on.

Various methods have been used to construct the student model. These methods have also been applied in adaptive hypermedia and hypertext applications. They include :

- Overlay model
- Differential student model
- Perturbation student model
- Cognitive model
- Constraint-based Modelling
- Machine Learning (ML) techniques
- Logic-based methods
- Stereotype-based methods

The subsequent sections elaborate some of the methods above, together with some ITSs and their approaches to student modelling. Systems that use a combination of different techniques are also reviewed.

In the 80s, student models were devised to record misconceptions, missing conceptions or a combination of both. A misconception is described as knowledge that the student possessed but not by the domain expert. Missing conceptions are knowledge which the expert has but not the student. (VanLehn, 1988). The student model was classified as

overlay model, differential student model and perturbation student model or buggy student model.

### 2.2.1 Overlay Model (Carbonell, 1970; Carr & Goldstein, 1977)

In the overlay models, the student's knowledge is regarded as a subset of expert's knowledge and is extendable (Figure 2.3). It is particularly applicable when the teaching materials can be presented as a prerequisite hierarchy. In strict overlay model, incorrect knowledge is not included. In other words, it does not cater for misconceptions or bugs that the student may have or acquire during the tutoring process. Among the systems that use overlay model for student modelling are *GUIDON* (Clancey, 1992), *Grace Tutor* (Gray & Atwood, 1992), *CALAT* (Nakabayashi, et al, 1997) and listed in (Kavčič, 2000).

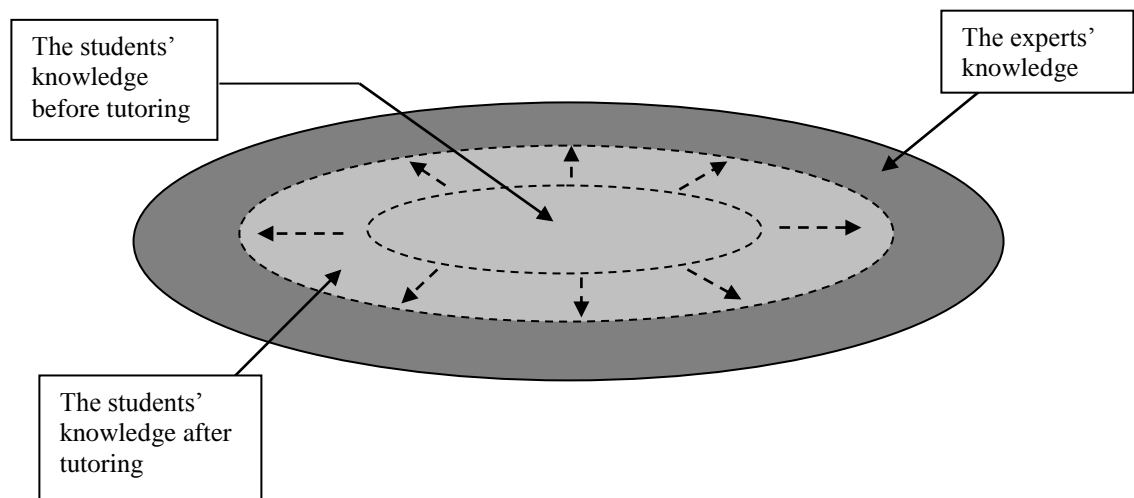


Figure 2.3 An illustration of the overlay model (Adapted from Smith, 1998)



### 2.2.2 Differential Student Model

To differentiate between knowledge the student has not yet acquired and knowledge the student has not yet been presented with, differential student model was introduced. It is used in the *WEST* (Burton & Brown, 1982) tutor. The model is depicted in Figure 2.4. The model partitions the domain knowledge into knowledge already exposed and that which has not yet been exposed to the student. Like the overlay model, the differential model does not consider the mistakes made by students.

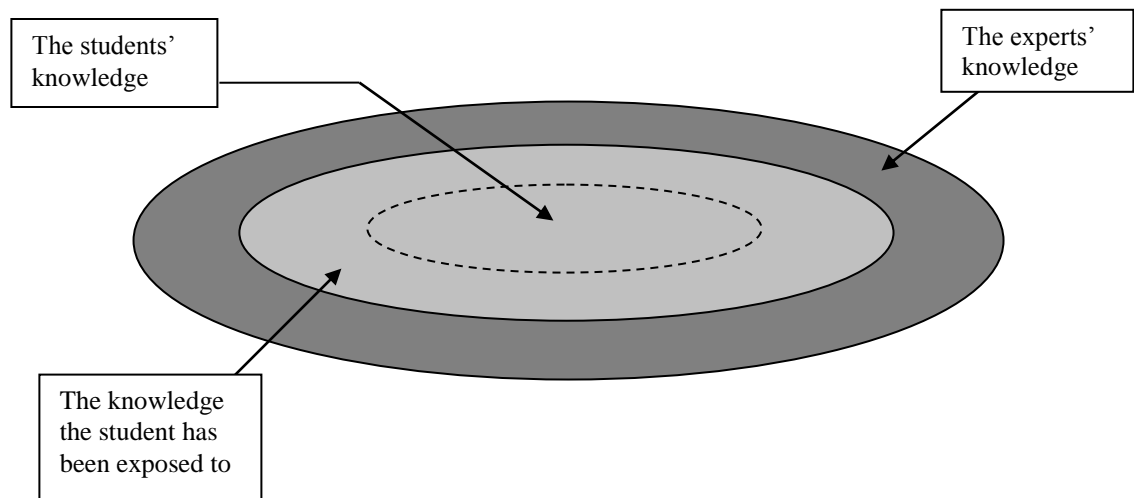


Figure 2.4 An illustration of the differential model (Adapted from Smith, 1998)

### 2.2.3 Perturbation Model or Buggy Student Model

The perturbation model or buggy student model is illustrated in Figure 2.5. This model considers knowledge which is not in the expert domain knowledge known as misconceptions or bugs. A bug library is included in the expert knowledge. The

list of possible bugs can be produced through analysis of the problem domain and errors made by students. This process is known as enumeration. The bug library can also be generated from an underlying cognitive theory. The goal of tutoring is to increase the student's knowledge, as in the overlay model, and at the same time eliminate any bugs.

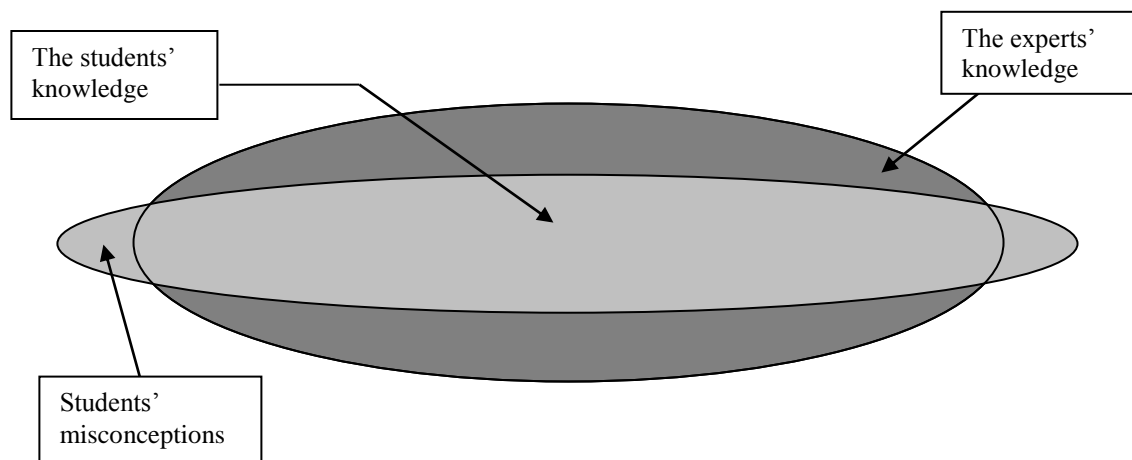


Figure 2.5 An illustration of the perturbation model or buggy student model  
(Adapted from Smith, 1998)

The first ITS based on the study of bugs was developed in 1975 by Brown and Burton, known as BUGGY. The developers attempt to enumerate the different possible procedural bugs students might acquire while trying to solve algebra problems. Using its bug library, BUGGY could produce general diagnostic tests to identify students' mistakes. The use of a static bug catalog restricts the application of BUGGY to well defined procedural domains. Extending the BUGGY model, the bug library in the *IDEBUGGY* tutoring system (Brown & Burton, 1978) is developed by enumerating the bugs present in a database of student tutor

interactions. It contains knowledge about frequent irregularities in the students' behaviour to treat misconceptions. Both systems fail to represent the semantic nature of a bug or to explain how a bug was generated.

The *PROUST* system (Johnson, 1986) performs a sophisticated diagnosis of student errors based on a bug catalogue and acts as a consultant for the novice programmer. In *ASSERT* (Baffes & Mooney, 1996), student model is automatically constructed and the bug library is refined during the tutoring process using a refinement theory.

#### 2.2.4 *Cognitive Model*

Cognitive Model is the basis for two well-known student modelling techniques : model tracing and knowledge tracing (Anderson et al, 1995).

Model tracing consists of matching every problem-solving action taken by the student with the steps of the expert's solution model of the problem being solved. The result of this matching is used to decide when to provide feedback and the type of feedback as the student progress through the problem.

Applying the model tracing paradigm, *SINT* - a Symbolic Integration Tutor (Mitrovic, 1996), is capable of solving problems step-by-step along with the students. *SINT* monitors the students while solving problems, informs the student of errors and provides individualized help and timely advice. The approach is not only incremental but interactive, since it involves the students in explicit dialogues about

their goals. The student model is used to guide the generation of instructional actions, like generation of explanations and new problems.

*Miss Lindquist* (Heffernan, 2001) is an ITS designed to engage students on a tutorial dialog about symbolization. In *Miss Lindquist*, the model tracing paradigm is expanded so that it does not only has a model of the student, but also has a model of tutorial strategies. *Ms Linquist* has tutored over 600 students at [www.AlgebraTutor.org](http://www.AlgebraTutor.org).

Crowley and her team (2003) have developed the foundations of model-tracing ITS in their work on *SlideTutor* for teaching microscopic diagnosis in Dermatopathology. *SlideTutor* is designed to provide individualized tutoring to students as they search, and interpret virtual pathology slides.

Knowledge tracing employs a simple two-state learning model and Bayesian updates. This student modelling facility is used in the LispTutor to implement mastery learning (Anderson & Reiser, 1985). The Bayesian probability is also used to estimate the probability that the student had learned each of the rules in the cognitive model.

### 2.2.5 Constraint-based Modelling (CBM)

Constraint-based Modelling (Ohlsson, 1994) represents both domain and student knowledge in the form of constraints, where constraints represent the basic principles underlying the domain. Symbolically, the constraint is represented in the form  $\langle Cr, Cs \rangle$  where  $Cr$  is the *relevance condition* and  $Cs$  is the *satisfaction condition*. The constraints define which problem states are consistent (or correct), and which are not. A constraint is relevant to a problem if the  $Cr$  is true. All relevant constraints must also be satisfied for the problem state to be correct. Otherwise, the problem state is incorrect and feedback can be given depending on which relevant constraints had their satisfaction condition violated.

In the *CFG-MINTS* - Context-Free Grammar Multimedia Intelligent Tutoring System, CBM reduces complexity of student modelling by focusing on faults only and the analysis is reduced to pattern matching (Reyes et al, 2000).

*CAPIT* - Capitalisation And Punctuation Intelligent Tutor, is designed for, and evaluated with, school children in the 10-11 year old age group (Mayo, et al, 2000). The system represents the domain as a set of constraints specifying the correct patterns of punctuation and capitalisation, and feedback is given on violated constraints. *CAPIT* is the second ITS to implement Ohlsson's CBM, the other is a tutor for the SQL database language. (Mayo & Mitrovic, 2000) (Mitrovic & Ohlsson, 1999).

### 2.2.6 Machine Learning (ML) Techniques

Machine Learning is an area of AI research concerned with developing computational theories of learning process and building machines that *think* and learn. An informal definition of ML is “the ability of a machine to improve its performance based on previous results” (dli.grainger.uiuc.edu). Tom Mitchell (1997) provides a formal definition of ML in his book on Machine Learning :

“A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ ”.

In the definition, Mitchell explained that to have a well-defined learning problem, three features must be identified : the class of tasks  $T$ , the measure of performance to be improved  $P$ , and the source of experience  $E$ .

Some learning strategies or techniques of ML include rote learning, inductive reference, Stochastic Bayesian inference, deductive inference, reinforcement learning, neural network learning, clustering, analogical learning and case-based reasoning. Multi-strategy learning is commonly practiced. The applications of some of these techniques in building ITS are discussed in Section 2.3. Sison and Shimura (1996) have also reviewed and compared main applications of machine learning to student modelling in the 80s and early 90s.

### 2.2.7 *Stereotype-based methods*

Stereotype-based methods consist of rules for triggering stereotypes on the basis of students' actions. Individual student is assigned to one or more stereotypes according to his or her level of knowledge. There are four stereotypes that concern the knowledge of student: novice, beginner, intermediate and advanced.

*Web PVT* (Virvou & Tsiriga, 2001) uses stereotypes in three dimensions to initialise the student model. One dimension concerns the knowledge level of the student. A second dimension concerns the degree of carelessness of a student. In this category, there are three stereotypes: very careful, averagely careful and careless. Finally, a third dimension concerns the student's knowledge of other languages: student's mother tongue and foreign languages the student may already know. Based on their performance on a preliminary test and their answers to a questionnaire, students are initially assigned to one of the stereotypes relating to each dimension.

*HyperTutor* (Perez et al, 1995) an adaptive hypermedia system employs a pure stereotype for student modelling and the student can be categorized as novice, medium or expert through some exercises.

### 2.2.8 *Combination of Methods*

A combination of two or more methods is commonly applied to construct the student model. Different methods are used for initializing and maintaining the student model to provide more accurate modelling and allow better analysis of gathered information.

Multi-layered overlay models and Episodic User Modelling are used in the web-based *ELM-ART II* for learning LISP. Individual student learning history in former solving situations is collected in a database of episodes. The episodes represent student's individual learning history, their behaviour and former problem solving situations.

The *Conceptual Helper ITS* (Albacete & VanLehn, 2000) tutor is basically a model-tracing ITS enhanced by the use of probabilistic assessment to guide the remediation. It is designed to coach students through physics homework problem solving of a qualitative nature, i.e., those problems that do not need the use of algebraic manipulation to be solved but require the application of conceptual knowledge.

A combination of a stereotype and an overlay model is used to represent the student's knowledge of the domain in the web-based ITS using hybrid rules by Prentzas et al (2002). The student model consists of personal data, interaction parameters, knowledge concepts and student characteristics. The student



characteristics are represented with two stereotypes: directly obtainable (multimedia type preferences, available Internet connection and educational content preferences) and inferable (knowledge level, concentration level and experience in using ITS). Its overlay model is based on the concepts associated with the course learning units on Internet technologies.

### 2.3 Machine Learning Techniques in ITS

Beck and Stern (1999) described several promising technologies that have the potential to greatly impact the Artificial Intelligence (AI) and Education community. They have provided a summary of how these techniques can be used to solve a variety of problems relevant to the community. One of the applications of AI is in the construction of self-improving tutors. The tutor has the ability to draw conclusions about effectiveness of particular parts of an Intelligent Learning Environment (ILE) and learn about teaching strategies. Currently, specific teaching strategies are hard coded into ITSs. It is then assumed that the teaching style is suitable for all students under all circumstances. However, a good tutor must apply various teaching strategies depending on the student. It was pointed by Beck and Stern that one of the biggest challenges is how to get different teaching strategies to work together in a single ITS. Moreover, it is essential for the teaching model in the ITS to have the ability to direct students to the most appropriate teaching strategies. Undoubtedly, it is suggested that machine learning is a very suitable technique to learn the interactions between the student and the ITS, guiding the student to the appropriate tutorial sessions. This section discusses the application of some well-known machine learning techniques in the development of ITS.

### 2.3.1 Case-based Reasoning

Case-based reasoning (CBR) (Hopgood, 2001) is a problem solving paradigm that is used to solve a new problem by remembering a previous similar situation and by reusing or recalling information and knowledge of that situation. CBR involves two difficult problems: determining the relevant case and adapting the case to the current situation. Two ITSs that applied the CBR are described below – *PROWIZ* (ETU Projects, 2001) and *RMT* (Wiemer-Hastings & Malatesta, 2001).

*PROWIZ* (Programming Wizard) uses CBR approach for identifying errors and providing solutions to the problems. It aims at guiding Java programmers to solve syntax errors. The main tasks that *PROWIZ* have to deal with are to identify the problem situation, find a similar past case, use the case found to suggest a solution, evaluate the solution and finally update the system by learning from this experience.

Taking a case-based approach, the *RMT* (Research Methods Tutor) presents a research question to the student, and asks the student to address the question through in-depth discussions. It also allows the system to develop the student's analogical reasoning. *RMT* brings in related research paradigms to help the student infer both similarities and differences with their approach. *RMT* also employs the Structured Latent Semantic Analysis (SLSA) language analysis system (Wiemer-Hastings, 2000). This system uses part-of-speech tagging, anaphora resolution, and shallow parsing to split input sentences into their subject, verb, and object segments

and to replace pronouns with their antecedents. Compared to the standard LSA (Wiemer-Hastings & Zipitria, 2001), this technique provides a better match to human similarity judgments. Additionally, this allows the tutoring system to know what part of the student's answer matched an expected good answer, and what part did not match. This efficient matching allows *RMT* to have a more effective dialog with the student, thus leading the student to the complete correct answer.

### 2.3.2 *Theory Refinement*

Baffes and Mooney (1993) approach in applying a machine learning technique called *theory revision* is a novel idea. The key application of the theory refinement to student modelling is to facilitate the automatic construction of a bug library. The bug library can contain both common and unique misconceptions. This is very useful in guiding students to different teaching strategies.

### 2.3.3 Fuzzy Logic, Neural Network or Hybrid

Fuzzy logic techniques have been used to manage uncertainties in student modelling, such as behaviour, understanding and cognitive abilities. Fuzzy logic uses the continuum of logical values between 0 and 1, instead of just 2 Boolean values. Zadeh (1965), the founder of fuzzy logic provides the following definition for fuzzy logic :

“Fuzzy logic is determined as a set of mathematical principles for knowledge representation based on degrees of membership rather than on crisp membership of classical binary logic”.

In classical set theory, crisp set  $A$  for a universe  $X$  is defined as function  $f_A(x)$  called the characteristic function of  $A$  (Negnevitsky, 2005) :

$$f_A(x) : X \rightarrow 0,1 \quad (\text{two-valued Boolean logic}) \quad (2.1)$$

where

$$f_A(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases}$$

Crisp set theory is represented by two-valued Boolean logic: true (1) or false (0). This logic cannot represent vague concepts. It can answer the question, ‘Does the student understand the topic?’, and say ‘yes’, but not the question, ‘How much does the student understand?’ Fuzzy logic is able to answer the latter question. Therefore, it is very suitable for student modelling in ITS.

Fuzzy logic is built upon fuzzy set which has the capability to model a smooth transition across a boundary. In the fuzzy set theory, fuzzy set A of universe X is formally defined by function  $\mu_A(x)$  called the membership function of set A (Negnevitsky, 2005) :

$$\mu_A(x) : X \rightarrow [0,1] \quad (2.2)$$

where

$$\mu_A(x) = 1 \text{ if } x \text{ is totally in A}$$

$$\mu_A(x) = 0 \text{ if } x \text{ is not in A}$$

$$0 < \mu_A(x) < 1 \text{ if } x \text{ is partly in A}$$

The membership function assigns a membership number to each element  $x$  of the universe. Applying a suitable membership function to the set of values will produce a graceful transformation across a boundary.

Fuzzy sets theory is used in fuzzy inference process to map a given input to an output. In student modelling, the input comprises information such as student's actions and knowledge, and the desired output is knowledge gained and problem solving abilities.

In Brilliant Scholar Series 1 (BSS1) (Warendorf & Tsao, 1997) for tutoring subjects such as mathematics and sciences, a general fuzzy logic engine was developed to better manage student's learning. The main elements in the engine are: a knowledge base, a fuzzification unit, a fuzzy logic reasoning unit and a defuzzification unit. The engine is able to read in a knowledge base at runtime and

use this knowledge to perform inference on input variable. As the required internal parameters are fixed, an average engineer would be able to design a simple expert system using the inference engine.

Stathacopoulou and team (1999) proposed a *neuro-fuzzy* synergism for student modelling. Fuzzy logic techniques are used to provide human-like approximate diagnosis of student's knowledge and cognitive abilities. On the other hand, neural networks are trained to imitate human teacher's decisions regarding student's characteristics, knowledge and cognitive abilities in a domain. The combination of these two techniques enhances the student model enabling the ITS to mimic a human tutor.

Neural network was used to assess whether the student is struggling with a problem and predict the appropriate problem for the student to solve (Wang & Mitrovic, 2002). The assessment of errors performed well but the selection of problems was less successful.

#### 2.3.4 *Bayesian Network or Probabilistic Model*

A Bayesian Network or probabilistic model (Heckerman, 1996) is a graphical model or representation of a probability distribution among a set of variables. With a rigorous formalism, the model is able to learn the parameters and structure of a Bayesian network from data or a combination of data and prior knowledge. It has

become a popular representation for encoding uncertain knowledge in expert systems.

The model is commonly expressed as a directed acyclic graph. Each node in the graph corresponds to a piece of conceptual knowledge that the student is expected to learn or a misconception that the tutor can help remedy. The arcs represent the underlying dependencies of the domain knowledge. Each node has a value attached to it that indicates the probability that the student will apply the piece of knowledge when it is applicable. As the student solves a problem or applies knowledge, the probabilities are updated according to the actions taken by the student.

*AnimalWatch* (Arroyo et al, 2003) maintains Bayesian-probabilistic overlay student model that allows it to make inferences about each student's knowledge during the problem solving tasks. Based on these estimations about how students perform in relation to the problems that are given to them, *AnimalWatch* adjusts its problem selection to give students problems that will challenge them, so that they improve with the guidance provided by the system.

*WITS - Whole-course Intelligent Tutoring System* is essentially an expert system which tries to simulate the expertise of the tutor (Calleary, 1997). The student model aims to be the type that tutors use, and is built up using multiple choice questions, with probabilities attached to the possible answers. Besides awarding marks to student's correct answers, each answer is significant and is assigned a

corresponding probability that contributes to an overall probability that the student is benefiting from the course.

The student modelling based on Bayesian Network is also applied in the *VITAL* architecture for teaching statistics (Madigan et al, 1995) and in (Fernandez & Sison, 2001) to model the student's understanding in a programming domain.

In *VITAL*, the domain knowledge is represented with a Bayesian network graphical model. Each node in the graph corresponds to a facet with probabilistic links. A facet is a piece of content knowledge used by the student. Hence, the student model is represented by a probability distribution over the facets. Teaching operators or instructional module are associated with each facet to update the probabilities during interactions with individual students. The design of the domain model is a cumbersome task due to the need to identify all the concepts and misconceptions that students have. Moreover, for each group of facets, appropriate teaching operators and instructional strategy need to be devised.

The Bayesian student model proposed by Fernandez and Sison (2001) represents a programming learning hierarchy for novice programming. The model depicts the prerequisite relationships among topics with nodes containing probabilities that the student has acquired those skills. A number of issues are still outstanding in the research: establishing the rules for setting and updating prior probabilities and dependencies among the network nodes, customising the updating algorithm, identifying type of student misconceptions and employing a learning algorithm.



## 2.4 Neural Networks vs Bayesian Networks

Artificial Neural Networks (ANNs) are biologically inspired structures of generalized mathematical models of human cognition. An ANN architecture consists of neurons that are connected together either in single layers or in multiple layers, and processing information in parallel. Figure 2.6 shows an ANN topology with 5 input neurons, 4 hidden neurons and 2 output neuron. Signals are passed between neurons over connection links. Each connection link has an associated weight which multiplies the input. The weight is altered throughout the “training” period. A given neural network can be trained either in the supervised or unsupervised mode. In supervised mode, training is achieved by presenting the network with the input data and its associated target output values. The weights of the network are then adjusted according to a given learning algorithm. The weights are modified to make the network output as close as possible to the expected output. Only input values are provided in an unsupervised training mode. Similar input data are grouped together and trained with various weights to produce the same output values.

Each neuron applies a non-linear activation function to the sum of its weighted inputs to derive its output signal. Initially, the weights are assigned randomly. The network learns by propagating the prediction error backwards through the network, to modify the weights. This is the most widely used learning algorithm known as the backpropagation learning algorithm.

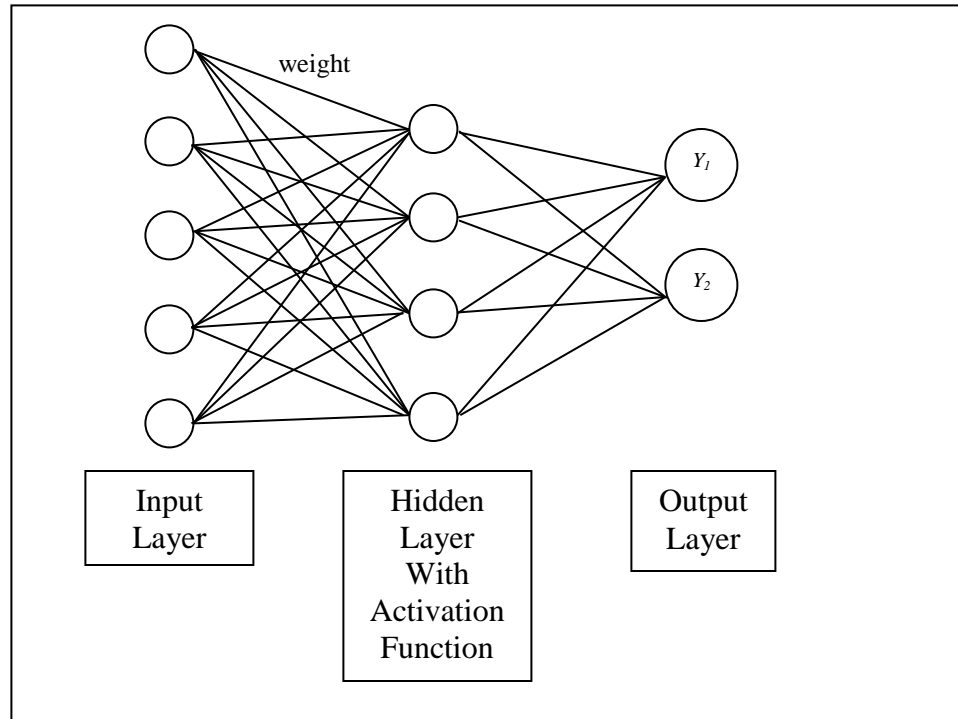


Figure 2.6 A Neural Network Topology

NN has the ability to deal with variance in the data provided. When presented with incomplete data, NNs are capable of producing an approximate answer rather than an incorrect one. Similarly, when presented with data that was not previously used in the training, the network will generally produce a reasonable output. They are frequently used in classification tasks and its parallelism makes them highly suited to parallel processing computers.

However, much time and effort are required to design and train the network with various weights to produce the optimal result. The NN designer has to perform a large amount of trial and error to configure the network for a particular problem. The configuration includes the number of inputs, appropriate number of hidden nodes and the desired output. There is

no clear rule to determine the number of hidden nodes in the network topology. Much training effort is then required to obtain the optimal number of hidden nodes. The learning rate, momentum rate and other parameters have to be determined as well during the training. In most cases, the design is still poorly understood and it is treated like a black box. (Beck & Woolf, 1998, Prentzas et al, 2002). The neural systems are just not intuitive for end users to understand the reasoning process.

Bayesian Networks (BNs) are named after Rev. Thomas Bayes, an 18th-century Presbyterian minister and mathematician. He devised a mathematical formula for explaining how existing beliefs should be changed in the light of new evidence for or against the belief using probability theory. The basic probability theory is examined in the next chapter.

A BN consists of directed acyclic graph and a corresponding set of conditional independence probability distributions. The probabilities are encoded into the nodes in the graph representing the strength of the relation among the nodes. To handle uncertainty, the probability of a belief or assertion is updated based on new evidence on the assertion.

Bayesian updating is based upon a well proven statistical theorem – Bayes’ Theorem. The technique employs deductive probabilities which are easier to estimate and understand than abductive one. Initial values for the probability of an event (effect) occurring given another event (cause) are included in the graphical model. The theorem is applied to update an effect in response to one or more causes.

Both Neural networks (NN) and Bayesian networks (BN) have the ability to deal with uncertain knowledge. Both networks can automatically generate predictions or decisions even when key pieces of information are missing. They are also both represented diagrammatically with a graph-like structure containing nodes. In NN, each node holds a value corresponding to the domain problem, whereas in BN each node contains a probability. Weights in the NN are randomly generated. However, the probabilities in the nodes of a BN are assigned in an ad hoc manner. BN maps out a cause-effect relationship among the key variables in a problem. An Artificial NN (ANN) models a similar kind of relationship whereby the input neurons in an ANN are generalized to produce the desired output.

NNs need to be trained exhaustively before it can reach an optimal prediction. One could never get sufficient data to train a network. Its dependency on historical data is an apparent limitation. Hence, it cannot give an optimal prediction with unforeseen data. Without the need of training previous data, BN is able to generate a good predication as the heart of BNs is the Bayes' theorem which is a proven statistical theorem. This enables the BN to offer an efficient method to handle ambiguity or lack of information. In the *VITAL* architecture (Madigan et al, 1995), the Bayesian graphical model approach is shown to be more straightforward and transparent than a NN approach.

Table 2.1 summarises the findings for the comparison of NN and BN. Overall, BN bypasses NN as the more popular and beneficial technique in the development of ITS.

Table 2.1 Similarities and Differences in Neural Networks and Bayesian Networks

NEURAL NETWORKS	BAYESIAN NETWORKS
<b>Similarities</b>	
Ability to deal with uncertain knowledge.	
Generate predictions even with incomplete data.	
Represented diagrammatically with graph-like structures.	
Ad hoc node values.	
Models cause-effect relationships.	
<b>Differences</b>	
Often regarded as a “black box”	Model is inspectable
Exhaustive training required to learn first and design the optimal network – pre-learning required.	Learns as it updates the network – on-going learning.

## 2.5 Summary

The following table summarises the student modelling techniques discussed above, together with the ITSs that employ the technique and brief evaluation. The techniques are categorized as Primary models, Cognitive model and Machine Learning. The overlay, buggy and stereotype-based models are classified as Primary models. These models are classic and applied in most of the ITSs. Machine Learning techniques employed include neural network, fuzzy logic, Bayesian network or probabilistic model, case-based reasoning, constraint-based model, theory refinement and neuro-fuzzy.

Table 2.2 Summary of Student Modelling Techniques

STUDENT MODELLING TECHNIQUES	INTELLIGENT TUTORING SYSTEMS	BRIEF DESCRIPTION AND EVALUATION
<i>Primary Model</i>		
Overlay Model	GUIDON GraceTutor CALAT WILEDS (Kassim, et al, 2001)	The goal of tutoring is to enlarge the student's knowledge which is assumed to be a subset of the expert's knowledge. Most widely employed. Student's misconceptions are not considered. Lack accuracy.
Buggy Model	BUGGY IDEBUGGY PROUST ELM-ART II ASSERT	Maintains a bug library – static and dynamic.  Time consuming to build. In a static bug library, it is not possible to consider all bugs or misconceptions. ASSERT overcomes this problem by refining the bug library as students progress in their learning.
Stereotype-based	Web PVT Hypertutor	Assigns a stereotype to categorise students. Simple and straightforward. Powerful in providing considerable information based on a few observations.  There are uncertainties in the classification. Need to assume an initial stereotype. Lack accuracy.
<i>Cognitive Model</i>		
<ul style="list-style-type: none"> <li>Model tracing</li> <li>Knowledge tracing</li> </ul>	LispTutor Geometry Tutor (Anderson et al, 1986) Miss Linguist SINT SlideTutor PAT (Algebra Tutor) (Ritter et al, 1998)	Well-established and successful.  Drawback – Teaching strategies are directive.

Table 2.2 (continued)

<i>Machine Learning</i>		
Neural Network	<p>On the use of NN in ITSs</p> <p>On the effectiveness of a NN for Adaptive External Pacing</p>	<p>Behaves like an expert if trained successfully.</p> <p>Used to simulate student's cognitive process and for adaptive external control of student's learning speed.</p> <p>Produce an approximate answer if data is incomplete or noisy.</p> <p>Time consuming to design a suitable network – need trial and error.</p> <p>Reasoning is obscure.</p>
Fuzzy Logic	<p>BSS1 Tutoring System (Warendorf &amp; Tsao, 1997)</p> <p>Sherlock II (Katz et al, 1992, quoted in Stathacopoulou et al 1999)</p>	<p>Able to handle imprecise and vague information, such as student's actions, knowledge and performance.</p> <p>Difficult to model the problem suitably, define fuzzy parameters and suitable membership functions for fuzzy sets, and develop a comprehensive set of rules relating input and output parameters.</p>
Bayesian Network or Probabilistic Model	AnimalWatch WITS VITAL	<p>Technique is based on a proven statistical theorem.</p> <p>An ad hoc technique.</p> <p>Need to assume a probability to a knowledge acquired or action taken.</p> <p>Difficult to obtain accurate estimates of likelihood of events and combinations of events.</p>
Case-based Reasoning	PROWIZ RMT	<p>Encourage students to refer to previously solved problems for feedback.</p> <p>Used to improve adaptability of ITS.</p> <p>Determining and adapting a case involves a certain complexity.</p>
Constraint-based Model	CFG-MINTS CAPIT	<p>Natural representation for multiple correct solutions to the same problem.</p> <p>Constraints need to be carefully designed to meet required objectives.</p>

Table 2.2 (continued)

Theory Refinement	ASSERT	<p>Used to introduce errors into an initially correct knowledge base to model incorrect student behaviour. Automatically makes revisions to improve accuracy of knowledge.</p> <p>Limited domain application. Most theory refinement algorithms are designed for classification.</p>
<p>Hybrid</p> <ul style="list-style-type: none"> <li>• Neuro-fuzzy</li> <li>• Neurules</li> </ul>	<p>Neural Network-based Fuzzy Modelling of the Student</p> <p>A Web-based ITS using Hybrid Rules</p>	<p>Fuzzy logic techniques are used to evaluate student's knowledge and cognitive abilities. Neural Networks are trained to evaluate student's characteristics and decide appropriate teaching strategy. Permits representation and processing of incomplete, imprecise and vague information about the student.</p> <p>Hybrid rules integrating symbolic rules with neurocomputing. Used to derive values of inferable student characteristics (knowledge, concentration level and experience).</p> <p>Benefits: Time-efficient, space-efficient, able to conclude based on partial inputs, easy to update, produce natural explanations on conclusions and enable exploitation of various knowledge sources.</p>

Table 2.3 below furnishes the type of applications that are suitable for a specific student modelling technique. In general, overlay model, stereotype-based and cognitive model have a wider scope of applications. The other techniques are more application-specific. Another trend observed is that almost all of these techniques have been applied in the



domain of programming. It shows that intelligent tutoring systems are commonly implemented to teach the syntax of a programming language.

Table 2.3 Area of Applications of Student Modelling Techniques

<b>STUDENT MODELLING TECHNIQUES</b>	<b>AREA OF APPLICATION</b>	<b>CURRENT APPLICATIONS</b>
Overlay Model	General. Applicable to all domain knowledge.	<ul style="list-style-type: none"> <li>• Programming Language</li> </ul>
Buggy Model	Most applicable to programming language domain knowledge.	<ul style="list-style-type: none"> <li>• Programming Language</li> </ul>
Stereotype-based	General. Applicable to all domain knowledge.	<ul style="list-style-type: none"> <li>• Language – Passive Voice</li> <li>• Programming Language</li> </ul>
Cognitive Model	General. Applicable to all domain knowledge.	<ul style="list-style-type: none"> <li>• Programming Language</li> <li>• Mathematics</li> <li>• Medical</li> </ul>
Neural Network	Applications that contain a significant large amount of data that can be trained to act like an expert.	<ul style="list-style-type: none"> <li>• Programming Language</li> </ul>
Fuzzy Logic	Applications that need to manage uncertainties or vagueness.	<ul style="list-style-type: none"> <li>• Programming Language</li> </ul>
Bayesian Network or Probabilistic Model	Applications that need to manage uncertainties based on existing probabilities.	<ul style="list-style-type: none"> <li>• Programming Language</li> <li>• Statistics</li> </ul>
Case-based Reasoning	Law and Medical – Previous cases can be identified, analysed and discussed.	<ul style="list-style-type: none"> <li>• Programming Language</li> <li>• Research Methods</li> </ul>
Constraint-based Model	Pattern matching.	<ul style="list-style-type: none"> <li>• Grammar</li> <li>• Database Language</li> </ul>
Theory Refinement	Limited applications. Mostly designed for classification.	<ul style="list-style-type: none"> <li>• Programming Language</li> </ul>

## Chapter 3 ITS for Computer Programming

The application of ITS encompasses a wide range of education and training domain. These include flight training simulation, medical sciences, law, engineering and computer science. Its extensive coverage manifests the effectiveness of ITS in providing one-to-one tutoring. One major area is the research and development of ITS for computer programming. ITSs to teach various programming languages from the 1980s to the current millennium have matured in the areas of the choice of programming languages, level of programming, tutoring goals and application of Artificial Intelligence techniques.

### 3.1 Prior Work – Overview

#### 3.1.1 *LispTutor (Reiser, Anderson & Farell, 1985)*

LispTutor was developed to teach the basic principles of programming in LISP. Students are provided with a problem description and are required to enter their code. The LispTutor guides their left-to-right, top-down attempts in a highly directed fashion by interpreting their code as a correct or buggy solution and advising immediately when errors occur.

LispTutor is an application of the Adaptive Control of Thought (ACT\*) theory of cognition (Anderson, 1983); and now the ACT-R theory (Anderson, 1993). The theory distinguishes between declarative knowledge (eg. knowing a theorem) and procedural knowledge (eg. an ability to apply the theorem). Much of the theory has

been concerned with the acquisition of cognitive skills which refer to a set of production rules acquired in the domain. The expert model was created as a series of correct and buggy production rules of a Goal-Restricted Production Systems Architecture (GRAPES). It has 375 correct production rules and about 475 buggy production rules to diagnose the student's errors. The effectiveness of these rules depends on the complexity of the lesson and how well the lessons were implemented. Obviously, developing the rules is a tedious and time consuming task.

To model the student, LispTutor employs model tracing or the knowledge tracing model. Model tracing systems analyze problem solving activities and maintain a model of problem solving which is traced against the students' actions. Applying an overlay model, any differences between the students' solution and the systems can be acted upon and suitable action taken. Three components are utilized in the implementation of the model tracing methodology: expert's model, bug catalogue and pedagogical strategies.

The model tracing method has the advantages of early diagnosis of students' misconceptions, providing immediate feedback to the students. The students are always guided closely to the correct solution. However, the steps are rather rigid and the students are not encouraged to explore other possibilities or alternative solutions.

The evaluation results demonstrated that the LispTutor has achieved its tutorial goals and is more effective than the standard teaching strategies. However, its

highly directive nature is more suitable for less experienced students. Better students prefer more control of the problem-solving tasks and not being constrained in some unnecessary way.

### 3.1.2 *PROUST (Johnson, 1986)*

PROUST (Program Understanding for Students) involve students entering code to solve a given problem description. PROUST is an intention-based diagnosis system for novice Pascal programmers. In intention-based diagnosis, errors in a program are found through a process of understanding the intended program design, and determined whether those intentions were satisfied. It has the capability to identify a wider range of errors because it allows detailed diagnosis of design errors. Johnson (1986) defined intention-based error diagnosis as follows:

“A system for diagnosing errors in artifacts is intention-based if it finds errors based on an interpretation of the cognitive process which generated the artifact, rather than on an interpretation of the structure or behaviour of the artifact itself.”

Based on the program description and a given solution, PROUST analyses novice programs, diagnoses each program's non-syntactic bugs, and explains the bugs and their causes to the students. It is also a buggy-based system driven by a bug-catalogue that identifies the misconceptions that students may have in solving Pascal programs.

As a program analyser, PROUST analyses student programs and gives a printed feedback on the errors found, but it is not integrated within a real learning environment. Therefore, students are not allowed to request feedback as programming progresses. It is not a programming tutor, and lacks a pedagogical model. Moreover, the problem descriptions to analyse a program are limited. Writing a problem description of a program is also time consuming and not easy for an ordinary tutor. Another limitation of PROUST is that it has difficulty in dealing with syntactic variabilities. There are still bugs that are not encountered for, and PROUST cannot explain them appropriately to the students.

### 3.1.3 *BRIDGE (Bonar and Cunningham, 1988)*

In BRIDGE, the student is presented with an introductory programming problem and passes through three phases while solving the problem in Pascal.

In the first phase, the student constructs a set of step-by-step instructions by selecting the English phrases from the Natural Language Plan Selection menu. In Phase 2, the student matches these instructions to programming plans which are standard concepts and techniques for programming, and builds a program. Briefly, the student matches the plans to programming language constructs and uses these to develop a programming language solution to the original problem.

BRIDGE is highly interactive and it has been designed for students who have some familiarity with programming plans. The terminology and complex screen display of BRIDGE is rather overwhelming. The initial evaluation showed that students had difficulty in using BRIDGE, particularly in Phase 2. They lost track in the matching of their English phrases from Phase 1 with a menu selection of programming plan names. In addition, the teaching strategy is rather inflexible.

#### 3.1.4 ASSERT (*Baffes and Mooney, 1996*)

ASSERT (Acquiring Stereotypical Student Errors by Refining Theories) is a generic program for building ITS using the idea of refinement-based modelling and refinement-based remediation. ASSERT was used to develop a tutor for teaching concepts in C++ programming. The C++ Tutor tests a student's ability to classify program segments rather than the ability to write a program. It is more like an intelligent testing system.

The theory refinement implemented in ASSERT is based on general machine language technique. It is a method for automatically revising a knowledge base using a database of classified examples. The NEITHER (Baffes and Mooney, 1993) theory refinement system, a successor to the EITHER system (Ourston & Mooney, 1994), is used to provide a correct representation of the domain using propositional Horn-clauses knowledge representation (Baffes, 1994). The representation takes two inputs, a propositional rule base called the *theory*, which is refined using a set of input *examples*. The examples are lists of *feature-value pairs*

chosen from a set of *observable* domain features. Each example has an associated label or *category* which is provable using the theory given with the feature values in the example. NEITHER can generalize or specialize a theory, without user intervention, and is guaranteed to produce a set of refinements that are consistent with the training examples.

A summary of the theory refinement technique is as follows. Propositional Horn-clause theories can have four types of errors. An overly-general theory is one that causes an example to be proven in an incorrect category, which is a false positive. NEITHER adds new antecedents and deletes rules to fix such problems. An overly-specific theory causes an example not to be proven in its own category, which is a false negative. NEITHER deletes existing antecedents and learns new rules to fix these problems. By making these four kinds of syntactic rule changes, NEITHER can correct the semantics of the theory by altering the conditions under which rules are satisfied (Baffes, 1994).

Refinements produced by NEITHER are used to generate explanations for the correct use of the rule which is changed, and examples which use the rule. Various student errors detected by NEITHER are used to automatically construct a bug library. The bug library algorithm is able to model both common and unique misconceptions. It is a major improvement to earlier systems like LisTutor and PROUST which has a static bug library.

Experiments conducted show that ASSERT can be used to construct tutorials which would significantly improve students' performance (Baffes, 1994). The theory refinement is able to generate more accurate student model and support individualized feedback.

As pointed out by Baffes (1994), most mature theory refinement algorithms developed thus far are designed for classification. Baffes quoted that enhancements in first-order logic refinement methods (Richards & Mooney, 1995) allow ASSERT to address a wider range of applications.

#### *3.1.5 ELM-ART (Weber & Specht, 1997)*

ELM-ART II (Episodic Learner Model Adaptive Remote Tutor) is an intelligent interactive and adaptive learning textbook to support learning programming in introductory LISP on the WWW. It was developed to overcome the shortcomings in its predecessor ELM-ART (Brusilovsky et al, 1996) and built, based on ELM-PE (Programming Environment) (Weber & Möllenberg, 1995). ELM-PE is an on-site intelligent learning environment that supports example-based programming, intelligent analysis of problem solutions, and advanced testing and debugging facilities.

ELM-ART II represents knowledge about units to be learned in terms of a conceptual network. Units are organized hierarchically into lessons, sections, subsections, and test pages. The knowledge-based component of the system uses a



combination of a multi-layered overlay model and an episodic user model. The multi-layered overlay model (Weber, 1999) assumes that a student's knowledge is a subset of the expert's knowledge that allows for adaptive link annotation and individual curriculum sequencing. The model consists of four layers relating to the concepts in the domain knowledge as depicted below :

Table 3.1 Multi-layered Overlay Model in the ELM-ART II

<b>LAYERS</b>	<b>CONCEPTS REPRESENTED</b>
First Layer	Page of a corresponding concept already visited
Second Layer	Exercise or test of a concept attempted and the results
Third Layer	Inferred concepts
Fourth Layer	Known concepts

To allow individual curriculum sequencing, links on each page are updated based on five learning states of a corresponding concept : 'already learned', 'inferred', 'stated as known by the user', 'ready and suggested to be visited' and 'not ready to be visited'. The NEXT button in the navigation bar of the web pages allows the student to ask the system for the best next step. The next page is intelligently derived from the general learning goal and the learning state of the concepts. The next suggested page will refer to the concept that is not annotated to one of the first three learning states and that is the next one ready to be learned.

In an episodic user model, knowledge about the student is stored in terms of a collection of episodes or cases. It encourages the student to reuse previously analysed sample code to solve a current problem. ELM-ART II can predict the student's way of solving a particular problem and find the most relevant example from the individual learning history. To construct the student model, the code

produced by a student is analysed in terms of the domain knowledge and a task description.

Results of an empirical study show that individual adaptive guidance by the system is especially helpful for the novices. Students who are familiar with Web browsers welcome the link annotation and maintain a longer interest in the learning system when links are annotated adaptively. The experiment also shows that the number of navigation steps is reduced by both adaptive navigation support and individual curriculum sequencing with the NEXT navigation button.

#### *3.1.6 Virtual Campus PROLOG Tutor (Peylo et al, 2000)*

The VC PROLOG Tutor analyses the problem solving tasks through the analysis of solutions (or partially completed solutions). It compares the layout and structure of the student's code with one or more example solutions. The error analysis module has to assume that the student will use the basic layout conventions. The analyzer interprets the error dictated and presents them to the user with the corrections and explanations. A graphical display of the program execution is available to enable the students to visualize and understand the way a PROLOG program works.

The domain knowledge is organized in a concept lattice to represent the dependencies between concepts. To support intelligent problem solving, concepts are related to skills. Concepts and skills are fundamental for understanding and learning to write programs in PROLOG.

To judge the student's understanding of PROLOG, two information sources are used: URL-tracking to detect visited PROLOG concepts and the results from the intelligent analysis of the problem solving assignments. The problem arises when student browses the lecture notes but does not work on the assignment. In this situation, the system is not able to accurately determine the degree of the student's understanding. Information from both sources must be available to identify the transfer of theoretical knowledge taken from the learning material to problem solving.

### *3.1.7 Tutor on C++ Programming (Kumar, 2002)*

The model-based reasoning is used for domain modelling, explanation generation and program animation to teach students to analyze and debug C++ programs for semantic and run-time errors. A model of the domain is first constructed. In this case, it is the model of the C++ language. The model is used to simulate the expected behaviour predicted by the student. The discrepancies between these two behaviours are used to predict the student model and to generate feedback to teach the student.

The model knows the correct answer and therefore it doubles as the runnable expert module. Another main advantage of model based reasoning is its contribution to dynamic generation of problems. Unlimited number of problems are generated by randomly instantiating BNF-like grammar. Each rule of grammar can be carefully

designed with specific pedagogical objectives in mind. However, building the domain model is an expensive task both in terms of time and expertise.

In general, the evaluation of the tutor shows that the tutor does help the students to improve their performance.

#### *3.1.8 JITS - Java Intelligent Tutoring System (Sykes, 2003)*

The web-based JITS prototype is designed using cognitive science and Artificial Intelligence techniques. It models a small subset of the Java programming language for students in the College and University level.

There are four modules in the JITS architecture – curriculum design, the AI module, the distributed web-based infrastructure, and the user interface design. JITS focuses on the methodology by which a student attempts to solve a problem. Giving the correct answers to a given program specification is not the final goal in the tutoring system. Pedagogical issues including conventions, style and professional programming techniques are modelled in JITS.

To provide intelligent feedback, the JITS AI Module examines the student's code after being analyzed by the Java Parser. If the parser fails, the Fuzzy Scanner module is invoked to construct feasibly-sound variations of the student's code and proceeds to compile and run them. The information is then passed into the syntactic\_decision\_tree to determine appropriate feedback. A Java Parse Tree is

constructed upon a successful parsing. The code is then compiled and executed. In this situation, `semantic_decision_tree` is used to determine feedback or hints. The two decision trees represent the strategic and judgmental knowledge for a specific programming problem (Scott et al, 1991).

The project is in progress and it is hypothesized that the completed prototype will be sufficient to prove that the completed JITS will provide an interactive learning environment to improve student's performance (Sykes, 2003).

#### *3.1.9 Pseudocode Tutor (Chad Lane & VanLehn, 2003)*

Pseudocode Tutor is based on Coached Program Planning (CPP), a dialogue-based style of tutoring for novice program design. A similar idea is adopted in BRIDGE which allows the student to build the natural language solutions via menu selections.

The system employs a four-step pattern found in the dialogues and uses keywords to understand student input. The tutor repeatedly asks the student to (1) identify a programming goal, (2) describe a technique to achieve this goal, (3) suggest pseudocode steps to implement the technique, and (4) place the steps correctly within the pseudocode. This dialogue pattern continues until all the programming goals are satisfied and the pseudocode completed. The collaborative nature of the dialogue allows the refinement of the answers. Other issues that may arise from the interactions are the layout and organization of the pseudocode. These sub-dialogues

include the improper sequencing steps that produce a logic error and improper indentation.

A computer-mediated, human-to-human study to evaluate the usability of the environment and the effect of CPP on novices revealed positive results. It showed that students exhibit less unpredictable programming behaviour during the implementation, more productive with comments in their programs and commit fewer structural mistakes, like indentation.

The system is still under development. The results from the controlled-experiment revealed that using natural language to derive the pseudocode to prepare students in writing the program is effective. Pseudocode is a well-established approach to teach programming and design. On the other hand, the effectiveness and naturalness of the techniques adopted in the dialogue remains to be seen. There are other open issues identified by Chad Lane and VanLehn (2003) – representation of the wide variabilities of solutions, appropriate reasoning method to derive the sub-dialogues and more analysis of the data. The analysis includes the quality of designs, quality of identifiers, content and quality of the comments, the timing when comments are added, and overall attitude on their work.

#### *3.1.10 BITS – Bayesian Intelligent Tutoring System (Butz et al, 2004)*

BITS is built to teach students using the elementary topics in the C++ programming language on the web. These include concepts such as variables, assignments and

control structures. The architecture of BITS consists of four main modules – Bayesian networks, the knowledge base, the user interface and the study mode.

Bayesian networks (BN) are used to model the prerequisite information and to keep track of student knowledge regarding each concept. The first task is to identify the set of concepts, followed by the construction of the BN by specifying the conditional probability distributions of each node which represents a concept. Figure 3.1 illustrates the prerequisite concepts of the “For Loop” construct which consists of “Variable Assignment”, “Relational Operators” and “Increment/Decrement Operators”.

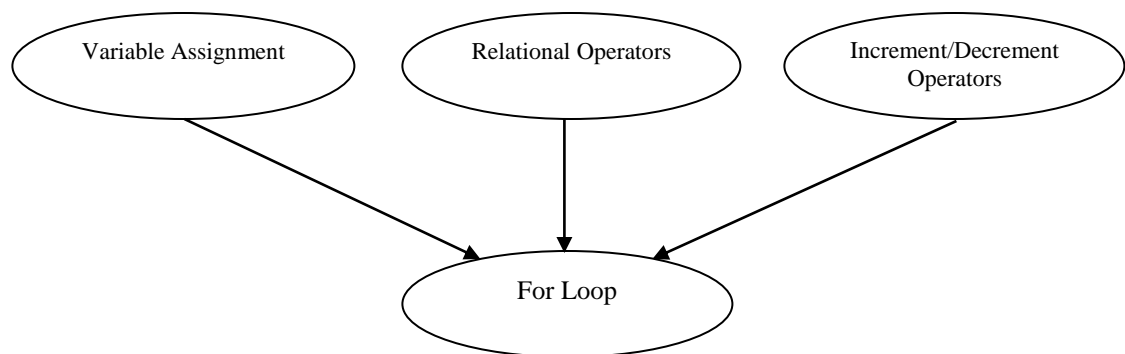


Figure 3.1 A partial Directed Acyclic Graph implemented in the BITS

The conditional probability distributions for the graph were obtained from the previous examination results. Each concept is either known or not known based on the student’s answer. Subsequently, the probability of each concept and its corresponding prerequisite concepts being known can be determined for the entire Bayesian network.

The knowledge base contains the class lecture notes and quizzes which are organized by concept for efficient indexing and retrieval. The class lecture notes are displayed during the learning session. On the other hand, a corresponding quiz is presented when BITS attempts to determine whether the student has understood a concept or not.

The User Interface module consists of two sub-modules: an Input module and an Output module for data flow between the BITS and a student. The Input module updates the BN using data collected from answers to quizzes, choices of study goal and understanding of concept. The output from BITS to the student includes lecture notes, sample quizzes and recommendations.

The study module provides three sub-modules called Regular Study, Problem Study and Quick Study to guide the students in the tutoring session. Similarly to the ELM-ART II (Weber & Specht, 1997), the Regular Study displays a set of lecture notes that are labelled using the traffic light signs on a Navigation Menu: yellow (already known), green (ready to learn) and red (not ready to learn). A concept is considered known if the BN indicates a probability greater or equal to 0.70. A concept is marked ready to learn if the probability is less than 0.70 and all of the parent concepts are known. If at least one parent concept is not known, then the concept is labelled as red - not ready to learn.

After reading a green topic, the student indicates their understanding by selecting Understand or Don't Understand or Not Sure if Understand (Figure 3.2). The BN is



updated if the student understands the notes and the Navigation Menu is redisplayed. If the student selects ‘Don’t Understand’, then the Problem Study module is invoked to advise the students to revise the notes and revisit the prerequisites. If the student is unsure of his/her understanding, the student is quizzed and immediate feedback is provided. The Quick Study module organizes a chosen topic in a learning sequence which allows a student to learn a concept for an impending exam or assignment deadline.

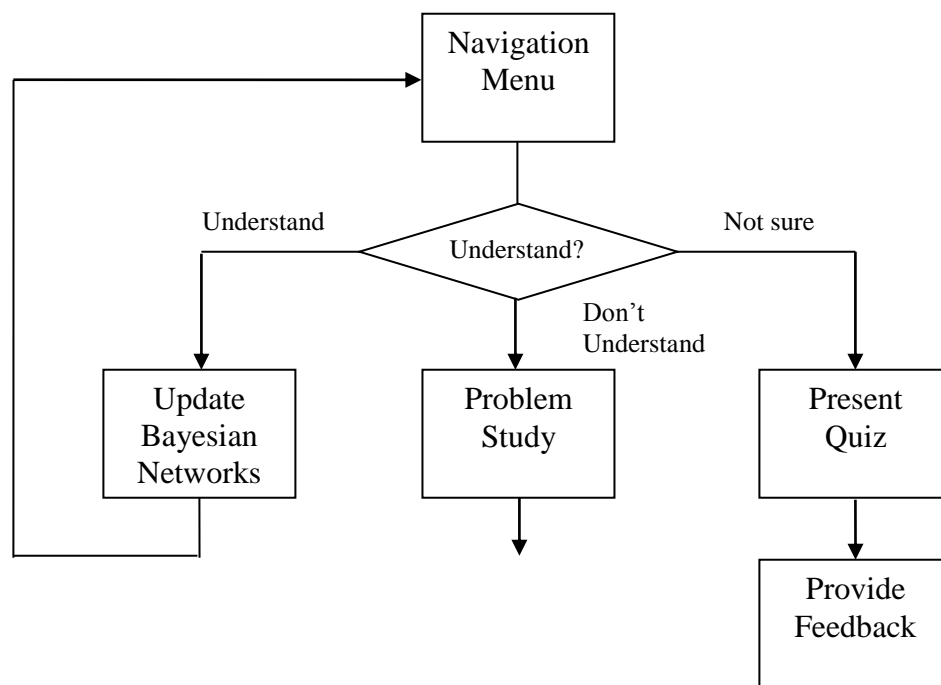


Figure 3.2 Routing of Student's Understanding

Last but not least, BITS provides a positive environment for learning by incorporating an animated study agent “genie” to convey appropriate emotion and encouragement to the student. Through the agent, the student receives feedback in the form of voice animation and dialog boxes.

The primary objective of BITS is to intelligently guide students in navigating the course materials. Analyzing the student's interactions with the system, it is also able to assess what a student knows. However, the threshold of 0.70 to indicate a known concept is subjective and it does not give an accurate assessment of the student's understanding. It is also noted by Butz et al (2004) that problem solving is outside the focus of BITS even though it is essential in learning computer programming. Hence, BITS functions more like an intelligent Computer Aided Instruction rather than ITS.

## 3.2 ITS for Programming – Comparative Analysis

### *3.2.1 Domain Knowledge*

It is observed that the earliest work on the domain knowledge in the ITS for programming falls in the field of Artificial Intelligence (AI). LISP, PROLOG, C++ and Java are programming languages of AI. LISP (LISt Processing) is a functional programming language that uses the theory of mathematical functions as the basis for programming. PROLOG is a well-known programming language in the logic programming paradigm, whereas C++ and Java are object-oriented.

The other popular domain knowledge is Pascal which is a structured programming language. Numerous interests in ITSs to teach Pascal seem to indicate that it is widely taught in introductory programming courses. The syntax and semantics of the language are less complex as compared to other programming languages like C.

It is suitable for teaching in a basic course in programming and design. However, it lacks dominance in the industry.

As AI has matured and demonstrated its applicability to a range of practical problems, its almost exclusive reliance on LISP and PROLOG has diminished. Consequently, ITSs for programming shift to object-oriented programming (OOP) as their domain knowledge (Luger, 2005). OOP languages incorporate many of the features found in frame-based knowledge representations, including class inheritance and the ability to represent structured knowledge. For these reasons, object-oriented languages such as Smalltalk, CLOS (OO LISP) and C++ are popular in AI applications.

### *3.2.2 Level of Programming*

The level of programming taught in the current ITSs such as ASSERT, C++ Tutor, JITS, BITS mostly covered elementary topics that are typically found in an introductory course to Computer Programming. Undoubtedly, there is a need for tutoring materials that target the second level of programming for a course. These include application of the programming language to create data structures and solve more complex problems.

### 3.2.3 Tutoring Goals

Another observation in the ITSs to teach programming is the tutoring goals. This is summarized in Table 3.2. The table shows the progression of ITSs for programming from the 80s to the current millennium in the area of domain knowledge as well as tutoring goals.

Table 3.2 Tutoring goals of ITSs to teach programming

ITS	DOMAIN KNOWLEDGE	TUTORING GOALS
<i>1980s</i>		
LispTutor	LISP	Focus on teaching the syntax of LISP cognitively.
PROUST	Pascal	Focus on teaching the syntax of Pascal by analyzing student's program.
BRIDGE	Pascal	Focus on teaching the syntax of Pascal using a Natural Language Plan Selection Menu.
<i>1990s</i>		
ASSERT	C++	Tutor tests a student's ability to classify program segments rather than the ability to write a program.
ELM-ART	LISP	Focus on teaching the syntax of LISP by analyzing and diagnosing problem solutions.
VC PROLOG Tutor	PROLOG	Focus on teaching the syntax of PROLOG by analyzing student's solutions
<i>2000s</i>		
C++ Tutor	C++	Teach by analyzing and debugging C++ code segments. Focus on tutoring programming constructs, semantic and run-time errors instead of syntax errors.
JITS	Java	Focus on the approach used by a student to solve a program rather than just the program correctness. Promote better programming style.

Table 3.2 (continued)

Pseudocode Tutor	Structured programming (eg. Pascal)	Help novices to understand and solve problems in their own words, rather than implementation and programming language specific issues.
BITS	C++	Teach elementary C++ utilizing prerequisite concepts. Primary objective is to help students navigate the course material.

Most of the above ITSs have focused on teaching students the syntax of the programming language as opposed to application which uses programming constructs in new situations or solves problems using knowledge. The common teaching strategy adopted is to provide a program specification to solve, followed by intelligent analysis of the students' solutions. However, weaker students do not adapt well to this strategy. Beck and Stern (1999) suggested coding various learning theories and teaching strategies into a system, and then letting the system determine which works under which circumstances for a particular student.

In the 2000s, the ITSs for computer programming have shifted the tutoring goals from teaching the syntax to improving programming styles. This is an important progression as students should not be equipped with only syntax of a programming language but also acquire skills to improve the quality of programs. Students need to learn to write programs that are readable, reliable and maintainable.

#### 3.2.4 *Student Modelling Techniques*

Techniques in the implementation of the ITSs range from application of the cognitive theory to Artificial Intelligence (AI) techniques. Earlier work shows a strong influence from cognitive modelling and less contribution of AI techniques. However, ITS researches (Beck & Stern, 1999) have realized the promising benefits of bringing back AI into AI and Education. They have pointed out that AI has the capabilities to employ more complex models of student behavior, and has the potential to decrease the cost and complexity of building educational systems.

### 3.3 Summary

The prior work on ITSs to teach computer programming has demonstrated progress in the tutoring goals from teaching purely syntax to promoting better programming style. However, the main focus on the level of programming is on the fundamental level. One obvious reason is that researchers realised the importance of strengthening the foundation of students before proceeding to the next level. Nevertheless, it is time researches turn their interest into teaching higher level of programming to students and improve the students' application skills. ITS for programming should not just end at teaching elementary syntax to students but look beyond that into guiding students to apply what they have learned. For example, knowledge gained in the syntax for iteration can be applied to show when and where a particular iteration construct can be most appropriately utilized.

ITSs for programming have shown maturity in the techniques that were employed in the domain modelling, student modelling and interface. The Table 3.3 below summarizes the techniques employed by current ITSs. Two main techniques are compared here – domain modelling and student modelling. The domain knowledge is the heart of an ITS and the student model is essential to enable personalized tutoring. The third aspect considered is the user interface and platform of the ITSs.

Table 3.3 Techniques employed in current ITSs

<b>ITS</b>	<b>TECHNIQUES</b>		<b>INTERFACE</b>
	<i>Domain Knowledge</i>	<i>Student Modelling</i>	
<b>1980s</b>			
LispTutor	Production Rules based on the Adaptive Control of Thought theory of Cognition	Model Tracing	Text-based Dialogue
PROUST	Bug catalogue	Intention-based Diagnosis	Text-based providing diagnosis of student's program
BRIDGE	Organized using programming plans ie. Concepts	Overlay Model	Natural Language Plan Selection via menu
<b>1990s</b>			
ASSERT	Dynamic bug library entries. Domain rules to classify problems.	Theory Refinement to Student Modelling	Text-based Dialogue
ELM-ART	Represented in terms of a conceptual network. Units are organized hierarchically into lessons, sections, subsections and unit pages.	Adaptive techniques: <ul style="list-style-type: none"> <li>• Multi-layered Overlay Model</li> <li>• Episodic Learner Model</li> </ul>	Web-based Adaptive Learning

Table 3.3 (continued)

VC PROLOG Tutor	Organized in concept lattice - hierarchical	Overlay Model based on intelligent analysis of solutions - compare student's solution with one or more example solutions	Web-based
<b>2000s</b>			
C++ Tutor	Model-based Reasoning - simulate the correct behaviour of a C++ construct	Overlay Model based on behavioural discrepancies	Windows-based
JITS	A database of records with problem, solutions, classifiable incorrect responses and appropriate hints.	Performance history based on statistics	Web-based
Pseudocode Tutor	Program specifications organized in HTML pages	Coached Program Planning Dialogue : i) identify programming goal ii) describe technique to achieve goal iii) suggest pseudocode steps to achieve goal iv) place steps appropriately to complete pseudocode.	Web-based. Coached Program Planning – dialogue-based style
BITS	Bayesian Networks to model structure of problem domain. Lecture notes in the form of web pages, organized by concept.	Bayesian Networks to track student knowledge	Web-based



Domain knowledge modelling has progressed from predefined rules and bug catalogues to hierarchical structure and dynamic bug library entries. Combination of overlay modelling and other techniques were employed in student modelling.

Prior work also suggested that natural language has played an important role in the interface of the ITSs. However, this interaction style is too rigid and directive causing the system to be unsuitable for better students.

The trend shows a lack of contribution from Artificial Intelligence (AI) techniques. It is believed that AI can still act as a vital avenue in the design and implementation of ITSs particularly in the areas of domain and student modelling.

AI techniques for uncertainties management play an important role in student modelling. In general, the student model is loaded with uncertainty associated with the student's behaviour and understanding. Jameson (1995) has reviewed several frameworks for managing uncertainty in expert systems, including Bayesian networks, the Dempster-Shafer theory of evidence, and fuzzy logic. It is noted that Pearl (1988) has shown that Bayesian networks have certain benefits over the other two frameworks. The application of Bayesian probability reasoning and fuzzy logic techniques is described in the next chapter.

Finally, an important milestone in the implementation of ITS is the deployment of the system on the World Wide Web (WWW). Since the introduction of the WWW in the early 1990s, most efforts focused on Web-based ITS. Researchers began to port existing stand-alone ITS to the web and works were carried out to provide adaptive learning and intelligent sequencing on the web. The borderless world created by the advent of technological advancement has become a platform for independent teaching and learning with respect to time and space.

## **Chapter 4 C++ STL and Bayesian-Fuzzy Student Modelling**

Most of the current Intelligent Tutoring Systems (ITSs) for programming focus on teaching students the syntax of a programming language as opposed to application, for example, LispTutor, PROUST, BRIDGE, ELM-ART, ASSERT, JITS and BITS. The main tutoring approach is to present a problem specification for the student to solve, followed by intelligent analysis of the solution with various feedback. It is also observed that existing ITSs suffer from static domain knowledge and are restricted to the tutoring session.

The review of prior work motivated this research to shift the tutoring of programming language from teaching elementary programming syntax to application. Following this motivation is the choice of the programming language for implementation. The interest in C++ is most obvious as it is widely taught in the first year of a Computing degree course. It is considered as one of the de facto programming languages to computer scientists and engineers.

Stroustrup (1999) pointed out that there is a need to adopt a paradigm shift in the way we write and teach C++ programs. In his article on “Learning Standard C++ as a New Language”, he argued his preferred approach to teaching C++ along with its Standard Library using some simple examples. Consequently, the aim of the proposed ITS is to teach students the application of the C++ Standard Template Library (STL) to problem solving.

## 4.1 The Domain Knowledge – C++ STL

Data structures and algorithms are inherently building blocks of a program. For many years, programmers everywhere have been writing their own data structures and defining fundamental algorithms such as sorting and searching. The tasks are tedious and error-prone. The other problems include incompatibility among vendors, use of virtual functions which reduces performance, difficulty in adding new algorithms, not type-safe and use of heap memory (ObjectSpace, 1996). Recognizing that thousands of C++ programmers have been reinventing the wheel, the Standard Template Library (STL) was accepted in 1994 as part of the C++ Standard Library by the ANSI/ISO C++ Standard Committee.

The Standard Template Library is a framework collection of data structures (called *containers* in STL) and algorithms designed by Alexander Stephanov and Meng Lee (1995) at Hewlett-Packard based on their research on generic programming. Hence, STL is not about object-oriented programming but generic programming. Musser's (2003) working definition of generic programming is "programming with concepts," where a concept is defined as a family of abstractions that are all related by a common set of requirements. An important aspect in generic programming is defining abstract concepts (for example, containers and iterators) and writing algorithms and data structures in terms of abstract concepts (Austern, 1999). In simpler terms, generic programming is about generalizing software components so that they can be easily reused in a wide variety of situations – different algorithms should work for different containers.

Stephanov (1995) has verified that the STL is as efficient as a non-generic version written in the same language by studying the performance of the assembly code generated by different compilers on different architectures. Furthermore, the generic algorithms and containers had been handcrafted for one specific type (Austern, 1999).

The STL is divided into three key components – containers, iterators and algorithms. The containers are data structures capable of storing any type of data element (Table 4.1). They are further divided into sequence containers (vector, list and deque), associative containers (map, multimap, set and multiset) and container adapters (stack, queue and priority queue). STL iterators can be viewed as pointers to manipulate the containers elements. It consists of five different flavours – input, output, forward, bidirectional and random access. STL algorithms are function templates that operate on and are parameterized by iterator types. Because STL algorithms are decoupled from containers for efficiency, all containers within the same iterator category can utilize the same algorithms. This also reduces the software development tremendously. For example, the single *sort* routine in the STL algorithm can work on (almost) all of the STL containers such as vectors, lists. New algorithms can be added easily without modifying the containers (Stephanov & Lee, 1995). Future extensions of the STL are discussed in the ANSI/ISO C++ Standard Committee.

Table 4.1 STL Components

STL COMPONENTS	CATEGORIES	
Containers	Sequence containers	vector list deque
	Associative containers	map multimap set multiset
	Container adapters	stack queue priority queue
Iterators	Input Output Forward Bidirectional Random access	
Algorithms	Mutating-sequence algorithms Non-mutating-sequence algorithms Numerical algorithms	

Stroustrup (1999, p. 1), the creator of C++ commented

“to get the most out of Standard C++, we must rethink the way we write C++ programs. An approach to such a ‘rethink’ is to consider how C++ can be learned and taught”.

In software development, our goals are ultimately to have programs that are easy to write, correct, maintainable and acceptably efficient. Comparing examples of standard C++ programs and traditional C-style solutions, Stroustrup demonstrated that through designing and programming at a higher level of abstraction with the use of libraries, these goals are achievable. This also leads to a reduction of size and complexity of the code we write and subsequently reduces development time, eases maintenance, and decreases the cost of testing. Importantly, the task of learning C++ is also simplified.

“Thus, work on more libraries, on more consistent implementation of widely-used libraries (such as the C++ STL), and on making libraries more widely available can yield great benefits to the C++ community” (Stroustrup, 1999, p. 10).

Education must play a vital role in this move to make the C++ STL more accessible and gearing towards cleaner and higher-level programming styles. Since becoming a standard, C++ STL has been widely taught in C++ programming courses.

Budd (1998) has moved away from the traditional approach of teaching data structures by including the power of the STL. The increased emphasis on the use of standard components and decreased emphasis on implementation represent a significant paradigm shift indicating the maturation of the computer science field. He added that many authors have predicted that in the future most programs will be constructed piece by piece like building a Lego set rather than developing them from scratch. He concluded that although it is important for students to know how to implement a linked list, it would be more important to know how to use the list container in the STL (Budd, 1998).

Working with the STL, students will realize the power of C++ in applying the ideas to data structures and learn the power of C++, thus allowing them to carry their knowledge to later courses on advanced data structures and eventually into their careers. STL provides most of the classic data structures (list, queue, stack) as basic abstract concepts, so it is relatively easy to create even quite complex programs. Knowledge acquired in the ITS should be transferable to a larger set of curriculum objectives. In the case of the STL, it provides a platform of expression for data structures. For example, the application of member

functions in the STL list forms a foundation for students to understand the traditional user defined linked list data structure.

To conclude, two main points have been considered in the selection of this domain knowledge of C++ STL. Firstly, it is to encourage the use of C++ STL to produce easy to write, correct, maintainable, and acceptably efficient programs. Secondly, to show that using STL adds more experience to augment students' knowledge in data structures and C++.

## 4.2 Difficulties in Learning and Teaching C++ STL

### *4.2.1 Learning the C++ STL*

Having justified the selection of the domain, a discussion will be made here to ascertain the problems students face in using or applying the STL. In general, two views will be considered: difficulties with the prerequisites of STL and the common mistakes when using the STL.

From experience, it is discovered that students find the C++ STL difficult due to their weaknesses in understanding various object-oriented concepts. These concepts form the prerequisites in learning the C++ STL. This is different from the idea of breaking down a problem into sub problems or components in previous work like the JITS and BITS. Both of these ITSs decomposed the domain knowledge consisting of language basics into sub-problems. For example, a for-loop structure

---



is decomposed into three parts – initialization, test and incrementation. However, to master the application of the C++ STL, students need a good foundation in some prerequisites.

One of the main prerequisite concepts is parameterization. From experience, STL can be difficult to understand due to a high degree of parameterization. In parameterization, a type description is parameterized with another type. This allows us to describe the idea of a *vector* of T, where T represents a yet to-be-known type. In C++, this idea is expressed by a construct called a *template*. Template is a very powerful tool for reusability of object code and very useful for implementing generic structures such as lists, queues and stacks to manipulate on arbitrary types (Deitel & Deitel, 2003). C++ provides two types of templates: function template and class template. Function templates are commonly used to implement generic functions like sorting and searching routines. The class *vector* is an example of a class template. Class templates are called parameterized types. They encourage software reusability by enabling type-specific versions of generic classes to be created. For example, one *vector* class template could thus become the foundation for creating many *vector* classes such as *vector of int*, *vector of double*, *vector of string*, *vector of student*, and so on used in a program.

In any course on C++ programming, students start with non-template functions which work on specific data types. When templates are later introduced, they tend to struggle with the idea because they could not ‘visualize’ the data type the functions are manipulating. The STL generic algorithms are implemented as

function templates and the containers (vector, list, queue, stack) are implemented as class templates. Therefore, difficulties in understanding templates prevent students in applying the STL effectively.

The concept of function overloading is another difficult abstraction to grasp. In C++, several functions with the same name can be defined, as long as these functions have different sets of parameters. It could be different parameter types or number of parameters or the order of the parameter types. These functions are known as overloaded functions. The C++ compiler differentiates these functions through the parameter list.

The STL containers and algorithms are bundled with functions which are heavily overloaded. For example, the vector *insert()* has three flavours – two arguments specifying the position and value to insert, three arguments where the first argument specifies the destination, the second and third arguments refer to the range of values in the source, and another version that allows inserting multiple copies of the same value starting at a particular position in the container. STL actually includes approximately 70 standard algorithms. Having to decide on which one to use and how to use it are issues students face.

Other than the problems in understanding the concepts of templates and function overloading, there are many common mistakes when using the STL containers, iterators, and algorithms. Most of the mistakes involve the difference between applying an algorithm to a container and applying an algorithm to the range of

elements in a container (Austern, 2000). Indirectly, it is related to function overloading.

The ITS for the C++ STL hopes to address some of the difficulties highlighted above and give proof that programming skills can be improved through the system.

#### 4.2.2 *Teaching the C++ STL - Curriculum Planners vs Implementer-tutors*

For an ITS to achieve its learning goals successfully, both curriculum planners and implementer tutors need to be actively involved in the overall architectural design of the ITS. The terms that Kinshuk et al (2001) presented in their paper are *ITS designer teacher* and *ITS implementer teacher*. They recommended the concept of Human Teacher Model in the research and design of ITSs which would consider the different personality attributes, styles and preferences of a human teacher – both as a designing collaborator and a teaching collaborator. Their research was motivated by the need for adaptation to local contexts where ITS may be used in different parts of the world. Moreover, current research on ITS has been focusing only on the tutoring process, neglecting the roles of teacher. As quoted by Kinshuk, evaluation results have also confirmed that implementing teacher/tutor plays an important role in the success of a tutoring system (Stoner & Harvey, 1999).

Kinshuk (2002) interestingly argued the various reasons for the failure of the adoption of ITS research. He discussed that there seem to be a gap between the delivered research outcome and the needs of actual learning environment. The

culprits highlighted included no intervention from human tutors, very little possibility of customization and mismatch of preferred teaching style between developer teacher and implementing teacher. He concluded with the suggestion that the distinction and incorporation of teacher/tutor role allow the ITS to identify the different styles, record the teaching style/s adopted in the design and enable the ITS to adapt themselves to implementer tutor's style.

In this research, curriculum planners refer to a designer teacher who designs the curriculum for the tutoring and generates assessment methods for students. This person could be the lecturer of the module who plans the teaching schedule and prepare the teaching materials. Implementer tutors refer to tutors who facilitate the laboratory or practical sessions of the module. There are situations whereby both the curriculum planner and the implementer tutor refer to a single person. Nevertheless, the roles that they play are different – one delivers, another directs. Therefore, it is beneficial to distinguish between these two roles in the design and implementation of the ITS.

In teaching the C++ STL, both curriculum planner and implementer tutor face challenges in dealing with the diversified learning pace and knowledge of students. Curriculum planners need tools that can assist them in the assessment of students' prerequisite knowledge and compile them for the implementer tutor to identify suitable teaching strategies for students. Their tasks are different from those who teach elementary concepts in C++ such as variable declarations, expressions, operators, selection constructs and iteration constructs. At the foundation level or in

the first year of a computer-related course, students are assumed to have no knowledge at all in programming. Though the students may have different learning pace, their knowledge starts at an 'equal' level. Strictly speaking, some students may have already learned programming but the majority of these freshmen knows little or nothing at all about programming.

On the other hand, C++ STL is covered in a higher level which is after students have acquired the foundation in programming. Tutors who are teaching the C++ STL cannot assume that all students have the same understanding in the elementary topics, and these topics form the prerequisites in learning the STL. Some may have already forgotten and just need hints to prompt them to recall. Others may have lack of understanding, and hence require exercises to drill the concepts into them again. Besides imparting skills on applying the STL, it is necessary for both the curriculum planner and the implementer tutor to put in effort to understand the needs of individual student. However, traditional classroom tutoring is not able to meet this demand and satisfy the students. Intelligent tools such as a curriculum authoring tool, assessment authoring tool and tutoring authoring tool are vital to assist the tutors. The integration of these tools together with the application of Artificial Intelligence techniques form a powerful system to aid teaching and learning.

### 4.3 Dynamic Domain Knowledge Modelling

The lack of customization in the domain knowledge is one reason why most current ITSs still remain as prototype. Their domain knowledge is static, and the problem specifications are limited and not updatable. There has been some previous work done on random generation of C++ code to teach program debugging (Kumar, 2002). Using BNF-like grammar, the randomizing can produce interesting and non-trivial variation of a problem. However, it would be more useful and effective if program specifications could be created adaptively, updated and randomly generated. The benefits include providing problems to students until they have mastered a particular topic, wider range of problems and individualized problems to deter plagiarism.

Most ITSs for computer programming are built for a specific programming language and then used in a particular environment for a certain group of students. Therefore, the ITSs are limited to the area of their application. To enlarge the scope of the application, ITSs for computer programming need to provide flexible tools to allow dynamic domain knowledge modelling.

In summing up, curriculum planners need the flexibility to design their own problem specifications and set assessments that cater to their own students. It is also necessary for implementer tutors to design appropriate tutoring sessions incorporating various teaching strategies. An authoring tool for the problem specifications and assessment coupled with

authoring of the tutoring sessions as part of the ITS will definitely resolve some of problems raised above.

#### 4.4 Student Modelling with Transparency

Much work is related to the involvement of students in the diagnosis of the student model. The term *cooperative* student model (Beck, Stern & Woolf, 1997) has been coined to refer to a model which is constructed by collaborating the student's and the tutor's beliefs. In the tutoring system, the tutor prompts the student to provide information that it cannot detect by itself. The model is student-centered design, giving students some control over the student model. In learning, the student knows better, what they know and what they do not know. Therefore, it is appropriate that students assist the ITS to capture their knowledge and other information to form the student model.

It is believed that by having a transparent student model, student reflection is promoted and this will encourage students to use feedback in future problem solving (Bull, 1997, Dimitrova et al, 2000). The model is also known as an inspectable student model. An inspectable model is constructed for each student based on the feedback given by the tutor. In *See Yourself Write*, the student model encourages the student to think of ways to improve their work by viewing and interacting with the model, allowing easy access to useful comments on earlier work, being prompted to explain their progress and being encouraged to take advantage of self-explanation (Bull, 1997). This is very useful for domain which is difficult to get useful feedback computationally.

LacePro – Learning, Applying and Consulting Established Procedures (de Buen et al, 1999) applies a collaborative approach to student modelling. LacePro is a system in which professional engineers can learn, apply and consult established procedures. The collaborative user modelling approach provides students with two choices: a directive system with adaptive characteristics, and a flexible system in which the student can express his or her own approach. This is within a collaborative environment in which the student provides the information the student model requires, has the possibility to inspect the student model at any time, and can reject or accept the system's suggestions. Two major considerations behind LacePro's collaborative approach are:

- (i) In principle, the system will accept as true all that the student says (e.g., that she knows a step or concept), and
- (ii) Most actions proposed by the system (e.g., what to explain) can be accepted or rejected by the student.

The scenarios and empirical evaluation showed that engineers preferred this open and flexible approach.

Involving students in student modelling as described above (cooperative, inspectable or collaborative) requires some consideration. Weaker students or slower learners need more guidance and in most cases, they are not able to determine or decide on the appropriate teaching strategy for themselves. Thus, the information that they provide may not be accurate. Over-confident students pose similar problems. The student modelling technique must have the intelligence to overcome these issues. Dimitrova et al (2000) has also discussed a number of potentials and problems in involving students in diagnosis. One issue is the degree of power for students to participate in diagnosis and how the conflicts



between the system and the student's views can be resolved. An experienced tutor is required to give judgment in this area. Moreover, Artificial Intelligence techniques have the capability to deal with these diagnosis and uncertainties.

#### 4.5 Uncertainties Management

A number of problems were highlighted above. These include problems in learning and teaching the C++ STL, and the deficiencies within current ITSs in general and specifically in dynamic domain modelling and transparent student modelling.

The key problem in using the C++ STL lies in the lack of capability in prerequisite concepts. Therefore, a model is required to capture the student's knowledge in the prerequisite concepts. To find out the student's level of understanding in the prerequisite concepts, a test can be conducted. However, the test results create an uncertainty in determining whether the student has really understood the topic or has guessed the answer correctly. In other words, the system needs to know whether the student knows or do not know the topic. Subsequently, better guidance can be provided for the student.

Besides modelling the student's prior knowledge, the ITS also needs to track the student's interactions as he or she uses the ITS. There are also various uncertainties that need to be managed to model the student behaviour in the learning process of the C++ ITS. For example, vague problem specifications could lead students to respond incorrectly. A wrong answer could also mean that students have forgotten a particular topic instead of no knowledge of it at all.

In summary, the ITS is required to deal with two main sources of uncertainties:

- i)       Uncertainties in the student's answers to the test
- ii)      Uncertainties in the student's behaviour during the learning.

There are three common Artificial Intelligence techniques for managing uncertainties – Bayesian reasoning, certainty theory and fuzzy logic (Hopgood, 2001). Bayesian reasoning is based on a sound probability theory, but in some applications, it may not be practically true to assume the conditional independence of evidence. Certainty theory lacks mathematical basis, but offers a simple approach to overcome some of the limitations in Bayesian reasoning. Lastly, fuzzy logic or possibility theory allows the representation of vague language in a precise manner. There are a number of design issues that need to be considered in the building of a fuzzy expert system. These include the identification of input and output parameters, definition of membership functions and rules to improve the performance of the expert system.

Considering the problem in the prerequisite knowledge of the individual student, a method that is able to model a cause-effect relationship with uncertainty management is required. Therefore, Bayesian reasoning which encodes a corresponding set of conditional probability distributions is highly appropriate for the student modelling. The Bayesian approach also supports a transparent student model as it has a strongly proven derivation probability theory basis, and the probabilities associated are intuitively understandable. On the other hand, certainty factor approach, though simpler, appears ad hoc compared with Bayesian reasoning. Certainty factors are used in situations where reliable statistical

information is not available or the probabilities are difficult or expensive to obtain (Negnevitsky, 2002). Fuzzy logic, in this case, allows us to deal with vagueness in categorizing student's performance.

#### *4.5.1 Bayesian Reasoning*

To avoid the computation complexities in Bayesian network, which is used in current ITSs, the Bayesian Theorem is used directly in this project for student modelling. Though Bayesian network does not require extensive computation to update and assign initial probabilities (Lauritzen & Spiegelhalter, 1988), there are considerable design issues in building the network. These include determining the number of nodes for the problem and specifying the parent-child relationships (Fernandez & Sison, 2001). The application of Bayesian Theorem is discussed in this section. Bayesian reasoning or updating provides a probabilistic approach to represent uncertainty. It is based on the assumption that it is possible to assign a probability to every hypothesis, and that this probability can be updated in the presence or absence of an evidence (Hopgood, 2001).

The technique of Bayesian updating is based upon the application of Bayesian Theorem, also known as Bayes Rule. Bayesian Theorem provides an expression for conditional probability which is denoted by  $P(H/E)$ ,

where

$H$  is the Hypothesis

$E$  is the Evidence or observed data

$P(H/E)$  is the conditional probability of the hypothesis  $H$  holds given the evidence  $E$ .  $P(H/E)$  is also known as the posterior probability of  $H$ , because it reflects the confidence that the hypothesis  $H$  holds given that the evidence  $E$  has been obtained (Mitchell, 1997). The conditional probability of  $H$ , given  $E$  is then defined by

$$P(H | E) = \frac{P(H \cap E)}{P(E)} \quad \text{if } P(E) > 0 \quad (4.1)$$

where

$P(H \cap E)$  is the probability of both the  $H$  and  $E$  occurring, which is also called the joint probability.

Similarly, the conditional probability of  $E$ , given  $H$  is defined by

$$P(E | H) = \frac{P(E \cap H)}{P(H)} \quad \text{if } P(H) > 0 \quad (4.2)$$

The joint probability is commutative, thus

$$P(H \cap E) = P(E \cap H) \quad (4.3)$$

Therefore,

$$P(H \cap E) = P(E | H) \times P(H) \quad (4.4)$$

Substituting Equation (4.4) into Equation (4.1), yields the Bayesian Theorem

$$P(H | E) = \frac{P(E | H) \times P(H)}{P(E)} \quad (4.5)$$

The Theorem provides a method to calculate the posterior probability  $P(H | E)$ , from the prior probability  $P(H)$ , together with  $P(E)$  and  $P(E | H)$ .

The Bayesian Theorem can then be expanded as follows:

$$P(H | E) = \frac{P(E | H) \times P(H)}{P(E | H) \times P(H) + P(E | \sim H) \times (1 - P(H))} \quad (4.6)$$

where

$\sim H$  means “not  $H$ ” or false hypothesis.

The probability of ( $\sim H$ ) is equivalent to one minus the probability of  $H$  occurring which is

$$P(\sim H) = 1 - P(H)$$

The denominator for Equation (4.6) is easily derived as follows:

$$\begin{aligned} P(E) &= P(E \cap H) + P(E \cap \sim H) \\ &= P(E | H) \times P(H) + P(E | \sim H) \times P(\sim H) \\ &= P(E | H) \times P(H) + P(E | \sim H) \times (1 - P(H)) \end{aligned}$$

Equation (4.6) provides the cornerstone for the application of probability theory to manage uncertainties in the student modelling of the ITS to tutor C++ STL.

#### 4.5.2 Bayesian-based Predictive Initial Student Model

To model the student’s competence in the prerequisites after taking the pre-test, the desired conditional probability is that the student understands the prerequisite given that he or she answered it correctly.

Based on the example given by Ross (2003), the conditional probability is obtained by

$$\begin{aligned}
 P(U | C) &= \frac{P(U \cap C)}{P(C)} \\
 &= \frac{P(C | U) \times P(U)}{P(C | U) \times P(U) + P(C | \sim U) \times (1 - P(U))} \quad (4.7)
 \end{aligned}$$

where

$U$  is the event that the student understands the prerequisite and

$C$  is the event that the student answers correctly.

Now, let  $p$  represent the probability that the student understands the prerequisite,  $P(U)$  and  $(1 - p)$  the probability that he or she guesses,  $P(\sim U)$ . Two assumptions are made here. Firstly, a student who guesses at the answer will be correct with probability  $1/m$ , where  $m$  is the number of multiple choice alternatives. This is denoted by  $P(C/\sim U)$ . The second assumption is that a correct answer shows the student's understanding of the prerequisite. Therefore, the conditional probability of student answers correctly given that he or she understands the prerequisite is 1:

$$P(C | U) = 1$$

Applying the assumptions and substituting  $p$  and  $m$  into Equation (4.7) yields:

$$\begin{aligned}
 P(U | C) &= \frac{p}{p + (1/m) \times (1 - p)} \\
 &= \frac{mp}{1 + (m - 1) \times p} \quad (4.8)
 \end{aligned}$$

The value for  $p$  is obtainable from the results of the pre-test for a particular prerequisite. The prerequisites can be modelled using a Bayesian network structure as shown in Figure 4.1. The figure below shows a prerequisite *iteration* which consists of 3 sub-skills or sub-topics, namely ‘for loop’, ‘while loop’ and ‘do..while loop’. Understanding the three sub-skills, leads to the understanding of the topic *iteration* as a whole. Another way to view it is that, to have a complete understanding on *iteration*, the student must know the three different types of loop structure.

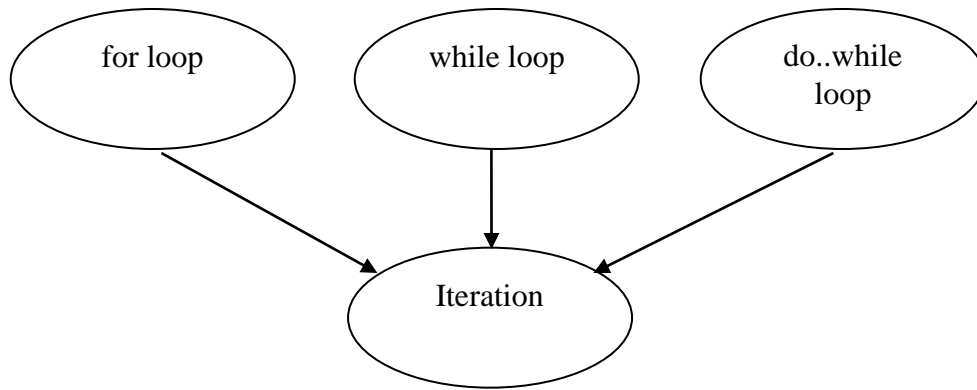


Figure 4.1 Prerequisite *Iteration* with its sub-skills

Each question in the pre-test is associated with a prerequisite sub-skill. The probability of student’s understanding in a prerequisite,  $P(U)$  or  $p$  can then be calculated as follows :

$$P(U) = \frac{\text{Total number of correct answers}}{\text{Total number of prerequisite sub - skill questions}}$$

Suppose out of 3 questions in the prerequisite *iteration*, the student has answered only 2 correctly. Therefore,

$$P(U) = 2/3 = 0.67$$

Let the multiple choice alternatives,  $m$  be 4. Substituting all the values into Equation (4.8), produces the following result:

$$\begin{aligned} P(U | C) &= \frac{mp}{1 + (m - 1) \times p} \\ &= \frac{4 \times 0.67}{1 + (4 - 1) \times 0.67} \\ &= 0.89 \end{aligned}$$

The conditional probability,  $P(U / C)$  value obtained needs to be further redefined to provide useful analysis. Comparing this value with a threshold allows the student's level of understanding to be quantified and categorised as well. To determine a suitable threshold, range samples of student's performance in some prerequisites are tabulated in Table 4.2. This table shows four prerequisite sub-skills together with their range of conditional probabilities calculated based on the total number of correct answers and total number of questions. The complete table of conditional probabilities is in Appendix A.



Table 4.2 Examples of Student's Pre-Test Performance

PREREQUISITE	TOTAL NUMBER OF CORRECT ANSWERS  ( <i>c</i> )	TOTAL NUMBER OF PREREQUISITE SUB-SKILL QUESTIONS  ( <i>q</i> )	$P(U)$  $p = c/q$	$P(U / C)$
User Defined Function	2	7	0.28	0.62
	4	7	0.57	0.84
	6	7	0.86	0.96
Array	2	6	0.33	0.67
	4	6	0.67	0.89
Class	1	5	0.20	0.50
	2	5	0.40	0.73
	3	5	0.60	0.86
Function Template	1	4	0.25	0.57
	2	4	0.50	0.80
	3	4	0.75	0.92

From the sample range of  $P(U/C)$  values shown in the table above, a threshold of 0.80 is a suitable value to indicate the student's performance level. This threshold also shows mastery of the prerequisite concept. Therefore, using an initial threshold of 0.80, the prerequisite concept is considered understood if the conditional probability  $P(U/C)$  indicates a probability greater than 0.80. The probability is then employed to guide the students in the tutoring of the C++ STL problems. This threshold will be adjusted according to the progress made by the student during the tutoring process. Since students' performance may either improve or decline as they solve the given problems, the values of the conditional probability can be updated regularly. This can be done in the light of new evidence such as time taken to solve a problem, total number of hints consulted, trials attempted and errors made.

#### 4.5.3 Tracking Student's Progress during Tutorial Sessions

Based on the conditional probabilities obtained after the pre-test, the tutoring session is then tailored to suit the individual student's understanding. Learning a programming language requires much drill and practice in problem solving. It was observed by McCracken et al (2001), as quoted by Kelly et al (2004), that problem solving difficulty is caused by the an inability to break programming problems up into smaller manageable tasks. Subsequently, the tutoring session for the C++ STL consists of problems that are organized according to sub-problems. In solving a problem, students are often taught to decompose the problem into sub-problems to reduce complexity. The two most common approaches in problem solving are top down and bottom up. Given a problem, students can break it into sub-problems in a top down manner. On the other hand, students can choose to solve the sub-problems independently and eventually combine them to solve the main problem. This is the bottom up approach.

The ITS for C++ STL employs a *sandwich* view of a given problem which combines top down and bottom up approaches. This allows more flexibility to students. The proposed interface is to provide students the problem specifications together with a program framework and sub-tutorials, which represents the sub-problems. Hence, students are shown the big picture of the solution, which is a top down view – from the problem specification to the program framework, and the sub-problems that form the main program giving a bottom up view. A sample layout of the interface is depicted in Figure 4.2. The *Program Framework* frame represents the top down view, where as the *Sub Tutorial* frame gives the bottom up

view. It is believed that showing two views of a problem to students gives them a better understanding of the requirements, and as a result, their problem solving tasks are simplified. Moreover, the top down and bottom up approaches are elementary forms of problem solving skills (Arvanitis, et al, 2001).

**Problem Specification**  
Write a program that ...

Program Framework – Fill in the answers below:

```

#include <iostream>
#include  // Header Declaration – Step 1

using namespace std;
int main()
{
     // Create a vector – Step 2

    int i;
    cout << "Please enter 10 integers: " << endl;
     // Populate the vector – Step 3
    cout << "The integers are: " << endl;
     // Output the vector – Step 4
    cout << endl;
    return 0;
}

```

**Sub-Tutorials**

Sub-Tutorial 1

Sub-Tutorial 2

Sub-Tutorial 3

Sub-Tutorial 4

Answer

Figure 4.2 A Sample Layout of a Problem Specification during the Tutoring

Each step in the sub-tutorial is related to a series of teaching strategies or hints, and prerequisite skills. In order for students to solve the sub-tutorials, they must possess the necessary prerequisite concepts for that problem. The results obtained from the

pre-test will determine the student's level of understanding in a particular prerequisite sub-skill.

Figure 4.2 illustrates a problem specification that instructs the student to create a *vector*, populate it with some values and finally output the results. To perform the population and output, knowledge in iteration is required. From the previous example, suppose a student obtains a conditional probability  $P(U/C)$  of 0.89 for the prerequisite *iteration* in the pre-test. This indicates that the student understands the prerequisite sub-skill concept reasonably well. Therefore, the ITS will direct the student to the problem specification mentioned above, to solve. If the student fails to obtain a probability greater than 0.80, remedial lessons will be recommended to the student. This will enable them to improve the prerequisite concept before trying to solve the STL problems.

The algorithm for directing the student after the pre-test is outlined below:

```
Calculate P(U|C)
If P(U|C) > 0.80 Then
    Display the Problem Specification
Else
    Display choice – Remedial Lessons or Problem Specification
    Read choice
    If Remedial Lessons Then
        If conditional probability between 0.6 and 0.8 Then
            Display Pre-Test Review
        Else If conditional probability less than 0.6 Then
            Display Pre-Tutoring
        End If
    End If
End If
```

Figure 4.3 Directing Student after Pre-Test

During the tutoring session, the student's progress needs to be tracked and updated. On completion of each problem specification, the acquired skills will be added to the student's domain knowledge. The time taken to solve the sub problems will also be recorded. Another item to be taken into account is the number of times the student accesses the hints. All this information is useful in leading the students to various teaching strategies to suit their needs and abilities. The teaching strategies here also refer to different levels of hints to guide the students progressively to solve the problem. Four levels of hints (Table 4.3) are proposed for this C++ STL ITS:

- i) Brief explanation
- ii) Pre-Test Review
- iii) Pre-Tutoring
- iv) Demonstration

Table 4.3 Four Levels of Hints

HINTS	DESCRIPTION	EXAMPLE – Output the vector
Brief Explanation	Provide brief explanation of sub-tutorial	<i>vector</i> is similar to array with elements accessed using subscript or index. Like array, a loop is required to output the contents of a <i>vector</i> .
Pre-Test Review	Show performance of student in the pre-test for a required prerequisite to promote reflective learning	Below are your pre-test question answers related to this sub-tutorial : .... (The system will show the student's answer and the correct answer.)
Pre-Tutoring	Guide student step-by-step to complete the sub-tutorial	A <i>for</i> loop consists of 3 parts – initialization, test and incrementation separated by semicolons. (A series of sub-tutorials follow.) Write the statement for the initialization. ... Now, enter the test condition. ... Finally, add the incrementation part. ...

		<p>The statement within the <i>for</i> loop should be a <i>cout</i> statement that outputs the element in the <i>vector</i>. Write the code to access the element in the <i>vector</i> using the subscript.</p> <p>...</p>
Demonstration	Demonstrate i.e. present solution of the sub-tutorial based on actions taken in the Pre-Tutoring	<p>A <i>for</i> loop consists of 3 parts – initialization, test and incrementation separated by semicolons.</p> <p>The correct solution is:</p> <pre>for (i = 0; i &lt; 10; i++)</pre> <p>Within the <i>for</i> loop, a <i>cout</i> statement should be used to output the contents of the <i>vector</i>:</p> <pre>cout &lt;&lt; v[i] &lt;&lt; " ";</pre>

In the event that the student is unable to solve the sub-tutorial, he/she has the option to refer to the hints. The type of hint displayed depends on the performance of the student for the prerequisite sub-skill in the pretest. Suppose the student obtains a conditional probability of 0.89 for the prerequisite *iteration*, he/she will be directed to the first hint, which is, a brief explanation or notes on the sub-tutorial. Referring to Figure 4.2, an example of a sub-tutorial is *Step 3 – Populate the vector*. If the student still has difficulty in completing the sub-tutorial, then the *Pre-Test Review* will be invoked to show the student how he/she has responded to the pre-test question with respect to the prerequisite expected in the sub-tutorial. *Pre-Tutoring* will be offered next when the student still fails to answer correctly. Finally, if all fails, the solution will be demonstrated to the student. On the other hand, a conditional probability between 0.60 and 0.80, will direct student straight to the *Pre-Test Review*, whereas a value less than 0.60 will lead them to the *Pre-Tutoring* hint which guides the student step by step to solve the sub-tutorial. The selected threshold of 0.60 is justified from the range of sample results in Table 4.2. The algorithm below clarifies the various path during the tutoring session:

```

If Failed or Hint Selected Then
  If conditional probability > 0.80 Then
    Display Brief Explanation
    Update Student Model
    If Problem Unable to Solve-1 Then
      Display Pre-Test Review
      Update Student Model
      If Problem Unable to Solve-2 Then
        Display Pre-Tutoring
        Update Student Model
        If Problem Unable to Solve-3 Then
          Demonstrate solution
          Update Student Model
        End If
      End If
    End If
  End If
Else If conditional probability between 0.6 and 0.8 Then
  Display Pre-Test Review
  Update Student Model
  If Problem Unable to Solve-1 Then
    Display Pre-Tutoring
    Update Student Model
    If Problem Unable to Solve-2 Then
      Demonstrate solution
      Update Student Model
    End If
  End If
Else If conditional probability less than 0.6 Then
  Display Pre-Tutoring
  Update Student Model
  If Problem Unable to Solve-1 Then
    Demonstrate solution
    Update Student Model
  End If
End If
Else
  Proceed to next sub problem
End If

```

Figure 4.4 Various Paths during Tutoring Session

#### 4.5.4 *Updating Student Model*

The student model which includes the conditional probability for each topic,  $P(U/C)$  will be updated for every path that is taken by the student. The key idea in the update process is that the student's knowledge level progresses gradually. The adjustment is based on the different levels of performance of the students as they attempt to solve various problems. See Tables 4.4 (a), 4.4 (b) and 4.4 (c). The highlighted area represents the entry point to the recommended teaching strategy based on the conditional probability obtained after the pre-test.

Referring to Table 4.4 (a), students who obtained a conditional probability,  $P(U/C)$  less than 0.6 will be directed to the Pre-Tutoring as explained in section 4.5.3. If the student solved the problem correctly on the first attempt, no update is performed. The conditional probability for the topic is updated when the student has obtained the correct solution after the second attempt. If the student did not consult the hint and answered correctly, the conditional probability is increased accordingly. Similarly, the conditional probability is decreased if the student requires a second attempt with the same hint to obtain the correct answer. The conditional probability is set to 0.01 if student fails in all attempts with the given hints. The non-zero value of 0.01 used is based on the assumption that student has obtained some understanding after having been given the hints even though he or she fails to answer correctly. The updated value which is the new  $P(U/C)$  in Table 4.4 is based on the table of conditional probabilities in Appendix A. The inner workings of this algorithm will be explained and evaluated in Chapter 6.



Table 4.4 (a) How the Student Model is Updated when  $P(U/C) < 0.6$

$P(U/C) < 0.6$				Updated Values	
Brief Explanation	Pre-Test Review	Pre-Tutoring	Demonstrate	Total Attempts	New $P(U/C)$
No Hint Selected				1	No update
No Hint Selected				2	0.80
-	-	Y	-	1	No update
		Y	-	2	0.60
-	-	N	Y	1	0.60
		N	Y	2	Lowest
-	-	N	N	-	0.01

Table 4.4 (b) How the Student Model is Updated when  $0.6 \leq P(U/C) \leq 0.8$

$0.6 \leq P(U/C) \leq 0.8$				Updated Values	
Brief Explanation	Pre-Test Review	Pre-Tutoring	Demonstrate	Total Attempts	New $P(U/C)$
No Hint Selected				1	No update
No Hint Selected				2	Next Higher Value
-	Y	-	-	1	No update
	Y	-	-	2	0.60
-	N	Y	-	1	0.60
	N	Y	-	2	Next Lower Value or Lowest
-	N	N	Y	1	Next Lower Value or Lowest
	N	N	Y	2	Lowest
-	N	N	N	-	0.01

Table 4.4 (c) How the Student Model is Updated when  $P(U/C) > 0.8$ .

$P(U/C) > 0.8$				Updated Values	
Brief Explanation	Pre-Test Review	Pre-Tutoring	Demonstrate	Total Attempts	New $P(U/C)$
No Hint Selected				1	No update
No Hint Selected				2	1.00
Y	-	-	-	1	No update
Y	-	-	-	2	0.60
N	Y	-	-	1	0.60
N	Y	-	-	2	Next Lower Value 1 or Lowest
N	N	Y	-	1	Next Lower Value 1 or Lowest
N	N	Y	-	2	Next Lower Value 2 or Lowest
N	N	N	Y	1	Next Lower Value 2 or Lowest
N	N	N	Y	2	Lowest
N	N	N	N	-	0.01

The reasoning behind the update is that the performance of students normally progresses or weakens gradually and not abruptly. Therefore, the conditional probability is updated after the second attempt and not the first attempt. This is also an assurance that the students truly understood the knowledge acquired and not guessing his or her way through.

Figure 4.5 below depicts a graph that shows a gradual increase of a performance for a particular pre-requisite topic. The performance level is represented by the set of conditional probability,  $P(U/C)$  values  $\{0, 0.44, 0.67, 0.80, 0.89, 0.95, 1\}$ , calculated using the Bayesian Theorem. The updated values which is the new  $P(U/C)$  in Tables 4.4 correspond to the set of conditional probability values for each topic.

For example, if the student’s initial  $P(U|C)$  is less than 0.6, and he/she is able to answer correctly without hint twice consecutively, then his/her  $P(U|C)$  will be updated to 0.80.

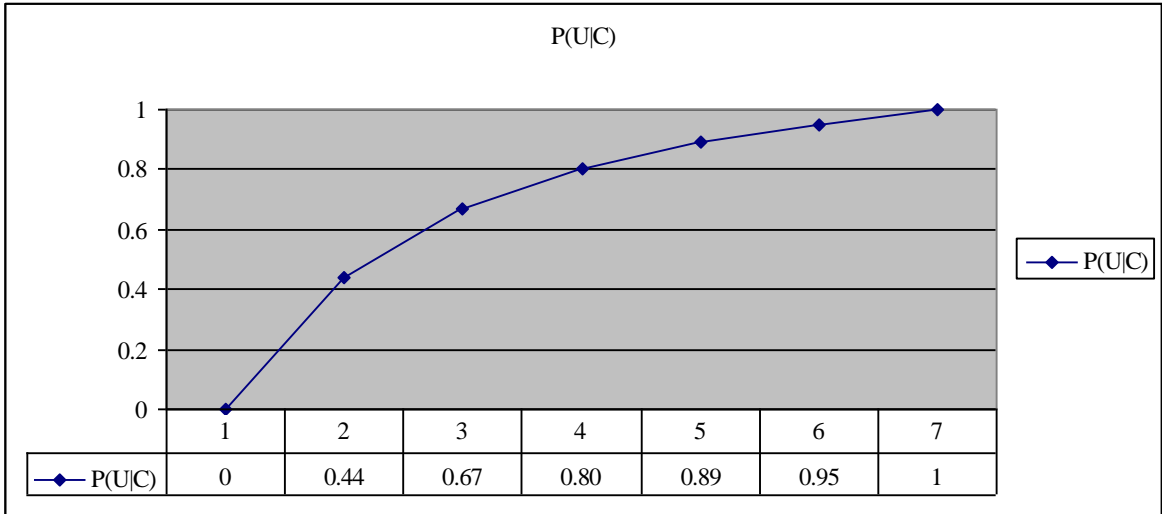


Figure 4.5 Gradual Change in Conditional Probability

This gradual update algorithm is similar to the idea in fuzzy logic. Fuzzy logic deals with degrees of membership and degrees of truth. The transition from membership to non-membership is gradual rather than abrupt (Zadeh, 1965). Similarly, the update algorithm models the student’s understanding gradually from less understanding to more understanding.

In the C++ STL ITS student model, a student obtaining an initial conditional probability of 0.67 will not be increased suddenly to 1.00 by attempting the problem correctly the first time. The application of the Bayesian Theorem captures a realistic progression of the student’s performance.

#### 4.5.5 *Evaluating and Categorising Student's Behaviour*

Upon completing each tutorial, it is useful to assign a stereotype according to the student's understanding to allow tutors to assess the student further. Information required for the stereotyping is obtained from the student's behaviour as he/she interacts with the ITS. Sison and Shimura (1996) defined student's behaviour as *any observable response that is used as input to the student modelling process*. In the C++ STL ITS, various information measured during the interactions is evaluated, and subsequently used to categorise the student. The pre-test performance results include the number of correct and incorrect answers, and conditional probabilities of understanding for each topic. As the student attempts the tutorial, the time spent to complete the tutorial, number of attempts to answer correctly, number of hints selected and the updated conditional probabilities are measured. The information observed is employed to categorise the students into four common stereotypes that concern the knowledge of student: novice, beginner, intermediate and advanced.

The Bayesian approach efficiently calculates the conditional probabilities of student's understanding. However, it is not capable of dealing with uncertainties in categorizing the student's understanding given the variables like tutorial duration, number of attempts and number of hints selected.

The proposal is to apply fuzzy logic techniques to provide human-like diagnosis of the student's knowledge. A general fuzzy expert system based on the application of fuzzy set theory will be implemented to perform the approximate reasoning. The

technique proposed to build fuzzy expert system is the Sugeno method. The method was named after Michio Sugeno (Sugeno, 1985). The operation of the fuzzy expert system involves the execution of four major tasks: fuzzification, rule evaluation, aggregation and defuzzification (see Figure 4.6)

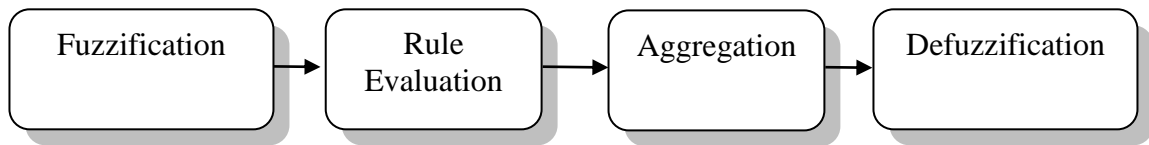


Figure 4.6 The Four Tasks in the Fuzzy Expert System

*Fuzzification* involves the choice of parameters, fuzzy input and output variables and defuzzified output variable(s), definition of membership functions for the input parameters and the description of fuzzy rules. The main purpose is to convert crisp values into a fuzzy set. The membership functions defined on the input parameters are applied to their actual values to determine the degree of membership for each fuzzy rule.

A fuzzy rule is an IF .. THEN rule of the standard form:

IF x is A THEN y is B

where

- x and y are linguistic variables
- A and B are linguistic values determined by fuzzy sets.

The  $x$  is  $A$  part is known as the antecedent or premise, whereas  $y$  is  $B$  is called the consequent or conclusion. In the Sugeno-style fuzzy inference, a singleton is used as the membership function of the rule consequent. A singleton is a fuzzy set with a membership function that is unity at a single particular point on the universe of discourse and zero everywhere else (Negnevitsky, 2005). The most commonly used zero-order Sugeno fuzzy model applies a constant value to the linguistic value (i.e.  $B$ ) of the consequent.

Fuzzy sets can have a variety of shapes represented by various membership function. A regular membership function will be applied to the C++ STL ITS fuzzy model. In most cases, a triangle or a trapezoid is adequate to represent the expert's knowledge.

To categorise the students, the C++ STL ITS will adopt the following input as the linguistic variables with their corresponding linguistic values:

- Conditional probabilities - high, medium and low
- Time to complete a tutorial - short, medium and long
- Number of attempts - high, medium and low
- Number of hints selected - high, medium and low
- Understanding - novice, beginner, intermediate and advanced

The linguistic variable *Understanding* is the desired consequent, where as, the rest are the antecedent.

Subsequently, the student's understanding is captured in fuzzy rules:

Examples:

Rule 1:

IF conditional\_probability is high

AND time\_taken is short

AND number\_of\_attempts is low

AND hints\_selected is low

THEN understanding is advanced

Rule 2:

IF conditional\_probability is low

AND time\_taken is very long

AND number\_of\_attempts is very high

AND hints\_selected is very high

THEN understanding is novice

*Rule Evaluation:* With the definition of the rules, membership function and fuzzified input, the truth value for the antecedent of each rule is computed, and applied to the consequent part of each rule. The output is one fuzzy set assigned to each output parameter for each rule. The max\_min operator will be applied to evaluate the rules (Hopgood, 2001).

In the *Aggregation* process, all the fuzzy sets assigned to each output variable are combined together to form a single fuzzy set for each output variable. The membership functions of all rule consequents from the previous stage are combined into a single fuzzy set.

*Defuzzification* is the process of converting fuzzy output set into the desired crisp value. The input for the defuzzification process is the aggregate output fuzzy set and the output is a single crisp number. The output is simply calculated by finding the weighted average of the singletons obtained from the aggregation operation.

Another method for fuzzy inference is known as the Mamdani method. It was introduced by Professor Ebrahim Mamdani of London University who built one of the first fuzzy systems to control a steam engine and boiler combination (Mamdani & Assilian, 1975). This method uses the *centroid* technique for defuzzification which relies on using the centre of gravity of the membership function to calculate the crisp value of the output parameter. Although the Mamdani method is widely accepted, its defuzzification process is computationally intensive. On the other hand, the Sugeno method is computationally efficient, works well with optimisation and adaptive techniques (Negnevitsky, 2005). The singleton output functions are able to satisfy the requirement of C++ STL ITS to categorise the student. This makes the Sugeno method an attractive choice for this ITS.

The implementation details of the fuzzy expert system is covered in Chapter 5.



#### 4.5.6 *Acquired Knowledge*

On completion of the tutoring session, the student will be directed to a post-test to find out whether he/she has gained the knowledge in the C++ STL satisfactory. This post-test also allows us to gauge the effectiveness of the tutoring system. The Bayesian Theorem will again be employed to determine whether the student has achieved the learning outcome of the STL.

### 4.6 Summary of Problems and Solutions

The main focus in this chapter is to ascertain the difficulties in learning and teaching the C++ STL. The findings have led to the proposed solution that models the problem specification based on prerequisite sub-skills. The ITS will also be equipped with the authoring of the domain knowledge dynamically which includes the authoring of pre-test, tutoring session and post-test. Hence, curriculum planner and implementer tutor can fully participate in the design of the curriculum and tutoring sessions as well as in the implementation of the tutorials respectively for their students for effective teaching and learning. Table 4.5 highlights the major problems discussed, in relation to the C++ STL, along with their proposed solution.

Table 4.5 Summary of Identified STL Problems and Proposed Solutions

IDENTIFIED PROBLEMS IN STL	PROPOSED SOLUTION
<p>Difficulties in learning STL</p> <ul style="list-style-type: none"> <li>• Poor understanding in prerequisite concepts like parametrization, templates and function overloading.</li> <li>• Unable to determine the most appropriate STL algorithm to apply.</li> </ul>	<ul style="list-style-type: none"> <li>→ • Model the problem specification based on prerequisite sub-skill. Strengthening prerequisites through drill and practice.</li> <li>→ • Provide tutoring sessions with various teaching strategies.</li> </ul>
<p>Difficulties in teaching STL</p> <ul style="list-style-type: none"> <li>• Dealing with diversified learning pace and knowledge.</li> <li>• Lack customization in the domain knowledge.</li> </ul>	<ul style="list-style-type: none"> <li>→ • Distinguish between Curriculum Planner and Implementer Tutor.</li> <li>→ • Dynamic Domain Modelling using various Authoring Tools</li> </ul>

Uncertainties in identifying student's knowledge in the prerequisite concepts have led to the application of the Bayesian Theorem to update the student model dynamically, providing a transparent student model that can be explained intuitively. With Bayesian reasoning, the probability of a hypothesis can be updated in the light of new evidences. Finally, the novelty of the student modelling is in the adjustment of the prerequisite sub-skill threshold during the tutoring session based on a gradual scale. A fuzzy expert system is also proposed to categorise the students for further assessment. Integrating the Bayesian Theorem and fuzzy logic will produce a dynamic student model that changes smoothly as the conditional probability is updated.

## Chapter 5 STL Tutor Architecture and Development

The C++ Standard Template Library (STL) Intelligent Tutoring System (ITS) provides tutoring in the domain of the C++ STL through drill and practice. Unlike previous ITS for programming which models the domain knowledge hierarchically based on elementary topics, C++ STL Tutor models the problem specifications instead based on prerequisite sub-skills. Its domain knowledge is modelled in a 2-level hierarchical structure based on topics and sub-topics of prerequisite concepts.

Authoring Tools are included in the design and development of the C++ STL Tutor. These include authoring and editing of the pre-test questions, tutorial sessions and post-test questions.

### 5.1 Overall Architecture

The overall design of the C++ STL ITS constitutes three main components – Student Modelling Module, Tutoring Module and Users Administration Module, as depicted in Figure 5.1. The modularity of the implementation allows more effective participation of both the curriculum planner and implementer tutor. The Student Modelling Module is a core component which contains the knowledge of the student. The Teaching Strategies Module includes the Pre-Test, Tutorial and Post-Test modules. Lastly, the Student, Tutor and Administrator modules are part of the Users Administration Module. The Student module is decomposed into Tutorial and Non-Tutorial.

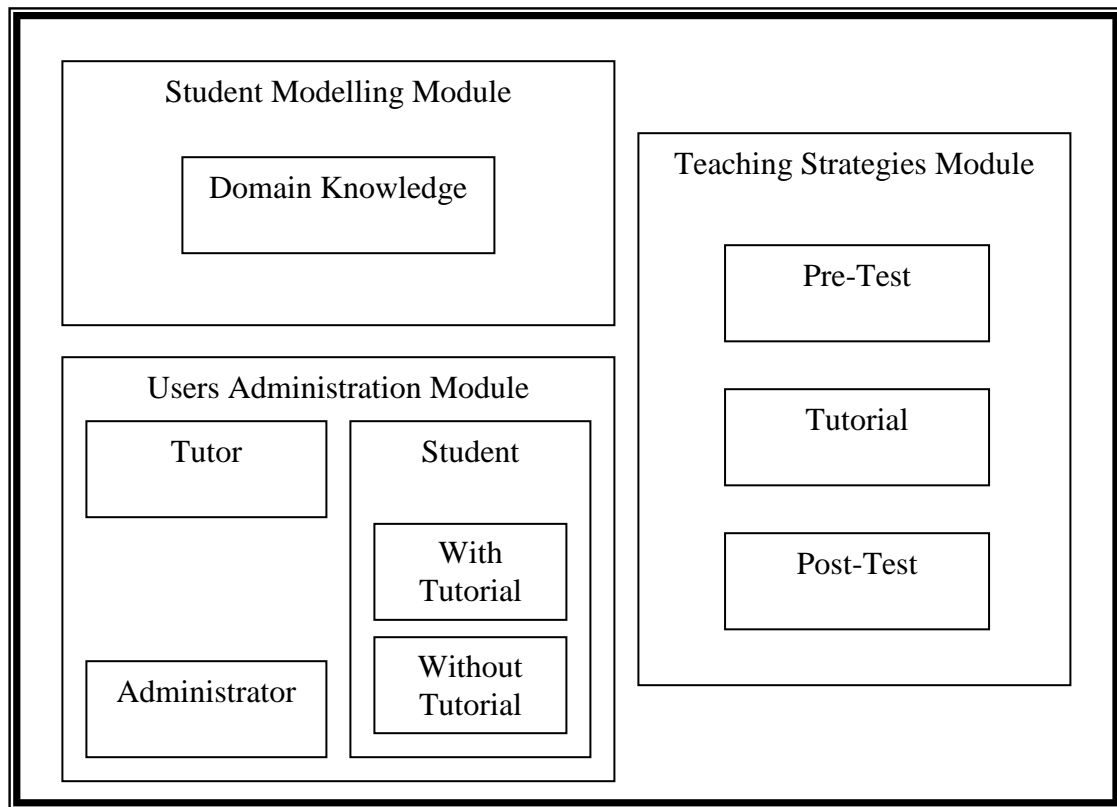


Figure 5.1 Components of C++ STL ITS

Figure 5.2 depicts the system architecture of the C++ STL ITS which is based on a typical ITS architecture. The four main modules are:

- i) Graphical User Interface Module
- ii) Student Modelling Module
- iii) Teaching Strategies Module
- iv) Domain Knowledge Module

The Graphical User Interface (GUI) Module handles the interaction between the user (tutor or student) and the system. It accepts input from the user and directs the information to the other modules for processing. The Student Module contains the dynamic model of a student which stores the personal details and the knowledge of the students. Input obtained

from the GUI Module through the Domain Knowledge Module and Teaching Strategies Module is used to update the student model. The Domain Knowledge Module stores tutorials, program specifications and topics on the C++ STL. This module interacts with the Teaching Strategies Module to allow authoring of the various teaching strategies. The responsibilities of the Teaching Strategies Module include the authoring and maintenance of pre-test, alternative teaching strategies and post-test.

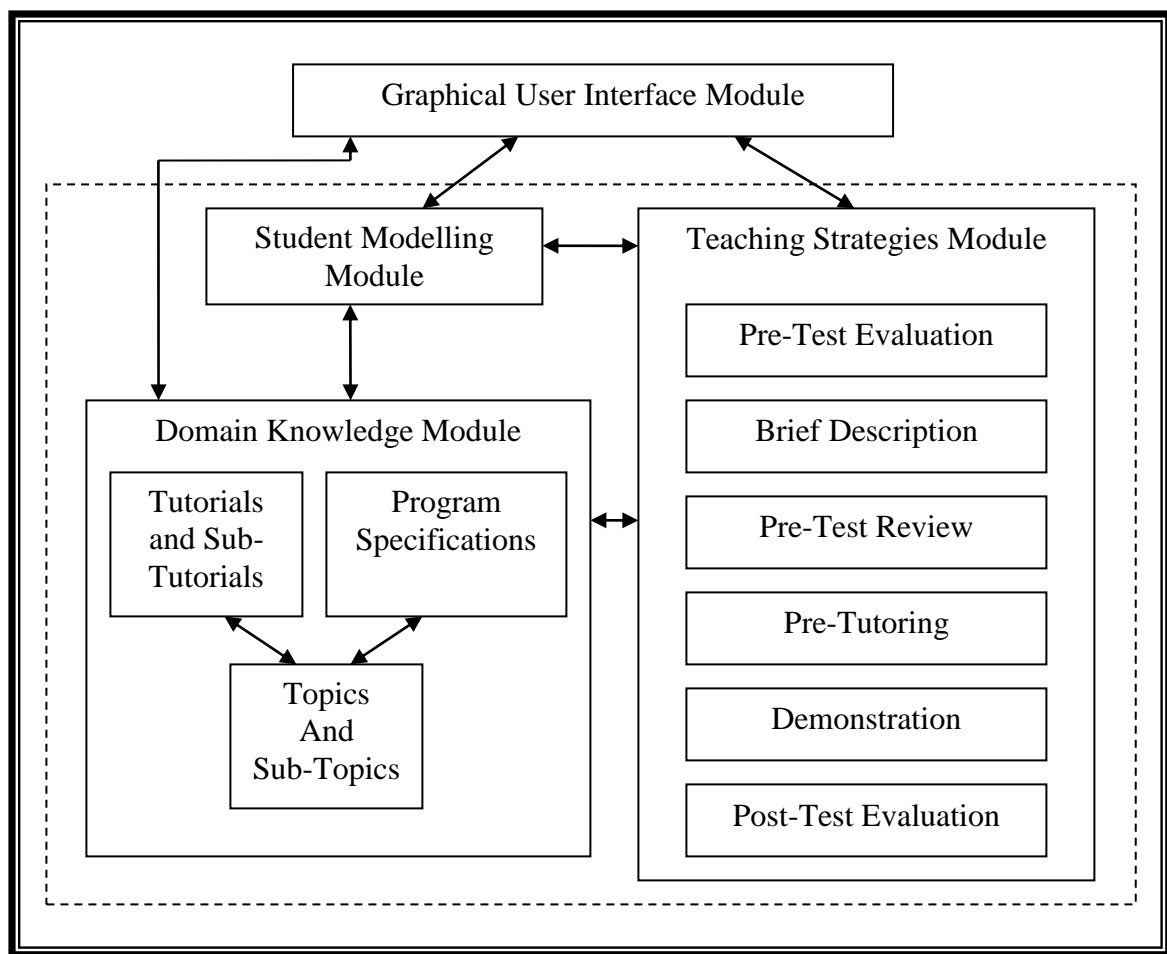


Figure 5.2 System Architecture of the ITS

## 5.2 Student Modelling Module

The Student Modelling Module is responsible for building the student model from information obtained through the student's interaction with the system. The student model consists of information obtained from three components. The first component is from the pre-test assessment which stores the prerequisite knowledge of the student. The second source is the tutorial module which keeps track of the student's learning path during the problem solving session and updates the student model accordingly. The third category of information for the student model is derived from the post-test evaluation. This information indicates the acquired skills of the student from the tutorials completed.

The student's knowledge of the domain is evaluated using the overlay model and represented using the Bayesian theorem. Tracking information measured from the problem solving session is then directed to the fuzzy expert system to categorize the students.

## 5.3 Domain Knowledge Module

The domain knowledge is divided into 3 parts:

- i) Topics and Sub-Topics
- ii) Program Specifications
- iii) Tutorials and Sub-Tutorials

C++ STL Tutor models the problem specifications based on prerequisite sub-skills. The prerequisite sub-skills are organized into topics and sub-topics, which is a 2-level hierarchical structure. Tutorials include the program specifications and are decomposed

into sub-tutorials to provide a bottom up view of the given problem. Authoring tools incorporated allow different knowledge domain to be designed for the tests and tutorials. Therefore, the programming language is not fixed on C++ but can be customized to Java, for example, and subsequently changed to Java Standard Template Library (JSTL).

### 5.3.1 Topic and Sub-Topic Repository

Tutors maintain the repository by creating new topics and sub-topics, editing and deleting existing ones. One topic has many sub-topics associated to it. This 2-level simple structure allows the tutors to group the related sub-topics together for easy reference. The structure also reduces the complexity of the mapping between the topics and Tutoring Module which includes pre-test, tutorial and post-test. Table 5.1 lists three main topics and their respective sub-topics. These are typical topics taught in the first year of a programming module in a Computing degree course. The remaining list is in Appendix B.

Table 5.1 Partial List of Topics and Sub-Topics

NO	TOPIC	SUB-TOPICS
1	Data	Data Type Variable Constant
2	Expression	Assignment Expression Conditional Expression
3	Input-Output	Output Input
4	Selection	if...else else if... switch...case
5	Iteration	for while do...while

### 5.3.2 Program Specifications Repository

The most common method to learn a programming language is through problem solving. Butz et al (2006) acknowledged that problem solving is an integral part of computer programming. In the C++ STL Tutor, students learn to apply the STL through drill and practice. The tutoring sessions are carried out after a pre-test evaluation of the student's prerequisites sub-skills. Each tutorial question includes a program specification for the student to solve. The program specification gives a top view of the program. Below is an example of a program specification for the STL vector domain:

“Declare a vector of 10 integers called *myVector*. Read 10 integers from the keyboard to populate *myVector* and then output them.”

The program specification above is designed to be clear and concise. It starts with the need for declaration, then listing out the tasks required. Tutors will have the flexibility to author the specifications based on the needs of their students. With the authoring tool, tutors are encouraged to be creative in setting of the specifications to enhance learning. The specifications can be maintained and updated when the need arises.



### 5.3.3 Tutorials and Sub-Tutorials Repository

The tutorial session demonstrates that the program specification can be divided into sub-tutorials, forming the bottom up view of a program. Each tutorial is linked to a set of required set of prerequisites and acquired skills which are the topics and sub-topics. To illustrate this, the same example as in Section 5.3.2 is used. Table 5.2 lists the prerequisites and acquired skills for the given specification above:

Table 5.2 An Example of a Program Specification with Prerequisites and Acquired Skills

<u>Program Specification</u> Declare a vector of 10 integers called <i>myVector</i> . Read 10 integers from the keyboard to populate <i>myVector</i> and then output them.			
Prerequisites Skills		Acquired Skills	
TOPICS	SUB-TOPICS	TOPICS	SUB-TOPICS
Class Template	Creating Instance	STL vector	Parametized Declaration
Constructor	Declaration – Parametized	STL vector	Populate
Fundamental	Input	STL vector	Output
Fundamental	Output		
Iteration	for		
Array	Range		
Array	Assignment		

The following table 5.3 shows the sub-tutorials that can be designed for the tutorial above. Three sub-tutorials are created with the prerequisites and acquired skills associated to each. Both skills are organized into topics and sub-topics.

Table 5.3 An Example of Sub-Tutorial Problems

NO	SUB-TUTORIAL	PREREQUISITE (TOPIC – SUB-TOPIC)	ACQUIRED SKILL (TOPIC – SUB-TOPIC)
1	Write the statement to create a vector called <i>myVector</i> that can store up to 10 integers.	Class Template – Creating Instance Constructor – Declaration parametized	STL vector - Declaration
2	Read 10 integers from the keyboard and populate the vector.	Fundamental – input Iteration – for Array – range Array – assignment	STL vector – Populate
3	Output the contents of the vector.	Fundamental – output Iteration – for Array – range Array – assignment	STL vector - Output

The purpose of the sub-tutorials is to reduce the complexity of the given problem through decomposition. It demonstrates to students that a problem can be decomposed into sub-problems and each problem can be solved independently. An ethnographic field study to identify problem solving skills used in Software Design (Arvanitis, et al, 2001) showed that students tend to avoid attempting a conceptual design after they received a problem specification. Most students applied a build-and-fix strategy and had difficulties verifying their programs. A further problem was discovered where students struggled to differentiate between syntactic and semantic errors. The findings concluded that it is beneficial to integrate problem solving design strategies into programming knowledge. The combination of the top down and bottom up views, which is the program specification and the sub-tutorials respectively, provided by the C++ STL ITS seeks to satisfy this requirement. Therefore, software design skills are established and reinforced into the

programming knowledge through the tutorials. The skills gained are desirable and transferable to more advanced or complex programming problems.

## 5.4 Teaching Strategies Module

The Teaching Strategies Module consists of three main modules:

- i) Pre-Test Module
- ii) Tutorial Module
- iii) Post-Test Module

The Pre-Test Module involves the setting of questions to test the prerequisite concepts of a student and the Pre-Test evaluation session. On completion of the Pre-Test, students are directed to the Tutorial Module. The Tutorial Module allows the tutor to design the tutorial sessions and set different teaching strategies. Lastly, the purpose of the Post-Test module is to test the effectiveness of the tutorial sessions. This includes questions related to the C++ STL.

### 5.4.1 *Pre-Test Module*

The Pre-Test Module is composed of two components: the Pre-Test Editor and the Pre-Test Evaluation Session. The Pre-Test Editor provides the tutor the tool to design pre-test questions to assess student's prerequisite sub-skills. The questions are based on the topics and sub-topics added into the repository. The Pre-Test Evaluation Session is responsible to provide the interface to evaluate the student's prerequisites and produce the necessary reports on completion.

The proposed pre-test editor includes an interface to enter the question and multiple choice based answers. There are two key reasons for employing a multiple choice test format. Firstly, simplicity – simplicity in assessment, design of questions as well as implementation. More importantly, the multiple choice test facilitates the application of the Bayesian theorem for the student model. The results simplify the calculation of the conditional probabilities of the student's understanding of a particular prerequisite sub-skill. The calculation was demonstrated in Chapter 4.

The Pre-Test repository includes the pre-test questions, answers and answers record. This information is used during the Pre-Test conducted for the students. The Pre-Test Evaluation Session provides a simple interface for the students. A sample screen shot of the interface is shown in Figure 5.3.

The 'Question Contents' button allows the student to view the entire pre-test questions repository (see Figure 5.4). Questions that are answered are highlighted. From this view, the student are allowed to choose to view all the questions or answered questions only or unanswered questions only. The purpose of this filtering is to promote flexibility during this self-assessment. From the question contents, students can choose which questions they wish to answer next. The different views show which questions are still left unanswered or answered already but need to be reviewed.

## Question 1.

Template operator << Question 1

Write the function declaration to overload the template version of the operator <<.

### Available Answers

Answer	Choice
template < class T > ostream & << (ostream &out, const int &x);	<input type="radio"/>
template < class T > ostream &operator << (ostream &out, const T &x);	<input type="radio"/>
template < class T > operator << (ostream &out, const T &x);	<input type="radio"/>
template < class T > ostream &operator << (out, const T &x);	<input type="radio"/>

[Next](#)

[Question Contents](#)

Intelligent Tutoring System, ITS ® is a registered trade mark of Christine Lee.  
Copyright © 2004 KBU International College, All rights reserved.

Figure 5.3 A Screen Shot of an Interface during the Pre-Test Evaluation Session

The order of the questions is generated randomly so that no students will have the same sequence of questions presented to them. This can avoid memorization of answers if the student chooses to repeat the test.

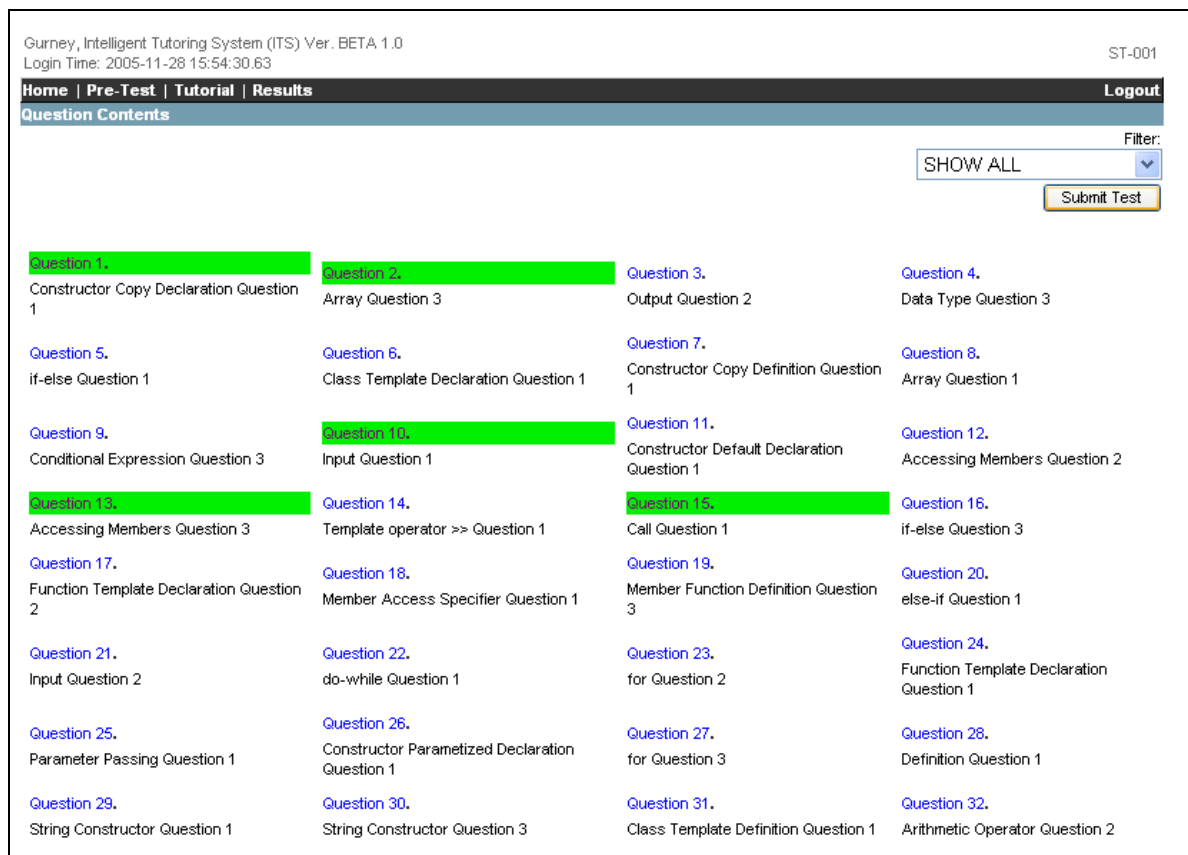


Figure 5.4 Partial Screen Shot of the View from 'Question Contents'

At the end of the Pre-Test, the conditional probabilities of the student's understanding for each prerequisite skill are generated and stored. The values are then used to direct the student's learning path during the tutorial session.

As the emphasis is on the performance of the prerequisite skills, a tabulated result showing the total number of questions answered correctly and incorrectly for each topic is produced. This will present an overall picture to the students on the topics that they are strong and weak at. The tutors will also benefit from this analysis.

### 5.4.2 *Tutorial Module*

The Tutorial Module allows the authoring of tutorial sessions and four main teaching strategies used during the tutoring process: The strategies are:

- i) Brief Explanation
- ii) Pre-Test Review
- iii) Pre-Tutoring
- iv) Demonstration

The teaching strategies are linked to the sub-tutorials. This means that each sub-tutorial consists of these 4 teaching strategies as described in Chapter 4.

### 5.4.3 *Post-Test Module*

The interface of the Post-Test module is similar to the Pre-Test Module. Editing tools are included to design the multiple choice test. The questions are related to the C++ STL.

Like the Pre-Test repository, the Post-Test repository also includes the post-test questions, answers and answers record. The interface and functionalities are also similar to the Pre-Test module. The main objective of the Post-Test is to evaluate the effectiveness of the tutoring session. Reports are generated for both tutors and students.

## 5.5 Users Administration Module

The responsibility of the Users Administration Module is to manage the various users of the system. The users include:

- i) Tutor
- ii) Tutorial Student
- iii) Non-Tutorial Student
- iv) Administrator

The Administration Module includes the management of various users of the system. The users consist of administrators, tutors (curriculum planner and implementer tutor), tutorial and non-tutorial students. Non-tutorial students refer to those who are involved in the tests but did not sit through the Tutorial Module. Its student model is simpler containing fewer results. The purpose of the distinction is purely for evaluation only.

The basic operations to manage the users are create, add, update, search, view and delete. Users are authenticated before the access to the system is granted. Upon successful authentication, their login sessions are logged for monitoring purpose. The data recorded includes user identification number, session identification, last IP address and date.

The next sections discuss the development details of the architecture described in the previous sections. The methodologies applied are also presented and rationalized.



## 5.6 The eXtreme Programming Methodology

eXtreme Programming (XP) is an agile software development methodology for delivering quality software while emphasizing the requirements. The application of XP in the C++ STL ITS allows the software to be implemented in a shorter period of time with tests created even before the software is written. XP is not just a process. It contains principles and practices that guide a project development. The beauty of XP is that it does small design continuously throughout the project one task at a time (see Figure 5.5). Testing is performed repeatedly until the task is fully implemented and working. Refactoring is carried out to make small changes that do not affect external behaviour. It improves the design and makes the code easier to understand.

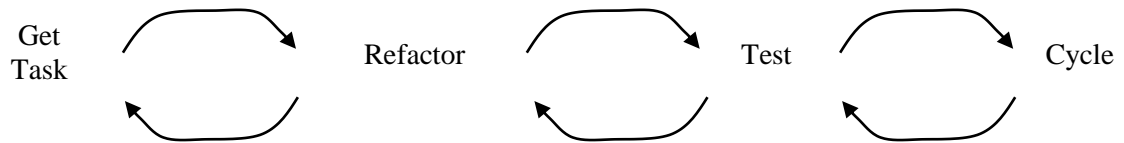


Figure 5.5 Cycles in XP (Astels, et al, 2002)

In conceptualizing the ITS, the following main tasks in XP were performed:

i) Creating a vision (Appendix C)

The vision card describes the purpose and objective of the system. It defines the scope of the system and provides the areas of focus.

ii) Writing User Stories (Appendix D)

A user story is the smallest unit of information necessary to allow the definition of various paths in the system. The main components are identified and described briefly.

### iii) Writing Acceptance Tests (Appendix E)

Acceptance tests are generated based upon user stories. The Appendix E includes test cases to test the Exception class and the Student session beans.

White box testing is applied on each unit to test the behaviour of the code.

The tasks above were iterated throughout the development stage to refine the system through a series of small releases. The first component released was the Administration Module, this was followed by the authoring tools for the pre-test, tutorial framework and post-test. The final release was the Tutoring Module.

Using the user stories from XP, the main classes in the system are identified and graphically designed using Unified Modelling Language. UML provides the visual notation to analyze the problem and its possible solutions using object-oriented concepts. The diagrams designed for the C++ STL ITS are class diagrams, general package diagram and general module package diagram depicted in Appendix F.

## 5.7 The 3-Tier System Architecture

A lot of research has been focused on Web-based education which gives rise to more web-based ITS systems on the Internet. In Brusilovsky's (1999) paper entitled "Adaptive and Intelligent Technologies for Web-based Education", he gave an impressive review of adaptive and intelligent technologies in the context of web-based distance education. He compared three categories of systems – hybrid of adaptive hypermedia and ITS, web-based

adaptive hypermedia systems and web-based ITS systems. Web-based ITS first appeared during the period 1995 – 1996. Examples include RAPITS (Woods et al, 1995) and ELM-ART (Brusilovsky et al, 1996). This list continues to grow with ELM-ART-II (Weber et al, 1997), CALAT ITS (Nakabayashi et al, 1997), ILESA (López et al, 1998), AlgeBrain (Alpert et al, 1999), VC PROLOG (Peylo et al., 2000), Web PVT (Virvou & Tsiriga, 2001a, b), WILEDS (Kassim et al, 2001), Web-EasyMath (Tsiriga & Virvou, 2002), SQLT-Web (Mitrovic & Hausler, 2003), JITS (Sykes & Franek, 2003), BITS (Butz et al, 2004) (see Table 5.4) and many more.

The latter systems used a two-tier or client/server architecture implemented using Java. Earlier systems employed the Common Lisp Hypermedia Server CL-HTTP completely written in Lisp (Mallery, 1994). CL-HTTP is a full-featured server for the Internet Hypertext Transfer Protocol applied in numerous Artificial Intelligence systems. CL-HTTP offers a Common Gateway Interface (CGI) to handle incoming URLs which are associated with a response function implemented in Lisp. In response, the appropriate HTML page is generated. With this architecture, CL-HTTP is a flexible and powerful tool for intelligent applications implementation on the web (Brusilovsky et al, 1998). Unfortunately, CGI-based applications are subject to these limitations: platform-specific, resource intensive, slow and difficult to maintain (J2EE BluePrints, 2001). Moreover, Common LISP is no longer one of the best choices available for today. The HTML-CGI architecture employed by some of the web-based ITS does not scale to more complex enterprise applications due to these reasons (Matena & Stearns, 1991): unstructured encapsulation of business process, difficult to develop, maintain and manage, intertwine business and presentation logic and difficult to maintain integrity of business rules.

Table 5.4 Overview of Implementation of Web-based Intelligent Tutoring Systems

INTELLIGENT TUTORING SYSTEM	IMPLEMENTATION
RAPITS (Woods et al., 1995) <ul style="list-style-type: none"> <li>Electronic book environment</li> <li>Made available to students to use on their off-campus machines</li> </ul>	<ul style="list-style-type: none"> <li>Asymetrix ToolBook</li> <li>Microsoft Word</li> <li>Microsoft Excel</li> <li>Microsoft Windows Dynamic Data Exchange</li> </ul>
ELM-ART (Brusilovsky et al, 1996) <ul style="list-style-type: none"> <li>Intelligent problem solving support</li> </ul>	Common Lisp Hypermedia Server CL-HTTP <ul style="list-style-type: none"> <li>HTTP server implemented in Common LISP</li> <li>Offers Common Gateway Interface (CGI) to handle incoming URLs</li> </ul>
ELM-ART-II (Weber & Specht, 1997) <ul style="list-style-type: none"> <li>Intelligent interactive textbook</li> </ul>	Programmable WWW-server CL-HTTP <ul style="list-style-type: none"> <li>Client – Web browser</li> <li>Server – Web server</li> </ul>
CALAT ITS (Nakabayashi et al, 1997) <ul style="list-style-type: none"> <li>Provides individual adaptation capability</li> </ul>	Client-server <ul style="list-style-type: none"> <li>Client – WWW browser</li> <li>Server consists of a WWW daemon process and back-end ITS processes.</li> </ul> HTML-CGI architecture.
ILESA (López et al, 1998) <ul style="list-style-type: none"> <li>Intelligent Learning Environment</li> <li>Knowledge and Task sequencing</li> </ul>	Client-server Used JDK 1.1.4
AlgeBrain (Alpert et al, 1999) <ul style="list-style-type: none"> <li>Support problem solving activities</li> </ul>	Client-server <ul style="list-style-type: none"> <li>Java Client – Web browser and AlgeBrain Java applet.</li> <li>Server – HTTP server and AlgeBrain Application Server</li> </ul> Distributed architecture.
VC PROLOG (Peylo et al, 2000) <ul style="list-style-type: none"> <li>Intelligent analysis of student's solutions</li> </ul>	Client-server <ul style="list-style-type: none"> <li>Java client</li> <li>Server written in ANSI-C, a POSTGRESS database and application process (PROLOG or LISP)</li> </ul>
Web PVT (Virvou & Tsiriga, 2001a,b) <ul style="list-style-type: none"> <li>Adaptive tutoring</li> </ul>	Client-server <ul style="list-style-type: none"> <li>Client – Web browser</li> <li>Server – contains web server and application server</li> </ul> HTML-CGI architecture.

Table 5.4 (continued)

<p>WILEDS (Kassim et al, 2001)</p> <ul style="list-style-type: none"> <li>Intelligent learning environment</li> </ul>	<p>Three-tier Java client-server architecture</p> <ul style="list-style-type: none"> <li>Client – Web browser and run Java Applets.</li> <li>Server – contains WWW server, student model and domain knowledge.</li> </ul> <p>Java servlet for communication between client-side Java applets and student model.</p>
<p>Web-EasyMath (Tsiriga &amp; Virvou, 2002)</p> <ul style="list-style-type: none"> <li>Individualized assessment</li> </ul>	<p>Client-server</p> <ul style="list-style-type: none"> <li>Client – Web browser</li> <li>Server – contains web server and application server</li> </ul> <p>HTML-CGI architecture.</p>
<p>SQLT-Web (Mitrovic &amp; Hausler, 2003)</p> <ul style="list-style-type: none"> <li>Adaptive tutoring</li> </ul>	<p>CL-HTTP server (Common Lisp Hypermedia Server)</p> <ul style="list-style-type: none"> <li>Client – Web browser</li> <li>Server – Web server</li> </ul> <p>Centralized architecture.</p>
<p>JITS (Sykes &amp; Franek, 2003)</p> <ul style="list-style-type: none"> <li>Support problem-solving activities with focus on methodology applied</li> </ul>	<p>Three-tier Java client-server architecture</p> <ul style="list-style-type: none"> <li>J2EE compliant server with web server</li> <li>Presentation layer users JSP</li> <li>JDBC connection from the EJBs to database.</li> </ul>
<p>BITS (Butz et al, 2006)</p> <ul style="list-style-type: none"> <li>Hypermedia-structured learning material</li> </ul>	<p>General framework developed using Microsoft Visual Studio.net. Developed instructional materials on Macromedia Studio MX. Developed ASP, XML files and HTML pages.</p>

Generally, in two-tier architecture or client/server architecture, the user interface client on the web invokes services from the web server. In most two-tier designs, most of the application part of processing is in the client environment. The web server usually provides the part of the processing related to accessing the database. The client (web browser) requests execution of some job, while the application on the server executes this job. The main advantage of two-tier architecture is that it reduces the complexity in implementation,

because the presentation logic and business logic reside in the same process. Ironically, this is also the main disadvantage of two-tier architecture. Matena and Stearns (1991) discussed that because developers cannot cleanly and clearly separate business logic from the user interface, a number of problems arise: “easily compromised database integrity, difficult to administer, difficult to maintain, exposure to security violations, limited scalability, restricted client architecture requirements, and limitation to one presentation type.” This could be one of the reasons ITSs are not adopted commercially.

The C++ STL ITS adopts a three-tier system architecture which is based on the Java 2 Platform, Enterprise Edition (J2EE) architecture. The three-tier architecture overcomes the limitations of the two-tier architecture by splitting the presentation logic from the business logic. The multi-tiered ITS architecture shown in Figure 5.6 consists of the following tiers:

- i) Client-tier layer – contains Presentation Logic
- ii) Middle-tier layer – contains Business Logic
- iii) Data Source-tier layer – contains Database

J2EE is a platform for developing distributed software applications. The J2EE platform supports servlets and Java Servlet Pages technologies that overcome the limitations in CGI applications. The key benefits include:

- reduces development time to concentrate on business logic
- promotes scalability and manageability in development

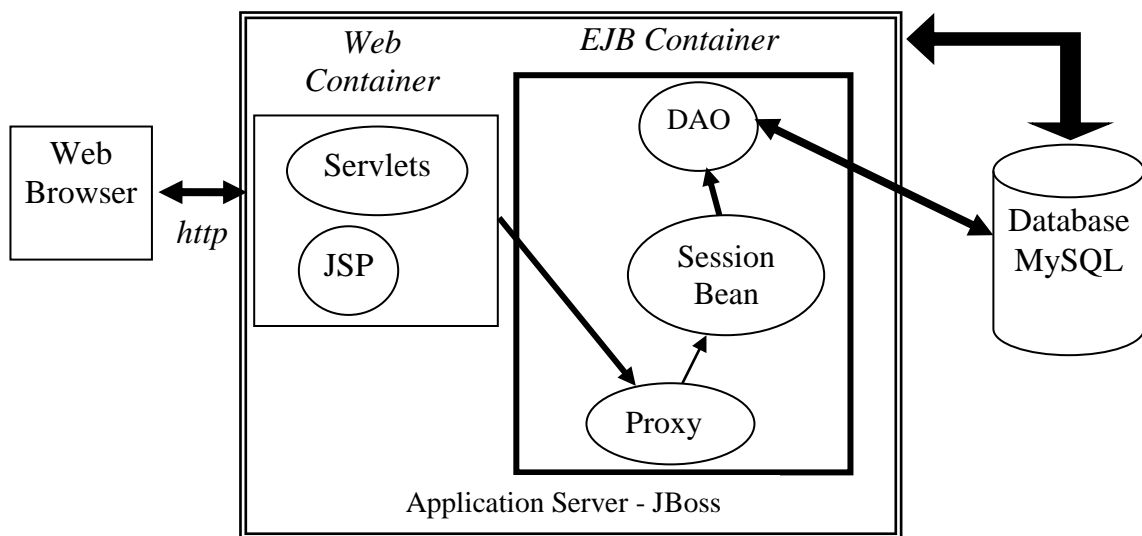


Figure 5.6 3-Tier System Architecture of C++ STL ITS

Kinshuk (2002) observed that most ITS ended up as prototypes and were not deployed in the actual learning environment. There is also no evidence of commercialization of ITS even though various evaluation studies confirmed the effectiveness of using these systems. This shows the need to reduce the gap between implementation of ITS and profit-driven organizations. One way of achieving this is the application of industry standards for software engineering. The combination of XP and J2EE forms a strong foundation in the web development of the C++ STL ITS. Their industry practices are applied to enhance the value and contribution of this research. It is an attempt to bring the C++ STL ITS closer to the industry level and promote its adoption.

### *5.7.1 The Client-tier Layer*

The client tier which represents the Graphical User Interface Module in the ITS consists of two parts:

- i) dynamic Web pages containing HTML, which are generated by the Web Container components running in the Middle-tier, and
- ii) a Web browser, which renders the pages received from the application server and interacts with the user.

### *5.7.2 The Middle-tier Layer*

The middle-tier accepts user responses from the client-tier and generates the appropriate presentation logic. This tier also handles the core business logic of the application implemented as Enterprise Java Bean (EJB) components. It consists of the:

- Application Server
- Java Servlet and Java Servlet Page Server
- Enterprise Java Bean (EJB) components
  - Data Access Objects (DAO)
  - Session Beans
  - Proxy



Servlets are Java programs with HTML embedded that run on the application server. Java Servlet Pages (JSP) are HTML pages with Java code embedded. Both servlet and JSP are responsible for generating and transmitting web browser markup language to the http client through the http channel dynamically.

The EJB components provide for persistence management, business processing, transaction processing and distributed processing capabilities. The EJB architecture provides a standard for developing reusable Java server components that run in the application server. In the C++ STL ITS, the EJB specification defines session beans which are stateless. These beans are distributed transactional component which provide a single-use service, do not maintain any state, do not survive server crashes, and are relatively short lived. Modules are encapsulated within respective session façade patterns implemented as session beans. The modules are admin, authentication, estudent, posttest, pretest, student, topic, tutor and tutorial. (Appendix G). A session façade encapsulates business-tier components and expose services to remote clients. Clients access a session façade instead of accessing business components directly. This will improve manageability, centralize logic, increase flexibility, and improve ability to cope with changes (Alur et al, 2003).

The Data Access Object (DAO) manages the connection with the data source to retrieve and store data. It encapsulates and abstracts access to the data from the MySQL database. The DAO design pattern is one of the core J2EE patterns documented in the Core J2EE Pattern Catalog at <http://java.sun.com>. Using DAO and Hibernate allows for complete transparency between the system and MySQL.

Hibernate is a Java-based middleware designed to complete the Object Relational mapping model and handles the persistency of those objects. Hibernate also provides data querying and retrieval functions in the Java environment (Pugh & Gradecki, 2004).

The listing below shows the mapping for the Java class UserDO.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 2.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">

<hibernate-mapping>
    <class name="com.its.module.user.model.UserDO" table="ITS_USER">
        <id name="sysId" column="SYS_ID" type="long">
            <generator class="native"/>
        </id>
        <property name="username" column="USERNAME"
            type="java.lang.String" not-null="true"/>
        <property name="password" column="PASSWORD"
            type="java.lang.String" not-null="true"/>
        <property name="name" column="NAME"
            type="java.lang.String" not-null="true"/>
        <property name="email" column="E_MAIL"
            type="java.lang.String" not-null="true"/>
        <property name="department" column="DEPARTMENT"
            type="java.lang.String" not-null="false"/>
        <property name="type" column="TYPE"
            type="java.lang.String" not-null="true"/>
        <property name="createdTimestamp" column="CREATED_TIMESTAMP"
            type="java.sql.Timestamp" not-null="true"/>
        <property name="lastModifiedTimestamp"
            column="LAST_MODIFIED_TIMESTAMP"
            type="java.sql.Timestamp" not-null="true"/>
        <many-to-one name="statusDO"
            class="com.its.module.status.model.StatusDO"
            column="STATUS_SYS_ID" not-null="true"/>
    </class>
</hibernate-mapping>
```

In the previous listing, the mapping document is XML based and includes elements for specifying an identifier as well as the attributes of the mapped objects. Once a mapping document has been created, the appropriate database table can be added to MySQL, thus completing an Object Relational mapping from a Java class to the database.

### *5.7.3 The Data Source-tier Layer*

The Student Module, Teaching Strategies Module and Domain Knowledge Module reside in the data source-tier layer. In the data source-tier, the DAO is used to invoke the MySQL database. The tables implemented are described in Appendix H.

## **5.8 XML Syntax Parser and Student Model Update Algorithm**

The XML syntax parser and the Student Model update algorithm are part of the Tutorial Module which resides in the Middle-tier layer. The parser and the algorithm play a major role in the workings of the tutorial sessions. The parser is required to check the correctness of the student's tutorial answers, where as the update algorithm is responsible in maintaining the student model dynamically during the tutorial.

### 5.8.1 XML Syntax Parser

In the authoring of the tutorials, the answers are represented in a simple XML-based format. The data type definition for the format is displayed below:

```
<?xml version='1.0' encoding='UTF-8'?>
<!ELEMENT syntax (eval)*>
<!ELEMENT eval (#PCDATA|eval)*>
<!ATTLIST eval
  join (and|or) #IMPLIED
  allow (string|number) #IMPLIED
  ignorecase (true|false) #IMPLIED>
```

The XML elements used are *syntax* and *eval*. The element *syntax* defines the start of the answer syntax using the tags `<syntax>` and `</syntax>`. The element *eval* represents the actual C++ syntax to be evaluated, within the tags `<eval>` and `</eval>`. Blank spaces will be trimmed. The attributes consist of *join*, *allow* and *ignorecase*. The valid *join* values are *and* and *or*. The allowed values are *string* and *number*. The third attribute is used to indicate whether a statement or variable is case sensitive or not.

An example is shown in Figure 5.7. The example shows the XML describing the answer to populate a vector from the keyboard using a *for* loop. Each element of the C++ syntax is entered with the `<eval>` and `</eval>` tags on a new line to allow blank spaces to be trimmed during the syntax validation process. Some special characters are also required within the syntax. Table 5.5 lists the allowable special characters used within the *eval* tags. The characters are `<`, `>` and `&`.

Table 5.5 Special Characters

SPECIAL CHARACTER	ACTUAL CHARACTER
&lt;	<
&gt;	>
&amp;	&

```

<syntax>
  <eval>for</eval>
  <eval>(</eval>
  <eval>i</eval>
  <eval>=</eval>
  <eval allow="number">0</eval>
  <eval>;</eval>
  <eval>i</eval>
  <eval>&lt;</eval>
  <eval allow="number">10</eval>
  <eval>;</eval>
  <eval>i++</eval>
  <eval>)</eval>
  <eval>{</eval>
  <eval>cin</eval>
  <eval>&gt;&gt;</eval>
  <eval>aVector</eval>
  <eval>[</eval>
  <eval>i</eval>
  <eval>]</eval>
  <eval>;</eval>
  <eval>}</eval>
</syntax>

```

Figure 5.7 XML-based format describing the answer to populate a vector from keyboard.

Figure 5.8 illustrates the three main phases in the XML Syntax Parser to validate the student's answer during the tutorial session. The phases are Validate Sub-Tutorial Answer, Build ITS Syntax Model and Validate Answer Syntax.

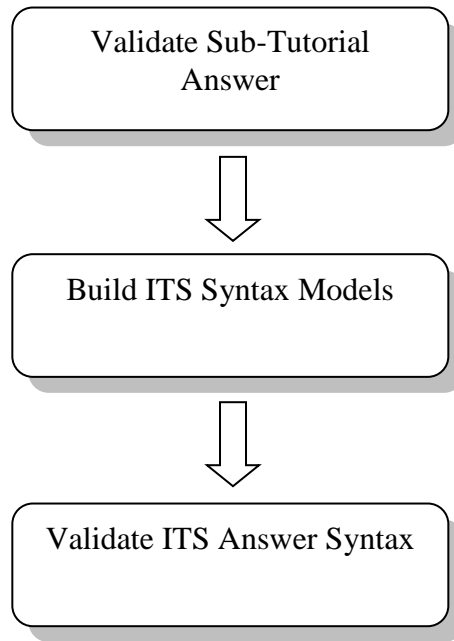


Figure 5.8 Phases in the XML Syntax Parser

In the first phase, two parameters are required – student’s answer and XML syntax answer. The purpose of this phase is to extract the elements in the XML syntax answer required for a particular sub-tutorial that the student is solving. The elements are fed into the second phase to build a list recursively forming the syntax model. Allowable elements are checked within this phase. The list is then recursively validated in the third phase against the required answers, matching element by element. The code is furnished in Appendix I.

### 5.8.2 Student Model Update Algorithm

The student model update is handled by a class called *TutorialTrxnSBBean* in the Tutorial Module. The update is based on the algorithm outlined in Table 4.4 discussed in Chapter 4. Figure 5.9 provides a clearer flow of the algorithm using a flowchart and supplemented with pseudocodes in Figure 5.10 and Figure 5.11.

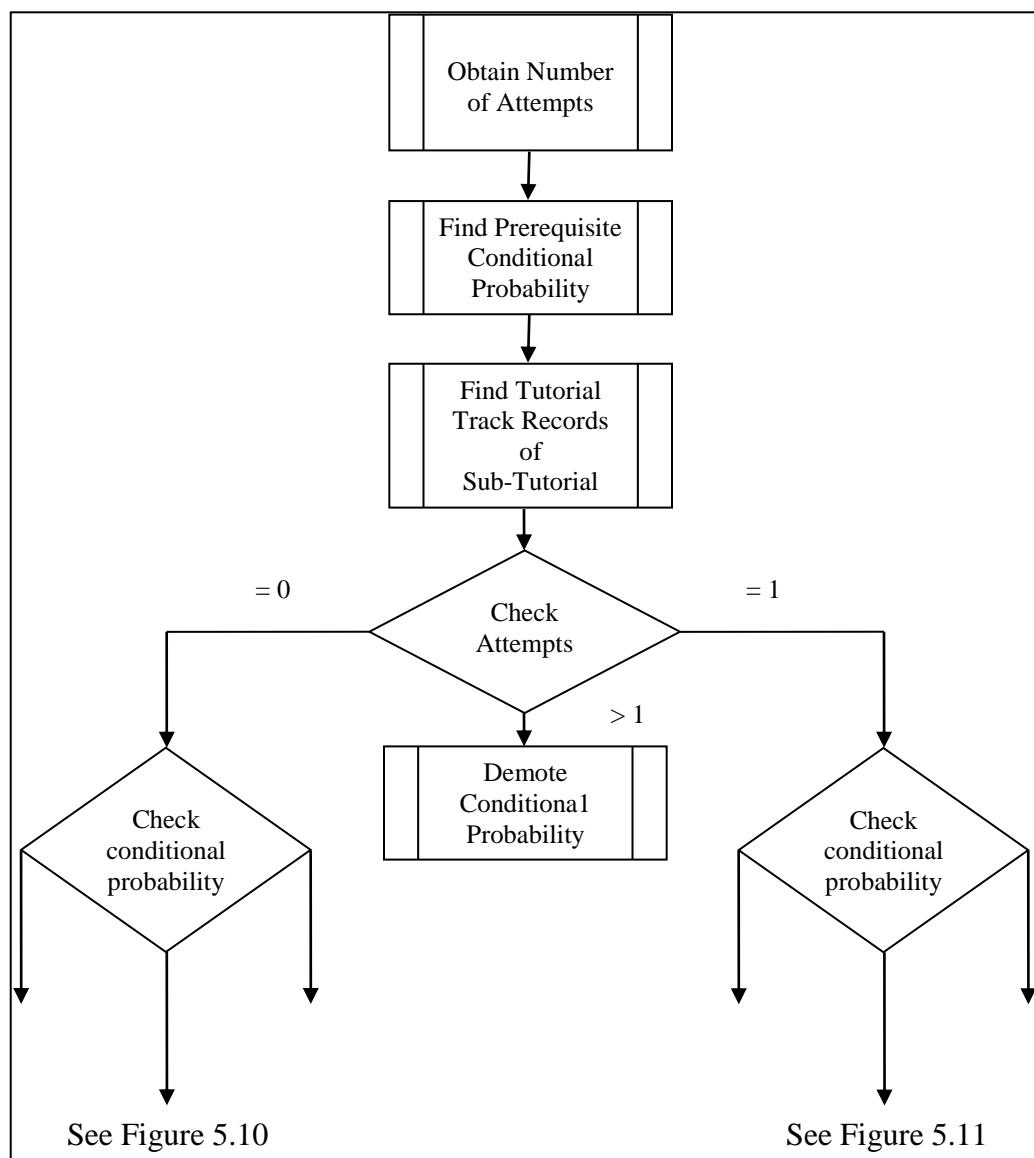


Figure 5.9 Flowchart of Student Model Update

The initial conditional probabilities for each topic were obtained from the Pre-Test results. Each tutorial is linked to a set of prerequisite topics with conditional probabilities attached to the topics. An average of these conditional probabilities is calculated and checked against the various thresholds. Table 5.6 shows the prerequisite topics and their respective initial conditional probabilities achieved by a student, for a tutorial question entitled “vector – populate using push\_back()”. The average of 0.88 is obtained. Therefore, the first hint displayed for the student will be *Brief Explanation*. These conditional probabilities are then updated during the tutorial session. Generally, the update is performed according to the number of attempts to solve the problem and type of hint presented to the students. Brief explanation, Pre-Test Review, Pre-Tutoring and Demonstration are the various hints or teaching strategies offered to the students during the tutorial session.

Table 5.6 Example of Conditional Probabilities Achieved by a Student

<b>PREREQUISITE TOPIC</b>	<b>CONDITIONAL PROBABILITY</b>
Class Template	0.57
Constructor	0.89
Class	0.94
Array	1.00
Iteration	0.89
Input-Output	1.00
<b>AVERAGE</b>	<b>0.88</b>



```
// Number of attempts = 0
If current conditional probability > 0.8 Then
    If from Pre-Tutoring or from Demo Then
        Demote conditional probability
    Else If from Pre-Test Review
        Set new conditional probability to 0.6
    End if
Else If 0.6 <= current conditional probability <= 0.8
    If from Demo Then
        Demote conditional probability
    Else If from Pre-Tutoring
        Set new conditional probability to 0.6
    End If
Else
    If from Demo Then
        Set new conditional probability to 0.1
    End If
End If
```

Figure 5.10 Update Pseudocode for Number of Attempts = 0

```

// Number of attempts = 1
If current conditional probability > 0.8 Then
    If not from Brief Explanation and not from Pre-Test Review
        and not from Pre-Tutoring and not from Demo Then
        Set new conditional probability to 1.0
    Else If from Pre-Test Review or from Pre-Tutoring or from Demo
        Demote conditional probability
    Else If from Brief Explanation
        Set new conditional probability to 0.6
    End If
Else If 0.6 <= current conditional probability <= 0.8
    If from Pre-Tutoring or from Demo Then
        Demote conditional probability
    Else If from Pre-Test Review
        Set new conditional probability to 0.6
    End If
Else
    If from Demo Then
        Demote conditional probability
    Else If from Pre-Tutoring
        Set new conditional probability to 0.6
    End If
End If

```

Figure 5.11 Update Pseudocode for Number of Attempts = 1

The flowchart and the pseudocode above are for the *updateCP* ( ) function in the class *TutorialTrxnSBBean*. The function takes in two arguments: sub-tutorial and student data objects. It is invoked after the student has submitted the answer for a sub-tutorial.

The entire code for the class *TutorialTrxnSBBean* is furnished in Appendix I.

## 5.9 Fuzzy Expert System

The purpose of the Fuzzy Expert System is to categorize students by a stereotype based on their interactions with the C++ STL ITS and performance during the tutoring sessions. The system shall be called the Fuzzy Stereotyping of Students (FSS).

A typical process in developing the FSS incorporates the following steps (Negnevitsky, 2005) :

- i) Specify the problem and define linguistic variables.

The objective here is to categorize the students understanding of the C++ STL into novice, beginner, intermediate and advanced.

The linguistic variables are defined in Table 5.7.

- ii) Determine fuzzy sets.

The Gaussian fuzzy membership function which produces a very smooth output has been selected to represent the fuzzy sets. Figures 5.12 to Figures 5.15 show the fuzzy sets for all the linguistic variables defined for this problem. Figure 5.12 depicts the membership function plots for the input variable *Conditional Probabilities* from very low to high. The key point is to ensure sufficient overlap between neighbouring fuzzy sets for the fuzzy system to respond smoothly.

- iii) Derive and construct fuzzy rules.

For this problem, there are four input and one output variables. A detailed analysis of the tutorial session produces 128 rules that represent complex relationships between the variable used in the FSS. The output was elicited from an expert's knowledge. These rules are listed in Appendix J.

- iv) Encode the fuzzy sets, fuzzy rules and procedures to perform fuzzy inference into the FSS.

This process is described below using the MATLAB Fuzzy Logic Toolbox.

- v) Evaluate and tune the system.

This last step is the most tedious tasks. The objective is to determine whether FSS meets the requirements specified at the beginning. The evaluation details are discussed in section 6.6.

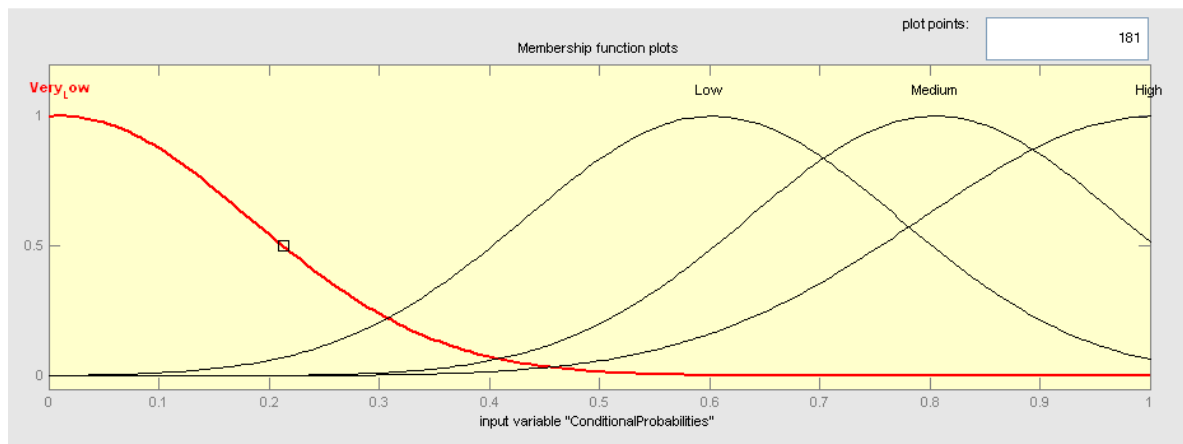


Figure 5.12 Fuzzy Sets of Conditional Probabilities

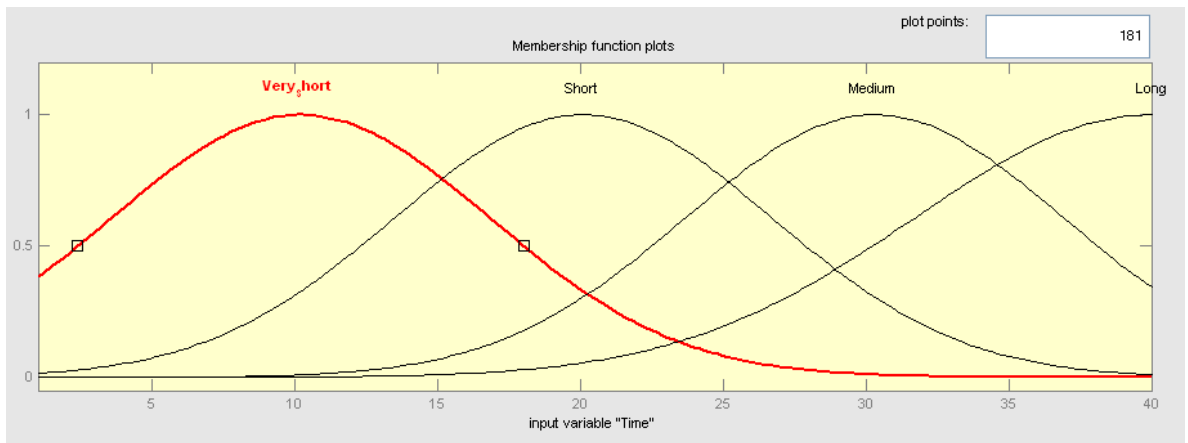


Figure 5.13 Fuzzy Sets of Time

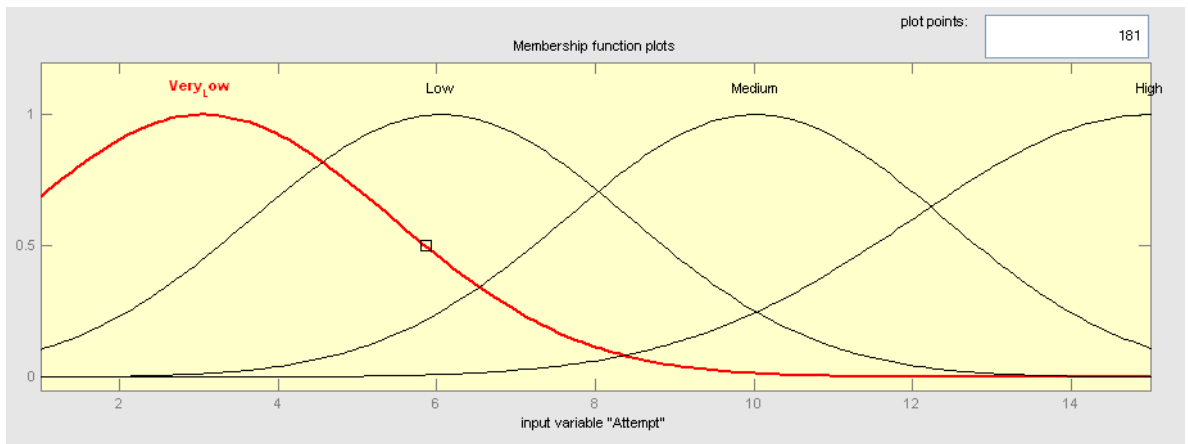


Figure 5.14 Fuzzy Sets of Number of Attempts

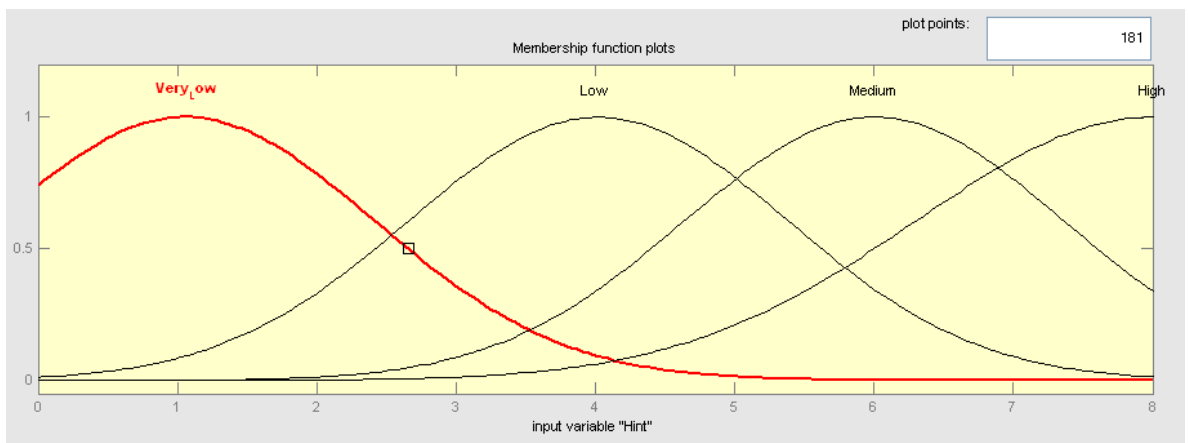


Figure 5.15 Fuzzy Sets of Number of Hints

For rapid development of FSS, one of the most popular tools, the MATLAB Fuzzy Logic Toolbox from the MathWorks is used. The toolbox provides graphical interactive tools to create and edit fuzzy inference systems within the framework of MATLAB. There are five main integrated Graphical User Interface (GUI) tools for building, editing, and observing fuzzy inference systems in the Fuzzy Logic Toolbox: the Fuzzy Inference System or FIS Editor, the Membership Function Editor, the Rule Editor, the Rule Viewer, and the Surface Viewer. The tools are easy to use and reduce the laborious tasks to tune the expert system to meet the desired performance.

The FIS Editor handles the specification of the input and output parameters. In FSS, there are four input variables or parameters and one output, known as the linguistic variables. See Table 5.7 for the list of the linguistic variables with their corresponding linguistic values and numerical range. These values were obtained from the Tutorial Performance Report described in section 6.4.

The Membership Function Editor is used to define the shapes of all the membership functions associated with each variable. Some of the membership functions provided include Triangular, Trapezoidal, Gaussian, Bell, S and Z membership functions. The membership functions chosen for FSS are Gaussian for the input variables, and Linear for the output variables. Both functions provide an adequate representation of the expert knowledge.

Table 5.7 Linguistic variables, values, range and membership function

Linguistic Variable: Conditional Probabilities, <i>Conditional Probabilities</i> (Type: Input)		
Membership Function: Gaussian		
LINGUISTIC VALUE	NOTATION	NUMERICAL RANGE
Very Low	VL	[0, 0.01]
Low	L	[0.4, 0.6]
Medium	M	[0.62, 0.8]
High	H	[0.84, 1]
Linguistic Variable: Time taken, <i>Time</i> (Type: Input)		
Membership Function: Gaussian		
LINGUISTIC VALUE	NOTATION	NUMERICAL RANGE
Very Short	VS	[1, 10]
Short	S	[11, 20]
Medium	M	[21, 30]
Long	L	[31, 40]
Linguistic Variable: Number of Attempts, <i>Attempts</i> (Type: Input)		
Membership Function: Gaussian		
LINGUISTIC VALUE	NOTATION	NUMERICAL RANGE
Very Low	VL	[1, 3]
Low	L	[4, 6]
Medium	M	[7, 10]
High	H	[11, 15]
Linguistic Variable: Number of Hints, <i>Hints</i> (Type: Input)		
Membership Function: Gaussian		
LINGUISTIC VALUE	NOTATION	NUMERICAL RANGE
Very Low	VL	[0, 1]
Low	L	[2, 4]
Medium	M	[5, 6]
High	H	[7, 8]
Linguistic Variable: Stereotype of Understanding, Understanding (Type: Output)		
Membership Function: Linear		
LINGUISTIC VALUE	NOTATION	NUMERICAL RANGE
Very Low	VL	[0, 0.01]
Low	L	[0.4, 0.6]
Medium	M	[0.62, 0.8]
High	H	[0.84, 1]

The Rule Editor allows the construction of the fuzzy rules in FSS using the fuzzy linguistic variables specified previously. The fuzzy rules defined in FSS are found in Appendix J.

The Rule Viewer and the Surface Viewer are used for viewing the FIS results. The Rule Viewer is used as a diagnostic tool to show which rules are active and how individual membership function shapes are influencing the results. The Surface Viewer is used to display the dependency of one of the outputs on any one or two of the inputs — that is, it generates and plots a three-dimensional output surface map for the FSS. Both views allow the FSS to be evaluated and tuned to meet the specified requirements.

## 5.10 Development Tools

The development tools used for the C++ STL ITS project are listed below:

- Apache Tomcat (part of JBoss application server) serves as the platform for the JSP and Java Servlet implementation.
- Apache Ant provides the framework that enables rapid processes configuration for all phases of the software life cycle. An Ant build process is described in an XML file, called a buildfile (Appendix K).
- JBoss Application Server is a J2EE certified platform for developing and deploying the application.
- The NetBeans Integrated Development Environment (IDE) together with the Java 2 Platform Standard Edition Development Kit (JDK) provides the environment for writing, compiling, testing, and debugging the web application.
- MATLAB Fuzzy Logic Toolbox



## **Chapter 6 C++ STL ITS System Evaluation – Methodologies and Results**

The adoption of eXtreme Programming methodology and J2EE standards has been successful in ensuring the completeness of the implementation of the C++ STL Intelligent Tutoring System (ITS). Vigorous testing was carried out to verify and validate the components in the ITS using black box and white box testing strategies. Functional testing was carried out modularly, starting with the Users Administration Module, followed by the Tutoring Module and lastly the Student Modelling Module. The next important stage is the evaluation of the ITS. The main purpose of this process is to evaluate the effectiveness of the Tutoring Module to guide students in learning and applying the C++ STL.

As more ITSs are developed and employed in education, business and government, the evaluation of ITSs is being taken more seriously to ensure its effectiveness (Iqbal et al, 1999, Mark & Greer, 1993). Mark and Greer (1993) noted that most ITS researchers have in the past concerned themselves only with envisioning the potential of ITSs and investigating the implementation issues involved in constructing actual components and systems. However, times are changing. Ainsworth (2005) observed that now, more and more researchers are reporting on formative and summative evaluation results, and statistical analyses. Ainsworth also strongly commented that “without evaluation, there is no point in doing anything...”

## 6.1 Evaluation Methodologies and Evaluation Requirements

Iqbal et al (1999) proposed a classification of evaluation methods based on two primary questions relating to the target of evaluation and learning environment:

- i) What is being evaluated: the whole system or just a component?
- ii) Is it possible to experimentally adjust conditions in the evaluation, and how many users are available for the purpose of evaluation?

Iqbal et al (1999) classified evaluation methods into two dimensions according to the above questions. The first dimension focuses on the *degree of evaluation* – whole or part of a system. If the method evaluates just a component of a system, it can be considered for *internal evaluation*. On the other hand, methods that test the whole system are suitable for *external evaluation*.

The second dimension is concerned with the *environment of evaluation* – *experimental research* or *exploratory research*. According to Iqbal et al (1999, p. 2): “*Experimental research* requires experiments varying systematically the independent variable(s) while measuring the dependent variable(s) and ensuring random assignment of participants to conditions and requires statistically significant groups. *Exploratory research* includes in-depth study of the system in a natural context using multiple sources of data, usually where sample size is small and the area is poorly understood.” Table 6.1 shows the classification of the evaluation methods reviewed and proposed. This classification simplifies the selection for a suitable method. The methods are adapted by various ITS researches from expert systems development, computer-based instruction, education, evaluation, computer science, engineering and psychology disciplines (Iqbal et al, 1999). Methods 1 to 7 are

categorized under *experimental research*. The remaining methods from 8 to 20 are for *exploratory research* (Iqbal et al, 1999). Based on the evaluation requirements of the C++ STL ITS, a combination of evaluation methods from the table below will be selected.

Table 6.1 Classification Table of Evaluation Methods

NO	EVALUATION METHODS	EXPLORATORY RESEARCH		EXPERIMENTAL RESEARCH	
		INTERNAL	EXTERNAL	INTERNAL	EXTERNAL
1	<i>Proof of Correctness</i>	-	-	✓	-
2	<i>Additive Experimental Design</i>	-	-	✓	-
3	<i>Diagnostic Accuracy</i>	-	-	✓	-
4	<i>Feedback/instruction Quality</i>	-	-	✓	-
5	<i>Sensitivity Analysis</i>	-	-	✓	✓
6	<i>Experimental Research</i>	-	-	-	✓
7	<i>Product Evaluation</i>	-	-	-	✓
8	Expert Knowledge	✓	-	-	-
9	Level of Agreement	✓	-	-	-
10	Wizard of Oz Experiment	✓	-	-	-
11	Performance Metrics	✓	-	-	-
12	Internal Evaluation	✓	-	-	-
13	Criterion-based	✓	✓	-	-
14	Pilot Testing	✓	✓	-	-
15	Certification	-	✓	-	-
16	Outside Assessment	-	✓	-	-
17	Existence Proofs	-	✓	-	-
18	Observation and Qualitative classification	-	✓	-	-
19	Structured Tasks and Quantitative Classification	-	✓	-	-
20	Comparison Studies	-	✓	-	-

In the C++ STL ITS, the whole system and parts of the system are to be evaluated. Therefore, it is classified as both internal and external evaluation. The main aim of the whole system evaluation is to evaluate the effectiveness of the C++ STL ITS as compared to conventional tutoring in the lectures or laboratory. *Experimental research* which enables researchers to examine relationships between teaching intervention and outcomes will be employed. In addition, three key components of the system are to be evaluated: Pre-Test, Tutoring and Post-Test Modules. The purpose is two-fold. Firstly, there is a need to assess their effectiveness and secondly, their suitability. *Observation and qualitative* classification in the exploratory research category is appropriate to observe the pattern and trend in the student's behaviour within the three components. This is discussed in sections 6.3, 6.4 and 6.5. *Expert inspection* will then be used to assess the suitability of the domain knowledge represented in the three components. Section 6.2 provides the detail of the inspection.

The adaptiveness of the ITS to provide individualized learning during the tutorial session needs to be assessed as well. Mark and Greer (1993) suggested that *sensitivity analysis* and *certification* have the potential of evaluating the adaptiveness of ITSs. Sensitivity analysis is an experimental approach suitable for both internal and external evaluations. It examines the behaviour of a component or system in response to various information supplied to it. Such information in the C++ STL ITS includes tutorial completion time, number of hints, number of attempts and the conditional probabilities of understanding. *Certification* which falls under exploratory research is based on techniques to identify competent human teachers. Experts will be approached to appraise the overall accuracy of the ITS, making certification adequate for external evaluations. This is discussed in section 6.4. These

techniques are also adopted to evaluate the Fuzzy Stereotyping of Students Expert System described in section 6.6.

## 6.2 Evaluation of the C++ STL ITS Architecture

The system architecture of the C++ STL ITS contains four main modules as described in Chapter 5. The modules are Domain Knowledge, Teaching Strategies, Student Modelling and Graphical User Interface. Different modules require different evaluation approaches (Mark & Greer, 1993). This section discusses the formative evaluation methods applied and the results recorded. Summative evaluations are documented in the subsequent sections. Formative evaluation occurs during design and early development of the ITS. By contrast, summative evaluation is concerned with the evaluation of the completed ITS and producing formal claims about the ITS (Mark & Greer, 1993).

### 6.2.1 *Domain Knowledge Module*

In the C++ STL ITS, the domain knowledge module contains pre-test questions and descriptions, problem specifications and teaching strategies, post-test questions and descriptions. The accuracy of this component is ensured before the actual learning environment takes place. Formative evaluation using expert inspection was employed to verify the domain. The experts consulted for the evaluation have about 10 years in teaching the C++ programming language.

Initially, the pre-test covered 12 main prerequisite topics – Fundamentals, Selection, Iteration, Array, User Defined Function, Class, Class Member Function, Constructor, Function Template, Class Template, Operator Overloading and Class string. Topics were selected from the elementary programming level that represents the prerequisites of C++ STL. The pre-test evaluation results using the 12 topics were discussed in (Lee & Sapiyan, 2005a). It was discovered that a single Fundamental topic is unable to specifically identify a particular student's understanding. For the second evaluation, the topic Fundamental was decomposed into four other topics – Data, Operator, Expression and Input-Output, increasing the total number of prerequisite topics to 15. This gives a better breakdown of the prerequisite skills to reflect more accurate results on the student's understanding. The total number of questions was subsequently increased from 70 to 76, with a minimum of four questions for each topic. Consequently, the prerequisite topics need to be carefully designed by the tutor to model the elementary programming topics required to learn the C++ STL. The choice of minimum of four questions per topic is to facilitate the algorithm to gradually update the student model discussed in Chapter 4 and Section 6.2.3.

Currently, 11 tutorial questions related to the STL vector are presented. The 15 sub-topics covered are listed in Appendix L. It comprises declaration, population, output, iterator and member function.

The post-test contains 50 multiple choice questions on STL vector, STL list and iterators. Even though STL list is not covered in the tutorial session, its inclusion is to analyze the knowledge transfer of students from STL vector to STL list. The topics tested are based on the acquired skills from the tutorial sessions.

Expert inspection through a questionnaire was carried out to evaluate three aspects of the domain knowledge: understandability, appropriateness and accuracy. Both experts either agree or strongly agree with the following aspects of the pre-test questions and post-test questions :

- i) The questions are understandable
- ii) The questions are relevant prerequisites to C++ Standard Template Library
- iii) The multiple choice answers provided are appropriate
- iv) The number of multiple choice answers is sufficient

The domain knowledge presented in the tests is accurate according to the required standard in a computer programming course.

Students also participated in the evaluation of the understandability of the pre-test and post-test questions. Their responses are depicted in Figures 6.1 and 6.2 below. The majority of them agree that the questions are understandable – 71% in pre-test and 51% in post-test. Students commented that questions on iterators in the post-test are less understandable. Some questions were similar but worded differently. This may have confused some of the students.

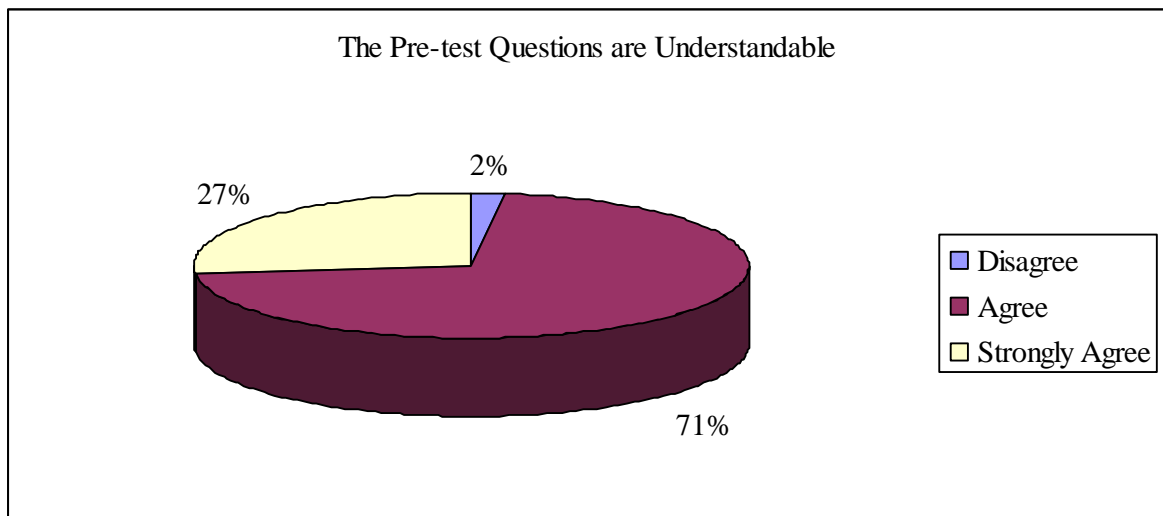


Figure 6.1 Understandability of Pre-Test Questions

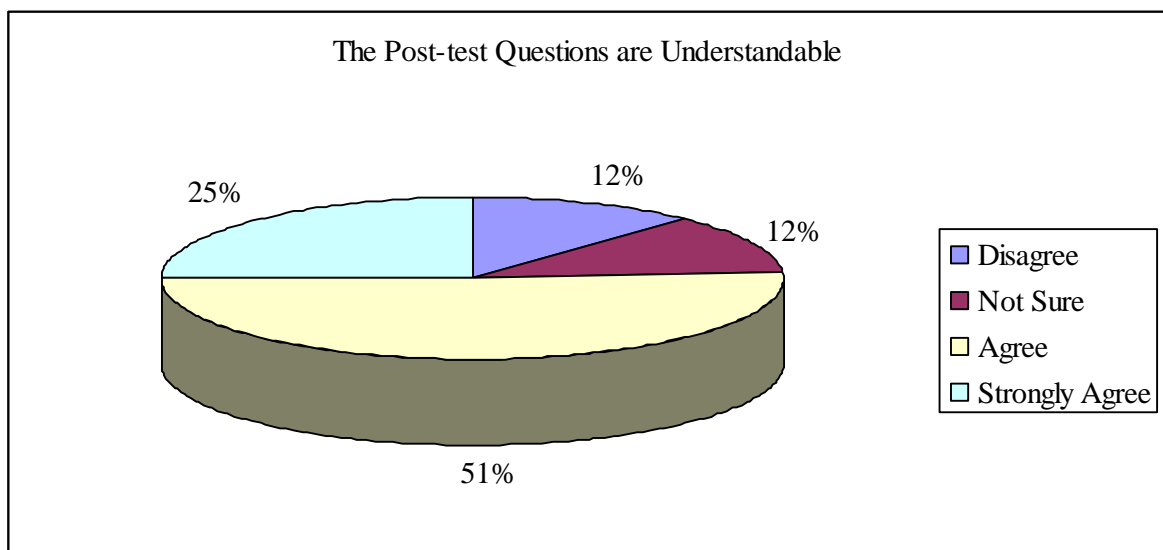


Figure 6.2 Understandability of Post-Test Questions



### 6.2.2 *Teaching Strategies Module*

Four teaching strategies or hints were proposed during the tutorial sessions to guide students – brief description, pre-test review, pre-tutoring and demonstration. According to Mark and Greer (1993), the standards by which teaching knowledge can be assessed are instructional theory and the expert human tutor. In the C++ STL ITS Teaching Strategies Module, the evaluation criteria include the range of the teaching strategies and the degree to which the ITS can adapt its behaviour to individual students. Expert inspection is again employed to evaluate the former criteria. The appraisal of the latter criteria is discussed in Section 6.4.

Experts commented that the level of teaching strategies is appropriate. The level matches the student's understanding. Students with a higher conditional probability of understanding receive less hints than students with a lower conditional probability of understanding. However, one observation is that some students may have the tendency to take the easy path by continuously selecting the hints until the solution is demonstrated to them. It is explained that even though the solution is illustrated to them, they still need to enter the answer before they can proceed to the next tutorial problem. The system does not solve the problem for the students. Students are encouraged to make efforts to enter the solution and solve each sub-tutorial.

### 6.2.3 *Student Modelling Module*

Two important dimensions of the student model are considered for evaluation: validity and reliability. The validity of the student model is measured through the pre-test. In the experimental research, a total of 56 students sat for the pre-test. The test was conducted in the computer laboratory where the C++ STL ITS was installed. On average, students took about one hour to one and a half hour to complete the test. The conditional probabilities of student's understanding in the prerequisites were calculated by the system at the end of the pre-test.

Table 6.2 shows the pre-test performance of various students generated at the end of the pre-test. It lists down the total number of correct and wrong answers for each topic. This gives awareness to the student on the topic that he or she is weak at. Based on the total number of correct and wrong answers, the conditional probability produces a useful result which reflects the student's understanding after the pre-test. This result is accessible to the students during the course of the tutorial session, allowing them to compare their performance before and after the tutorial.

Analysing the results in Table 6.2 further shows that a student needs to get more than half of the answers correct for a topic to obtain a value greater than 0.80. This value indicates the student's competency in a particular topic. Furthermore, a value less than 0.60 shows a weakness in the student's understanding. Based on the total number of correct and wrong answers, the conditional probability produces a useful

result which reflects the student's understanding after the pre-test (Lee & Sapiyan, 2005b).

The total number of questions per topic was taken into consideration in the calculation of the conditional probabilities. A minimum of 4 questions per topic gives a better range of conditional probabilities to demonstrate student's understanding. The distribution of questions does affect the student model but not critical as the topics are independent of each other. The range of conditional probability values produced is based on the number of questions per topic. Therefore, 5 questions will produce 5 different conditional probabilities. A range of 4 to 6 questions per topic enables the threshold to be applied more effectively.

Table 6.2 Sample - Pre-Test performance of various students

STUDENT	Student X			Student Y			Student Z		
TOPIC TITLE	Total Correct	Total Wrong	CP	Total Correct	Total Wrong	CP	Total Correct	Total Wrong	CP
Class String	3	1	0.92	4	0	1	1	3	0.57
Operator Overloading	3	1	0.92	3	1	0.92	2	2	0.80
Class Template	2	2	0.80	0	4	0	1	3	0.57
Function Template	4	0	1	1	3	0.57	2	2	0.80
Constructor	6	0	1	3	3	0.80	2	4	0.67
Class Member Function	2	3	0.73	2	3	0.73	2	3	0.73
Class	4	2	0.86	2	3	0.73	3	2	0.86
User Defined Function	6	1	0.96	5	2	0.91	2	5	0.62
Array	6	0	1	4	2	0.89	3	3	0.80
Iteration	6	0	1	6	0	1	4	2	0.89
Selection	5	0	1	5	0	1	4	1	0.94
Input-Output	5	0	1	4	1	0.94	5	0	1
Expression	5	0	1	4	1	0.94	2	3	0.73
Operator	4	0	1	4	0	1	3	1	0.92
Data	6	0	1	4	2	0.89	5	1	0.95
TOTAL	66	10	14.19	51	25	12.32	41	35	11.85

Table 6.3 summarizes the overall performance for each topic. It is consistent with the fact that students are better at fundamental topics (Data, Operator, Expression, Input-Output) as compared to object-oriented topics and templates. From the average values, a threshold of 0.80 is able to show a student's mastery of a prerequisite concept. A value below 0.60 indicates a student's weakness. Using these thresholds, students were directed to different remedial lessons during the tutorial session.

Table 6.3 also compares the results between the Tutorial and Non-Tutorial students. Tutorial students are students from the BSc (Hons) Software Engineering (SE) programme whereas the Non-Tutorial students are enrolled in the BEng (Hons) Electronics and Computing (EC). From past experience, the Engineering (BEng) students have been consistently better in programming than the Computing (BSc). This is clearly reflected in the table below as the Engineering students performed better in most of the topics tested. One contributive factor is that Engineering students have a stronger foundation in Mathematics. This is a useful analysis when comparing class performances.

Table 6.3 Overall students' Pre-Test performance for each topic

TOPIC	TUTORIAL	NON-TUTORIAL
Class String	0.68	0.79
Operator Overloading	0.76	0.84
Class Template	0.58	0.62
Function Template	0.69	0.81
Constructor	0.72	0.86
Class Member Function	0.82	0.81
Class	0.82	0.86
User Defined Function	0.78	0.86
Array	0.93	0.90
Iteration	0.83	0.92
Selection	0.95	0.98
Input-Output	0.92	0.97
Expression	0.84	0.90
Operator	0.94	0.93
Data	0.95	0.95
<b>CLASS AVERAGE</b>	<b>0.81</b>	<b>0.87</b>

The conditional probabilities obtained from the pre-test reflected the student's understanding in a particular prerequisite. It provided a form of automated self-assessment for the students as it indicates the competency of each topic. It showed that the application of the Bayesian theorem strongly demonstrated the student's prerequisite knowledge. The student model produced by the Bayesian theorem is a valid representation of the student's understanding. This result together with the threshold identified is useful in directing the student during the tutoring session, as the student model will then be updated based on the initial values obtained.

The objective of the validation testing has been satisfied and the results showed to be beneficial for the tutoring session to direct the students intelligently. The reliability of this result is assessed during the summative evaluation of the tutorial sessions.

### 6.2.4 Graphical User Interface Module

The module presents information to the student and acquires responses from the student. A survey was conducted with the students to examine two dimensions of the GUI: interface and learning experience.

Most students agree that the interface is easy to use (see Figure 6.4). The system presents the program specification together with the program skeleton which is divided into sub-problems. Students are required to enter their answers in the Sub Tutorial section as shown in Figure 6.3, and submit the answers or refer to the hints for help.

Gurney, Intelligent Tutoring System (ITS) Ver. BETA 1.0  
Login Time: 2005-09-14 21:09:17.585

ST-001

Home | Pre-Test | Tutorial | Results Logout

Title : STL vector Question 1

Please analyze the tutorial framework below and answer each related sub-tutorial questions.

```
int i;

cout << "Please enter 10 numbers" << endl;


cout << "You have entered" << endl;


return 0;
}
```

Sub Tutorials

Sub Tutorial 1 Sub Tutorial 2 Sub Tutorial 3

Output a vector using subscript  
Output the contents of the vector.

```
for (i = 0; i < 10; i++)
{
    cout << myVector[i];
}
```

Hint ? Submit Answer

Intelligent Tutoring System, ITS ® is a registered trade mark of Christine Lee.  
Copyright © 2004 KBU International College, All rights reserved.

Figure 6.3 Tutorial Framework and Sub-Tutorial Questions

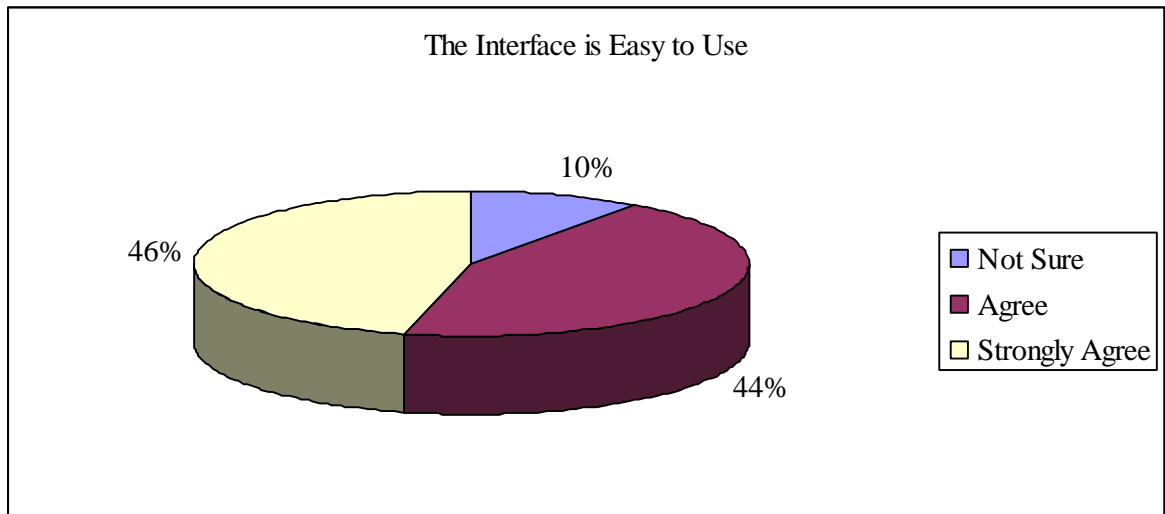


Figure 6.4 Student's Evaluation of the Interface for the Pre-Test Module

The 'Question Content' functionality allows students to view answered and unanswered questions during the pre-test and post-test. It also enables the students to review their answers and change them. From the survey (Figure 6.5), it is apparent that students did not use the functionality much. Most of them did not really bother to check their answers. Hence, they might not see the need to use the function.

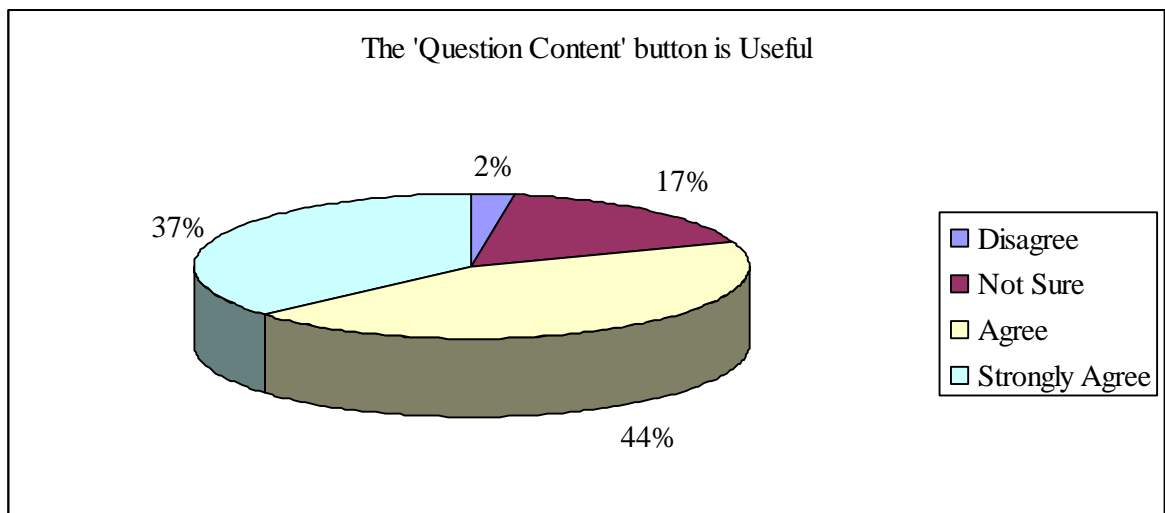


Figure 6.5 Student's Evaluation of the 'Question Content' Functionality

### 6.3 Pre-Test Results and Analyses

This section analyzes the pre-test answer records in detail. The analysis performed is on the common mistakes students made in programming. The multiple choice answers in the pre-test cover the common programming mistakes or misconceptions. The purpose is not to ‘trick’ the students but to identify and confirm the common programming errors. For each question in the pre-test, the percentage of each answer chosen was calculated. In most cases, the highest percentage represented the correct answer chosen. The next highest percentage reveals the common mistake of students. A few examples are highlighted below. The tables show the percentage of each chosen answer for the Tutorial students (TS), Non-Tutorial students (NTS) and both combined. The correct answer is identified with a tick next to it.

Example 1 in Table 6.4 indicates that students often misunderstood between *char* and *string*. *Strings* are denoted by the data type “*char []*” and enclosed within double quotes (“”). Some students very often chose the data type “*char*” and single quotes to represent *strings* or array of characters.



Table 6.4 Common Programming Mistakes Example 1

TOPIC	AVAILABLE ANSWERS	TS	NTS	BOTH
Data Type	char	15.2	13.0	14.3
	✓ char[]	81.8	82.6	82.1
	double	3.0	4.3	3.6
	int	0.0	0.0	0.0
Assignment Expression	start = "a";	21.2	8.7	16.1
	start = a;	9.1	8.7	8.9
	✓ start = 'a';	69.7	82.6	75.0
	start = 'a'	0.0	0.0	0.0
Accessing Members	aDog.setBreed('German Shepherd');	6.1	0.0	3.6
	✓ aDog.setBreed("German Shepherd");	78.8	82.6	80.4
	aDog.setBreed(German Shepherd);	9.1	13.0	10.7
	setBreed.aDog("German Shepherd");	6.1	4.3	5.4

Example 2 is a classic error where students mixed up the operators >> and << for input and output.

Table 6.5 Common Programming Mistakes Example 2

TOPIC	AVAILABLE ANSWERS	TS	NTS	BOTH
Output	cin << "Please enter an integer";	3.0	0.0	1.8
	cin >> "Please enter an integer";	15.2	8.7	12.5
	✓ cout << "Please enter an integer";	78.8	91.3	83.9
	cout >> "Please enter an integer";	3.0	0.0	1.8
Input	cin << "age";	3.0	0.0	1.8
	cin << age;	12.1	4.3	8.9
	✓ cin >> age;	84.8	95.7	89.3
	cin >> age	0.0	0.0	0.0

Example 3 reveals the programming mistakes made for function call. Students are usually weak in specifying the required parameters in calling a function.

Table 6.6 Common Programming Mistakes Example 3

TOPIC	AVAILABLE ANSWERS	TS	NTS	BOTH
Call	cout << calculateArea(int);	15.2	8.7	12.5
	cout << calculateArea();	9.1	0.0	5.4
	✓ cout << calculateArea(radius);	69.7	87.0	76.8
	cout >> calculateArea(radius);	6.1	4.3	5.4
Member Function Call	cout << aDate.getYear(int);	15.2	8.7	12.5
	cout << aDate.getYear(year);	9.1	8.7	8.9
	cout << getYear();	0.0	0.0	0.0
	✓ cout << aDate.getYear();	75.8	82.6	78.6

The errors highlighted in the examples above are not just relevant for this particular cohort but for novice students studying programming in general. It is believed that such analysis is beneficial for tutors to guide the students. The awareness of the common mistakes made is a first step to help students to improve their programming.

Another obvious trend from the result is that as the difficulty of the questions increases, the variance of the selected answers increases. Table 6.7 illustrates this point by comparing the percentage of results for two questions. The question on Data Type is easier than the question on Class Template Member Function, and it is clearly shown in the columns marked **TS**, **NTS** and **BOTH**. With this analysis, tutors can guide the students more effectively by setting tutorial questions for topics that students are struggling in.

Table 6.7 Example – Variance of Percentage in Selection of Answers

TOPIC	AVAILABLE ANSWERS	TS	NTS	BOTH
Data Type	✓ char	100.0	95.7	98.2
	float	0.0	0.0	0.0
	double	0.0	0.0	0.0
	int	0.0	4.3	1.8
Class Template Member Function	template < class T > T Complex<T>::Display()	15.2	13.0	14.3
	template < class T > void Complex<T>::Display;	24.2	26.1	25.0
	Complex<class>::Display();	24.2	8.7	17.9
	✓ template < class T > void Complex<T>::Display()	36.4	52.2	42.9

#### 6.4 Tutorial Sessions Results and Analyses

All the students' actions performed during the tutorial session were logged, and subsequently used to analyze the effectiveness of the teaching strategies and the student model update algorithm.

The log files are summarized into individual Student Tutorial Performance Report. Each report contains tutorial title, sub-tutorial title, tutorial route, correctness and duration to complete the sub-tutorial. From the reports generated, a few analyses were carried out. These include the duration to complete each tutorial, the number of attempts and the number of hints referred. The average duration to complete each tutorial is approximately 30 minutes. The average number of attempts is 7 times. This figure includes those who did not refer to the hints and submitted the incorrect answer. It is also observed that some students referred to the hints each time even before attempting the problem. This has caused their conditional probabilities of understanding to be demoted each time. The demotion may be discouraging to some students but it reflects their actual learning

progression. The updated conditional probabilities is important in directing the students throughout the tutorial.

It is noted that for the student model update algorithm to be effective, there must be sufficient sub-tutorial questions related to the same problem. This is due to the fact that the update is performed based on the number of hints and attempts. As the student solves similar problems, a clearer trend can be observed – whether the student’s conditional probabilities of understanding are being demoted or promoted. In other words, the problem solving support applies a ‘drill and practice’ approach. In the C++ STL ITS, students were required to declare, populate and output a vector in most of the tutorial questions. It is observed that as they progress in the tutorial, their problem solving skills in those topics improved.

Figure 6.6 depicts the prerequisites for the first tutorial question – STL vector Question 1. It shows an example of conditional probabilities,  $P(U/C)$  for each prerequisite achieved by a student after the pre-test. The question is decomposed into three sub-tutorials – declare a vector, populate a vector using subscript and output a vector using subscript. Initially, an average conditional probability is calculated to obtain the type of hints or teaching strategies to be presented to the student. In this case, the average is  $(0.94+0.95+1+0.8+1) / 5$ , giving the value 0.94. Since the value is greater than 0.8, the student will be directed to the brief explanation of the sub-tutorial upon failure to solve the problem. The levels in the figure refer to the four levels of hints available to the students – brief explanation, pre-test review, pre-tutoring and demonstration. Based on the student model update algorithm described in section 5.8.2, the conditional probability of the student will be demoted if the

problem is not solved after two attempts to submit the answer. On completion of the tutorial, students can compare their initial conditional probabilities achieved from the pre-test to the conditional probabilities obtained during the tutorial. Generally, the conditional probabilities showed improvement.

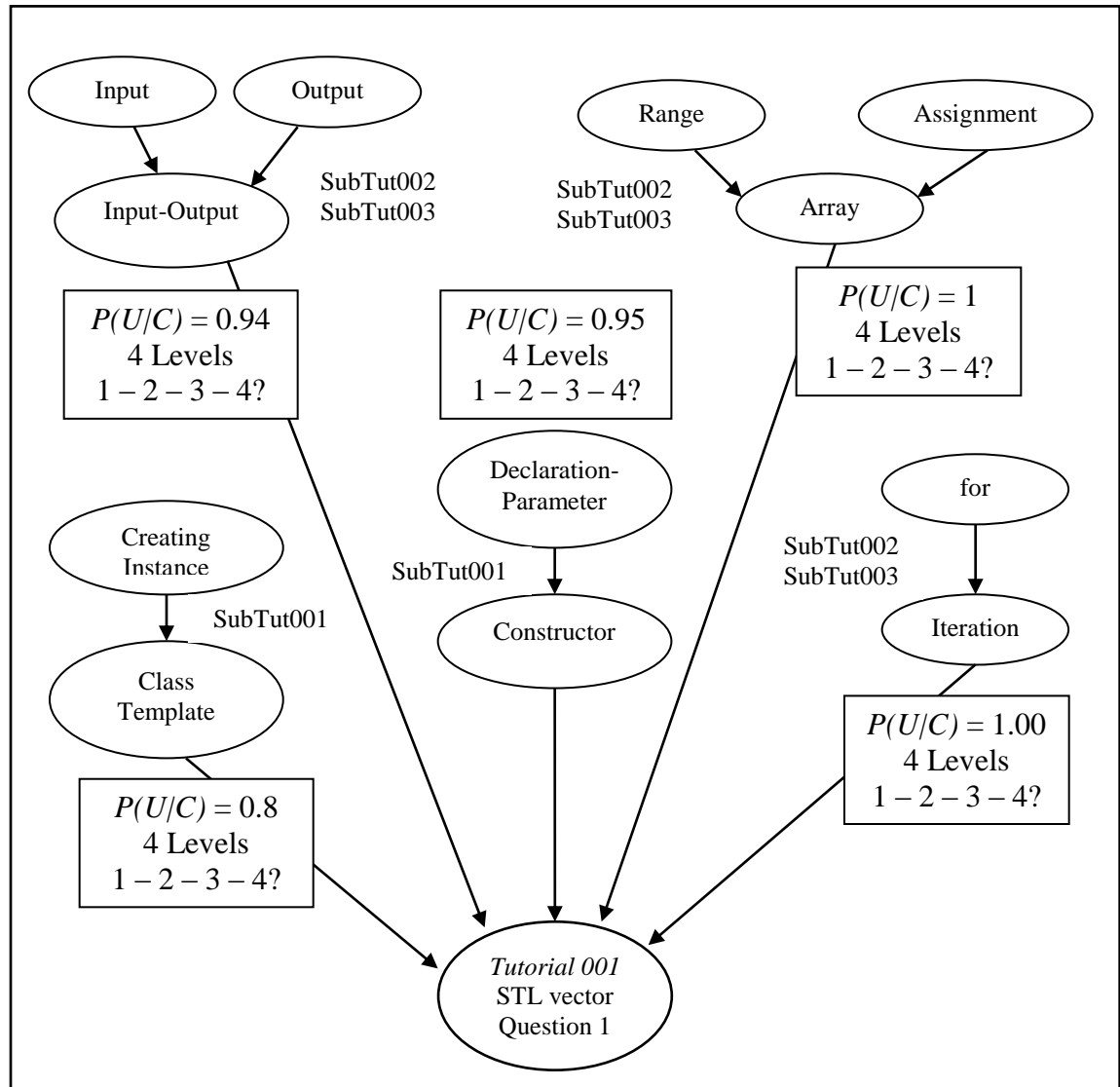


Figure 6.6 Prerequisites for STL vector Question 1

The results of the questionnaire on the tutorial sessions are summarized in Table 6.8 and Table 6.9. Overall, the responses from the students have been positive. The percentage for both *agree* and *strongly agree* is high. Though the feedback is supportive and encouraging, it does not reflect the effectiveness of the tutorial. Nevertheless, it provides a general view of the tutorial sessions and the students' learning experience in C++ STL ITS. The logged files which are the tutorial performance reports and the post-test results are the two methods used to evaluate the effectiveness of the tutorial.

Table 6.8 Survey – Questions on Tutorial Session in C++ STL ITS

NO	QUESTION	RATING (%)				
		Strongly Disagree	Disagree	Not sure	Agree	Strongly Agree
1	The program specifications are understandable.	-	-	5 %	65 %	30 %
2	The hints provided are useful.	-	-	-	45 %	55 %
3	The function to view my current progress is useful.	-	-	15 %	35 %	50 %
4	The interface is easy to use.	-	-	-	60 %	40 %

Table 6.9 Survey - General Questions on Learning Experience with C++ STL ITS

NO	QUESTION	RATING (%)				
		Strongly Disagree	Disagree	Not sure	Agree	Strongly Agree
1	I learned more about C++ STL from the ITS than the lectures.	-	-	20 %	30 %	50 %
2	I learned more about C++ STL from the ITS than the labs.	-	-	20 %	40 %	40 %
3	I would like to use C++ STL ITS more.	-	-	5 %	55 %	40 %
4	I enjoyed the learning experience with C++ STL ITS.	-	-	-	60 %	40 %
5	I would recommend C++ STL to other students.	-	-	5 %	40 %	55 %

## 6.5 Post-Test Results and Analyses

The main advantage of post-tests is to enable quick evaluation of student's knowledge. A total of 50 multiple-choice questions related to the sequence containers *vector* and *list*, and *iterators* were designed. The post-test class average conditional probability results are tabulated in Table 6.10 and illustrated using pie charts in Figure 6.7 and Figure 6.8.

At first glance of the post-test results, the tutorials seem to be ineffective as the difference between the non-tutorial and tutorial results is minimal. Section 6.2.3 discussed that the EC students achieved a higher average than the SE students in the pre-test as the former have a stronger foundation in Mathematics. SE students who did the tutorial on the STL vector performed better than the EC students in the post-test. Though the tutorial did not cover topics on STL list, the SE students still performed better. This shows that they have the ability to transfer their knowledge gained for STL vector and apply it successfully to the STL list. Questions on iterators were not included in the tutorial. Therefore, being the *weaker* group, the SE students performed less satisfactorily.

Table 6.10 Class Average for Post-Test Results

TOPIC	EC : NON-TUTORIAL	SE : TUTORIAL
STL vector	0.87	0.94
STL list	0.85	0.96
Iterator	0.85	0.77

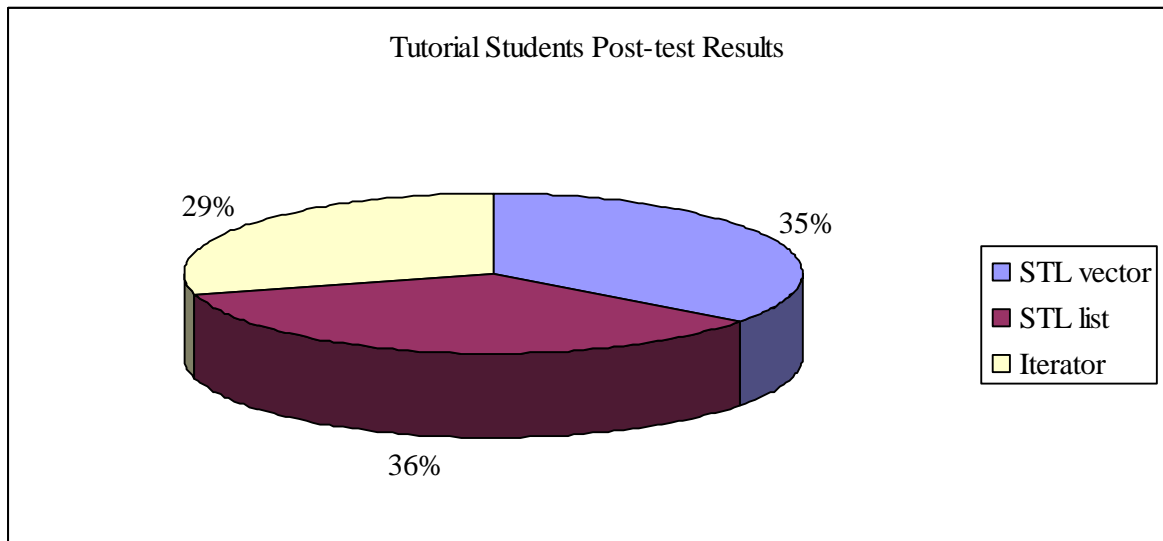


Figure 6.7 Tutorial Students Post-Test Results

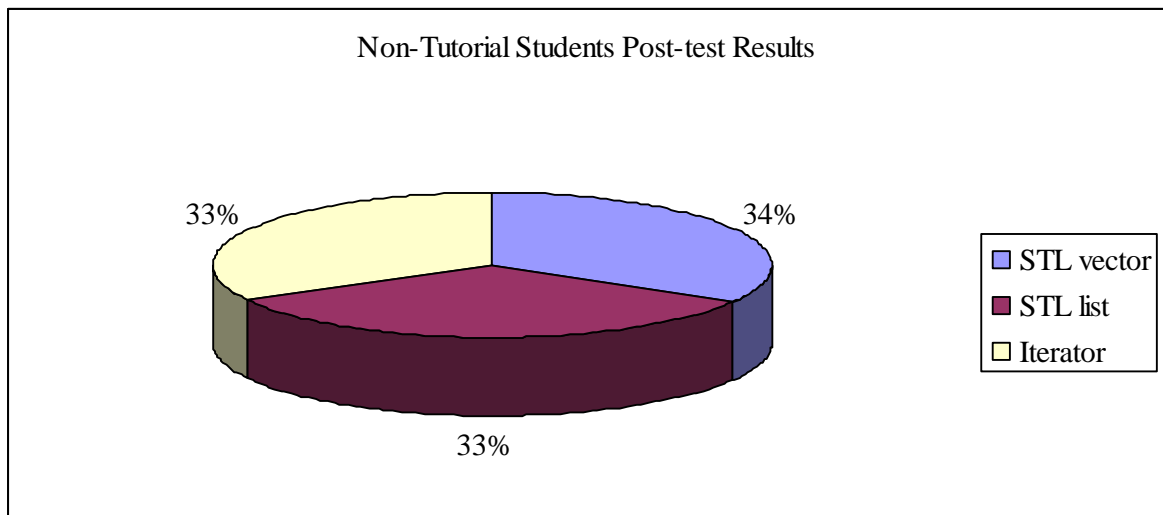


Figure 6.8 Non-Tutorial Students Post-Test Results

The detailed results for each post-test question reveal that the answers selected by the SE students contain less variance. Some examples are illustrated in Table 6.11. For the topic on *vector Member Function*, SE students selected choices with the *end* function, whereas, EC students selected the *last* function as well. The topic on *vector Declaration* showed



more variations. All the answers were selected by EC but SE chose only three answers. These examples indicate that SE students are more confident in answering the questions.

Table 6.11 Partial Post-Test Results (Figures are shown in percentage)

TOPIC	AVAILABLE ANSWERS	CORRECT	EC	SE	BOTH
	<code>fitr = vec.last();</code>		14	0	7
vector	<code>fitr = vec.end();</code>	✓	73	95	83
Member	<code>fitr = vec.end;</code>		14	5	10
Function	<code>fitr = vec.End();</code>		0	0	0
	<code>fitr = vec.End;</code>		0	0	0
	<code>vector v(10);</code>		9	0	5
vector	<code>vector &lt; int &gt; v(10);</code>	✓	32	84	56
Declaration	<code>vector ( int ) v(10);</code>		5	11	7
	<code>vector &lt; int &gt; v[10];</code>		50	5	29
	<code>Vector &lt; int &gt; v(10);</code>		5	0	2
	<code>( vector )</code>		9	5	7
vector	<code>&lt; VECTOR &gt;</code>		5	0	2
Header	<code>&lt; Vector &gt;</code>		0	0	0
File	<code>&lt; vector &gt;</code>	✓	77	95	85
	<code>"vector"</code>		9	0	5

Like the pre-test, common mistakes students made in applying the C++ STL can be identified by taking a closer look at the selection of answers. Tutors can then set additional tutorial questions to address these misconceptions.

In conclusion, further analysis of the post-test results has revealed positive outcome from the tutorial sessions. The problem solving support with various teaching strategies has benefited the students.

## 6.6 Evaluation of the Fuzzy Stereotyping of Students Expert System

The MATLAB Fuzzy Logic Toolbox provides two functionalities to analyse the performance of the Fuzzy Stereotyping of Student Expert System (FSS) – the fuzzy inference viewer (Figure 6.9) and the output surface viewer (Figures 6.10 (a) – 6.10 (c)).

Figure 6.9 shows the output for the following input:

Conditional Probabilities = 0.5            (Low)  
Time = 20.5                                    (Short)  
Attempt = 8                                    (Medium)  
Hint = 4                                        (Low)

The output obtained is 0.736, which falls into the category, *Medium* and subsequently stereotyped as *Intermediate*. The numerical range for the output is specified in Table 5.7 of Chapter 5. The range including the stereotype is summarized in Table 6.12 below. The numerical range is determined based on the list of conditional probabilities in Appendix A.

Table 6.12 Numerical Range for Stereotype

STEREOTYPE	LINGUSTIC VALUE	NUMERICAL RANGE
Novice	Very Low	[0, 0.01]
Beginner	Low	[0.4, 0.6]
Intermediate	Medium	[0.62, 0.8]
Advanced	High	[0.84, 1]

Using the Rule Viewer, the FSS has successfully categorized the students into the required stereotype.

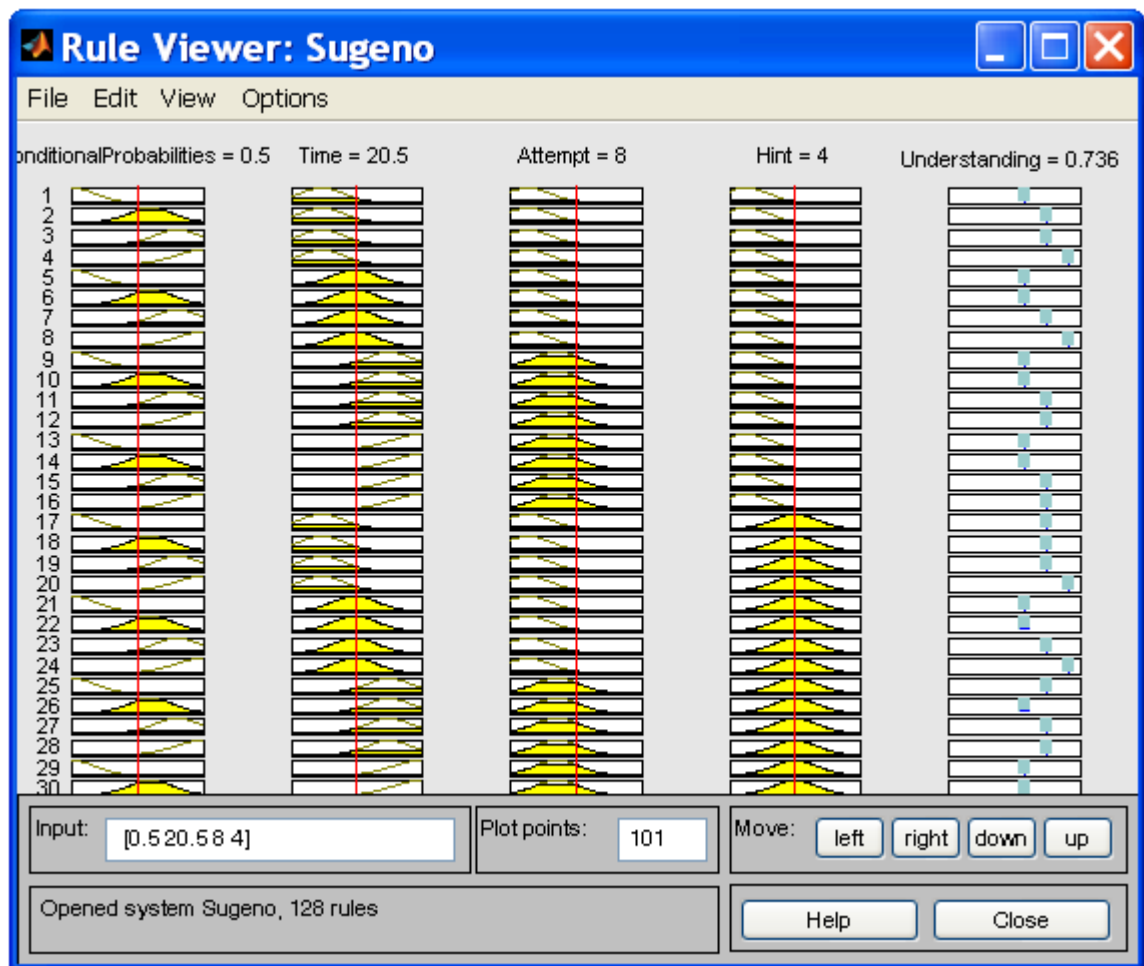


Figure 6.9 Rule Viewer

The Fuzzy Logic Toolbox can generate a three-dimensional output surface by varying any two of the inputs and keeping other inputs constant. In the FSS, three output surfaces are generated. Figure 6.10 (a) depicts the three-dimensional plot for *Understanding-Time-Conditional Probabilities* relationship. As the conditional probability increases and time decreases, understanding grows. Figure 6.10 (b) shows the three-dimensional plot for *Understanding-Attempt-Conditional Probabilities* relationship. Lastly, Figure 6.10 (c) illustrates the three-dimensional plot for *Understanding-Hint-Conditional Probabilities* relationship.

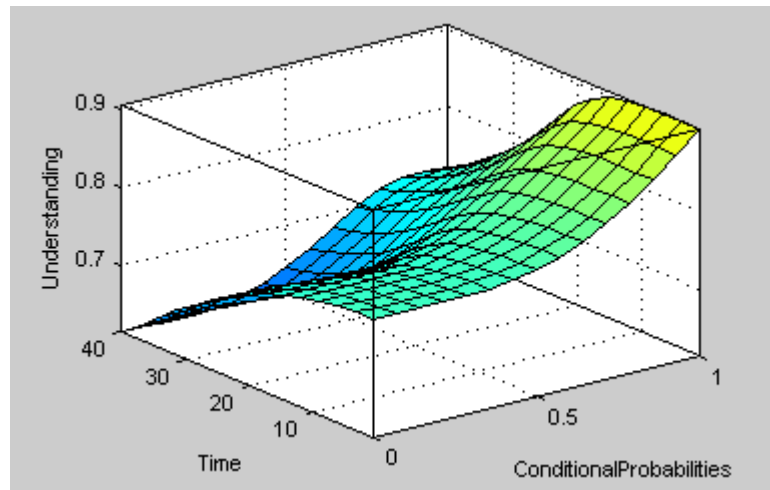


Figure 6.10 (a) Three-dimensional Plot for Understanding-Time-Conditional Probabilities Relationship

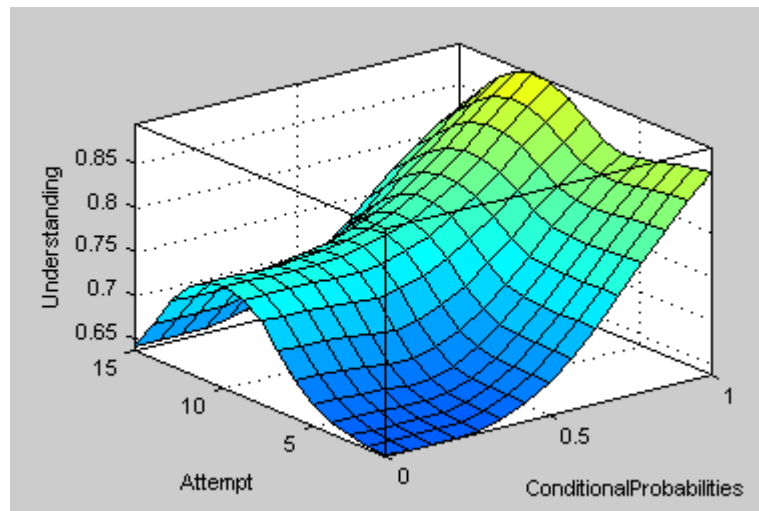


Figure 6.10 (b) Three-dimensional Plot for Understanding-Attempt-Conditional Probabilities Relationship

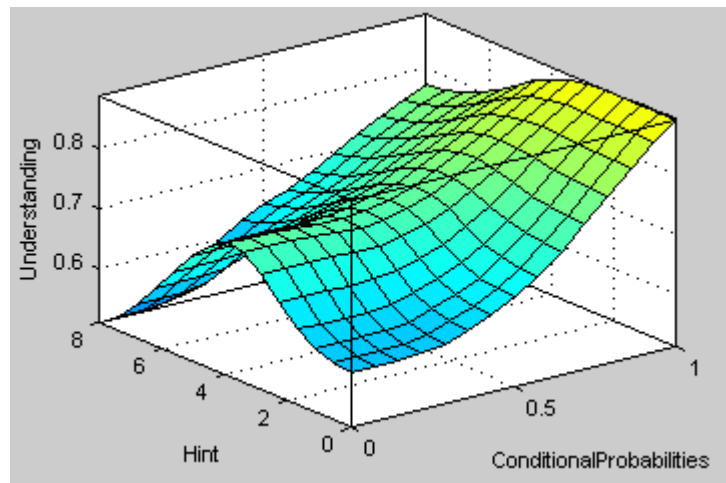


Figure 6.10 (c) Three-dimensional Plot for Understanding-Hint-Conditional Probabilities Relationship

The 128 rules defined in the fuzzy system works well. The linguistic variables and values are sufficient and appropriate to achieve the objective in stereotyping the students. The Sugeno-style inference system is computationally effective and there is no strong reason to try different techniques to tune the FSS.

## 6.7 Related Work

One of the most recent research work that applies the Bayesian Network is BITS (Butz et al, 2006). BITS uses a fixed value of 0.70 to indicate whether a concept is known or otherwise. This value is not only subjective but also does not reflect the actual understanding of the student as tutoring takes place. As students progress in their learning, their performance may become inconsistent. Hence, a variable threshold is more appropriate to direct students' learning. Moreover, the choice of 0.70 is arbitrarily chosen, whereas the threshold in the C++ STL ITS is chosen based on the conditional probabilities generated by the pre-test questions.

BITS distinguishes between two states of a knowledge concept, *known* and *unknown*. Butz et al (2006) noted that this might not be useful in practical applications as student may partially know a concept. In the C++ STL ITS, the conditional probability of understanding for a prerequisite concept is calculated. The direct application of the Bayesian Theorem gives a more realistic model of the student's understanding of the domain knowledge.

Currently, BITS provides only intelligent navigation of course materials. Butz et al (2006) acknowledged that problem solving is an important component of computer programming. They plan to enhance BITS to offer such guidance.

In Chapter 3, the review on ITSs for teaching computer programming has unfolded a few facts. The level of programming taught in existing ITSs such as PROUST (Johnson, 1986), BRIDGE (Bonar & Cunningham, 1988), ASSERT (Baffes & Mooney, 1996), C++ Tutor (Kumar, 2002), JITS (Sykes & Franek, 2003) and BITS (Butz et al, 2006) covers elementary topics that are typically found in an introductory course to Computer Programming. On the other hand, the domain knowledge in C++ STL ITS encompasses a higher level programming.

Most of the current ITSs for programming focuses on teaching students the syntax of a programming language as opposed to application of the programming. The main tutoring approach is to present a problem specification for the student to solve, followed by intelligent analysis of the solution with various feedback. Correcting syntax is not the

focus of C++ STL ITS. The main purpose is to guide students in applying the STL using a variety of teaching strategies.

Unlike the assessment system proposed by Martin and VanLehn (1995), C++ STL ITS does not just assess what a student knows, it also provides problem solving support. Two assessments are included – a pre-test to evaluate a student's prerequisite knowledge and a post-test to examine his/her understanding on completion of the tutorial. The Bayesian theorem is applied in both tests to determine the conditional probabilities of understanding.

Section 5.7 examined various web-based ITSs which employ two-tier or client/server architecture implemented using Java and the Common Lisp Hypermedia Server CL-HTTP completely written in Lisp. The C++ STL ITS adopts a three-tier system architecture which is based on the Java 2 Platform, Enterprise Edition (J2EE) architecture. This 3-tier architecture overcomes some of the limitations of two-tier architectures.

## Chapter 7 Conclusions and Future Work

### 7.1 Conclusions

The web-based C++ Standard Template Library (STL) Intelligent Tutoring System (ITS) provides a complete teaching and learning tool for both tutors and students. For tutors, assessment tools and authoring tools are available to complement their teaching. The domain knowledge can be modified to suit the current curriculum. Tutors can design their own program specifications according to the needs of the students. The ITS also provides self-assessment to students and adaptively helps them in solving C++ STL problems using various teaching strategies. The teaching strategies can form a supplement to existing classroom teaching materials.

The Bayesian theorem is applied to model the student's understanding. The application centers around the prerequisite knowledge of the students. The conditional probabilities produced by Bayesian reveal strengths and weaknesses of the students. The student model is simple and yet effective enough to direct students during the tutorial sessions. The student model update algorithm which is based on the range of conditional probabilities obtained by the students during the pre-test simplifies the computational complexities in Bayesian networks.

The C++ STL ITS has the potential to be extended to other domain knowledge. The authoring tools facilitate the flexibilities of incorporating other programming languages in the ITS. One unique feature is that more than one programming languages can exist at the



same time in the ITS. This enables students to use the same system to learn different computer programming.

The Fuzzy Stereotyping of Students Expert System built works well and has successfully categorized the students. Knowing the ability of the students in the class will enable the tutors to provide better guidance and use different teaching approaches to help the students.

## 7.2 Future Work

Two fields of future work are considered: short term and long term. Short term extensions can be achieved in a shorter time frame. These include extending the tutorial questions, domain knowledge and accommodating more feedback on the programming syntax. One major short term future work is to incorporate the fuzzy expert system into the C++ STL ITS. Three areas of research proposed for long term are application of alternative knowledge acquisition techniques, integration of learning styles into the student model, and representation of domain knowledge using ontologies.

### *7.2.1 Short Term*

The C++ STL ITS will be used in the module Software Design and Implementation 2 for the courses BSc (Hons) Software Engineering and BEng (Hons) Electronics and Computing. The system is currently installed in the Computer Center at KBU International College. Students will be able to use the system during the laboratory sessions. The tutorial questions will be extended to cover higher level topics such as the STL associative containers, adaptive containers and algorithms. Besides

---

topics on STL, learning materials on advanced data structures like trees, deap, hash tables can be incorporated too.

Another extension is to include the domain Java Standard Template Library (JSTL). Prerequisite knowledge required to learn the JSTL can be modelled by the experts who would subsequently design pre-test questions, tutorial questions and post-test questions based on the prerequisites.

Currently, the XML syntax parser does not focus on the syntax of the programming language but emphasizes more on application of certain standard and style of programming. For example, the system insists that students use braces for the begin ('{') and end ('}') block within a *for* loop, and apply certain variable naming conventions. The parser can be extended to provide more specific feedback on the student's misconceptions and check the correctness of the code. More tokens on the programming syntax can be added to improve the parser.

A more significant enhancement is to integrate fuzzy logic into the system to stereotype the students directly after each path in the system, which is the pre-test, tutorial and post-test sessions. It is believed that combining Bayesian and fuzzy logic techniques can further improve the adaptivity of the ITS during the tutorial sessions.

### 7.2.2 Long Term

Currently, the C++ STL ITS models the student according to the knowledge of the prerequisite sub-skills. For future enhancement, it would be beneficial to model other type of student model attributes such as learning styles and learning preferences attributes. From the preferences in interaction, various content presentation can be engaged. Recent learner characteristics and preferences examples are the approaches presented in INSPIRE (Papanikolaou et al, 2003) and the NEMO project (Manouselis and Sampson, 2003).

The aim of INSPIRE is to support a more *learning-focused* model of instruction by providing a sequence of authentic and meaningful tasks that matches students' knowledge level and preferred way of learning. INSPIRE, throughout its interaction with the student, dynamically generates personalized lessons that gradually lead to the attainment of student's learning goals. Furthermore, it supports several levels of adaptation, which range from full system-control to full student-control. The student model is accomplished in multiple ways: curriculum sequencing, adaptive navigation support, adaptive presentation, and supports system's adaptable behavior. It is worth mentioning that Papanikolaou et al (2003) reviewed several aspects in Adaptation in Educational Hypermedia Systems: adaptation source, adaptation technology, adaptivity, teaching theories and teaching approaches.

The NEMO “Non-Excluding Models for Web-based Education” project aims in designing and developing a web-based platform for empowering the education and training methodology of learning communities with special needs, in an adaptive and individualized way. The learner model was modelled and specified using the IMS Learner Information Package (LIP) Specification, and the domain knowledge was also modelled and specified using the IEEE Learning Object Metadata (LOM) specification. Then, a rule-based expert shell, and a multi-criteria evaluation model were presented to recommend e-learning courses most appropriate to the learner’s needs and preferences.

Much could be learned from the work done in INSPIRE and NEMO on modelling various student’s attributes. It is foreseen that the application can improve the adaptivity of the teaching strategies in the C++ STL ITS. It would be interesting to review how standards from the IMS LIP and IEEE LOM specifications can be employed in the C++ STL ITS. Moreover, the advantages of using learning technologies standards promote reusability of information and interoperability with other systems.

In the C++ STL ITS, a multiple-choice single answer pre-test has been used to initialize the student model. The multiple-choice format facilitates the application of the Bayesian Theorem directly. It would be interesting to consider other types of format in the pre-test to acquire the student’s prerequisite knowledge. For example, Fill-in-the-blank and True/False type of questions. To describe a proposal on the

application of Bayesian Theorem for alternative question format, Equation (4.8) explained in Chapter 4 is further considered:

$$P(U | C) = \frac{mp}{1 + (m-1) \times p} \quad (4.8)$$

The value of  $p$  is easily attainable as it is calculated based on the number of correct answers for the prerequisite sub-skills. The value of  $m$  in the equation refers to the total number of multiple choice alternatives. If the question format is True/False type, then  $m$  will be 2, and the  $P(U/C)$  value can be computed for each student. However, to facilitate the Bayesian for Fill-in-the-blank type of questions, the simplest method is to offer multiple options for the blank. The value of  $m$  can vary based on the number of choices provided. Further evaluation can be conducted to confirm the appropriateness of the Bayesian for other types of pre-test question format.

As quoted by Kumar (2003), several researches have proposed using a pre-test to initialize the student model for adaptive learning (Aimeur et al, 2002, Czarkowski and Kay, 2003). He also briefly reviewed various improvements to the pre-test by other researches:

- Adaptive pre-tests to reduce the number of problems the student must solve (Arroyo et al, 2001, Millan et al, 2000)
- Stereotypes to generate a shortened pre-test (Aimeur et al, 2002, Kay, 2000)
- Schema-based assessment (Kalyuga, 2003) to devise tests rapidly wherein the student fills in incomplete intermediate stages in a solution rather than providing the entire solution

The pre-test in the C++ STL has a large set of 76 questions which is necessary to cover the various prerequisite sub-skills. In future work, it is worth researching how the pre-test questions can be minimized and how work by the researches above can be adapted to improve the pre-test.

The domain knowledge in the C++ STL ITS is modelled based on prerequisite sub-skills which is a 2-level hierarchical model. The domain is not structured to provide intelligent adaptive navigation and adaptive sequencing. This was not part of the objective of this research. However, it is beneficial to look into the relationships among the prerequisite concepts and consider how these relationships can enhance adaptive learning.

Karampiperis and Sampson (2004) have addressed the issue of adaptive learning object sequencing problem in intelligent learning management systems proposing a methodology based on the ontologies and learning object metadata. Ontologies are specifications of the conceptualization and corresponding vocabulary use to describe a domain (Karampiperis and Sampson, 2004). They added that ontologies typically consist of definitions of concepts relevant for the domain, their relations, and axioms about these concepts and relationships. In their instructional planning process, they have identified four classes of concept relationships, namely:

- *Consists of*
- *Similar to*
- *Opposite of*
- *Related with*

The result of their research is a generic Instructional planner capable of providing both Adaptive and Dynamic Courseware Generation. The main contribution of this method is that it is fully automatic and can be applied independently of the knowledge domain.

For future work, the hypothesis is that in-depth study in identifying relationships among the prerequisite concepts will enable intelligent sequencing of the problems. Subsequently, problem specifications can be organized according to the level of difficulty. An example of a relationship is between the user defined function and class member function. It can be described with the relationship *related with*, which means class member function is related with user defined function. Therefore, the program specification can be designed to test that students understand the relation that class member functions are user defined functions that exist within a *class* and can be either *public* or *private*.

In conclusion, the long term future work proposed focuses on a wider development scope that can be taken by the ITS community. The C++ STL ITS has achieved the three main tasks of an ITS (Shute and Psotka, 1996):

- It accurately diagnoses the student's knowledge using the Bayesian Theorem
- Based on the student's knowledge, it provides adaptive teaching strategies
- The system provides feedback through the various teaching strategies

## References

Aimeur, E., Brassard, G., Dufort, H., and Gambs, S. (2002). CLARISSE: A Machine Learning Tool to Initialize Student Models. S. Cerri, G. Gouarderes, F. Paraguacu (eds.), In *Proceedings of ITS 2002*, Springer (2002). pp 718-728.

Albacete P.L., VanLehn, K. (2000). The conceptual helper: An intelligent tutoring system for teaching fundamental physics concepts. In G. Gauthier, C. Frasson & K. VanLehn (Eds), In *Proceedings of 5th International Conference, Intelligent Tutoring Systems*, Berlin: Springer (Lecture Notes in Computer Science, Vol. 1839), Montreal, Canada, pp 564-573.

Alpert, Sherman R., Singley, Mark K. and Fairweather, Peter G. (1999). "Deploying Intelligent Tutors on the Web: An Architecture and an Example" *International Journal of Artificial Intelligence in Education*, 10(2), pp 183-197.

Alur, D., Crupi, P. and Malks, D. (2003). *Core J2EE Patterns, Best Practices and Design Strategies*, Second Edition, Sun Microsystems, Prentice Hall.

Ainsworth, Shaaron. (2005). Evaluation Methods for Learning Environments : A Tutorial for *AIED 2005 – T1*, Amsterdam.

[URL:

[http://www.psychology.nottingham.ac.uk/staff/Shaaaron.Ainsworth/aied\\_tutorialslides2005.pdf](http://www.psychology.nottingham.ac.uk/staff/Shaaaron.Ainsworth/aied_tutorialslides2005.pdf)]

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R. (1988). The Expert Module. In Polson M.C. and Richardson J.J. (Eds.), *Handbook of Intelligent Training Systems*. Hillsdale, NJ: Erlbaum, pp 21-53.

Anderson, J. R. (1993). *Production Systems and the ACT-R Theory. Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum & Associates, Inc., pp 1-10.

Anderson, J. R., Boyle, C. F., and Yost, G. (1986). The geometry tutor. *The Journal of Mathematical Behavior*, pp 5-20.



Anderson, J. R., Corbett, A. T., Koedinger, K. and Pelletier, R. (1995). Cognitive Tutors: Lessons Learned. In *The Journal of the Learning Science*, 4(2), Lawrence Erlbaum & Associates, Inc., pp 167-207.

Anderson, J. R. and Reiser, B. J. (1985). *The LISP tutor*. In: BYTE, Volume 10, No. 4, S. pp 159-175.

Arroyo, I., Beck, Joseph. E, Beal, Carole. R. and Woolf, Beverly. P. (2003). Learning within the ZPD with the AnimalWatch Intelligent Tutoring System, *AERA 2003*, Chicago.

Arroyo, I., Conejo, R., Guzman, E. and Woolf, B. P. (2001). An Adaptive Web-Based Component for Cognitive Ability Estimation., In *Proceedings of International Conference on Artificial Intelligence in Education (AIED 2001)*, IOS Press. pp 456-466.

Arvanitis, T. H., Todd, M. J., Gibb, A. J. and Orihashi, E. (2001). Understanding Students' Problem-Solving Performance in the Context of Programming-In-The-Small: An Ethnographic Field Study. Paper presented at the *31th ASEE/IEEE Frontiers in Education Conference*, Session F1D, Reno, NV, 10–13 October.

Astels, D., Miller, G. and Novak, M. (2002). *A Practical Guide to eXtreme Programming*, The Coad Series, Peter Coad, Series Editor, Upper Saddle River, NJ, USA, Prentice Hall.

Austern, Matthew H. (1999) *Generic Programming and the STL. Using and extending the C++ Standard Template Library*. Professional Computing Series. Addison-Wesley, USA.

Austern, Matthew H. (2000). *C++ Report : The Standard Librarian : Algorithms and Containers*, In: ADTmag.com.

Baffes, P. (1994). Automatic student modelling and bug library construction using theory refinement. Ph.D. Dissertation, Department of Computer Sciences, The University of Texas at Austin.  
[URL: <http://net.cs.utexas.edu/users/ml/>]

Baffes, P. and Mooney, R. (1993). Symbolic revision of theories with M-of-N rules. In *Proceedings of the Thirteenth International Joint Conference on Artificial intelligence*, pp 1135-1140. Chambery, France.

Baffes, P. and Mooney, R. (1996). A Novel Application of Theory Refinement to Student Modeling, *In the Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*.

Beck, J. E. and Woolf, B. P. (1998). Using a learning agent with a student model. In *Proceedings of Intelligent Tutoring Systems*, pp 6-15.  
[URL: <http://citeseer.ist.psu.edu/beck98using.pdf>]

Beck, Joseph E. and Stern, Mia K. (1999). Bringing back the AI to AI & ED, *9th World Conference of the AIED Society*.

Beck, J., Stern, M., and Haugsjaa, E. (1996). Applications of AI in Education, ACM Crossroads, September.

Beck, J., Stern, M. and Woolf, B. P. (1997). Cooperative Student Models, In B. du Boulay & R. Mizoguchi (eds), *Artificial Intelligence in Education*, IOS Press, Amsterdam, pp 127-134.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher* 13(6), pp 4-16.

Bonar, J.G, and Cunningham, R. (1988). Bridge: Tutoring the Programming Process. In *Intelligent Tutoring Systems: Lessons Learned*, J. Psotka, L.D. Massey, and S A. Mutter (Eds.), Lawrence Erlbaum Associates, Hillsdale, New Jersey, pp 409-434.

Brown J. S. and Burton R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, vol. 2, pp. 155-92.

Brusilovsky, P. (1999). Adaptive and Intelligent Technologies for Web-based Education, In C. Rollinger and C. Peylo (eds.), *Special Issue on Intelligent Systems and Teleteaching*, Künstliche Intelligenz, 4, pp 19-25.

Brusilovsky, P., Eklund, J. and Schwarz, E. (1998). Web-based Education for All: A Tool for Development Adaptive Courseware. *Computer Networks and ISDN Systems*. Proceedings of Seventh International World Wide Web Conference, 14-18 April 1998 30 (1-7), pp 291-300.

Brusilovsky, P., Schwarz, E., and Weber, G. (1996). *ELM-ART: An Intelligent Tutoring System on World Wide Web*, Frasson, C., Gauthier, G., Lesgold, A. (Eds). In Proceedings of the Third International Conference on Intelligent Tutoring Systems, *ITS'96*, Berlin: Springer, pp 261-269.

Budd, Timothy (1998). *Data Structures in C++ using Standard Template Library*. Addison Wesley Longman, Inc., USA.

Bull, Susan (1997). See Yourself Write: A Simple Student Model to Make Students Think. In Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.), *User Modelling: Proceedings of the Sixth International Conference, UM97*. Vienna, New York: Springer Wien New York. © CISM.

Burton, R. R., Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In: Sleeman, D., Brown, J. (eds.): *Intelligent Tutoring Systems*, Chap. 4, pp. 79-98, London: Academic Press.

Butz, C. J., S. Hua, S. and Maguire, R.B. (2004). "A Web-based Intelligent Tutoring System for Computer Programming", *IEEE/WIC/ACM Conference on Web Intelligence (WI04)*, pp 159-165.

Butz, C. J., S. Hua, S. and Maguire, R.B. (2006). A Web-based Bayesian Intelligent Tutoring System for Computer Programming, *Web Intelligence and Agent Systems: An International Journal*, to appear Vol. 4, No. 1.

Callear, D. (1997). A Course-oriented Approach to Intelligent CAL Based on the Teacher. [URL: <http://www.media.uwe.ac.uk/masoud/cal-97/posters/callear.htm>]  
Last accessed date: 21 November 2005.

Carbonell, J. R. (1970). AI in CAI: an artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11(4), pp 190-202.

Carr, B. and Goldstein, I. (1977). Overlays: a theory of modelling for computer-aided instruction. Technical Report A. I. Memo 406, Cambridge, MA: MIT.

Chad Lane, H. and VanLehn, K. (2003). Coached Program Planning: Dialogue-Based Support for Novice Program Design. In *Proceedings of the SIGCSE '03*, February 19-23, 2003 Reno, Nevada, USA.

Clancey, William J. (1992). Guidon-Manager Revisited: A Socio-Technical Systems Approach. In Claude Frasson, Gilles Gauthier, Gordon I. McCalla (Eds.): *Intelligent Tutoring Systems, Second International Conference, ITS '92*, Montréal, Canada, June 10-12, 1992, Proceedings. Lecture Notes in Computer Science 608 Springer, pp 21-36.

Crowley, R. S., Medvedeva, O. and Jukic, D. (2003). SlideTutor – A model-tracing Intelligent Tutoring System for teaching microscopic diagnosis. U. Hoppe, F. Verdejo and J. Kay (eds.), In *Proceedings of the 11th International Conference on Artificial Intelligence in Education* (AIED 2003), July 20 – 24, Sydney, Australia, IOS Press.

Czarkowski, M. and Kay, J. (2003). Challenges of Scrutable Adaptivity. U. Hoppe, F. Verdejo and J. Kay (eds.), In *Proceedings of the 11th International Conference on Artificial Intelligence in Education* (AIED 2003), July 20 – 24, Sydney, Australia, IOS Press. pp 404-406.

de Buen, P.R., Vadera, S. and Morales, E.F. (1999). A Collaborative Approach to User Modeling within a Multi-Functional Architecture, in J. Kay (ed), UM99: In *User Modelling, Proceedings of the Seventh International Conference*, Springer Wien New York, pp 291-293.

[URL: [http://www.cs.usask.ca/UM99/Proc/short/BuenRodriguez\\_032414.pdf](http://www.cs.usask.ca/UM99/Proc/short/BuenRodriguez_032414.pdf)]

Deitel, H. M. and Deitel, P. J. (2003). *C++ How to Program*, Fourth Edition, Pearson Education Inc., Prentice Hall.

Dimitrova, V., Self, J. A. and Brna, P. (2000). *Involving the Learner in Diagnosis – Potentials and Problems*. In Web Information Technologies: Research, Education, Commerce (WITREC 2000), May 2-5 2000, Montpellier, France.

Fernandez, K. and Sison, R. (2001). A Probabilistic Student Model in Novice Programming. In *Proceedings of the 9th International Conference on Computers in Education*, pp 242-249.

Gamboa, Hugo and Fred, Ana. (2001). Designing Intelligent Tutoring Systems: a Bayesian Approach. In *3rd International Conference on Enterprise Information Systems, ICEIS'2001*.

Gray, W. D and Atwood, M. E. (1992). Transfer, Adaptation, and Use of Intelligent Tutoring Technology: The Case of Grace. In M. Farr and J. Psotka (Eds.), *Intelligent Instruction by Computer: Theory and Practice*, New York: Taylor and Francis, pp 179-203.

Hartley, J. and Sleeman, D. (1973). Towards more intelligent teaching systems. *International Journal of Man-Machine Studies* 2, pp 215-236.

Heckerman, D. (1996). A Tutorial on Learning with Bayesian Networks. In *Learning in Graphical Models*, M. Jordan, ed.. MIT Press, Cambridge, MA, 1999. Also appears as Technical Report MSR-TR-95-06, Microsoft Research, March, 1995. An earlier version appears as Bayesian Networks for Data Mining, *Data Mining and Knowledge Discovery*, 1, pp 79-119.

Heffernan, N. T. (2001). Intelligent Tutoring Systems have Forgotten the Tutor: Adding a Cognitive Model of an Experienced Human Tutor. Dissertation. Carnegie Mellon University, Computer Science Department.  
[URL: <http://gs260.sp.cs.cmu.edu/diss>]

Hopgood, Adrian A. (2001). *Intelligent Systems for Engineers and Scientists*, Second Edition, CRC Press LLC.

Ito, J., Okazaki, Y., Watanabe, K., Kondo, H., and Okamoto, M. (1998). "Pen based user interface for an ITS on WWW client". In *Proceedings of ICCE'98*, Beijing, China, AACE (1998) pp 324-327.

Iqbal A., Oppermann R., Patel A. and Kinshuk (1999). A Classification of Evaluation Methods for Intelligent Tutoring Systems. *Software Ergonomie '99 - Design von Informationswelten* (Eds. U. Arend, E. Eberleh & K. Pitschke), B. G. Teubner Stuttgart, Leipzig, pp 169-181.

Jameson, A. (1995). Numerical uncertainty management in user and student modelling: an overview of systems and issues, *User Modelling and User-Adapted Interaction*, 5(3- 4) (1995) 193-251.

J2EE BluePrints (2001), Sun Microsystems, Inc.

[URL:

[http://java.sun.com/blueprints/guidelines/designing\\_enterprise\\_applications/web\\_tier/dynamic\\_content/](http://java.sun.com/blueprints/guidelines/designing_enterprise_applications/web_tier/dynamic_content/)]

Last accessed date: 21 November 2005.

Jameson, A. (1998). "What Can the Rest of Us Learn from Research on Adaptive Hypermedia – and Vice-Versa?", Comments on the book "Adaptive Hypertext and

Hypermedia". Peter Brusilovsky, Alfred Kobsa, and Julita Vassileva (Eds), Dordrecht: Kluwer.

[URL: <http://w5.cs.uni-sb.de/~jameson/ahh/ahh-comments.html>]

Johnson, W. L. (1986). *Intention-based diagnosis of errors in novice programs*. Palo Alto, CA: Morgan Kaufman.

Kalyuga, S. (2003). Rapid Assessment of Learner's Knowledge in Adaptive Learning Environments, U. Hoppe, F. Verdejo and J. Kay (eds.), In *Proceedings of the 11th International Conference on Artificial Intelligence in Education (AIED 2003)*, July 20 – 24, Sydney, Australia, IOS Press. pp 167-174.

Karampiperis P. and Sampson D. (2004). Adaptive Instructional Planning using Ontologies, In *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies (ICALT 04)*, ISBN: 0769521819, Joensuu, Finland, IEEE Computer Society, August 2004, pp 126-130.

Kassim, A. A., Ahmed, K. S. and Ranganath, S. (2001). A Web-based Intelligent Approach to Tutoring. Paper presented at the *International Conference on Engineering Education*, Session 8B4, Oslo, Norway, 6-10, August.

Katz, S., Lesgold, A. Eggan, G. and Gordin, M. (1992). Modelling the student in Sherlock II, *Journal of Artificial Intelligence in Education* Vol 3 No 4 pp 495.

Kavčič, Alenka. (2000). "The Role of User Models in Adaptive Hypermedia Systems", In *Proceedings of the 10th Mediterranean Electrotechnical Conference MEleCon 2000*, Lemesos, Cyprus, May.

Kay, J. (2000). Stereotypes, Student Models and Scrutability. In *Proceedings of ITS 2000*. G. Gauthier, C. Frasson and K. VanLehn (eds.). Springer. pp 19-30.

Kelly, J., Ghent, J., Bergin, S., Gaughran, P. and Mooney, A. (2004). "Initial Findings on the Impact of an Alternative Approach to Problem Based Learning in Computer Science", PBL International Conference, Mexico, June.

Kinshuk (2002). Does intelligent tutoring have future! In Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson & C.-H. Lee (Eds.) *Proceedings of the International Conference on Computers in Education*, Los Alamitos, CA: IEEE Computer Society.

Kinshuk, Patel A., Oppermann R., Russell D. (2001). Role of Human Teacher in Web-based Intelligent Tutoring Systems. *Journal of Distance Learning*, 6 (1), pp 26-35.

Kumar, A. N. (2002). Model-based Reasoning for Domain Modelling, Explanation Generation and Animation in an ITS to help Students Learn C++. In *Proceedings of the 6<sup>th</sup> International Conference, ITS 2002*, Cerri, S. A., Gouarderes, G. & Paraguacu, F., Eds., Biarritz, France and San Sebastian, Spain, June 2-7, Springer-Verlag Berlin Heidelberg New York.

Kumar, A. N. (2003). Rule-based Adaptive Problem Generation in Programming Tutors and its Evaluation. U. Hoppe, F. Verdejo and J. Kay (eds.), In *Proceedings of the 11th International Conference on Artificial Intelligence in Education (AIED 2003)*, July 20 – 24, Sydney, Australia, IOS Press.

Lajoie, S. P. and Derry, S. J., eds (1993). *Computer as cognitive tools*. Hillsdale, NJ: Erlbaum.

Lauritzen, S. L. and Spiegelhalter, D.J. (1988). “Local computations with probabilities on graphical structures and their application to expert systems”, *J. Royal Stat. Soc.* 50 (1988) pp 172 – 194.

Lee, Christine and Sapiyan, M. (2005). “Web-based C++ Standard Template Library Intelligent Tutoring System”, *18<sup>th</sup> Education Technology Conference*, Kuala Terengganu, 16-19 September 2005.

Lee, Christine and Sapiyan, M. (2005). “Bayesian-based Intelligent Tutoring System for teaching C++ STL”, *The Journal of Technology Management and Entrepreneurship (JTME)*, Institute of Technology Management and Entrepreneurship, November.

López, J. M., Millán, E., Pérez-de-la-Cruz, J. L., & Triguero, F. (1998). ILESA: a Web-based Intelligent Learning Environment for the Simplex Algorithm. In Alvegård, C. (ed.) *Proceedings of CALISCE'98*, 4th International conference on Computer Aided Learning and Instruction in Science and Engineering, Göteborg, Sweden, pp 399-406.

Luger, G. F. (2005). *Artificial Intelligence, Structures and Strategies for Complex Problem Solving*, 5<sup>th</sup> Edition, Addison Wesley, USA.

Madigan, David, Hunt, Earl, Levidow, Bjorn and Donnell, Deborah (1995). Bayesian Graphical Modeling for Intelligent Tutoring Systems, Technical Report.  
[URL: <http://citeseer.ist.psu.edu/article/madigan94bayesian.html>]  
Last accessed date: 21 November 2005.

Mallery, J.C. (1994). A Common Lisp Hypermedia Server, In *Proceedings of the 1st International WWW Conference*, Geneva.

Mamdani, E. H. and Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller, *International Journal Man-Machine Studies*, 7(1), pp 1-13.

Manouselis N. and Sampson D. (2003). "Agent-based e-Learning Course Discovery and Recommendation: Matching Learner Characteristics with Content Attributes", *International Journal of Computers and Applications (IJCA)*, Special Issue on Intelligence and Technology in Educational Applications, Volume 25, No. 1.

Mark, M. A. and Greer, J. E. (1993). Evaluation methodologies for intelligent tutoring systems. *Journal of Artificial Intelligence in Education* (Special Issue on Evaluation), 1993, 4 (2/3), pp 129-153.

Martin, J. and VanLehn, K. (1995). Student assessment using Bayesian nets. *International Journal of Human-Computer Studies*, 42, pp 575-591.

Matena, Vlada and Stearns, Beth. (1991). *Applying Enterprise JavaBeans: Component-Based Development for the J2EE Platform*. The Java Series Enterprise Edition, Sun Microsystems, Inc.

Mayo, M. and Mitrovic, A. (2000). Using a probabilistic student model to control problem difficulty. In *Proceedings of ITS 2000 Conference*, pp 524-533.

Mayo, M., Mitrovic, A. and McKenzie, J. (2000). CAPIT: an Intelligent Tutoring System for Capitalization and Punctuation. *International Workshop for Advanced Learning Technologies IWALT2000*, December 4-6, 2000, Palmerston North, pp 151-154.

McCracken, M., Almstrum, V., Diaz, D., Guzdia, M., Hagan, D., Kolikant, Y.B-D., Laxer, C., Thomas, L. and Utting, I. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin* 33 (4), pp 125-180.



Millan, E., Perez-de-la-Cruz, J.L., and Svazer, E. (2000). Adaptive Bayesian Networks for Multilevel Student Modeling. In *Proceedings of ITS 2000*. Springer, pp 534-543.

Mitchell, Tom. (1997). *Machine Learning*. International Edition, McGraw Hill Publications.

Mitrovic, A. (1996). SINT - a Symbolic Integration Tutor. Frasson, C., Gauthier, G., Lesgold, A. (Eds). In *Proceedings of ITS'96 conference, Montreal*, June 1996, Lecture Notes in Computer Science, Springer, pp 587-595.

Mitrovic, A. and Hausler, Kurt. (2003). "An Intelligent SQL Tutor on the Web", *International Journal of Artificial Intelligence in Education*, Vol. 13, Numbers 2-4, pp 173-197.

Mitrovic A. and Ohlsson S. (1999). Evaluation of a Constraint-Based Tutor for a Database Language. *Int. J. on A.I. in Education*, 10(3-4), 1999, pp 238-256.

Musser, David. (2003). Generic Programming.  
[URL: <http://www.cs.rpi.edu/~musser/gp/>]  
Last accessed date: 21 November 2005.

Nakabayashi, K, Maruyama, M., Koike, Y., Kato, Y., Touhei, H. and Fukuhara, Y. (1997). Architecture of an Intelligent Tutoring System on the WWW. In *Proceedings of the 8<sup>th</sup> World Conference of the AIED Society*, Kobe, Japan, 18-22, August.

Negnevitsky, Michael (2005). *Artificial Intelligence, A Guide to Intelligent Systems*, Second Edition, Addison-Wesley.

ObjectSpace Inc. (1996), "Standard Template Library".  
[URL: <http://wwwasd.web.cern.ch/wwwasd/lhc++/ObjectSpace/doc/2.1/stdusr/intro.1.html>]  
Last accessed date: 21 November 2005.

Ohlsson, S. (1987). *Some Principles of Intelligent Tutoring*. In Lawler & Yazdani (Eds.), *Artificial Intelligence and Education*, Volume 1. Ablex: Norwood, NJ, pp 203-238.

Ohlsson, S. (1994). Constraint-Based Student Modelling. *Student Modelling: The Key to Individualized Knowledge-Based Instruction*. pp 167-189, Springer-Verlag.

Ourston, D. and Mooney, R. (1994). Theory refinement combining analytical and empirical methods. *Artificial Intelligence*. 66, pp 311-394.

Papanikolaou K.A., Grigoriadou M., Kornilakis H., and Magoulas G.D. (2003). "*Personalising the Interaction in a Web-based Educational Hypermedia System: the case of INSPIRE*", *User-Modeling and User-Adapted Interaction*, 13 (3), pp 213-267.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann.

Pérez, T., Lopistéguy, P., Gutiérrez, J., and Usandizaga, I. (1995). HyperTutor: From hypermedia to intelligent adaptive hypermedia. In H. Maurer (Eds.), *Educational Multimedia and Hypermedia, Proceedings of ED-MEDIA'95, World conference on educational multimedia and hypermedia*, June 17-21, 1995. Graz, Austria, AACE. – pp 529-534.

Peylo, C., Thelen, T., Rollinger, C., and Gust, H. (2000). A Web-based intelligent educational system for PROLOG. In Peylo, C. (Ed.), *Proceedings of the International Workshop on Adaptive and Intelligent Web-based Educational Systems* (held in Conjunction with ITS 2000), Technical Report of the Institute for Semantic Information Processing, Osnabrück, pp 85-96.

Polson, M. C. & Richardson, J.J., eds. (1988). *Foundations of intelligent tutoring systems*. Hillsdale, NJ: Erlbaum

Prentzas, J., Hatzilygeroudis, I. and Garofalakis, J. (2002). A Web-based Intelligent Tutoring System Using Hybrid Rules as its Representational Basis, Cerri, S.A., Gourederes, G. and Paraguacu, F. (Eds): ITS 2002, LCNS 2363, Springer-Verlag Berlin Heidelberg, In *Proceedings of the ITS 2002 Conference*, Biarritz, France, June, pp 119-128.

Prentzas, J, Hatzilygeroudis I and Koutsojannis C. (2001). *A Web-based ITS Controlled by a Hybrid Expert System*, *Proceedings of the IEEE ICALT-2001*, Madison, Wisconsin, USA, pp 239-240.

Psotka, J. Massey, L.D. and Mutter, S. A. (1988). *Intelligent tutoring systems: lessons learned*. Hillsdale, NJ: Erlbaum.

Pugh, E. and Gradecki, J. D. (2004). *Professional Hibernate*, Wrox, Wiley Publishing, Inc., Indianapolis, Indiana.

Reiser, B., Anderson, J., and Farell, R. (1985). Dynamic student modelling in an intelligent tutor for LISP programming. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp 8-14.

Reyes, Rhodora L., Galvey, Carlo, Gocolay, Ma. Christine, Ordon, Eden and Ruiz, Conrado Jr.(2000). Multimedia Intelligent Tutoring System For Context-Free Grammar. In *Proceedings of the Philippine Computing Science Congress (PCSC) 2000*.

Ritter, S., Anderson, J. R., Cytrynowicz, M., and Medvedeva, O. (1998) Authoring Content in the PAT Algebra Tutor. *Journal of Interactive Media in Education*, 98 (9).

Ross, M. Sheldon. (2003). *Introduction to Probability Models*, 8<sup>th</sup> Edition, Academic Press, USA.

Scott, A. C., Clayton, T. E., and Gibson, E. L. (1991). *A Practical Guide to Knowledge Acquisition*. Menlo Park, CA: Addison-Wesley.

Shute, V.J and Regian, J.W. (1993). Principles for evaluating intelligent tutoring systems. In a special evaluation issue of *Journal of Artificial Intelligence & Education* 4(3), pp 245-71

Shute, V. J. and Psotka, J. (1996). Intelligent Tutoring Systems: Past, Present and Future. In D. Jonassen (ed.) *Handbook of Research on Educational Communications and Technology*. Scholastic Publications.

Sison, R. and Shimura, M. (1996). The Application of Machine Learning to Student Modelling (A 1996 Perspective): Toward a Multistrategic Learning Student Modelling System. World wide web document.

Sleeman, D. and Brown, J.S. (1982). *Intelligent Tutoring Systems*. London, England : Academic.

Smith, Serengul. (1998) "Intelligent Tutoring Systems", School of Computing Science Middlesex University. Revised: September 1998.

[URL:

<http://www.cs.mdx.ac.uk/staffpages/serengul/Traditional.Computer.Aided.Learning.Systems.htm>]

Last accessed date: 21 November 2005.

Sommerville, Ian (2000). *Software Engineering*, 6<sup>th</sup> Edition, Addison-Wesley, USA.

Stathacopoulou R. , Magoulas G.D. and Grigoriadou M. (1999). Neural network-based fuzzy modeling of the student in intelligent tutoring systems, In *Proceedings of the INNS-IEEE International Joint Conference on Neural Networks*, Washington, U.S.A., July 1999.

Stephanov, Alex. (1995) "Part of the draft C++ standard, STL provides the framework for building generic, highly reusable algorithms and data structures", BYTE.com, October.

[URL: <http://www.byte.com/art/9510/sec12/art3.htm>]

Last accessed date: 21 November 2005.

Stephanov, Alex and Lee, Meng. (1995) *The Standard Template Library*. Internet Distribution, Published at <ftp://butler.hpl.hp.com/stl>, July 7, 1995

[URL: <http://www.cs.rpi.edu/~musser/doc.ps>]

Last accessed date: 21 November 2005.

Stoner G. and Harvey J. (1999) *Integrating learning technology in a foundation level management accounting course: an e(in)volving evaluation*. CTI-AFM Annual Conference, Brighton, U.K., April.

Stroustrup, Bjarne (1999). "Learning Standard C++ as a New Language", AT&T Labs, *The C/C++ Users Journal*, pp 43-54, May.

Sykes, E. R. (2003). An Intelligent Tutoring System Prototype for Learning to Program Java. In *Proceedings of the 3<sup>rd</sup> IEEE International Conference on Advanced Learning Technologies (ICALT '03)*, July 9-11, 2003, Athens, Greece, pp 485.

Sykes, E. R and Franek, F. (2003). "An Intelligent Tutoring System Prototype for Learning to Program Java", In *Proceedings of the 3<sup>rd</sup> IEEE International Conference on Advanced Learning Technologies (ICALT '03)*, Athens, Greece, July, 2003, pp 485-486.

Sugeno, M. (1985). *Industrial Applications of Fuzzy Control*. North-Holland, Amsterdam.

Tsiriga, V. and Virvou, M. (2002). "Individualized Assessment in a Web-based Algebra Tutor". In *Proceedings of 2002 International Conference on Information Communication Technologies in Education (ICICTE)*, Samos.

VanLehn, K. (1988). Student Modeling. *Foundations of Intelligent Tutoring Systems*, Polson M.C. and Richardson J.J. (Eds).

Virvou, M. and Tsiriga, V. (2001a). "Adaptive Tutoring Based on the Student Model of a Web-based ICALL". In *Proceedings of TELEMATICA-2001 International Conference on Telematics and Web-Based Education*, pp 47-50.

Virvou, M. and Tsiriga, V. (2001b). "Web Passive Voice Tutor: an Intelligent Computer Assisted Language Learning System over the WWW", In Okamoto, T., Hartley, R., Kinshuk, and Klus, J. (eds.) *Proceedings of the IEEE International Conference on Advanced Learning Technologies: Issues, Achievements and Challenges*, IEEE Society Press, Los Alamitos, 2001, pp 131-134.

Wang, T. and Mitrovic, A. (2002) Using neural networks to predict student's performance. In *Proceedings of the International Conference on Computers in Education (ICCE)*.

Warendorf K. and Tsao S. J. (1997). Application of Fuzzy Logic Techniques in the BSS1 Tutoring System. *Journal of Artificial Intelligence in Education*, 8(1), pp 113-146.

Weber, Gerhard. (1999). Adaptive Learning Systems in the World Wide Web. In *Proceedings of the 7<sup>th</sup> International Conference on User Modelling*, June 20-24, 1999, Banff, Canada.

Weber, Gerhard and Specht, Marcus. (1997). User Modeling and Adaptive Navigation Support in WWW-Based Tutoring System. In Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.) In *User Modeling: Proceedings of the Sixth International Conference, UM97*, Cagliari, Italy, Vienna, New York: Springer Wien New York, pp 289-300.

Weber, G., and Möllenberg, A. (1995). ELM programming environment: A tutoring system for LISP beginners. In Wender, K. F., Schmalhofer, F., and Böcker, H.-D. (Eds), *Cognition and Computer Programming*. Norwood, NJ: Ablex Publishing Corporation, pp 373-408.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. Los Altos, CA: Morgan Kaufmann Publishers, Inc.

Wiemer-Hastings, P. (2000). Adding syntactic information to LSA. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, pp 989-993 Mahwah, NJ. Erlbaum.

Wiemer-Hastings, Peter and Malatesta, Kalloipi-Irini. (2001). Introducing RMT: A dialog-based tutor for research methods. In *Proceedings of Artificial Intelligence in Education 2001*.

Wiemer-Hastings, P., and Zipitria, I. (2001). Rules for Syntax, Vectors for Semantics. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society* Mahwah, NJ. Erlbaum.

Wise, G. Bowden (1995). "An overview of the Standard Template Library", Dr Dobb's Journal, December.

[URL: <http://www.cs.rpi.edu/~wiseb/xrds/ovp2-3b.html>]

Last accessed date: 21 November 2005.

Wong, Danny. (2004). "Early Chinese Schooling", Daily Express, 29<sup>th</sup> March, p 9.

Woods, P. and Warren, J. (1995). Rapid Prototyping of an Intelligent tutorial system. In *Proceedings of the ASCILITE'95*, Melbourne, pp 557-563.

Zadeh, L. (1965). Fuzzy Sets, Information and Control, 8(3), pp 338-353.

Zhang Delai @ Chong Tet Loi. (2002). The Hakkas of Sabah, A Survey on Their Impact on the Modernization of the Borneon Malaysian State, Sabah Theological Seminary, Kota Kinabalu.

Ziemer, S. (1994). Intelligent Tutoring Systems in general and Curricula in particular. ERASMUS-Student, Coventry.

## Appendix

## Appendix A – List of Conditional Probabilities

Prerequisites (Topic)	Total number of correct answers (c)	Total number of prerequisite sub-skill questions ( q )	P(U) $p = c/q$	$m \cdot p$	$1 + (m-1) \cdot p$	P(U C)
<i>Data</i>	1	6	0.17	0.67	1.50	0.44
	2	6	0.33	1.33	2.00	0.67
	3	6	0.50	2.00	2.50	0.80
	4	6	0.67	2.67	3.00	0.89
	5	6	0.83	3.33	3.50	0.95
	6	6	1.00	4.00	4.00	1.00
<i>Operator</i>	1	4	0.25	1.00	1.75	0.57
	2	4	0.50	2.00	2.50	0.80
	3	4	0.75	3.00	3.25	0.92
	4	4	1.00	4.00	4.00	1.00
<i>Expression</i>	1	5	0.20	0.80	1.60	0.50
	2	5	0.40	1.60	2.20	0.73
	3	5	0.60	2.40	2.80	0.86
	4	5	0.80	3.20	3.40	0.94
	5	5	1.00	4.00	4.00	1.00
<i>Input-Output</i>	1	5	0.20	0.80	1.60	0.50
	2	5	0.40	1.60	2.20	0.73
	3	5	0.60	2.40	2.80	0.86
	4	5	0.80	3.20	3.40	0.94
	5	5	1.00	4.00	4.00	1.00
<i>Selection</i>	1	5	0.20	0.80	1.60	0.50
	2	5	0.40	1.60	2.20	0.73
	3	5	0.60	2.40	2.80	0.86
	4	5	0.80	3.20	3.40	0.94
	5	5	1.00	4.00	4.00	1.00
<i>Iteration</i>	1	6	0.17	0.67	1.50	0.44
	2	6	0.33	1.33	2.00	0.67
	3	6	0.50	2.00	2.50	0.80
	4	6	0.67	2.67	3.00	0.89
	5	6	0.83	3.33	3.50	0.95
	6	6	1.00	4.00	4.00	1.00
<i>Array</i>	1	6	0.17	0.67	1.50	0.44
	2	6	0.33	1.33	2.00	0.67
	3	6	0.50	2.00	2.50	0.80
	4	6	0.67	2.67	3.00	0.89
	5	6	0.83	3.33	3.50	0.95
	6	6	1.00	4.00	4.00	1.00



<i>User Defined Function</i>	1	7	0.14	0.57	1.43	0.40
	2	7	0.29	1.14	1.86	0.62
	3	7	0.43	1.71	2.29	0.75
	4	7	0.57	2.29	2.71	0.84
	5	7	0.71	2.86	3.14	0.91
	6	7	0.86	3.43	3.57	0.96
	7	7	1.00	4.00	4.00	1.00
<i>Class</i>	1	5	0.20	0.80	1.60	0.50
	2	5	0.40	1.60	2.20	0.73
	3	5	0.60	2.40	2.80	0.86
	4	5	0.80	3.20	3.40	0.94
	5	5	1.00	4.00	4.00	1.00
<i>Class Member Function</i>	1	5	0.20	0.80	1.60	0.50
	2	5	0.40	1.60	2.20	0.73
	3	5	0.60	2.40	2.80	0.86
	4	5	0.80	3.20	3.40	0.94
	5	5	1.00	4.00	4.00	1.00
<i>Constructor</i>	1	6	0.17	0.67	1.50	0.44
	2	6	0.33	1.33	2.00	0.67
	3	6	0.50	2.00	2.50	0.80
	4	6	0.67	2.67	3.00	0.89
	5	6	0.83	3.33	3.50	0.95
	6	6	1.00	4.00	4.00	1.00
<i>Function Template</i>	1	4	0.25	1.00	1.75	0.57
	2	4	0.50	2.00	2.50	0.80
	3	4	0.75	3.00	3.25	0.92
	4	4	1.00	4.00	4.00	1.00
<i>Class Template</i>	1	4	0.25	1.00	1.75	0.57
	2	4	0.50	2.00	2.50	0.80
	3	4	0.75	3.00	3.25	0.92
	4	4	1.00	4.00	4.00	1.00
<i>Operator Overloading</i>	1	4	0.25	1.00	1.75	0.57
	2	4	0.50	2.00	2.50	0.80
	3	4	0.75	3.00	3.25	0.92
	4	4	1.00	4.00	4.00	1.00
<i>Class String</i>	1	4	0.25	1.00	1.75	0.57
	2	4	0.50	2.00	2.50	0.80
	3	4	0.75	3.00	3.25	0.92
	4	4	1.00	4.00	4.00	1.00

## Appendix B – List of Topics and Sub-Topics

NO	MAIN TOPICS	SUB-TOPICS	NO. OF QUES	SUB TOTAL
1	Data	Data Type Variable Constant	3 2 1	6
2	Operator	Arithmetic Operator Relational Operator Logical Operator	2 1 1	4
3	Expression	Assignment Expression Conditional Expression	2 3	5
4	Input-Output	Output Input	3 2	5
5	Selection	if...else else if ... switch...case	3 1 1	5
6	Iteration	for while do...while	3 2 1	6
7	Array	Array Declaration Array Range Array Assignment Array - Passing to Function	3 1 1 1	6
8	User Defined Function	Function - Prototype Function - Call Function - Definition Function - Parameter Passing	2 2 1 2	7
9	Class	Object Declaration Class - Accessing Members Member Access Specifier	1 3 1	5
10	Class Member Function	Member Function - Declaration Member Function - Call Member Function - Definition	1 1 3	5
11	Constructor	Declaration - Default Declaration - Parametized Declaration - Copy Definition - Default Definition - Parametized Definition - Copy	1 1 1 1 1 1	6

12	Function Template	Function Template - Declaration	2	4
		Function Template - Call	1	
		Function Template - Definition	1	
13	Class Template	Class Template Declaration	1	4
		Class Template Creating Instance	1	
		Class Template Definition	1	
		Class Template Member Function	1	
14	Operator Overloading	Operator <<	1	4
		Operator >>	1	
		Template operator <<	1	
		Template operator >>	1	
15	Class String	String Constructor	3	4
		String Assignment	1	
				76

## Appendix C – Vision

### System Vision

The Intelligent Tutorial System (ITS) is a computer aided tutoring platform, which provides tutorials to any students / users the usage knowledge of C++ Standard Template Library (STL).

ITS is intelligent enough to measure the fundamental skill of a single student, thus providing the student with the appropriate STL tutoring path. The measurement of skills and students' progress is based on the implementation of common Artificial Intelligence algorithms.

The ITS is also a platform for moderators and tutors to enter STL questions, their respective weighting when correctly answered and their alternative learning route when it is not correctly answered, based on the application of artificial intelligence expert system.

Students' performances will be recorded in every aspect of their learning period, including the time taken in answering the questions, how many trials before answering correctly and other relevant aspects, which lead to the understanding of the students' learning and knowledge progress.

Reports and monitors will be part of the platform, which provides the tutors with information of the students' progress.

The ultimate objective of the system platform is to ensure that the students will improve their skills in C++ STL from a novice level to an advanced level by solely depending on the tutoring platform, without the aid of human tutors.

## Appendix D – User Stories

### Title: User Roles

*Description:* There are altogether 3 main user roles.

- 1) The Administrator: A user who has the authority to add multiple users to the system and maintain their individual details (e.g. user name, password, e-mail, etc).
- 2) The Student: A user who will be using the platform as a means of learning. This is the user who will go through the tutorial provided by the system.
- 3) The Tutor: A user whose role is to input the tutorial questions and direct the learning path of the question depending on its answer given.
- 4) External Student: A user who will only be using the post-test as a knowledge comparison to the system users.

### Title: Tutorial Categories

*Description:* The tutorial for the student will be categorized as Pre-Test, Tutorial, and Post-Test.

### Title: Pre-Test

*Description:* The Pre-Test is a test, which determines the students' fundamental skills before taking the tutorial on STL. This is crucial as if the student is weak in fundamentals, the student will be routed to be tutored on its fundamentals before tutoring STL. This is called the Prerequisite tutorial. Pre-Test questions are made up of multiple choices, single answer questions.

### Title: Tutorial

*Description:* The tutorial will have two parts, both the prerequisite and the STL. The prerequisite tutorials will have the student tutored on the fundamentals, whereas the STL is the main tutorial in the system. Tutorial questions will consist of a general question framework, with blanks for answers to be filled. Sub-tutorials will be given to the student on each question framework.

Title: Sub-tutorials and question framework relationship

*Description:* For every question framework, there will be a series of sub-tutorials. For each sub-tutorial, it will consist of an answer box (either a test box or a drop down combo box) for the student to input the answer. Each sub-tutorial will be displayed in sequence. The next sub-tutorial will not be displayed until the previous sub-tutorial is answered. However, the answer to the previous sub-tutorials could be updated. Each sub-tutorial will have its own respective tracker. All sub-tutorials will be finalized when the “finalize” button is clicked and data will be recorded.

Title: Post-Test

*Description:* Post-Test format will be the same as the pre-test.

Title: User Login

*Description:* All users of the platform must login to the system by providing the user name and the password. The administrator has a ready set user name and password, which could be changed when it is logged-in. The others will depend on the user name and password given by their administrator. All users could log off after using the system. A specific time-out will be applied.

Title: Administrator Options

*Description:* When the administrator is logged in, the administrator has the option to add, update and delete all users of the system, excluding himself. A search engine will be provided to the administrator to search for the user by its user name, name, e-mail and role. The return results will be listed in pages. When the user has been clicked, the details will be shown with the above-mentioned option. Only the user name could not be updated.

Title: Administrator Front Page

*Description:* The front page of the administrator will display the summary of the system, including number of tutors, students, external students, both active and non-active and support messages.

Title: Administrator Tutor's Page

*Description:* This page will be displayed when the Users -> Tutors tab has been clicked. The tutor's tab will have two panes, mainly the side pane and the main pane. The side pane provides links to add a tutor and display all tutors. When the page is loaded, a list of all the tutors will be displayed with their respective options (such as: details, delete) at the side.

#### Title: Administrator Tutors Listing

*Description:* This page will list all the tutors with their respective options (such as: details, delete) at the side. The page will only list 20 tutors at one time, and a page pane will display the pages below. Once the page number is clicked, the next page of tutors will be displayed. When the details link is clicked, the details of the tutor will be displayed with all details field populated and with the update option visible.

#### Title: Administrator Tutor Addition

*Description:* This page will allow the administrator to add a tutor into the database. The tutor's data model will include two sections: Login Information and Professional Details. The Login Information section includes both user name and password. The Professional Details Section will include name, title, e-mail, and department. Duplication check field: user name.

#### Title: Tutor Options

*Description:* Any tutor will have the option of entering into the Pre-Test, Tutorial and Post-Test sections. These test sections will grant the tutor the ability to add, update and remove questions of their respective test category (discussed later). The tutor has the option to search for a particular student (Student/External Student) to view his/her performance. This is done through search engine.

#### Title: Pre-Test and Tutorial Prerequisite Topics and Sub-Topics

*Description:* Both the Pre-Test and Tutorial prerequisite topics and sub-topics are of the same kind. For each topic, it consists of multiple sub-topics, or what we call sub-skills.

#### Title: Entering the Pre-Test questions (for tutors only)

*Description:* When the tutor selects the pre-test section, a list of topics and their sub-topics will be displayed. Tutors are allowed to add, update and delete topics and the sub-topics. For each topic branch, an option to add a sub-topic will be visible. As for each sub-topic branch, an option to view the questions will be visible.

#### Title: Main Topic addition

*Description:* When the *add* main topic link is clicked on, a text box will be displayed for the input of the main-topic's title. When entered, it will check for duplication. If everything is fine, the main topic will be added and returned to the list of all main topics and sub-topics.

#### Title: Sub-topic addition

*Description:* When clicked on the *add* sub-topic link, a text box will be displayed for the input of the sub-topic's title. When entered, it will check for duplication. If everything is fine, the main topic will be added and returned to the list of all main topics and sub-topics.

#### Title: Sub-topic views

*Description:* From the main topic listing, when a tutor clicks on the sub-topics list link, a series of questions and their ID will be displayed. From here, the tutor will have the options to add, delete and update the sub-topic questions.

#### Title: Sub-topic questions composition

*Description:* All sub-topic questions are composed of a question and multiple-choice single answer.

#### Title: Addition of sub-topic questions

*Description:* The sub-topic addition page will consist of multiple text areas and text boxes. The main text area will be for the input of the question (HTML tag is allowed) followed by 4 text areas for the answers and their respective radio button, which indicates the correctness of the answer. If 4 text areas if not enough, the tutor could click on the “Add more answers boxes” button, which will then generate another 4 extra answers text box and so on. A box to enter the weight (score) of the question will be provided. Sub-Topic questions could either be Pre-Test or for Prerequisite. An option (check box or radio button) could trigger this. All answers allow HTML tags!

#### Title: Sub-topic question model structure

*Description:* Each addition of new sub-topic questions will not be checked for duplication. Any new sub-topic question will be given a unique ID generated by the system for referential integrity.



Title: Update of sub-topic

*Description:* In the list of sub-topics, the tutor may choose any sub-topic link. This will be followed by the same page format as the adding of sub-topic, expecting that the questions and all the answers will be populated on the text areas. Any changes made on the populated text areas will be saved after the “Update” button is clicked.

Title: Deletion of sub-topic

*Description:* After the delete link is clicked, a confirmation page will be displayed, asking for a confirmation. If confirmed, the sub-topic will be deleted and will no longer appear in the sub-topic list. If not, the operation is canceled.

## Appendix E – Acceptance Tests

```
package com.its.test;

import com.its.exception.*;
import junit.framework.*;

/**
 * Test Case for Exception Handling
 *
 * @author
 * @version $Revision: 1.1 $
 */
public class ExceptionTest extends TestCase
{
    /**
     * Creates a new ExceptionTest object.
     *
     * @param testName DOCUMENT ME!
     */
    public ExceptionTest( java.lang.String testName )
    {
        super( testName );
    }

    /**
     * Test suite
     *
     * @return DOCUMENT ME!
     */
    public static Test suite( )
    {
        TestSuite suite = new TestSuite( ExceptionTest.class );

        return suite;
    }

    /**
     * Test the ITSEException
     */
    public void test01( )
    {
        try
        {
            throw new NullPointerException( );
        }
        catch ( NullPointerException ex )
        {
            ITSEException iex = new ITSEException( ex );
            System.out.println( iex.toString( ) );
        }
    }
}
```

---

```

package com.its.module.student.test;

import com.its.base.constant.*;

import com.its.module.student.constant.IStudentSearchFields;
import com.its.module.student.model.*;
import com.its.module.student.service.sb.*;
import com.its.module.user.model.*;

import com.its.resource.ResourceLocator;

import com.vagrant.lossehelin.j2ee.*;

import junit.framework.*;

import java.util.*;

/**
 * Test Case for Student Session Bean
 *
 * @author
 * @version 1.1
 */
public class StudentSBTest extends TestCase implements IJNDI
{
    /**
     * Creates a new StudentSBTest object.
     *
     * @param testName DOCUMENT ME!
     */
    public StudentSBTest( java.lang.String testName )
    {
        super( testName );
    }

    /**
     * Test suite
     *
     * @return DOCUMENT ME!
     */
    public static Test suite( )
    {
        TestSuite suite = new TestSuite( StudentSBTest.class );

        return suite;
    }
}

```

```

/**
 * Test Case 1 - Create
 *
 * @throws Exception DOCUMENT ME!
 */
public void atest01( ) throws Exception
{
    ServiceLocator locator = ServiceLocator.newInstance
        ( ResourceLocator.getJBossLookupProperties( ) );
    StudentSBHome studentSBHome = ( StudentSBHome ) locator.getRemoteHome
        ( JNDI_STUDENT_SB, StudentSBHome.class );
    StudentSBRemote studentSBRemote = studentSBHome.create( );

    for ( int i = 0, j = 100; i < 100; i++, j-- )
    {
        UserDO userDO = new UserDO( );
        userDO.setUsername( "TSUU" + i );
        userDO.setPassword( "PW" + i );
        userDO.setName( "TSUNAME" + j );
        userDO.setDepartment( "TSUDEPT" + i );
        userDO.setEmail( "tsumail" + i + "@test.com" );
        userDO.setStatus( Istatus.ACTIVE );
        userDO.setType( IUserType.STUDENT );

        StudentDO studentDO = new StudentDO( );
        studentDO.setClassCode( "TSUCC" + i );
        studentDO.setStatus( Istatus.ACTIVE );
        studentDO.setUserDO( userDO );

        studentSBRemote.createStudent( studentDO );
    }
}

/**
 * Test Case 2 – Find and Display
 *
 * @throws Exception DOCUMENT ME!
 */
public void atest02( ) throws Exception
{
    ServiceLocator locator = ServiceLocator.newInstance
        ( ResourceLocator.getJBossLookupProperties( ) );
    StudentSBHome studentSBHome = ( StudentSBHome ) locator.getRemoteHome
        ( JNDI_STUDENT_SB, StudentSBHome.class );
    StudentSBRemote studentSBRemote = studentSBHome.create( );

    for ( int i = 0; i < 100; i++ )
    {
        AbstractStudentDO studentDO = studentSBRemote.findStudent( "TSUU" + i );
        System.out.println( studentDO );
    }
}

```

```

/**
 * Test Case 3 – Find and Update
 *
 * @throws Exception DOCUMENT ME!
 */
public void atest03( ) throws Exception
{
    ServiceLocator locator = ServiceLocator.newInstance
        ( ResourceLocator.getJBossLookupProperties( ) );
    StudentSBHome studentSBHome = ( StudentSBHome ) locator.getRemoteHome
        ( JNDI_STUDENT_SB, StudentSBHome.class );
    StudentSBRemote studentSBRemote = studentSBHome.create( );

    for ( int i = 0; i < 100; i++ )
    {
        AbstractStudentDO studentDO = studentSBRemote.findStudent( "TSUU" + i );

        AbstractUserDO userDO = studentDO.getUserDO( );
        userDO.setPassword( "PWU" + i );
        userDO.setName( "TSUNAMEU" + i );
        userDO.setDepartment( "TSUDEPTU" + i );
        userDO.setEmail( "tsumailu" + i + "@test.com" );
        userDO.setStatus( IStatus.ACTIVE );
        userDO.setType( IUserType.STUDENT );

        studentDO.setClassCode( "TSUTITLEU" + i );
        studentDO.setStatus( IStatus.ACTIVE );

        studentSBRemote.updateStudent( studentDO );
    }
}

/**
 * Test Cases 4 – Find All using Username
 *
 * @throws Exception DOCUMENT ME!
 */
public void test04( ) throws Exception
{
    ServiceLocator locator = ServiceLocator.newInstance
        ( ResourceLocator.getJBossLookupProperties( ) );
    StudentSBHome studentSBHome = ( StudentSBHome ) locator.getRemoteHome
        ( JNDI_STUDENT_SB, StudentSBHome.class );
    StudentSBRemote studentSBRemote = studentSBHome.create( );
    Collection col = studentSBRemote.findAllStudent( IStudentSearchFields.USERNAME );
    Iterator itr = col.iterator( );

    while ( itr.hasNext( ) )
    {
        StudentDO _DO = ( StudentDO ) itr.next( );
        System.out.println( _DO.toString( ) );
    }
}

```

```

/**
 * Test Case 5 – Find All with Criteria
 *
 * @throws Exception DOCUMENT ME!
 */
public void atest05( ) throws Exception
{
    StudentSearchDO studentSearchDO = new StudentSearchDO( );
    studentSearchDO.setUsername( "TSUU9%" );

    ServiceLocator locator = ServiceLocator.newInstance
        ( ResourceLocator.getJBossLookupProperties( ) );
    StudentSBHome studentSBHome = ( StudentSBHome ) locator.getRemoteHome
        ( JNDI_STUDENT_SB, StudentSBHome.class );
    StudentSBRemote studentSBRemote = studentSBHome.create( );
    Collection col = studentSBRemote.findAllStudentWithCriteria( studentSearchDO,
        IStudentSearchFields.USERNAME );
    Iterator itr = col.iterator( );

    while ( itr.hasNext( ) )
    {
        StudentDO _DO = ( StudentDO ) itr.next( );
        System.out.println( _DO.toString( ) );
    }
}

/**
 * Test Case 6 – Find All with Criteria
 *
 * @throws Exception DOCUMENT ME!
 */
public void atest06( ) throws Exception
{
    StudentSearchDO studentSearchDO = new StudentSearchDO( );

    studentSearchDO.setUsername( "TSUU9%" );
    studentSearchDO.setName( "TSUNAME%" );
    studentSearchDO.setDepartment( "TSUDEPT%" );

    ServiceLocator locator = ServiceLocator.newInstance
        ( ResourceLocator.getJBossLookupProperties( ) );
    StudentSBHome studentSBHome = ( StudentSBHome ) locator.getRemoteHome
        ( JNDI_STUDENT_SB, StudentSBHome.class );
    StudentSBRemote studentSBRemote = studentSBHome.create( );
    Collection col = studentSBRemote.findAllStudentWithCriteria( studentSearchDO,
        IStudentSearchFields.USERNAME );
    Iterator itr = col.iterator( );

    while ( itr.hasNext( ) )
    {
        StudentDO _DO = ( StudentDO ) itr.next( );
        System.out.println( _DO.toString( ) );
    }
}

```

```

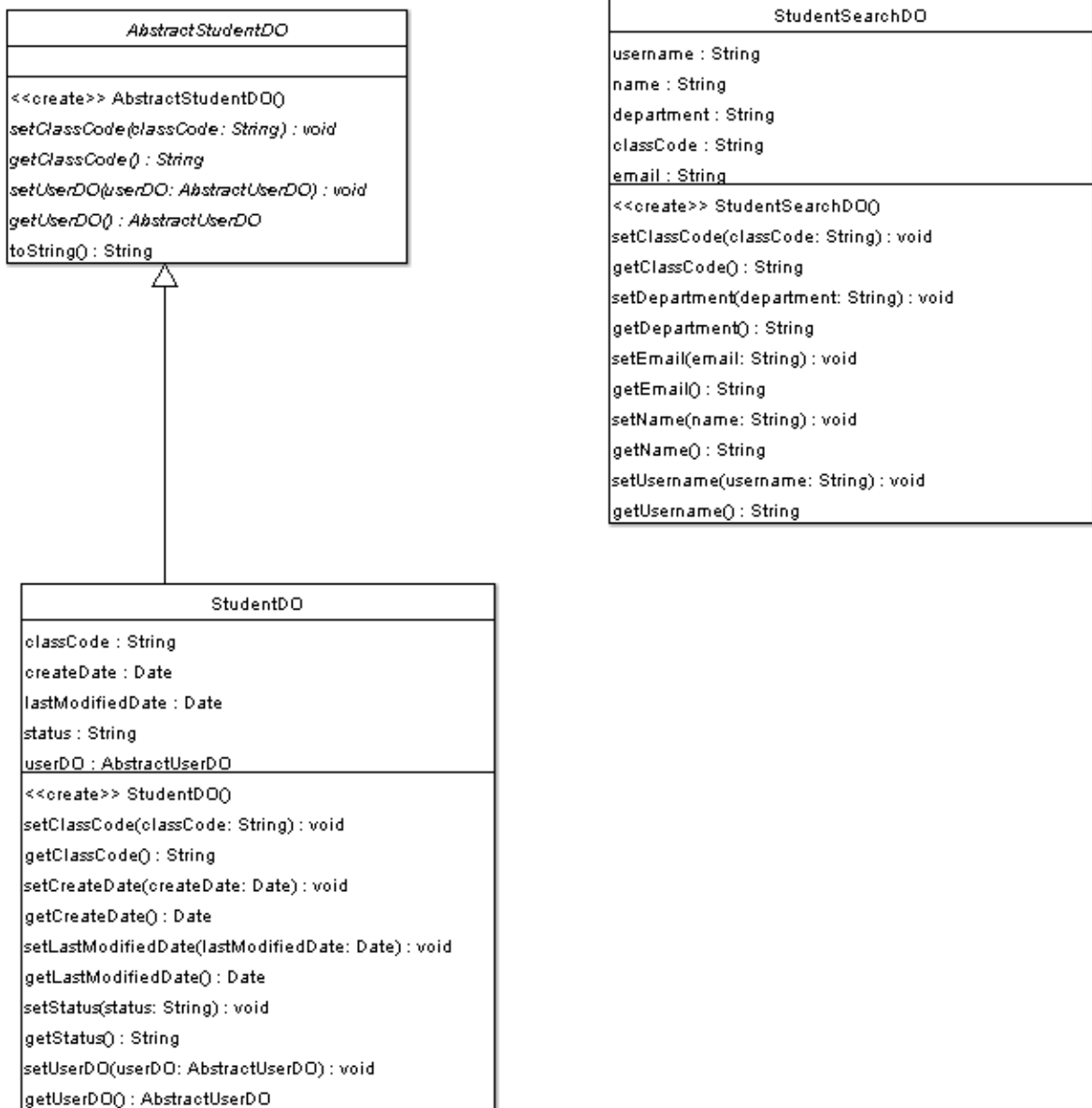
/**
 * Test Case 7 - Remove
 *
 * @throws Exception DOCUMENT ME!
 */
public void atest07( ) throws Exception
{
    ServiceLocator locator = ServiceLocator.newInstance
        ( ResourceLocator.getJBossLookupProperties( ) );
    StudentSBHome studentSBHome = ( StudentSBHome ) locator.getRemoteHome
        ( JNDI_STUDENT_SB, StudentSBHome.class );
    StudentSBRemote studentSBRemote = studentSBHome.create( );

    for ( int i = 0; i < 100; i++ )
    {
        studentSBRemote.removeStudent( "TSUU" + i );
    }
}
}

```

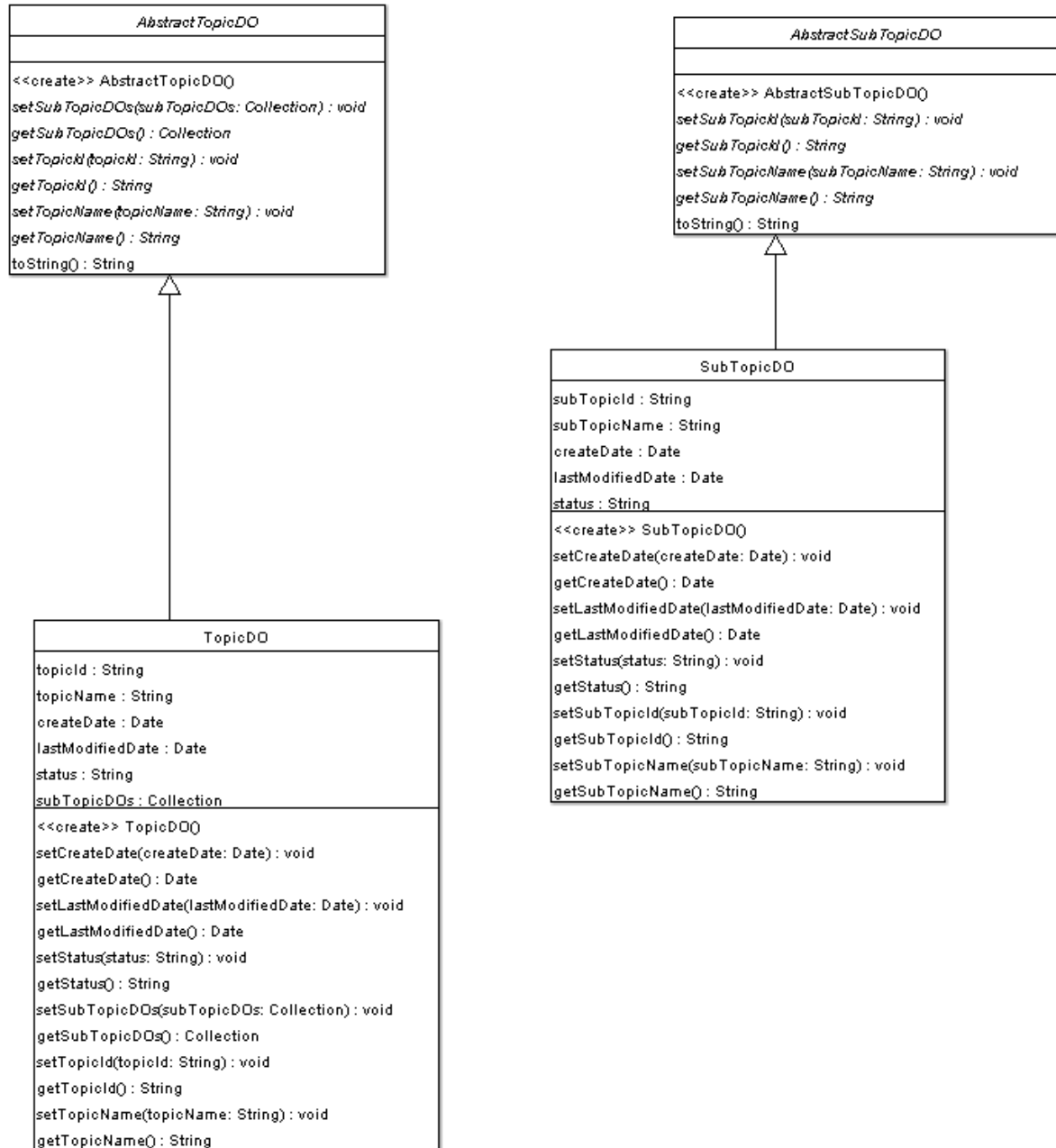
## Appendix F – UML Design Diagrams

### Class Diagrams – Student Module

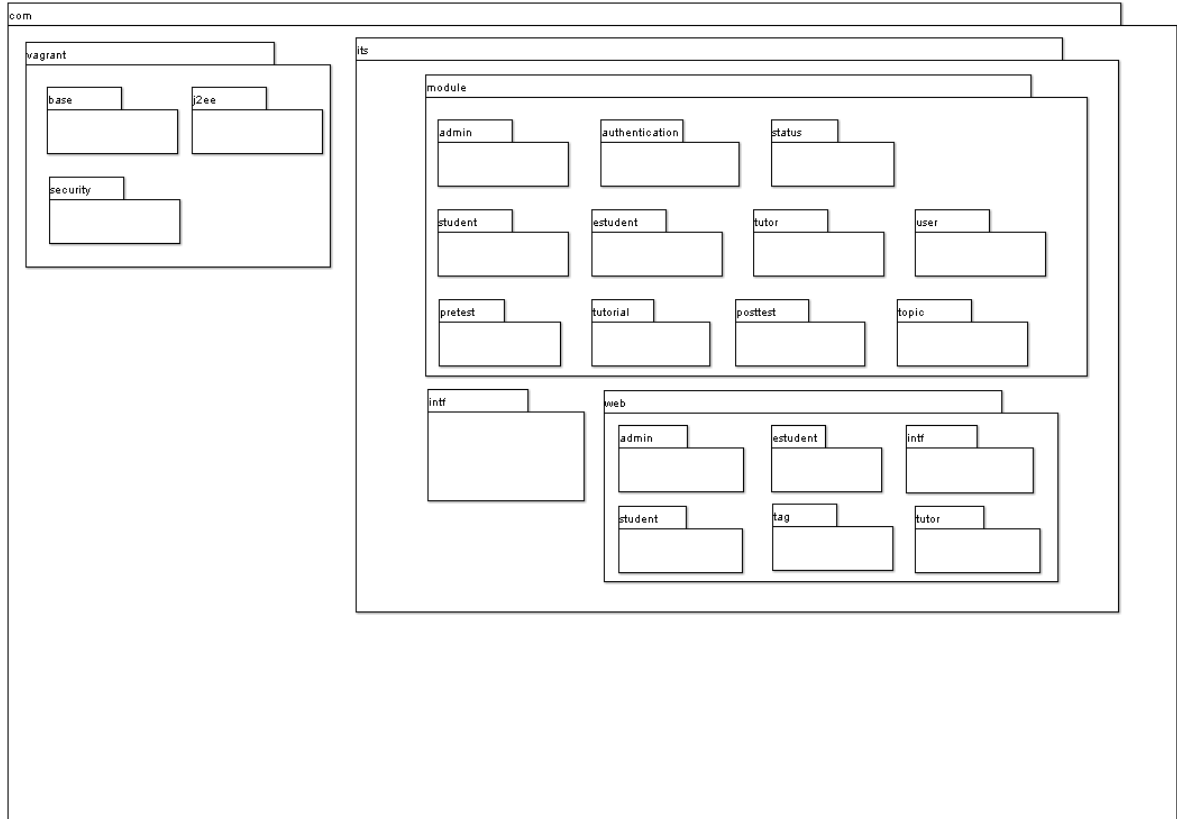




## Class Diagrams – Topic Module

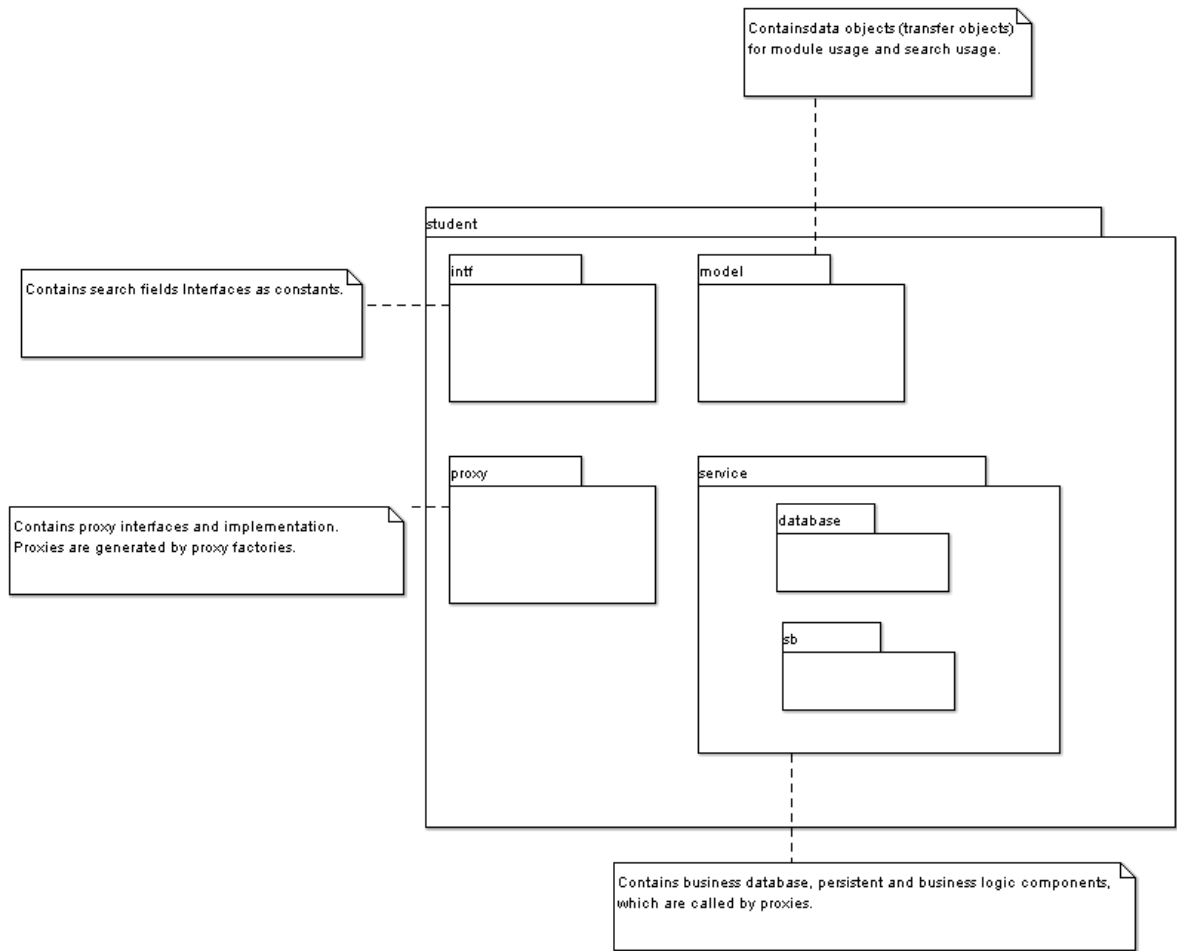


## General Package Diagram



## General Module Package Diagram

### Sample – Student Module



## Appendix G – Session Beans

### Admin

```
package com.its.module.admin.service.sb;

import com.its.module.admin.model.*;
import com.its.module.admin.service.database.*;
import com.its.module.user.model.*;

import com.vagrant.base.exception.*;
import com.vagrant.j2ee.ejb.*;

import org.apache.commons.logging.*;
import org.apache.commons.logging.impl.*;

import java.util.*;
import javax.ejb.*;

public class AdminSBBean extends AbstractCommonSBBean
{
    private Log log = LogFactory.getLog( AdminSBBean.class.getName( ) );
    private AdminDAO adminDAO = null;

    public AdminSBBean( ) throws EJBException
    {
        try
        {
            adminDAO = new AdminDAO( );
        }
        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw new EJBException( ex );
        }
    }

    public int getTotalActiveAdmin( ) throws CException
    {
        return adminDAO.getTotalActiveAdmin( );
    }

    public int getTotalActiveEStudent( ) throws CException
    {
        return adminDAO.getTotalActiveEStudent( );
    }

    public int getTotalActiveStudent( ) throws CException
    {
        return adminDAO.getTotalActiveStudent( );
    }
}
```

```

public int getTotalActiveTutor( ) throws CException
{
    return adminDAO.getTotalActiveTutor( );
}

public int getTotalAdmin( ) throws CException
{
    return adminDAO.getTotalAdmin( );
}

public int getTotalEStudent( ) throws CException
{
    return adminDAO.getTotalEStudent( );
}

public int getTotalStudent( ) throws CException
{
    return adminDAO.getTotalStudent( );
}

public int getTotalSuspendedAdmin( ) throws CException
{
    return adminDAO.getTotalSuspendedAdmin( );
}

public int getTotalSuspendedEStudent( ) throws CException
{
    return adminDAO.getTotalSuspendedEStudent( );
}

public int getTotalSuspendedStudent( ) throws CException
{
    return adminDAO.getTotalSuspendedStudent( );
}

public int getTotalSuspendedTutor( ) throws CException
{
    return adminDAO.getTotalSuspendedTutor( );
}

public int getTotalTutor( ) throws CException
{
    return adminDAO.getTotalTutor( );
}

public AdminDO findAdminByAdminSysId( long sysId ) throws CException
{
    return adminDAO.findAdminByAdminSysId( sysId );
}

public AdminDO findAdminByUsername( String username ) throws CException
{
    return adminDAO.findAdminByUsername( username );
}

```

```

public Collection findAllAdmin( ) throws CException
{
    return adminDAO.findAllAdmin( );
}

public Collection findAllAdminWithCriteria( AdminSearchDO adminSearchDO )
                                           throws CException
{
    return adminDAO.findAllAdminWithCriteria( adminSearchDO );
}

public void removeAdmin( AdminDO _DO ) throws CException
{
    adminDAO.removeAdmin( _DO );
}

public void updateAdmin( AdminDO _DO ) throws CException
{
    adminDAO.updateAdmin( _DO );
}
}

```

## Authentication

```
package com.its.module.authentication.service.sb;

import com.its.module.authentication.model.*;
import com.its.module.authentication.service.database.*;
import com.its.module.user.model.*;
import com.its.module.user.service.database.*;

import com.vagrant.base.exception.*;
import com.vagrant.j2ee.ejb.*;
import com.vagrant.security.*;

import org.apache.commons.logging.*;
import org.apache.commons.logging.impl.*;

import javax.ejb.*;

public class AuthenticationSBBean extends AbstractCommonSBBean
{
    private Log log = LogFactory.getLog( AuthenticationSBBean.class.getName( ) );
    private AuthenticationDAO authDAO = null;
    private UserDAO userDAO = null;

    public AuthenticationSBBean( ) throws EJBException
    {
        try
        {
            authDAO = new AuthenticationDAO( );
            userDAO = new UserDAO( );
        }
        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw new EJBException( ex );
        }
    }

    public boolean isSessionExists( String username ) throws CException
    {
        log.debug( "Executing isSessionExists( String username )" );

        boolean isSessionExists = false;

        try
        {
            try
            {
                authDAO.findLoginSessionByUsername( username );

                isSessionExists = true;
            }
            catch ( Exception ex )
            {
                log.debug( "Session does not exists" );
            }
        }
    }
}
```

---

```

        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw getExceptionHandler( ).generateError( "ERDB-003" );
        }

        return isSessionExists;
    }

    public boolean isSessionValid( LoginDO _DO ) throws CException
    {
        log.debug( "Executing isSessionValid( AbstractLoginDO _DO )" );

        boolean isSessionValid = false;

        try
        {
            LoginDO tmpDO = findLoginSession( _DO.getUserDO( ).getUsername( ) );

            if ( _DO.equals( tmpDO ) )
            {
                isSessionValid = true;
            }
        }
        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw getExceptionHandler( ).generateError( "ERDB-003" );
        }

        return isSessionValid;
    }

    public String authenticate( String username, String password ) throws CException
    {
        log.debug( "Executing authenticate( String username, String password )" );

        try
        {
            username = username.trim( ).toUpperCase( );
            password = password.trim( );

            //Authenticate username
            UserDO userDO = findUser( username );

            //Authenticate password
            if ( !userDO.getPassword( ).equals( password ) )
            {
                throw getExceptionHandler( ).generateError( "ERLG-002" );
            }

            return userDO.getType( );
        }
        catch ( CException ex )
        {
            log.error( ex.getMessage( ), ex );
            throw ex;
        }
    }

```



```

        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw getExceptionHandler( ).generateError( "ERDB-003" );
        }
    }

    public void createLoginSession( LoginDO _DO ) throws CException
    {
        log.debug( "Executing createLoginSession( AbstractLoginDO _DO )" );

        try
        {
            java.sql.Timestamp now = new java.sql.Timestamp(System.currentTimeMillis( ));
            _DO.setCreatedTimestamp( now );

            authDAO.createLoginSession( _DO );
        }
        catch ( CException ex )
        {
            ctx.setRollbackOnly( );
            log.error( ex.getMessage( ), ex );
            throw ex;
        }
        catch ( Exception ex )
        {
            ctx.setRollbackOnly( );

            log.error( ex.getMessage( ), ex );
            throw getExceptionHandler( ).generateError( "ERDB-003" );
        }
    }

    public void dismissSession( String username ) throws CException
    {
        log.debug( "Executing dismissSession( String username )" );

        try
        {
            try
            {
                LoginDO _DO = authDAO.findLoginSessionByUsername( username );
                authDAO.deleteLoginSession( _DO );
            }
            catch ( Exception ex )
            {
                log.debug( "Session Not Found" );
            }
        }
        catch ( Exception ex )
        {
            ctx.setRollbackOnly( );

            log.error( ex.getMessage( ), ex );
            throw getExceptionHandler( ).generateError( "ERDB-003" );
        }
    }
}

```

```

public LoginDO findLoginSession( String username ) throws CException
{
    try
    {
        return authDAO.findLoginSessionByUsername( username );
    }
    catch ( CException ex )
    {
        log.error( ex.getMessage( ), ex );
        throw ex;
    }
    catch ( Exception ex )
    {
        log.error( ex.getMessage( ), ex );
        throw getExceptionHandler( ).generateError( "ERDB-003" );
    }
}

public UserDO findUser( String username ) throws CException
{
    try
    {
        UserDO _DO = userDAO.findUserByUsername( username );

        /*
        PassPhraseEncryptor encryptor = new PassPhraseEncryptor(_DO.getUsername( ));
        String password = encryptor.decrypt( _DO.getPassword( ) );
        _DO.setPassword( password );
        */
        return _DO;
    }
    catch ( CException ex )
    {
        log.error( ex.getMessage( ), ex );
        throw ex;
    }
    catch ( Exception ex )
    {
        log.error( ex.getMessage( ), ex );
        throw getExceptionHandler( ).generateError( "ERDB-003" );
    }
}

public UserDO findUserBySysId( long sysId ) throws CException
{
    return userDAO.findUserBySysId( sysId );
}
}

```

## EStudent

```
package com.its.module.estudent.service.sb;

import com.its.module.estudent.model.*;
import com.its.module.estudent.service.database.*;
import com.its.module.user.model.*;

import com.vagrant.base.exception.*;
import com.vagrant.j2ee.ejb.*;

import org.apache.commons.logging.*;
import org.apache.commons.logging.impl.*;

import java.util.*;
import javax.ejb.*;

public class EStudentSBBean extends AbstractCommonSBBean
{
    private Log log = LogFactory.getLog( EStudentSBBean.class.getName( ) );

    EStudentDAO eStudentDAO = null;

    public EStudentSBBean( ) throws EJBException
    {
        try
        {
            eStudentDAO = new EStudentDAO( );
        }
        catch ( Exception ex )
        {
            throw new EJBException( ex );
        }
    }

    public long createEStudent( EStudentDO _DO ) throws CException
    {
        return eStudentDAO.createEStudent( _DO );
    }

    public Collection findAllEStudent( ) throws CException
    {
        return eStudentDAO.findAllEStudent( );
    }

    public Collection findAllEStudentWithCriteria( EStudentSearchDO eStudentSearchDO )
                                                throws CException
    {
        return eStudentDAO.findAllEStudentWithCriteria( eStudentSearchDO );
    }

    public EStudentDO findEStudentByEStudentSysId( long sysId ) throws CException
    {
        return eStudentDAO.findEStudentByEStudentSysId( sysId );
    }
}
```

```
public EStudentDO findEStudentByUserSysId( long userSysId ) throws CException
{
    return eStudentDAO.findEStudentByUserSysId( userSysId );
}

public EStudentDO findEStudentByUsername( String username ) throws CException
{
    return eStudentDAO.findEStudentByUsername( username );
}

public void removeEStudent( EStudentDO _DO ) throws CException
{
    eStudentDAO.removeEStudent( _DO );
}

public void updateEStudent( EStudentDO _DO ) throws CException
{
    eStudentDAO.updateEStudent( _DO );
}
}
```

## Posttest

```
package com.its.module.posttest.service.sb;

import com.its.module.posttest.model.*;
import com.its.module.posttest.service.database.*;
import com.its.module.posttest.service.helper.*;
import com.its.module.pretest.model.PreTestStatisticDO;
import com.its.module.pretest.service.helper.PreTestHelper;
import com.its.module.user.model.*;
import com.its.module.user.service.database.*;

import com.vagrant.base.exception.*;
import com.vagrant.j2ee.ejb.*;

import org.apache.commons.logging.*;
import org.apache.commons.logging.impl.*;

import java.sql.*;
import java.util.*;
import javax.ejb.*;

public class PostTestSBBean extends AbstractCommonSBBean
{
    private Log log = new Log4JLogger( PostTestSBBean.class.getName( ) );
    private PostTestDAO postTestDAO = null;
    private UserDAO userDAO = null;

    public PostTestSBBean( ) throws EJBException
    {
        try
        {
            postTestDAO = new PostTestDAO( );
            userDAO = new UserDAO( );
        }
        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw new EJBException( ex );
        }
    }

    public PostTestStatisticDO getPostTestStatisticsByUserSysId( long userSysId ) throws CException
    {
        try
        {
            UserDO userDO = userDAO.findUserBySysId( userSysId );

            //Calculate all the pass and failed stuff
            PostTestHelper postTestHelper = new PostTestHelper( );
            Collection postTestAnswerRecordDOs =
                findPostTestAnswerRecordsByUserSysId( userDO.getSysId( ) );

            return postTestHelper.generatePostTestStatistics( postTestAnswerRecordDOs,
                                                            userDO );
        }
    }
}
```

---

```

        catch ( CException ex )
        {
            log.error( ex.getMessage( ), ex );
            throw ex;
        }
        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw getExceptionHandler( ).generateError( "ERSYS-002", ex );
        }
    }

    public Collection findAllPostTestQuestionBySearchCriteria( PostTestQuestionSearchDO
                                                            postTestQuestionSearchDO ) throws CException
    {
        return postTestDAO.findPostTestQuestionsBySearchCriteria(postTestQuestionSearchDO);
    }

    public Collection findAllPostTestQuestions( ) throws CException
    {
        return postTestDAO.findAllPostTestQuestions( );
    }

    public PostTestAnswerDO findPostTestAnswerBySysId( long sysId ) throws CException
    {
        return postTestDAO.findPostTestAnswerBySysId( sysId );
    }

    public PostTestAnswerRecordDO findPostTestAnswerRecord( long postTestQuestionSysId,
                                                            long userSysId ) throws CException
    {
        return postTestDAO.findPostTestAnswerRecord( postTestQuestionSysId, userSysId );
    }

    public Collection findPostTestAnswerRecordDOByTopicSysId( long topicSysId )
                                                            throws CException
    {
        return postTestDAO.findPostTestAnswerRecordDOByTopicSysId( topicSysId );
    }

    public Collection findPostTestAnswerRecordDOByTopicSysIdUserSysId( long topicSysId,
                                                                        long userSysId ) throws CException
    {
        return postTestDAO.findPostTestAnswerRecordDOByTopicSysIdUserSysId( topicSysId,
                                                                                userSysId );
    }

    public Collection findPostTestAnswerRecordsByUserSysId( long userSysId ) throws CException
    {
        return postTestDAO.findPostTestAnswerRecordsByUserSysId( userSysId );
    }

    public Collection findPostTestAnswersByPostTestQuestionSysId( long sysId ) throws CException
    {
        return postTestDAO.findPostTestAnswersByPostTestQuestionSysId( sysId );
    }

```

```
public PostTestQuestionDO findPostTestQuestionBySysId( long sysId ) throws CException
{
    return postTestDAO.findPostTestQuestionBySysId( sysId );
}

public Collection findPostTestResultCPsByUserSysId( long userSysId ) throws CException
{
    return postTestDAO.findPostTestResultCPsByUserSysId( userSysId );
}

public long findTotalPostTestQuestions( ) throws CException
{
    return postTestDAO.findTotalPostTestQuestions( );
}
}
```

## Pretest

```
package com.its.module.pretest.service.sb;

import com.its.module.pretest.model.*;
import com.its.module.pretest.service.database.*;
import com.its.module.pretest.service.helper.*;
import com.its.module.user.model.*;
import com.its.module.user.service.database.*;

import com.vagrant.base.exception.*;
import com.vagrant.j2ee.ejb.*;

import org.apache.commons.logging.*;
import org.apache.commons.logging.impl.*;

import java.sql.*;
import java.util.*;
import javax.ejb.*;

public class PreTestSBBean extends AbstractCommonSBBean
{
    private Log log = LogFactory.getLog( PreTestSBBean.class.getName( ) );
    private PreTestDAO preTestDAO = null;
    private UserDAO userDAO = null;

    public PreTestSBBean( ) throws EJBException
    {
        try
        {
            preTestDAO = new PreTestDAO( );
            userDAO = new UserDAO( );
        }
        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw new EJBException( ex );
        }
    }

    public PreTestStatisticDO getPreTestStatisticsByUserSysId( long userSysId ) throws CException
    {
        try
        {
            UserDO userDO = userDAO.findUserBySysId( userSysId );

            //Calculate all the pass and failed stuff
            PreTestHelper preTestHelper = new PreTestHelper( );
            Collection preTestAnswerRecordDOs = findPreTestAnswerRecordsByUserSysId
                ( userDO.getSysId( ) );

            return preTestHelper.generatePreTestStatistics( preTestAnswerRecordDOs,
                                                            userDO );
        }
    }
}
```



```

        catch ( CException ex )
        {
            log.error( ex.getMessage( ), ex );
            throw ex;
        }
        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw getExceptionHandler( ).generateError( "ERSYS-002", ex );
        }
    }

    public Collection findAllPreTestQuestionBySearchCriteria( PreTestQuestionSearchDO
                                                            preTestQuestionSearchDO ) throws CException
    {
        return preTestDAO.findPreTestQuestionsBySearchCriteria( preTestQuestionSearchDO );
    }

    public Collection findAllPreTestQuestions( ) throws CException
    {
        return preTestDAO.findAllPreTestQuestions( );
    }

    public Collection findAllPreTestResultCPs( ) throws CException
    {
        return preTestDAO.findAllPreTestResultCPs( );
    }

    public Collection findFilteredPreTestResultCPs( long userSysId, Collection selectedTopicDOs )
                                                    throws CException
    {
        return preTestDAO.findFilteredPreTestResultCPs( userSysId, selectedTopicDOs );
    }

    public PreTestAnswerDO findPreTestAnswerBySysId( long sysId ) throws CException
    {
        return preTestDAO.findPreTestAnswerBySysId( sysId );
    }

    public PreTestAnswerRecordDO findPreTestAnswerRecord( long preTestQuestionSysId,
                                                            long userSysId ) throws CException
    {
        return preTestDAO.findPreTestAnswerRecord( preTestQuestionSysId, userSysId );
    }

    public Collection findPreTestAnswerRecordDOByTopicSysId( long topicSysId ) throws CException
    {
        return preTestDAO.findPreTestAnswerRecordDOByTopicSysId( topicSysId );
    }

    public Collection findPreTestAnswerRecordDOByTopicSysIdUserSysId( long topicSysId,
                                                                        long userSysId ) throws CException
    {
        return preTestDAO.findPreTestAnswerRecordDOByTopicSysIdUserSysId( topicSysId,
                                                                              userSysId );
    }

```

```

public Collection findPreTestAnswerRecordsByUserSysId( long userSysId ) throws CException
{
    return preTestDAO.findPreTestAnswerRecordsByUserSysId( userSysId );
}

public Collection findPreTestAnswersByPreTestQuestionSysId( long sysId ) throws CException
{
    return preTestDAO.findPreTestAnswersByPreTestQuestionSysId( sysId );
}

public PreTestQuestionDO findPreTestQuestionBySysId( long sysId ) throws CException
{
    return preTestDAO.findPreTestQuestionBySysId( sysId );
}

public Collection findPreTestResultCPsByUserSysId( long userSysId ) throws CException
{
    return preTestDAO.findPreTestResultCPsByUserSysId( userSysId );
}

public long findTotalPreTestQuestions( ) throws CException
{
    return preTestDAO.findTotalPreTestQuestions( );
}
}

```

## Student

```
package com.its.module.student.service.sb;

import com.its.module.student.model.*;
import com.its.module.student.service.database.*;
import com.its.module.user.model.*;

import com.vagrant.base.exception.*;
import com.vagrant.j2ee.ejb.*;

import org.apache.commons.logging.*;
import org.apache.commons.logging.impl.*;

import java.util.*;
import javax.ejb.*;

public class StudentSBBean extends AbstractCommonSBBean
{
    private Log log = LogFactory.getLog( StudentSBBean.class.getName( ) );

    StudentDAO studentDAO = null;

    public StudentSBBean( ) throws EJBException
    {
        try
        {
            studentDAO = new StudentDAO( );
        }
        catch ( Exception ex )
        {
            throw new EJBException( ex );
        }
    }

    public long createStudent( StudentDO _DO ) throws CException
    {
        return studentDAO.createStudent( _DO );
    }

    public void createStudentAcquiredSkillsMap( StudentAcquiredSkillsMapDO _DO )
                                                throws CException
    {
        studentDAO.createStudentAcquiredSkillsMap( _DO );
    }

    public Collection findAcquiredSkillsSubTopicDOsByStudentSysId( long studentSysId )
                                                                    throws CException
    {
        return studentDAO.findAcquiredSkillsSubTopicDOsByStudentSysId( studentSysId );
    }

    public Collection findAllStudent( ) throws CException
    {
        return studentDAO.findAllStudent( );
    }
}
```

```

    public Collection findAllStudentWithCriteria( StudentSearchDO eStudentSearchDO )
                                           throws CException
    {
        return studentDAO.findAllStudentWithCriteria( eStudentSearchDO );
    }

    public StudentDO findStudentByStudentSysId( long sysId ) throws CException
    {
        return studentDAO.findStudentByStudentSysId( sysId );
    }

    public StudentDO findStudentByUserSysId( long userSysId ) throws CException
    {
        return studentDAO.findStudentByUserSysId( userSysId );
    }

    public StudentDO findStudentByUsername( String username ) throws CException
    {
        return studentDAO.findStudentByUsername( username );
    }

    public void removeStudent( StudentDO _DO ) throws CException
    {
        studentDAO.removeStudent( _DO );
    }

    public void updateStudent( StudentDO _DO ) throws CException
    {
        studentDAO.updateStudent( _DO );
    }
}

```

## Topic

```
package com.its.module.topic.service.sb;

import com.its.module.topic.model.*;
import com.its.module.topic.service.database.*;

import com.vagrant.base.exception.*;
import com.vagrant.j2ee.ejb.*;

import org.apache.commons.logging.*;
import org.apache.commons.logging.impl.*;

import java.util.*;
import javax.ejb.*;
\

public class TopicSBBean extends AbstractCommonSBBean
{
    private Log log = LogFactory.getLog( TopicSBBean.class.getName( ) );
    private TopicDAO topicDAO = null;

    public TopicSBBean( ) throws EJBException
    {
        try
        {
            topicDAO = new TopicDAO( );
        }
        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw new EJBException( ex );
        }
    }

    public Collection findAllSubTopicBySearchCriteria( SubTopicSearchDO subTopicSearchDO )
        throws CException
    {
        return topicDAO.findSubTopicBySearchCriteria( subTopicSearchDO );
    }

    public Collection findAllTopic( ) throws CException
    {
        return topicDAO.findAllTopics( );
    }

    public Collection findAllTopicBySearchCriteria( TopicSearchDO topicSearchDO )
        throws CException
    {
        return topicDAO.findTopicBySearchCriteria( topicSearchDO );
    }

    public SubTopicDO findSubTopicBySubTopicSysId( long subTopicSysId ) throws CException
    {
        return topicDAO.findSubTopicBySysId( subTopicSysId );
    }
}
```

```

public TopicDO findTopicByTopicSysId( long topicSysId ) throws CException
{
    return topicDAO.findTopicBySysId( topicSysId );
}

public long findTotalTopics( ) throws CException
{
    return topicDAO.findTotalTopics( );
}

public void removeSubTopic( long subTopicSysId ) throws CException
{
    SubTopicDO subTopicDO = topicDAO.findSubTopicBySysId( subTopicSysId );
    topicDAO.removeSubTopic( subTopicDO );
}

public void removeTopic( long topicSysId ) throws CException
{
    TopicDO topicDO = topicDAO.findTopicBySysId( topicSysId );
    topicDAO.removeTopic( topicDO );
}

public void updateSubTopic( SubTopicDO _DO ) throws CException
{
    topicDAO.updateSubTopic( _DO );
}

public void updateTopic( TopicDO _DO ) throws CException
{
    topicDAO.updateTopic( _DO );
}
}

```

## Tutor

```
package com.its.module.tutor.service.sb;

import com.its.module.tutor.model.*;
import com.its.module.tutor.service.database.*;
import com.its.module.user.model.*;
import com.its.module.user.service.database.*;

import com.vagrant.base.exception.*;
import com.vagrant.j2ee.ejb.*;

import org.apache.commons.logging.*;
import org.apache.commons.logging.impl.*;

import java.util.*;
import javax.ejb.*;

public class TutorSBBean extends AbstractCommonSBBean
{
    private Log log = LogFactory.getLog( TutorSBBean.class.getName( ));
    private TutorDAO tutorDAO = null;
    private UserDAO userDAO = null;

    public TutorSBBean( ) throws EJBException
    {
        try
        {
            tutorDAO = new TutorDAO( );
            userDAO = new UserDAO( );
        }
        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw new EJBException( ex );
        }
    }

    public long createTutor( TutorDO _DO ) throws CException
    {
        return tutorDAO.createTutor( _DO );
    }

    public Collection findAllTutor( ) throws CException
    {
        return tutorDAO.findAllTutor( );
    }

    public Collection findAllTutorWithCriteria( TutorSearchDO tutorSearchDO ) throws CException
    {
        return tutorDAO.findAllTutorWithCriteria( tutorSearchDO );
    }

    public long findTotalEStudents( ) throws CException
    {
        return tutorDAO.findTotalEStudents( );
    }
}
```

---

```

public long findTotalStudents( ) throws CException
{
    return tutorDAO.findTotalStudents( );
}

public TutorDO findTutorByTutorSysId( long sysId ) throws CException
{
    return tutorDAO.findTutorByTutorSysId( sysId );
}

public TutorDO findTutorByUsername( String username ) throws CException
{
    return tutorDAO.findTutorByUsername( username );
}

public Collection findUsersBySearchCriteria( UserSearchDO searchDO ) throws CException
{
    return userDAO.findUsersBySearchCriteria( searchDO );
}

public Collection findUsersByUserTypes( List userTypes ) throws CException
{
    return userDAO.findUsersByUserTypes( userTypes );
}

public void removeTutor( TutorDO _DO ) throws CException
{
    tutorDAO.removeTutor( _DO );
}

public void updateTutor( TutorDO _DO ) throws CException
{
    tutorDAO.updateTutor( _DO );
}
}

```



## Tutorial

```
package com.its.module.tutorial.service.sb;

import com.its.module.pretest.model.*;
import com.its.module.pretest.service.database.*;
import com.its.module.pretest.service.helper.*;
import com.its.module.student.model.*;
import com.its.module.tutorial.model.*;
import com.its.module.tutorial.service.database.*;
import com.its.module.tutorial.service.helper.*;

import com.vagrant.base.exception.*;
import com.vagrant.j2ee.ejb.*;

import org.apache.commons.lang.builder.*;
import org.apache.commons.logging.*;
import org.apache.commons.logging.impl.*;

import java.util.*;
import javax.ejb.*;

public class TutorialSBBean extends AbstractCommonSBBean
{
    private Log log = new Log4JLogger( TutorialSBBean.class.getName( ) );
    private TutorialDAO tutorialDAO = null;
    private PreTestDAO preTestDAO = null;
    private PreTestHelper preTestHelper = null;
    private TutorialHelper tutorialHelper = null;

    public TutorialSBBean( ) throws EJBException
    {
        try
        {
            tutorialDAO = new TutorialDAO( );
            preTestDAO = new PreTestDAO( );
            preTestHelper = new PreTestHelper( );
            tutorialHelper = new TutorialHelper( );
        }
        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            throw new EJBException( ex );
        }
    }

    public boolean isTokenExists( long studentSysId, long token ) throws CException
    {
        //return tutorialDAO.isTokenExists( studentSysId, token );
        return false;
    }
}
```

```

public long createTutorialTracker( TutorialTrackerDO _DO ) throws CException
{
    _DO.setCreatedTimestamp( new java.sql.Timestamp( System.currentTimeMillis( ) ) );

    return tutorialDAO.createTutorialTracker( _DO );
}

public Collection findAcquiredSkillsSubTopicsByTutorialSysId( long tutorialSysId )
                                throws CException
{
    return tutorialDAO.findAcquiredSkillsSubTopicsByTutorialSysId( tutorialSysId );
}

public Collection findAllPreTutorialStackByStudentSysId( long studentSysId ) throws CException
{
    return tutorialDAO.findAllPreTutorialStackByStudentSysId( studentSysId );
}

public Collection findAllTutorials( ) throws CException
{
    return tutorialDAO.findAllTutorials( );
}

public Collection findFilteredTutorialCPs( long userSysId, Collection selectedTopicDOs )
                                throws CException
{
    return tutorialDAO.findFilteredTutorialCPs( userSysId, selectedTopicDOs );
}

public Collection findHintsBySubTutorialSysId( long sysId ) throws CException
{
    return tutorialDAO.findHintsBySubTutorialSysId( sysId );
}

public String findInitialPath( StudentDO studentDO, Collection preRequisiteSubTopicDOs )
                                throws CException
{
    Collection preRequisiteTopicDOs = tutorialHelper.getTopicsFromSubTopics(
                                                preRequisiteSubTopicDOs );
    Collection tutorialCPDOs = tutorialDAO.findFilteredTutorialCPs( studentDO.getUserDO
                                                ( ).getSysId( ), preRequisiteTopicDOs );

    if ( ( tutorialCPDOs != null ) && ( tutorialCPDOs.size( ) > 0 ) )
    {
        Iterator itr = tutorialCPDOs.iterator( );

        log.debug( "=====");

        while ( itr.hasNext( ) )
        {
            TutorialCPDO _DO = ( TutorialCPDO ) itr.next( );
            log.debug( "Topic Name: " + _DO.getTopicDO( ).getTopicName( ) );
            log.debug( "CP: " + _DO.getCp( ) );
        }

        log.debug( "=====");
    }
}

```

```

        double cp = tutorialHelper.getMinCP( tutorialCPDOs );

        log.debug( "obtained cp: " + cp );

        //return the choice page / next tutorial page
        String path = tutorialHelper.getInitialTutorialPath( cp );

        return path;
    }

    public TutorialTrackerDO findLastTutorialTrackerByStudentSysId( long studentSysId )
                                                                    throws CException
    {
        return tutorialDAO.findLastTutorialTrackerByStudentSysId( studentSysId );
    }

    public Collection findPreRequisiteSubTopicsByTutorialSysId( long tutorialSysId )
                                                                    throws CException
    {
        return tutorialDAO.findPreRequisiteSubTopicsByTutorialSysId( tutorialSysId );
    }

    public Collection findPreTestQuestionBySubTutorialSysId( long sysId ) throws CException
    {
        return tutorialDAO.findPreTestQuestionBySubTutorialSysId( sysId );
    }

    public SubTutorialDO findSubTutorialBySysId( long sysId ) throws CException
    {
        return tutorialDAO.findSubTutorialBySysId( sysId );
    }

    public Collection findSubTutorialsByTutorialSysId( long sysId ) throws CException
    {
        return tutorialDAO.findSubTutorialsByTutorialSysId( sysId );
    }

    public long findTotalTutorialQuestions( ) throws CException
    {
        return tutorialDAO.findTotalTutorialQuestions( );
    }

    public TutorialDO findTutorialBySysId( long sysId ) throws CException
    {
        return tutorialDAO.findTutorialBySysId( sysId );
    }

    public TutorialDO findTutorialByTutorialId( String tutorialId ) throws CException
    {
        return tutorialDAO.findTutorialByTutorialId( tutorialId );
    }

    public Collection findTutorialCPsByUserSysId( long userSysId ) throws CException
    {
        return tutorialDAO.findTutorialCPsByUserSysId( userSysId );
    }

```

```

public Collection findTutorialTrackersByUserSysId( long userSysId ) throws CException
{
    return tutorialDAO.findTutorialTrackersByUserSysId( userSysId );
}

public Collection findTutorialsByTopicSysId( long sysId ) throws CException
{
    return tutorialDAO.findTutorialsByTopicSysId( sysId );
}

public Collection findTutorialsWithCriteria( TutorialSearchDO searchDO ) throws CException
{
    return tutorialDAO.findTutorialsWithCriteria( searchDO );
}

public long generateNewToken( long studentSysId ) throws CException
{
    HashCodeBuilder builder = new HashCodeBuilder( );
    builder.append( System.currentTimeMillis( ) );
    builder.append( Math.random( ) );
    builder.append( studentSysId );

    long token = builder.toHashCode( );

    //check if the token exists
    while ( isTokenExists( studentSysId, token ) )
    {
        builder = new HashCodeBuilder( );
        builder.append( System.currentTimeMillis( ) );
        builder.append( Math.random( ) );
        builder.append( studentSysId );
        token = builder.toHashCode( );
    }

    return token;
}
}

```

## Appendix H – Data Tables Descriptions

### List of Tables

NO	TABLE
1	acquired_skills_map
2	admin
3	demo
4	estudent
5	hint
6	its_user
7	login_log
8	post_test_answer
9	post_test_answer_record
10	post_test_question
11	post_test_result_condition_probability
12	pre_requisite_map
13	pre_test_answer
14	pre_test_answer_record
15	pre_test_question
16	pre_test_result_condition_probability
17	pre_tutorial_stack
18	selection
19	status
20	student
21	student_acquired_skills_map
22	sub_topic
23	sub_tutorial
24	sub_tutorial_answer
25	sub_tutorial_pre_test_map
26	topic
27	tutor
28	tutorial
29	tutorial_condition_probability
30	tutorial_route
31	tutorial_token
32	tutorial_tracker

## Descriptions of Tables

Table: acquired\_skills\_map

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
TUTORIAL_SYS_ID	bigint(20)		PRI	0	
SUB_TOPIC_SYS_ID	bigint(20)		PRI	0	

Table: admin

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
ITS_USER_SYS_ID	bigint(20)		UNI	0	
TITLE	varchar(50)	YES		NULL	

Table: demo

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SUB_TUTORIAL_SYS_ID	bigint(20)		PRI	0	
DEMO_ID	varchar(50)		UNI		
DEMO_EXPLANATION	text	YES		NULL	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	

Table: estudent

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
ITS_USER_SYS_ID	bigint(20)		UNI	0	
CLASS_CODE	varchar(10)				
PRE_TEST_COMPLETED	tinyint(1)			0	
POST_TEST_COMPLETED	tinyint(1)	YES		0	
TUTORIAL_COMPLETED	tinyint(1)			0	

Table: hint

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
SUB_TUTORIAL_SYS_ID	bigint(20)		MUL	0	
HINT	text				
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	

Table: its\_user

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
USERNAME	varchar(20)		UNI		
PASSWORD	varchar(255)				
NAME	varchar(255)				
E_MAIL	varchar(30)				
DEPARTMENT	varchar(100)	YES		NULL	
TYPE	varchar(10)				
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)	YES	MUL	NULL	

Table: login\_log

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
ITS_USER_SYS_ID	bigint(20)		MUL	0	
SESSION_ID	varchar(50)				
LAST_IP	varchar(15)				
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	

Table: post\_test\_answer

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
QUESTION_SYS_ID	bigint(20)		MUL	0	
ANSWER_TEXT	varchar(255)				
CORRECT	tinyint(4)			0	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	

Table: post\_test\_answer\_record

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
ITS_USER_SYS_ID	bigint(20)		MUL	0	
POST_TEST_QUESTION_SYS_ID	bigint(20)		MUL	0	
SELECTED_POST_TEST_ANSWER_SYS_ID	bigint(20)	YES	MUL	NULL	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	

Table: post\_test\_question

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
QUESTION_ID	varchar(50)		UNI		
QUESTION_TITLE	varchar(255)				
QUESTION_TEXT	text				
SUB_TOPIC_SYS_ID	bigint(20)		MUL	0	
EXPLANATION	text	YES		NULL	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	



Table: post\_test\_result\_condition\_probability

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
ITS_USER_SYS_ID	bigint(20)		MUL	0	
TOPIC_SYS_ID	bigint(20)		MUL	0	
TOTAL_QUESTIONS	int(11)			0	
TOTAL_CORRECT	int(11)			0	
TOTAL_WRONG	int(11)			0	
CP	decimal(10,2)			0.00	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	

Table: pre\_requisite\_map

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
TUTORIAL_SYS_ID	bigint(20)		PRI	0	
SUB_TOPIC_SYS_ID	bigint(20)		PRI	0	

Table: pre\_test\_answer

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
QUESTION_SYS_ID	bigint(20)		MUL	0	
ANSWER_TEXT	varchar(255)				
CORRECT	tinyint(4)			0	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	

Table: pre\_test\_answer\_record

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
ITS_USER_SYS_ID	bigint(20)		MUL	0	
PRE_TEST_QUESTION_SYS_ID	bigint(20)		MUL	0	
SELECTED_PRE_TEST_ANSWER_SYS_ID	bigint(20)	YES	MUL	NULL	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	

Table: pre\_test\_question

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
QUESTION_ID	varchar(50)		UNI		
QUESTION_TITLE	varchar(255)				
QUESTION_TEXT	text				
SUB_TOPIC_SYS_ID	bigint(20)		MUL	0	
EXPLANATION	text	YES		NULL	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	

Table: pre\_test\_result\_condition\_probability

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
ITS_USER_SYS_ID	bigint(20)		MUL	0	
TOPIC_SYS_ID	bigint(20)		MUL	0	
TOTAL_QUESTIONS	int(11)			0	
TOTAL_CORRECT	int(11)			0	
TOTAL_WRONG	int(11)			0	
CP	decimal(10,2)			0.00	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	

Table: pre\_tutorial\_stack

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
STUDENT_SYS_ID	bigint(20)		MUL	0	
PRE_TUTORIAL_SYS_ID	bigint(20)		MUL	0	
PARENT_SUB_TUTORIAL_SYS_ID	bigint(20)			0	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	

Table: selection

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	0	
SUB_TUTORIAL_SYS_ID	bigint(20)		MUL	0	
SELECTION_TEXT	varchar(55)				
CORRECT	tinyint(4)			0	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	

Table: status

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
STATUS_TYPE	varchar(50)		UNI		
STATUS_DESC	varchar(255)				

Table: student

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
ITS_USER_SYS_ID	bigint(20)		UNI	0	
CLASS_CODE	varchar(10)				
PRE_TEST_COMPLETED	tinyint(1)			0	
TUTORIAL_COMPLETED	tinyint(1)			0	
POST_TEST_COMPLETED	tinyint(1)	YES		0	
CURRENT_TUTORIAL_TOKEN	bigint(20)	YES		NULL	

Table: student\_acquired\_skills\_map

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
STUDENT_SYS_ID	bigint(20)		PRI	0	
SUB_TOPIC_SYS_ID	bigint(20)		PRI	0	

Table: sub\_topic

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
SUB_TOPIC_ID	varchar(50)		UNI		
SUB_TOPIC_NAME	varchar(255)				
TOPIC_SYS_ID	bigint(20)			0	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	

Table: sub\_tutorial

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
SUB_TUTORIAL_ID	varchar(50)		UNI		
SUB_TUTORIAL_TITLE	varchar(255)				
SUB_TUTORIAL_QUESTION_TEXT	text				
TUTORIAL_SYS_ID	bigint(20)		MUL	0	
PRE_TUTORIAL_SYS_ID	bigint(20)	YES		NULL	
SEQUENCE	int(11)			0	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	

Table: sub\_tutorial\_answer

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SUB_TUTORIAL_SYS_ID	bigint(20)		PRI	0	
SYNTAX_STRING	text				
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	

Table: sub\_tutorial\_pre\_test\_map

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SUB_TUTORIAL_SYS_ID	bigint(20)		PRI	0	
PRE_TEST_QUESTION_SYS_ID	bigint(20)		PRI	0	

Table: topic

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
TOPIC_ID	varchar(50)		UNI		
TOPIC_NAME	varchar(255)				
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	

Table: tutor

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
ITS_USER_SYS_ID	bigint(20)		UNI	0	
TITLE	varchar(50)	YES		NULL	

Table: tutorial

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
TUTORIAL_ID	varchar(50)		UNI		
TUTORIAL_TITLE	varchar(255)				
TUTORIAL_QUESTION_TEXT	text				
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
LAST_MODIFIED_TIMESTAMP	datetime			0000-00-00 00:00:00	
STATUS_SYS_ID	bigint(20)		MUL	0	

Table: tutorial\_condition\_probability

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
ITS_USER_SYS_ID	bigint(20)		MUL	0	
TOPIC_SYS_ID	bigint(20)		MUL	0	
TOTAL_QUESTIONS	int(11)			0	
TOTAL_CORRECT	int(11)			0	
TOTAL_WRONG	int(11)			0	
CP	decimal(10,2)			0.00	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	

Table: tutorial\_route

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
ROUTE_NAME	varchar(200)				

Table: tutorial\_token

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
TUTORIAL_SYS_ID	bigint(20)		PRI	0	
TOKEN	bigint(20)		PRI	0	

Table: tutorial\_tracker

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
SYS_ID	bigint(20)		PRI	NULL	auto_increment
STUDENT_SYS_ID	bigint(20)		MUL	0	
TUTORIAL_SYS_ID	bigint(20)		MUL	0	
SUB_TUTORIAL_SYS_ID	bigint(20)		MUL	0	
TUTORIAL_ROUTE_SYS_ID	bigint(20)		MUL	0	
ANSWER	text	YES		NULL	
CORRECT	tinyint(1)	YES		NULL	
CREATED_TIMESTAMP	datetime			0000-00-00 00:00:00	
DURATION	bigint(20)			0	

## Appendix I

### XML Syntax Parser Code

```
package com.its.module.tutorial.service.helper;

import com.vagrant.base.exception.*;
import com.vagrant.base.helper.*;

import org.apache.commons.logging.*;
import org.apache.commons.logging.impl.*;
import org.w3c.dom.*;
import org.xml.sax.*;

import java.io.*;
import java.util.*;

import javax.xml.parsers.*;

public class TutorialAnswerSyntaxParser extends AbstractHelper
{
    private Log log = new Log4JLogger( TutorialAnswerSyntaxParser.class.getName( ) );
    private DocumentBuilder builder = null;

    /**
     * Creates a new instance of TutorialAnswerSyntaxParser
     *
     * @throws CException DOCUMENT ME!
     */
    public TutorialAnswerSyntaxParser( ) throws CException
    {
        try
        {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance( );
            factory.setValidating( true );
            factory.setNamespaceAware( true );
            factory.setIgnoringComments( true );
            factory.setIgnoringElementContentWhitespace( false );
            builder = factory.newDocumentBuilder( );
        }
        catch ( Exception ex )
        {
            log.error( ex.getMessage( ), ex );
            getExceptionHandler( ).generateError( "ERSYS-002", ex );
        }
    }
}
```

```

public boolean isAnswerCorrect( String syntax, String answer )
{
    if ( ( answer != null ) && ( syntax != null ) )
    {
        return validateSubTutorialAnswer( syntax, answer );
    }
    else
    {
        return false;
    }
}

public boolean validateSubTutorialAnswer( String xmlSyntax, String answer )
{
    try
    {
        StringBuffer xml = new StringBuffer( "<?xml version=\"1.0\"
                                                encoding=\"UTF-8\"?>" );
        xml.append( "<!DOCTYPE syntax PUBLIC \"-//ITS 1.0//EN\"
                    \"http://localhost:8080/its/itssyntax.dtd\">" );

        //xml.append("<!DOCTYPE syntax SYSTEM file:///C:/ITS/itssyntax.dtd>");
        xml.append( xmlSyntax );

        log.debug( "=====");
        log.debug( "Validating Answer Syntax" );
        log.debug( "=====");
        log.debug( xml.toString( ) );
        log.debug( "=====");

        InputSource source = new InputSource( new StringReader( xml.toString( ) ) );
        Document doc = builder.parse( source );

        Element rootElement = doc.getDocumentElement( );
        List itsSyntaxModels = buildITSSyntaxModels( rootElement );

        return validateITSAnswerSyntax( itsSyntaxModels, answer, false );
    }
    catch ( Exception ex )
    {
        log.error( ex.getMessage( ), ex );
        return false;
    }
}

```



```

public boolean validateSyntax( String xmlSyntax )
{
    try
    {
        StringBuffer xml = new StringBuffer( "<?xml version=\"1.0\"
                                                encoding=\"UTF-8\"?>" );
        xml.append( "<!DOCTYPE syntax PUBLIC \"-//ITS 1.0//EN\"
                                                            \"http://localhost:8080/its/itssyntax.dtd\">" );

        //xml.append("<!DOCTYPE syntax SYSTEM file:///C:/ITS/itssyntax.dtd>");
        xml.append( xmlSyntax );

        log.debug( "=====");
        log.debug( "Validating Answer Syntax" );
        log.debug( "=====");
        log.debug( xml.toString( ) );
        log.debug( "=====");

        InputSource source = new InputSource( new StringReader( xml.toString( ) ) );
        builder.parse( source );
        return true;
    }
    catch ( Exception ex )
    {
        log.error( ex.getMessage( ), ex );
        return false;
    }
}

private List buildITSSyntaxModels( Element element )
{
    ArrayList itsSyntaxModels = new ArrayList( );
    NodeList nodeList = element.getElementsByTagName( "eval" );

    log.debug( "nodelist size: " + nodeList.getLength( ) );

    for ( int i = 0; i < nodeList.getLength( ); i++ )
    {
        TutorialAnswerSyntaxParser.ITSSyntaxModel model = new
            TutorialAnswerSyntaxParser.ITSSyntaxModel( );
        Node node = nodeList.item( i );
        Element ele = ( Element ) node;

        if ( ( ele.getAttribute( "join" ) != null ) &&
            ele.getAttribute( "join" ).equalsIgnoreCase( "or" ) )
        {
            log.debug( "join spotted" );
            model.getOrValueList( ).addAll( buildITSSyntaxModels( ele ) );
        }
        else
        {
            //If more than one, then do recursive parsing
            if ( ele.getChildNodes( ).getLength( ) > 1 )
            {
                log.debug( "child nodes spotted" );
                model.getAndValueList( ).addAll( buildITSSyntaxModels(ele));
            }
            else
            {

```

```

        if ( ( ele.getAttribute( "ignorecase" ) != null )
            && ele.getAttribute( "ignorecase" ).equalsIgnoreCase( "true" ) )
        {
            log.debug( "ignore case spotted" );
            model.setIgnoreCase( true );
        }

        if ( ( ele.getAttribute( "allow" ) != null ) &&
            !ele.getAttribute( "allow" ).trim( ).equals( "" ) )
        {
            log.debug( "allow spotted" );
            log.debug( "allow type: " +
                ele.getAttribute( "allow" ).trim( ) );
            model.setAllow( ele.getAttribute( "allow" ) );
        }

        Text value = ( Text ) ele.getFirstChild( );
        model.setValue( value.getData( ) );

        log.debug( "Value: " + value.getData( ) );
    }

    itsSyntaxModels.add( model );
}

return itsSyntaxModels;
}

private boolean validateITSAnswerSyntax( List itsSyntaxModels, String answer,
                                         boolean orEnabled )
{
    log.debug( "validating answer..." );
    log.debug( "answer: " + answer );

    if ( ( itsSyntaxModels != null ) && ( itsSyntaxModels.size( ) > 0 ) )
    {
        boolean result = true;
        Iterator itr = itsSyntaxModels.iterator( );

        while ( itr.hasNext( ) )
        {
            TutorialAnswerSyntaxParser.ITSSyntaxModel model =
                ( TutorialAnswerSyntaxParser.ITSSyntaxModel ) itr.next( );

            if ( model.getAndValueList( ).size( ) > 0 )
            {
                boolean tmpResult = validateITSAnswerSyntax
                    ( model.getAndValueList( ), answer, false );
                if ( orEnabled )
                {
                    result |= tmpResult;
                }
                else
                {
                    result &= tmpResult;
                }
            }
        }
    }
}

```

```

    }
    else if ( model.getOrValueList( ).size( ) > 0 )
    {
        boolean tmpResult = validateITSAnswerSyntax
            ( model.getOrValueList( ), answer, true );
        if ( orEnabled )
        {
            result |= tmpResult;
        }
        else
        {
            result &= tmpResult;
        }
    }
    else
    {
        String compareValue = model.getValue( );
        log.debug( "compareValue: " + compareValue );
        if ( ( model.getAllow( ) != null ) &&
            model.getAllow( ).equals( "number" ) )
        {
            log.debug( "must be a number" );
            if ( ( compareValue != null ) &&
                !compareValue.equals( "" ) )
            {
                int index = -1;
                log.debug( "index: " + answer.indexOf
                    ( compareValue ) );
                if ( ( index =
                    answer.indexOf( compareValue ) ) != -1 )
                {
                    log.debug( "old answer: " + answer );
                    answer = answer.substring( index +
                        compareValue.length( ), answer.length( ) );
                    log.debug( "new answer: " +
                        answer );
                }
                else
                {
                    log.debug( "validation failed" );
                    result &= false;
                }
            }
        }
        else if ( ( model.getAllow( ) != null ) &&
            model.getAllow().equals( "string" ) )
        {
            log.debug( "must be a string" );
            if ( ( compareValue != null ) &&
                !compareValue.equals( "" ) )
            {
                int index = -1;
                log.debug( "index: " +
                    answer.indexOf( compareValue ) );

```

```

        if ( ( index =
            answer.indexOf( compareValue ) ) != -1 )
        {
            log.debug( "old answer: " + answer );
            answer = answer.substring( index +
                compareValue.length( ), answer.length( ) );
            log.debug( "new answer: " +
                answer );
        }
        else
        {
            log.debug( "validation failed" );
            result &= false;
        }
    }
}
else
{
    log.debug( "other free string" );
    if ( ( compareValue != null ) &&
        !compareValue.equals( "" ) )
    {
        int index = -1;
        log.debug( "index: " +
            answer.indexOf( compareValue ) );

        if ( ( index =
            answer.indexOf( compareValue ) ) != -1 )
        {
            log.debug( "old answer: " + answer );
            answer = answer.substring( index +
                ompareValue.length( ), answer.length( ) );
            log.debug( "new answer: " +
                answer );
        }
        else
        {
            log.debug( "validation failed" );
            result &= false;
        }
    }
}
}
return result;
}
else
{
    return false;
}
}

```

```

private class ITSSyntaxModel
{
    private String value = "";
    private String allow = "";
    private boolean ignoreCase = false;
    private ArrayList andValueList = new ArrayList( );
    private ArrayList orValueList = new ArrayList( );

    /**
     * Setter for property allow.
     *
     * @param allow New value of property allow.
     */
    public void setAllow( java.lang.String allow )
    {
        this.allow = allow;
    }

    /**
     * Getter for property allow.
     *
     * @return Value of property allow.
     */
    public java.lang.String getAllow( )
    {
        return allow;
    }

    /**
     * Setter for property andValueList.
     *
     * @param andValueList New value of property andValueList.
     */
    public void setAndValueList( java.util.ArrayList andValueList )
    {
        this.andValueList = andValueList;
    }

    /**
     * Getter for property andValueList.
     *
     * @return Value of property andValueList.
     */
    public java.util.ArrayList getAndValueList( )
    {
        return andValueList;
    }

    /**
     * Setter for property ignoreCase.
     *
     * @param ignoreCase New value of property ignoreCase.
     */
    public void setIgnoreCase( boolean ignoreCase )
    {
        this.ignoreCase = ignoreCase;
    }
}

```

```

/**
 * Getter for property isIgnoreCase.
 *
 * @return Value of property isIgnoreCase.
 */
public boolean isIgnoreCase( )
{
    return ignoreCase;
}

/**
 * Setter for property orValueList.
 *
 * @param orValueList New value of property orValueList.
 */
public void setOrValueList( java.util.ArrayList orValueList )
{
    this.orValueList = orValueList;
}

/**
 * Getter for property orValueList.
 *
 * @return Value of property orValueList.
 */
public java.util.ArrayList getOrValueList( )
{
    return orValueList;
}

/**
 * Setter for property value.
 *
 * @param value New value of property value.
 */
public void setValue( java.lang.String value )
{
    this.value = value;
}

/**
 * Getter for property value.
 *
 * @return Value of property value.
 */
public java.lang.String getValue( )
{
    return value;
}
}
}

```

## Student Model Update Code

```
package com.its.module.tutorial.service.sb;

import com.its.intf.*;

import com.its.module.pretest.model.*;
import com.its.module.pretest.service.database.*;
import com.its.module.pretest.service.helper.*;
import com.its.module.status.model.*;
import com.its.module.status.service.database.*;
import com.its.module.student.model.*;
import com.its.module.student.service.database.*;
import com.its.module.topic.model.*;
import com.its.module.topic.service.database.*;
import com.its.module.tutorial.intf.*;
import com.its.module.tutorial.model.*;
import com.its.module.tutorial.service.database.*;
import com.its.module.tutorial.service.helper.*;

import com.its.web.intf.*;

import com.vagrant.base.exception.*;

import com.vagrant.j2ee.ejb.*;

import org.apache.commons.logging.*;
import org.apache.commons.logging.impl.*;

import java.util.*;

import javax.ejb.*;

/**
 * TutorialTrxnSBBean
 *
 * @version 1.0
 */
public class TutorialTrxnSBBean extends AbstractTransactionSBBean
{
    private Log log = new Log4JLogger( TutorialTrxnSBBean.class.getName( ) );
    private TutorialDAO tutorialDAO = null;
    private StatusDAO statusDAO = null;
    private StudentDAO studentDAO = null;
    private PreTestDAO preTestDAO = null;
    private TutorialHelper tutorialHelper = null;
    private TutorialAnswerSyntaxParser answerSyntaxParser = null;
    private PreTestHelper preTestHelper = null;

    /**
     * Creates a new instance of TutorialTrxnSBBean
     *
     * @throws EJBException
     */
}
```

```

public TutorialTrxnSBBean( ) throws EJBException
{
    try
    {
        preTestDAO = new PreTestDAO( );
        tutorialDAO = new TutorialDAO( );
        statusDAO = new StatusDAO( );
        tutorialHelper = new TutorialHelper( );
        preTestHelper = new PreTestHelper( );
        answerSyntaxParser = new TutorialAnswerSyntaxParser( );
    }
    catch ( CException ex )
    {
        throw new EJBException( ex );
    }
}

/**
 * createBatchSubTutorialPreTestMap
 *
 * @param subTutorialPreTestMapDOs
 *
 * @throws CException
 */
public void createBatchSubTutorialPreTestMap( Collection subTutorialPreTestMapDOs )
                                                    throws CException
{
    try
    {
        beginTrxn( );

        tutorialDAO.createBatchSubTutorialPreTestMap( subTutorialPreTestMapDOs );

        commitTrxn( );
    }
    catch ( CException ex )
    {
        rollbackTrxnOnly( );

        log.error( ex.getMessage( ), ex );
        throw ex;
    }
    catch ( Exception ex )
    {
        rollbackTrxnOnly( );

        log.error( ex.getMessage( ), ex );
        throw getExceptionHandler( ).generateError( "ERSYS-001" );
    }
}

```



```

/**
 * createBatchTutorialCPscreateBatchTutorialCPs
 *
 * @param tutorialCPDOs
 *
 * @throws CException
 */
public void createBatchTutorialCPs( Collection tutorialCPDOs ) throws CException
{
    try
    {
        beginTrxn( );

        java.sql.Timestamp now = new java.sql.Timestamp( System.currentTimeMillis() );

        if ( ( tutorialCPDOs != null ) && ( tutorialCPDOs.size( ) > 0 ) )
        {
            Iterator itr = tutorialCPDOs.iterator( );

            while ( itr.hasNext( ) )
            {
                TutorialCPDO _DO = ( TutorialCPDO ) itr.next( );

                _DO.setCreatedTimestamp( now );
                _DO.setLastModifiedTimestamp( now );

                tutorialDAO.createTutorialCP( _DO );
            }

            commitTrxn( );
        }
        catch ( CException ex )
        {
            rollbackTrxnOnly( );

            log.error( ex.getMessage( ), ex );
            throw ex;
        }
        catch ( Exception ex )
        {
            rollbackTrxnOnly( );

            log.error( ex.getMessage( ), ex );
            throw getExceptionHandler( ).generateError( "ERSYS-001" );
        }
    }
}

```

```

/**
 * createTutorial
 *
 * @param tutorialDO
 * @param subTutorialDOs
 *
 * @return log
 *
 * @throws CException
 */
public long createTutorial( TutorialDO tutorialDO, Collection subTutorialDOs ) throws CException
{
    long pk = 0;

    try
    {
        beginTrxn( );

        java.sql.Timestamp now = new java.sql.Timestamp(System.currentTimeMillis( ));
        StatusDO statusDO = statusDAO.findStatusByStatusType( IStatus.ACTIVE );

        tutorialDO.setStatusDO( statusDO );
        tutorialDO.setCreatedTimestamp( now );
        tutorialDO.setLastModifiedTimestamp( now );

        pk = tutorialDAO.createTutorial( tutorialDO );
        tutorialDO = tutorialDAO.findTutorialBySysId( pk );

        if ( ( subTutorialDOs != null ) && ( subTutorialDOs.size( ) > 0 ) )
        {
            Iterator itr = subTutorialDOs.iterator( );

            while ( itr.hasNext( ) )
            {
                SubTutorialDO _DO = ( SubTutorialDO ) itr.next( );
                _DO.setTutorialDO( tutorialDO );

                _DO.setStatusDO( statusDO );
                _DO.setCreatedTimestamp( now );
                _DO.setLastModifiedTimestamp( now );

                long subTutorialSysId = tutorialDAO.createSubTutorial( _DO );

                //Create Sub Tutorial Answer
                StringBuffer syntaxString = new StringBuffer( "<syntax>\r\n" );
                syntaxString.append( "    <eval/>\r\n" );
                syntaxString.append( "</syntax>\r\n" );

                SubTutorialAnswerDO answerDO =
                    new SubTutorialAnswerDO( );
                answerDO.setSysId( subTutorialSysId );
                answerDO.setStatusDO( statusDO );
                answerDO.setSubTutorialDO( _DO );
                answerDO.setSyntaxString( syntaxString.toString( ) );
                answerDO.setCreatedTimestamp( now );
                answerDO.setLastModifiedTimestamp( now );

                tutorialDAO.createSubTutorialAnswer( answerDO );
            }
        }
    }
}

```

```

        //Create Demo
        DemoDO demoDO = new DemoDO( );
        demoDO.setSysId( subTutorialSysId );
        demoDO.setSubTutorialDO( _DO );
        demoDO.setDemoId( _DO.getSubTutorialId( ) );
        demoDO.setDemoExplanation( "" );
        demoDO.setStatusDO( statusDO );
        demoDO.setCreatedTimestamp( now );
        demoDO.setLastModifiedTimestamp( now );

        tutorialDAO.createDemo( demoDO );

    }

    }

    commitTrxn( );
}
catch ( CException ex )
{
    rollbackTrxnOnly( );

    log.error( ex.getMessage( ), ex );
    throw ex;
}
catch ( Exception ex )
{
    rollbackTrxnOnly( );

    log.error( ex.getMessage( ), ex );
    throw getExceptionHandler( ).generateError( "ERSYS-001" );
}

return pk;
}

/**
 * deleteBatchSubTutorialPreTestMap
 *
 * @param subTutorialPreTestMapDOs
 *
 * @throws CException
 */
public void deleteBatchSubTutorialPreTestMap( Collection subTutorialPreTestMapDOs )
    throws CException
{
    try
    {
        tutorialDAO.deleteBatchSubTutorialPreTestMap( subTutorialPreTestMapDOs );
    }
    catch ( CException ex )
    {
        rollbackTrxnOnly( );

        log.error( ex.getMessage( ), ex );
        throw ex;
    }
}

```

```

        catch ( Exception ex )
        {
            rollbackTrxnOnly( );

            log.error( ex.getMessage( ), ex );
            throw getExceptionHandler( ).generateError( "ERSYS-001" );
        }
    }

    /**
     * processFromBriefExplanationRoute
     *
     * @param parameters
     *
     * @return HashMap
     *
     * @throws CException
     */
    public HashMap processFromBriefExplanationRoute( HashMap parameters ) throws CException
    {
        HashMap result = new HashMap( );

        try
        {
            beginTrxn( );

            TutorialDO currentTutorialDO = ( TutorialDO )
                parameters.get( IWebConst.CURRENT_TUTORIAL );
            SubTutorialDO selectedSubTutorialDO = ( SubTutorialDO )
                parameters.get( IWebConst.SELECTED_SUB_TUTORIAL );
            Collection currentSubTutorialDOs = ( Collection )
                parameters.get( IWebConst.CURRENT_SUB_TUTORIALS );
            Collection allTutorialDOs = ( Collection )
                parameters.get( IWebConst.ALL_TUTORIALS );
            StudentDO studentDO = ( StudentDO )
                parameters.get( IWebConst.STUDENT_DO );
            long startTimeMillis = ( ( Long )
                parameters.get( IWebConst.START_TIME_MILLIS ) ).longValue( );
            long duration = System.currentTimeMillis( ) - startTimeMillis;

            //Create the new next tutorial tracker
            RouteDO routeDO = tutorialDAO.findRouteByRouteName(
                IWebConst.ROUTE_FROM_BRIEF_EXPLANATION );

            TutorialTrackerDO tutorialTrackerDO = new TutorialTrackerDO( );
            tutorialTrackerDO.setCreatedTimestamp( new java.sql.Timestamp(
                System.currentTimeMillis( ) ) );
            tutorialTrackerDO.setStudentDO( studentDO );
            tutorialTrackerDO.setSubTutorialDO( selectedSubTutorialDO );
            tutorialTrackerDO.setTutorialDO( currentTutorialDO );
            tutorialTrackerDO.setRouteDO( routeDO );
            tutorialTrackerDO.setDuration( duration );

            tutorialDAO.createTutorialTracker( tutorialTrackerDO );

            result.put( IWebConst.CURRENT_TUTORIAL, currentTutorialDO );
            result.put( IWebConst.CURRENT_SUB_TUTORIALS, currentSubTutorialDOs );
            result.put( IWebConst.SELECTED_SUB_TUTORIAL, selectedSubTutorialDO );

```

```

        result.put( IWebConst.DEST_PATH,
                    ITutorialPath.TP_DISPLAY_PROBLEM_SPEC );

        commitTrxn( );
    }
    catch ( CException ex )
    {
        rollbackTrxnOnly( );

        log.error( ex.getMessage( ), ex );
        throw ex;
    }
    catch ( Exception ex )
    {
        rollbackTrxnOnly( );

        log.error( ex.getMessage( ), ex );
        throw getExceptionHandler( ).generateError( "ERSYS-001" );
    }

    return result;
}

/**
 * processFromDemoRoute
 *
 * @param parameters
 *
 * @return HashMap
 *
 * @throws CException
 */
public HashMap processFromDemoRoute( HashMap parameters ) throws CException
{
    HashMap result = new HashMap( );

    try
    {
        beginTrxn( );

        TutorialDO currentTutorialDO = ( TutorialDO )
            parameters.get( IWebConst.CURRENT_TUTORIAL );
        SubTutorialDO selectedSubTutorialDO = ( SubTutorialDO )
            parameters.get( IWebConst.SELECTED_SUB_TUTORIAL );
        Collection currentSubTutorialDOs = ( Collection )
            parameters.get( IWebConst.CURRENT_SUB_TUTORIALS );
        Collection allTutorialDOs = ( Collection )
            parameters.get( IWebConst.ALL_TUTORIALS );
        StudentDO studentDO = ( StudentDO )
            parameters.get( IWebConst.STUDENT_DO );
        long startTimeMillis = ( Long )
            parameters.get( IWebConst.START_TIME_MILLIS ).longValue( );
        long duration = System.currentTimeMillis( ) - startTimeMillis;

        //Create the new next tutorial tracker
        RouteDO routeDO = tutorialDAO.findRouteByRouteName(
            IWebConst.ROUTE_FROM_DEMO );
    }
    catch ( CException ex )
    {
        rollbackTrxnOnly( );
        log.error( ex.getMessage( ), ex );
        throw ex;
    }

    return result;
}

```

```

        TutorialTrackerDO tutorialTrackerDO = new TutorialTrackerDO( );
        tutorialTrackerDO.setCreatedTimestamp( new java.sql.Timestamp(
            System.currentTimeMillis( ) ) );
        tutorialTrackerDO.setStudentDO( studentDO );
        tutorialTrackerDO.setSubTutorialDO( selectedSubTutorialDO );
        tutorialTrackerDO.setTutorialDO( currentTutorialDO );
        tutorialTrackerDO.setRouteDO( routeDO );
        tutorialTrackerDO.setDuration( duration );

        tutorialDAO.createTutorialTracker( tutorialTrackerDO );

        result.put( IWebConst.CURRENT_TUTORIAL, currentTutorialDO );
        result.put( IWebConst.CURRENT_SUB_TUTORIALS, currentSubTutorialDOs );
        result.put( IWebConst.SELECTED_SUB_TUTORIAL, selectedSubTutorialDO );
        result.put( IWebConst.DEST_PATH,
            ITutorialPath.TP_DISPLAY_PROBLEM_SPEC );

        commitTrxn( );
    }
    catch ( CException ex )
    {
        rollbackTrxnOnly( );

        log.error( ex.getMessage( ), ex );
        throw ex;
    }
    catch ( Exception ex )
    {
        rollbackTrxnOnly( );

        log.error( ex.getMessage( ), ex );
        throw getExceptionHandler( ).generateError( "ERSYS-001" );
    }

    return result;
}

/**
 * processFromPreTestReviewRoute
 *
 * @param parameters
 *
 * @return HashMap
 *
 * @throws CException
 */
public HashMap processFromPreTestReviewRoute( HashMap parameters ) throws CException
{
    HashMap result = new HashMap( );

    try
    {
        beginTrxn( );

        TutorialDO currentTutorialDO = ( TutorialDO )
            parameters.get( IWebConst.CURRENT_TUTORIAL );
        SubTutorialDO selectedSubTutorialDO = ( SubTutorialDO )
            parameters.get( IWebConst.SELECTED_SUB_TUTORIAL );

```

```

        Collection currentSubTutorialDOs = ( Collection )
            parameters.get( IWebConst.CURRENT_SUB_TUTORIALS );
        Collection allTutorialDOs = ( Collection )
            parameters.get( IWebConst.ALL_TUTORIALS );
        StudentDO studentDO = ( StudentDO )
            parameters.get( IWebConst.STUDENT_DO );
        long startTimeMillis = ( ( Long ) parameters.get(
            IWebConst.START_TIME_MILLIS ) ).longValue( );
        long duration = System.currentTimeMillis( ) - startTimeMillis;

        //Create the new next tutorial tracker
        RouteDO routeDO = tutorialDAO.findRouteByRouteName(
            IWebConst.ROUTE_FROM_PRE_TEST_REVIEW );

        TutorialTrackerDO tutorialTrackerDO = new TutorialTrackerDO( );
        tutorialTrackerDO.setCreatedTimestamp( new java.sql.Timestamp(
            System.currentTimeMillis( ) ) );
        tutorialTrackerDO.setStudentDO( studentDO );
        tutorialTrackerDO.setSubTutorialDO( selectedSubTutorialDO );
        tutorialTrackerDO.setTutorialDO( currentTutorialDO );
        tutorialTrackerDO.setRouteDO( routeDO );
        tutorialTrackerDO.setDuration( duration );

        tutorialDAO.createTutorialTracker( tutorialTrackerDO );

        result.put( IWebConst.CURRENT_TUTORIAL, currentTutorialDO );
        result.put( IWebConst.CURRENT_SUB_TUTORIALS, currentSubTutorialDOs );
        result.put( IWebConst.SELECTED_SUB_TUTORIAL, selectedSubTutorialDO );
        result.put( IWebConst.DEST_PATH,
            ITutorialPath.TP_DISPLAY_PROBLEM_SPEC );

        commitTrxn( );
    }
    catch ( CException ex )
    {
        rollbackTrxnOnly( );

        log.error( ex.getMessage( ), ex );
        throw ex;
    }
    catch ( Exception ex )
    {
        rollbackTrxnOnly( );

        log.error( ex.getMessage( ), ex );
        throw getExceptionHandler( ).generateError( "ERSYS-001" );
    }

    return result;
}

```

```

/**
 * processFromRemedialRoute
 *
 * @param parameters
 *
 * @return HashMap
 *
 * @throws CException
 */
public HashMap processFromRemedialRoute( HashMap parameters ) throws CException
{
    HashMap result = new HashMap( );

    try
    {
        beginTrxn( );

        TutorialDO currentTutorialDO = ( TutorialDO )
            parameters.get( IWebConst.CURRENT_TUTORIAL );
        SubTutorialDO selectedSubTutorialDO = ( SubTutorialDO )
            parameters.get( IWebConst.SELECTED_SUB_TUTORIAL );
        Collection currentSubTutorialDOs = ( Collection )
            parameters.get( IWebConst.CURRENT_SUB_TUTORIALS );
        Collection allTutorialDOs = ( Collection )
            parameters.get( IWebConst.ALL_TUTORIALS );
        StudentDO studentDO = ( StudentDO )
            parameters.get( IWebConst.STUDENT_DO );
        long startTimeMillis = ( Long )
            parameters.get( IWebConst.START_TIME_MILLIS ).longValue( );
        long duration = System.currentTimeMillis( ) - startTimeMillis;

        //obtain current sub-tutorial pre-requisites
        Collection preRequisiteSubTopicDOs =
            tutorialDAO.findPreRequisiteSubTopicsByTutorialSysId(
                currentTutorialDO.getSysId( ) );
        Collection preRequisiteTopicDOs = tutorialHelper.getTopicsFromSubTopics(
            preRequisiteSubTopicDOs );

        Collection tutorialCPDOs = tutorialDAO.findFilteredTutorialCPs(
            studentDO.getUserDO( ).getSysId( ), preRequisiteTopicDOs );

        //obtain the PC of the current sub-tutorial
        double cp = tutorialHelper.getMinCP( tutorialCPDOs );
        String path = tutorialHelper.getTutorialPathByChoice( cp );

        //If pre-tutorial was choose as route, obtain the pretutorial stuff
        if ( path.equals( ITutorialPath.TP_DISPLAY_PRE_TUTORING ) )
        {
            log.debug( "pre-tutorial enabled" );

            if ( selectedSubTutorialDO.getPreTutorialDO( ) != null )
            {
                currentTutorialDO = ( TutorialDO )
                    selectedSubTutorialDO.getPreTutorialDO( );

                log.debug( "Pre-Tutorial sys-id: " +
                    currentTutorialDO.getSysId( ) );
                log.debug( "Pre-Tutorial title: " +

```



```

        currentTutorialDO.getTutorialTitle( ));

currentSubTutorialDOs =
    tutorialDAO.findSubTutorialsByTutorialSysId(
        currentTutorialDO.getSysId( ));

//create a pretutorial stack record
PreTutorialStackDO stackDO = new PreTutorialStackDO( );
stackDO.setStudentSysId( studentDO.getSysId( ));
stackDO.setPreTutorialSysId( currentTutorialDO.getSysId( ));
stackDO.setParentSubTutorialSysId(
    selectedSubTutorialDO.getSysId( ));
stackDO.setCreatedTimestamp( new java.sql.Timestamp(
    System.currentTimeMillis( )) );

Boolean demoFlag = false;

//only create the pre-tutorial stack when it is not present
try
{
    stackDO = tutorialDAO.findPreTutorialStackDO(
        currentTutorialDO.getSysId( ), studentDO.getSysId( ));
}
catch ( Exception ex )
{
    tutorialDAO.createPreTutorialStack( stackDO );
}

selectedSubTutorialDO = ( SubTutorialDO )
    currentSubTutorialDOs.iterator( ).next( );
result.put( IWebConst.CURRENT_TUTORIAL,
    currentTutorialDO );
result.put( IWebConst.IS_PRE_TUTORIAL, Boolean.TRUE );
}
else
{
    log.debug( "No Pre-Tutorial, Changed to Demo" );
    path = ITutorialPath.TP_DEMO;
}
}
//set next route into the request
result.put( IWebConst.DEST_PATH, path );

//set all things into request
result.put( IWebConst.CURRENT_TUTORIAL, currentTutorialDO );
result.put( IWebConst.CURRENT_SUB_TUTORIALS, currentSubTutorialDOs );
result.put( IWebConst.SELECTED_SUB_TUTORIAL, selectedSubTutorialDO );

commitTrxn( );

return result;
}
catch ( CException ex )
{
    rollbackTrxnOnly( );
    log.error( ex.getMessage( ), ex );
    throw ex;
}

```

```

        catch ( Exception ex )
        {
            rollbackTrxnOnly( );

            log.error( ex.getMessage( ), ex );
            throw getExceptionHandler( ).generateError( "ERSYS-001" );
        }
    }

    /**
     * processFromTutorialRoute
     *
     * @param parameters
     *
     * @return HashMap
     *
     * @throws CException
     */
    public HashMap processFromTutorialRoute( HashMap parameters ) throws CException
    {
        HashMap result = new HashMap( );

        try
        {
            log.debug( "Check 1" );

            beginTrxn( );

            TutorialDO currentTutorialDO = ( TutorialDO )
                parameters.get( IWebConst.CURRENT_TUTORIAL );
            SubTutorialDO selectedSubTutorialDO = ( SubTutorialDO )
                parameters.get( IWebConst.SELECTED_SUB_TUTORIAL );
            Collection currentSubTutorialDOs = ( Collection )
                parameters.get( IWebConst.CURRENT_SUB_TUTORIALS );
            Collection allTutorialDOs = ( Collection )
                parameters.get( IWebConst.ALL_TUTORIALS );
            StudentDO studentDO = ( StudentDO )
                parameters.get( IWebConst.STUDENT_DO );
            long startTimeMillis = ( Long )
                parameters.get( IWebConst.START_TIME_MILLIS ).longValue( );
            long duration = System.currentTimeMillis( ) - startTimeMillis;

            log.debug( "Check 2" );

            //obtain the answer from student
            String answerText = ( String ) parameters.get( "answerText" );

            log.debug( "answerText: " + answerText );

            //validate the student answer
            boolean isAnswerRight = answerSyntaxParser.isAnswerCorrect(
                selectedSubTutorialDO.getSubTutorialAnswerDO( ).getSyntaxString( ),
                answerText );

            log.debug( "Check 3" );

            //Create the new next tutorial tracker
            RouteDO routeDO = tutorialDAO.findRouteByRouteName(

```

```

IWebConst.ROUTE_FROM_TUTORIAL );

TutorialTrackerDO tutorialTrackerDO = new TutorialTrackerDO( );
tutorialTrackerDO.setCreatedTimestamp( new java.sql.Timestamp(
    System.currentTimeMillis( ) ));
tutorialTrackerDO.setStudentDO( studentDO );
tutorialTrackerDO.setSubTutorialDO( selectedSubTutorialDO );
tutorialTrackerDO.setTutorialDO( currentTutorialDO );
tutorialTrackerDO.setRouteDO( routeDO );
tutorialTrackerDO.setDuration( duration );

log.debug( "Check 4" );

//if answer is right
if ( isAnswerRight )
{
    log.debug( "Check 5" );
    log.debug( "isAnswerRight: " + isAnswerRight );

    tutorialTrackerDO.setAnswer( answerText );
    tutorialTrackerDO.setCorrect( true );

    //Find Next sub tutorial
    SubTutorialDO nextSelectedSubTutorialDO =
    this.findNextSubTutorialDO( selectedSubTutorialDO, studentDO );

    if ( nextSelectedSubTutorialDO != null )
    {
        log.debug( "Check 6" );
        log.debug( "determining next route" );

        //determine next route by gathering of all next tutorial PC or
        // next sub-tutorial
        //get the next tutorial's pre-requisite sub-topics
        Collection preRequisiteSubTopicDOs =
        tutorialDAO.findPreRequisiteSubTopicsByTutorialSysId(
        nextSelectedSubTutorialDO.getTutorialDO( ).getSysId( ) );

        //return the choice page / next tutorial page
        String path = this.getTutorialSBLocal( ).findInitialPath(
            studentDO, preRequisiteSubTopicDOs );

        log.debug( "path check 1" );

        if ( nextSelectedSubTutorialDO.getTutorialDO( ).getSysId( )
            == currentTutorialDO.getSysId( ) )
        {
            log.debug( "path check 2" );
            path =
            ITutorialPath.TP_DISPLAY_PROBLEM_SPEC;
        }

        log.debug( "path: " + path );

        //set next route into the request
        result.put( IWebConst.DEST_PATH, path );
    }
}

```

```

        //set all things into request
        result.put( IWebConst.CURRENT_TUTORIAL,
                    nextSelectedSubTutorialDO.getTutorialDO( ) );
        result.put( IWebConst.CURRENT_SUB_TUTORIALS,
                    tutorialDAO.findSubTutorialsByTutorialSysId(
                    nextSelectedSubTutorialDO.getTutorialDO( ).getSysId( ) ) );
        result.put( IWebConst.SELECTED_SUB_TUTORIAL,
                    nextSelectedSubTutorialDO );
    }
    else
    {
        log.debug( "path: " + ITutorialPath.TP_COMPLETED );

        studentDO.setTutorialCompleted( true );
        studentDAO.updateStudent( studentDO );

        result.put( IWebConst.DEST_PATH,
                    ITutorialPath.TP_COMPLETED );
    }
}
else
{
    log.debug( "Check 9" );
    tutorialTrackerDO.setAnswer( answerText );
    tutorialTrackerDO.setCorrect( false );

    //obtain current sub-tutorial pre-requisites
    Collection preRequisiteSubTopicDOs =
        tutorialDAO.findPreRequisiteSubTopicsByTutorialSysId(
            currentTutorialDO.getSysId( ) );
    Collection preRequisiteTopicDOs =
        tutorialHelper.getTopicsFromSubTopics( preRequisiteSubTopicDOs );

    Collection tutorialCPDOs = tutorialDAO.findFilteredTutorialCPs(
        studentDO.getUserDO( ).getSysId( ), preRequisiteTopicDOs );

    log.debug( "tutorialCPDOs size: " + tutorialCPDOs.size( ) );

    //obtain the PC of the current sub-tutorial
    double cp = tutorialHelper.getMinCP( tutorialCPDOs );

    log.debug( "cp: " + cp );

    String path = tutorialHelper.getTutorialPathByChoice( cp );

    log.debug( "path: " + path );

    //If pre-tutorial was choose as route, obtain the pretutorial stuff
    if ( path.equals( ITutorialPath.TP_DISPLAY_PRE_TUTORING ) )
    {
        log.debug( "Check 10" );
        log.debug( "pre-tutorial enabled" );

        if ( selectedSubTutorialDO.getPreTutorialDO( ) != null )
        {
            currentTutorialDO = ( TutorialDO )
                selectedSubTutorialDO.getPreTutorialDO( );
            currentSubTutorialDOs =

```

```

        tutorialDAO.findSubTutorialsByTutorialSysId
            ( currentTutorialDO.getSysId( ) );

//create a pretutorial stack record
PreTutorialStackDO stackDO = new
    PreTutorialStackDO( );
stackDO.setStudentSysId( studentDO.getSysId( ) );
stackDO.setPreTutorialSysId(
    currentTutorialDO.getSysId( ) );
stackDO.setParentSubTutorialSysId(
    selectedSubTutorialDO.getSysId( ) );
stackDO.setCreatedTimestamp( new
    java.sql.Timestamp( System.currentTimeMillis( ) ) );

Boolean demoFlag = false;

//only creates it when the pre-tutorial is not in the stack
try
{
    stackDO =
        tutorialDAO.findPreTutorialStackDO
            ( currentTutorialDO.getSysId( ),
              studentDO.getSysId( ) );
}
catch ( Exception ex )
{
    tutorialDAO.createPreTutorialStack(stackDO);
}

selectedSubTutorialDO = ( SubTutorialDO )
    currentSubTutorialDOs.iterator( ).next( );
result.put( IWebConst.IS_PRE_TUTORIAL,
            Boolean.TRUE );
}
else
{
    log.debug( "No Pre-Tutorial, Changed to Demo" );
    path = ITutorialPath.TP_DEMO;
}
}

log.debug( "Check 11" );

//set next route into the request
result.put( IWebConst.DEST_PATH, path );

//set all things into request
result.put( IWebConst.CURRENT_TUTORIAL, currentTutorialDO );
result.put( IWebConst.CURRENT_SUB_TUTORIALS,
            currentSubTutorialDOs );
result.put( IWebConst.SELECTED_SUB_TUTORIAL,
            selectedSubTutorialDO );
}

log.debug( "Check 12" );
tutorialDAO.createTutorialTracker( tutorialTrackerDO );

```

```

        //Update cp
        if ( !isAnswerRight )
        {
            log.debug( "Check 13" );
            this.updateCp( selectedSubTutorialDO, studentDO );
        }

        commitTrxn( );

        return result;
    }
    catch ( CException ex )
    {
        rollbackTrxnOnly( );

        log.error( ex.getMessage( ), ex );
        throw ex;
    }
    catch ( Exception ex )
    {
        rollbackTrxnOnly( );

        log.error( ex.getMessage( ), ex );
        throw getExceptionHandler( ).generateError( "ERSYS-001" );
    }
}

/**
 * updateSubTutorial
 *
 * @param subTutorialDO
 * @param preTestQuestionDOs
 * @param hintDOs
 *
 * @return SubTutorialDO
 *
 * @throws CException
 */
public SubTutorialDO updateSubTutorial( SubTutorialDO subTutorialDO,
                                       Collection preTestQuestionDOs, Collection hintDOs ) throws CException
{
    try
    {
        beginTrxn( );

        java.sql.Timestamp now = new java.sql.Timestamp(System.currentTimeMillis( ));
        StatusDO statusDO = statusDAO.findStatusByStatusType( IStatus.ACTIVE );

        //Update SubTutorialDO
        subTutorialDO.setLastModifiedTimestamp( now );
        tutorialDAO.updateSubTutorial( subTutorialDO );

        //update subTutorialAnswerDO
        SubTutorialAnswerDO answerDO = subTutorialDO.getSubTutorialAnswerDO( );
        answerDO.setLastModifiedTimestamp( now );

        tutorialDAO.updateSubTutorialAnswer(
            subTutorialDO.getSubTutorialAnswerDO( ) );
    }
    catch ( CException ex )
    {
        rollbackTrxnOnly( );
        log.error( ex.getMessage( ), ex );
        throw ex;
    }
    catch ( Exception ex )
    {
        rollbackTrxnOnly( );
        log.error( ex.getMessage( ), ex );
        throw getExceptionHandler( ).generateError( "ERSYS-001" );
    }
}

```

```

//Remove and re-create preTestMap
Collection subTutorialPreTestMapDOs =
tutorialDAO.findPreTestMapsBySubTutorialSysId( subTutorialDO.getSysId( ) );
tutorialDAO.deleteBatchSubTutorialPreTestMap( subTutorialPreTestMapDOs );

if ( ( preTestQuestionDOs != null ) && ( preTestQuestionDOs.size( ) > 0 ) )
{
    ArrayList mapDOs = new ArrayList( );
    Iterator itr = preTestQuestionDOs.iterator( );

    while ( itr.hasNext( ) )
    {
        PreTestQuestionDO _DO = ( PreTestQuestionDO ) itr.next( );
        SubTutorialPreTestMapDO mapDO = new
            SubTutorialPreTestMapDO( );
        mapDO.setPreTestQuestionSysId( _DO.getSysId( ) );
        mapDO.setSubTutorialSysId( subTutorialDO.getSysId( ) );

        mapDOs.add( mapDO );
    }

    tutorialDAO.createBatchSubTutorialPreTestMap( mapDOs );
}

//Remove and re-create preTutorialMap

/*
Collection subTutorialPreTutorialMapDOs =
tutorialDAO.findPreTutorialMapsBySubTutorialSysId(
    subTutorialDO.getSysId( ) );
tutorialDAO.deleteBatchSubTutorialPreTutorialMap(
    subTutorialPreTutorialMapDOs );

if ( ( preTutorialDOs != null ) && ( preTutorialDOs.size( ) > 0 ) )
{
    ArrayList mapDOs = new ArrayList( );
    Iterator itr = preTutorialDOs.iterator( );

    while ( itr.hasNext( ) )
    {
        TutorialDO _DO = ( TutorialDO ) itr.next( );
        SubTutorialPreTutorialMapDO mapDO = new
            SubTutorialPreTutorialMapDO( );
        mapDO.setTutorialSysId( _DO.getSysId( ) );
        mapDO.setSubTutorialSysId( subTutorialDO.getSysId( ) );

        mapDOs.add( mapDO );
    }

    tutorialDAO.createBatchSubTutorialPreTutorialMap( mapDOs );
}
*/

```

```

//Remove and recreate hints
Collection tmpHintDOs = tutorialDAO.findHintsBySubTutorialSysId(
                                                                    subTutorialDO.getSysId( ) );
tutorialDAO.deleteBatchHints( tmpHintDOs );

if ( ( hintDOs != null ) && ( hintDOs.size( ) > 0 ) )
{
    Iterator itr = hintDOs.iterator( );

    while ( itr.hasNext( ) )
    {
        HintDO hintDO = ( HintDO ) itr.next( );
        hintDO.setSubTutorialDO( subTutorialDO );
        hintDO.setCreatedTimestamp( now );
        hintDO.setLastModifiedTimestamp( now );
        hintDO.setStatusDO( statusDO );

        tutorialDAO.createHint( hintDO );
    }
}

//update demo
DemoDO demoDO = subTutorialDO.getDemoDO( );
demoDO.setSysId( subTutorialDO.getSysId( ) );
demoDO.setLastModifiedTimestamp( now );

tutorialDAO.updateDemo( demoDO );

commitTrxn( );

return tutorialDAO.findSubTutorialBySysId( subTutorialDO.getSysId( ) );
}
catch ( CException ex )
{
    rollbackTrxnOnly( );

    log.error( ex.getMessage( ), ex );
    throw ex;
}
catch ( Exception ex )
{
    rollbackTrxnOnly( );

    log.error( ex.getMessage( ), ex );
    throw getExceptionHandler( ).generateError( "ERSYS-001" );
}
}

```



```

/**
 * updateTutorial
 *
 * @param tutorialDO
 * @param acquiredSkillsSubTopicDOs
 * @param preRequisiteSubTopicDOs
 * @param newSubTutorialDOs
 * @param deleteSubTutorialDOs
 *
 * @throws CException
 */
public void updateTutorial( TutorialDO tutorialDO, Collection acquiredSkillsSubTopicDOs,
    Collection preRequisiteSubTopicDOs, Collection newSubTutorialDOs,
    Collection deleteSubTutorialDOs ) throws CException
{
    try
    {
        beginTrxn( );

        java.sql.Timestamp now = new java.sql.Timestamp(System.currentTimeMillis( ));
        StatusDO statusDO = statusDAO.findStatusByStatusType( IStatus.ACTIVE );

        //Update the tutorialDO first
        tutorialDO.setLastModifiedTimestamp( now );
        tutorialDAO.updateTutorial( tutorialDO );

        //Re-map the acquiredSkills and Pre-Requisites
        Collection tmp = tutorialDAO.findAcquiredSkillsMapByTutorialSysId(
            tutorialDO.getSysId( ) );
        tutorialDAO.deleteBatchTutorialAcquiredSkillMap( tmp );

        tmp = tutorialDAO.findPreRequisiteMapByTutorialSysId( tutorialDO.getSysId( ) );
        tutorialDAO.deleteBatchTutorialPreRequisiteMap( tmp );

        ArrayList acquiredSkillsMapDOs = new ArrayList( );
        ArrayList preRequisiteMapMapDOs = new ArrayList( );

        if ( ( acquiredSkillsSubTopicDOs != null ) &&
            ( acquiredSkillsSubTopicDOs.size( ) > 0 ) )
        {
            Iterator itr = acquiredSkillsSubTopicDOs.iterator( );

            while ( itr.hasNext( ) )
            {
                SubTopicDO _DO = ( SubTopicDO ) itr.next( );
                TutorialAcquiredSkillMapDO mapDO = new
                    TutorialAcquiredSkillMapDO( );
                mapDO.setTutorialSysId( tutorialDO.getSysId( ) );
                mapDO.setSubTopicSysId( _DO.getSysId( ) );

                acquiredSkillsMapDOs.add( mapDO );
            }
        }
    }
}

```

```

if ( ( preRequisiteSubTopicDOs != null ) &&
      ( preRequisiteSubTopicDOs.size( ) > 0 ) )
{
    Iterator itr = preRequisiteSubTopicDOs.iterator( );

    while ( itr.hasNext( ) )
    {
        SubTopicDO _DO = ( SubTopicDO ) itr.next( );
        TutorialPreRequisiteMapDO mapDO = new
            TutorialPreRequisiteMapDO( );
        mapDO.setTutorialSysId( tutorialDO.getSysId( ) );
        mapDO.setSubTopicSysId( _DO.getSysId( ) );

        preRequisiteMapMapDOs.add( mapDO );
    }
}

tutorialDAO.createBatchTutorialAcquiredSkillMap( acquiredSkillsMapDOs );
tutorialDAO.createBatchTutorialPreRequisiteMap( preRequisiteMapMapDOs );

//Update the Sub-Tutorials
//Delete the mark for delete subTutorials
if ( ( deleteSubTutorialDOs != null ) && ( deleteSubTutorialDOs.size( ) > 0 ) )
{
    Iterator itr = deleteSubTutorialDOs.iterator( );

    while ( itr.hasNext( ) )
    {
        SubTutorialDO _DO = ( SubTutorialDO ) itr.next( );

        //Delete the preTestMapping
        Collection preTestMapDOs =
            tutorialDAO.findPreTestMapsBySubTutorialSysId(
                _DO.getSysId( ) );
        tutorialDAO.deleteBatchSubTutorialPreTestMap(
            preTestMapDOs );

        //Delete the hints
        Collection hintDOs =
            tutorialDAO.findHintsBySubTutorialSysId( _DO.getSysId( ) );
        tutorialDAO.deleteBatchHints( hintDOs );

        //Delete the SubTutorial
        tutorialDAO.deleteSubTutorial( _DO );
    }
}

//Create new sub-tutorials
if ( ( newSubTutorialDOs != null ) && ( newSubTutorialDOs.size( ) > 0 ) )
{
    Iterator itr = newSubTutorialDOs.iterator( );

    while ( itr.hasNext( ) )
    {
        SubTutorialDO _DO = ( SubTutorialDO ) itr.next( );
        _DO.setTutorialDO( tutorialDO );

        _DO.setStatusDO( statusDO );
    }
}

```

```

        _DO.setCreatedTimestamp( now );
        _DO.setLastModifiedTimestamp( now );

        long subTutorialSysId = tutorialDAO.createSubTutorial( _DO );

        //Create Sub Tutorial Answer
        SubTutorialAnswerDO answerDO = new
            SubTutorialAnswerDO( );
        answerDO.setSysId( subTutorialSysId );
        answerDO.setStatusDO( statusDO );
        answerDO.setSubTutorialDO( _DO );
        answerDO.setSyntaxString( "" );
        answerDO.setCreatedTimestamp( now );
        answerDO.setLastModifiedTimestamp( now );

        tutorialDAO.createSubTutorialAnswer( answerDO );

        //Create Demo
        DemoDO demoDO = new DemoDO( );
        demoDO.setSysId( subTutorialSysId );
        demoDO.setSubTutorialDO( _DO );
        demoDO.setDemoId( _DO.getSubTutorialId( ) );
        demoDO.setDemoExplanation( "" );
        demoDO.setStatusDO( statusDO );
        demoDO.setCreatedTimestamp( now );
        demoDO.setLastModifiedTimestamp( now );

        tutorialDAO.createDemo( demoDO );

    }

    }

    commitTrxn( );
}
catch ( CException ex )
{
    rollbackTrxnOnly( );

    log.error( ex.getMessage( ), ex );
    throw ex;
}
catch ( Exception ex )
{
    rollbackTrxnOnly( );

    log.error( ex.getMessage( ), ex );
    throw getExceptionHandler( ).generateError( "ERSYS-001" );
}
}

```

```

/**
 * getTutorialSBLocal
 *
 * @return TutorialSBLocal
 *
 * @throws CException
 */
private TutorialSBLocal getTutorialSBLocal( ) throws CException
{
    try
    {
        TutorialSBLocalHome tutorialSBLocalHome = ( TutorialSBLocalHome )
            getServiceLocator( ).getLocalHome( IJNDI.JNDI_TUTORIAL_SB_LOCAL );

        return tutorialSBLocalHome.create( );
    }
    catch ( Exception ex )
    {
        log.error( ex.getMessage( ), ex );
        throw getExceptionHandler( ).generateError( "ERSERV-001" );
    }
}

/**
 * demote
 *
 * @param subTutorialDO
 * @param studentDO
 *
 * @return double
 *
 * @throws CException
 */
private double demote( SubTutorialDO subTutorialDO, StudentDO studentDO ) throws CException
{
    double minCP = 0.0d;

    Collection prSubTopicDOs = tutorialDAO.findPreRequisiteSubTopicsByTutorialSysId(
        subTutorialDO.getTutorialDO( ).getSysId( ) );

    if ( ( prSubTopicDOs != null ) && ( prSubTopicDOs.size( ) > 0 ) )
    {
        double[] cps = new double[ prSubTopicDOs.size( ) ];
        Iterator itr = prSubTopicDOs.iterator( );

        for ( int i = 0; itr.hasNext( ); i++ )
        {
            SubTopicDO _DO = ( SubTopicDO ) itr.next( );
            TutorialCPDO cpDO = tutorialDAO.findTutorialCP( _DO.getSysId( ),
                studentDO.getUserDO( ).getSysId( ) );

            if ( cpDO.getTotalCorrect( ) != cpDO.getTotalQuestions( ) )
            {
                if ( cpDO.getTotalCorrect( ) == 0 )
                {
                    cps[ i ] = 0.01;
                }
                else

```

```

        {
            cps[ i ] = preTestHelper.getConditionalProbability(
                cpDO.getTotalQuestions( ),
                cpDO.getTotalCorrect( ) - 1, 4 );
        }
    }
    else
    {
        cps[ i ] = cpDO.getCp( );
    }
}
minCP = tutorialHelper.getAverageCP( cps );
}
return minCP;
}

/**
 * findNextSubTutorialDO
 *
 * @param currentSelectedSubTutorialDO
 * @param studentDO
 *
 * @return SubTutorialDO
 *
 * @throws CException
 */
public SubTutorialDO findNextSubTutorialDO( SubTutorialDO currentSelectedSubTutorialDO,
                                            StudentDO studentDO ) throws CException
{
    TutorialDO nextTutorialDO = null;
    SubTutorialDO nextSelectedSubTutorialDO = null;
    Collection nextSubTutorialDOs = null;

    //obtain next sub-tutorial or tutorial
    log.debug( "Obtaining next sub-tutorial or tutorial" );

    Collection currentSubTutorialDOs = tutorialDAO.findSubTutorialsByTutorialSysId(
        currentSelectedSubTutorialDO.getTutorialDO( ).getSysId( ) );
    Collection allTutorialDOs = tutorialDAO.findAllTutorials( );

    Iterator itr = currentSubTutorialDOs.iterator( );

    while ( itr.hasNext( ) )
    {
        SubTutorialDO _DO = ( SubTutorialDO ) itr.next( );

        if ( _DO.getSysId( ) == currentSelectedSubTutorialDO.getSysId( ) )
        {
            //next sub-tutorial
            if ( itr.hasNext( ) )
            {
                log.debug( "Obtaining Next Sub-Tutorial" );

                nextSelectedSubTutorialDO = ( SubTutorialDO ) itr.next( );
                nextTutorialDO = nextSelectedSubTutorialDO.getTutorialDO( );
                nextSubTutorialDOs = currentSubTutorialDOs;

                break;
            }
        }
    }
}

```

```

    }
    //next tutorial
    else
    {
        //Check if the student is at pre-tutorial
        try
        {
            PreTutorialStackDO stackDO =
            tutorialDAO.findPreTutorialStackDO(
            currentSelectedSubTutorialDO.getTutorialDO
            ( ).getSysId( ), studentDO.getSysId( ) );
            SubTutorialDO parentSubTutorialDO =
            tutorialDAO.findSubTutorialBySysId(
            stackDO.getParentSubTutorialSysId( ) );

            //clear the stackDO
            tutorialDAO.deletePreTutorialStack( stackDO );

            //return previous sub tutorial
            return parentSubTutorialDO;
        }
        catch ( Exception ex )
        {
            //move on...
        }

        log.debug( "Obtaining Next Tutorial" );

        Iterator itr2 = allTutorialDOs.iterator( );

        while ( itr2.hasNext( ) )
        {
            TutorialDO tDO = ( TutorialDO ) itr2.next( );

            if ( tDO.getSysId( ) ==
            currentSelectedSubTutorialDO.getTutorialDO
            ( ).getSysId( ) )
            {
                //next tutorial
                if ( itr2.hasNext( ) )
                {
                    log.debug( "Found Next Tutorial" );

                    nextTutorialDO = ( TutorialDO )
                    itr2.next( );
                    nextSubTutorialDOs =
                    tutorialDAO.findSubTutorialsByTuto
                    rialSysId( nextTutorialDO.getSysId(
                    ) );
                    nextSelectedSubTutorialDO = (
                    SubTutorialDO )
                    nextSubTutorialDOs.iterator( ).next(
                    );

                    break;
                }
                else
                {

```



```

/**
 * promote
 *
 * @param subTutorialDO
 * @param studentDO
 *
 * @return promote
 *
 * @throws CException
 */
private double promote( SubTutorialDO subTutorialDO, StudentDO studentDO ) throws CException
{
    double minCP = 0.0d;

    Collection prSubTopicDOs = tutorialDAO.findPreRequisiteSubTopicsByTutorialSysId(
        subTutorialDO.getTutorialDO( ).getSysId( ) );

    if ( ( prSubTopicDOs != null ) && ( prSubTopicDOs.size( ) > 0 ) )
    {
        double[] cps = new double[ prSubTopicDOs.size( ) ];
        Iterator itr = prSubTopicDOs.iterator( );

        for ( int i = 0; itr.hasNext( ); i++ )
        {
            SubTopicDO _DO = ( SubTopicDO ) itr.next( );
            TutorialCPDO cpDO = tutorialDAO.findTutorialCP( _DO.getSysId( ),
                studentDO.getUserDO( ).getSysId( ) );

            if ( cpDO.getTotalCorrect( ) != cpDO.getTotalQuestions( ) )
            {
                cps[ i ] = preTestHelper.getConditionalProbability(
                    cpDO.getTotalQuestions( ), cpDO.getTotalCorrect( ) + 1, 4 );
            }
            else
            {
                cps[ i ] = cpDO.getCp( );
            }
        }

        minCP = tutorialHelper.getAverageCP( cps );
    }

    return minCP;
}

```



```

/**
 * updateCp
 *
 * @param subTutorialDO
 * @param studentDO
 *
 * @throws CException
 */
private void updateCp( SubTutorialDO subTutorialDO, StudentDO studentDO ) throws CException
{
    log.debug( "===== " );
    log.debug( "Entering updateCp" );
    log.debug( "===== " );

    //Obtain number of attempts from previous tutorial tracker
    int noOfAttempts = tutorialDAO.findNoOfAttempts( subTutorialDO.getSysId( ),
                                                    studentDO.getSysId( ) );

    //Check no. of attempts and update the student model accordingly
    Collection preRequisiteSubTopicDOs =
        tutorialDAO.findPreRequisiteSubTopicsByTutorialSysId
            ( subTutorialDO.getTutorialDO( ).getSysId( ) );
    double currentCP = this.findPreRequisiteCp( preRequisiteSubTopicDOs, studentDO );

    //find tutorial track records of the same sub tutorial
    boolean fromBriefExplanation = false;
    boolean fromPreTestReview = false;
    boolean fromPreTutoring = false;
    boolean fromDemo = false;

    try
    {
        tutorialDAO.findTutorialTrackDO( studentDO.getSysId( ),
                                         subTutorialDO.getSysId( ),
                                         ITutorialPath.ROUTE_FROM_BRIEF_EXPLANATION );
        fromBriefExplanation = true;
    }
    catch ( Exception ex )
    {
        //Absorb
    }

    try
    {
        tutorialDAO.findTutorialTrackDO( studentDO.getSysId( ),
                                         subTutorialDO.getSysId( ),
                                         ITutorialPath.ROUTE_FROM_FROM_PRE_TEST_REVIEW );
        fromPreTestReview = true;
    }
    catch ( Exception ex )
    {
        //Absorb
    }
}

```

```

try
{
    tutorialDAO.findTutorialTrackDO( studentDO.getSysId( ),
    subTutorialDO.getSysId( ), ITutorialPath.ROUTE_FROM_PRE_TUTORIAL );
    fromPreTutoring = true;
}
catch ( Exception ex )
{
    //Absorb
}

try
{
    tutorialDAO.findTutorialTrackDO( studentDO.getSysId( ),
    subTutorialDO.getSysId( ), ITutorialPath.ROUTE_FROM_FROM_DEMO );
    fromDemo = true;
}

catch ( Exception ex )
{
    //Absorb
}

boolean update = false;
double newCP = currentCP;

log.debug( "noOfAttempts: " + noOfAttempts );
log.debug( "currentCP: " + currentCP );

//Bottom up
//First Attempt
if ( noOfAttempts == 0 )
{
    if ( currentCP > 0.8 )
    {
        if ( fromPreTutoring || fromDemo )
        {
            newCP = demote( subTutorialDO, studentDO );
            update = true;
        }
        else if ( fromPreTestReview )
        {
            newCP = 0.6d;
            update = true;
        }
    }
    else if ( ( currentCP >= 0.6 ) && ( currentCP <= 0.8 ) )
    {
        if ( fromDemo )
        {
            newCP = demote( subTutorialDO, studentDO );
            update = true;
        }
        else if ( fromPreTutoring )
        {
            newCP = 0.6d;
            update = true;
        }
    }
}

```

```

    }
    else
    {
        if ( fromDemo )
        {
            newCP = 0.01d;
            update = true;
        }
    }
}

//More than one attempts
else if ( noOfAttempts == 1 )
{
    if ( currentCP > 0.8 )
    {
        if ( !fromBriefExplanation && !fromPreTestReview &&
            !fromPreTutoring && !fromDemo )
        {
            newCP = 1.0d;
            update = true;
        }
        else if ( fromPreTestReview || fromPreTutoring || fromDemo )
        {
            newCP = demote( subTutorialDO, studentDO );
            update = true;
        }
        else if ( fromBriefExplanation )
        {
            newCP = 0.6d;
            update = true;
        }
    }
    else if ( ( currentCP >= 0.6 ) && ( currentCP <= 0.8 ) )
    {
        if ( fromPreTutoring || fromDemo )
        {
            newCP = demote( subTutorialDO, studentDO );
            update = true;
        }
        else if ( fromPreTestReview )
        {
            newCP = 0.6d;
            update = true;
        }
    }
}
else
{
    if ( fromDemo )
    {
        newCP = demote( subTutorialDO, studentDO );
        update = true;
    }
    else if ( fromPreTutoring )
    {
        newCP = 0.6d;
        update = true;
    }
}

```

```

        }
    }

    //The rest
    else
    {
        update = true;
        newCP = demote( subTutorialDO, studentDO );
    }

    //Update the cp
    if ( update )
    {
        Collection prSubTopicDOs =
            tutorialDAO.findPreRequisiteSubTopicsByTutorialSysId(
                subTutorialDO.getTutorialDO().getSysId() );

        if ( ( prSubTopicDOs != null ) && ( prSubTopicDOs.size() > 0 ) )
        {
            Iterator itr = prSubTopicDOs.iterator();

            while ( itr.hasNext() )
            {
                SubTopicDO _DO = ( SubTopicDO ) itr.next();

                TutorialCPDO cpDO = tutorialDAO.findTutorialCP(
                    _DO.getSysId(), studentDO.getUserDO().getSysId());
                cpDO.setCp( newCP );
                tutorialDAO.updateTutorialCPDO( cpDO );
            }
        }
    }
}

```

## Appendix J – Fuzzy Rules Table

Note: Linguistic Variable

$p$  = Conditional Probabilities  
 $t$  = Time  
 $a$  = Attempts  
 $h$  = Hints  
 $u$  = Understanding

R = Rule

Linguistic Value

VL = Very Low  
 VS = Very Short  
 S = Short  
 L = Low/Long  
 M = Medium  
 H = High

N = Novice  
 B = Beginner  
 I = Intermediate  
 A = Advanced

R	$p$	$t$	$a$	$h$	$u$	R	$p$	$t$	$a$	$h$	$u$	R	$p$	$t$	$a$	$h$	$u$	R	$p$	$t$	$a$	$h$	$u$
1	VL	VS	VL	VL	B	17	VL	VS	VL	L	I	33	VL	VS	VL	M	I	49	VL	VS	VL	H	B
2	L	VS	VL	VL	I	18	L	VS	VL	L	I	34	L	VS	VL	M	B	50	L	VS	VL	H	B
3	M	VS	VL	VL	I	19	M	VS	VL	L	I	35	M	VS	VL	M	I	51	M	VS	VL	H	I
4	H	VS	VL	VL	A	20	H	VS	VL	L	A	36	H	VS	VL	M	I	52	H	VS	VL	H	I
5	VL	S	VL	VL	B	21	VL	S	VL	L	B	37	VL	S	VL	M	B	53	VL	S	VL	H	B
6	L	S	VL	VL	B	22	L	S	VL	L	B	38	L	S	VL	M	B	54	L	S	VL	H	B
7	M	S	VL	VL	I	23	M	S	VL	L	I	39	M	S	VL	M	I	55	M	S	VL	H	B
8	H	S	VL	VL	A	24	H	S	VL	L	A	40	H	S	VL	M	I	56	H	S	VL	H	I
9	VL	M	L	VL	B	25	VL	M	L	L	I	41	VL	M	L	M	B	57	VL	M	L	H	N
10	L	M	L	VL	B	26	L	M	L	L	B	42	L	M	L	M	B	58	L	M	L	H	B
11	M	M	L	VL	I	27	M	M	L	L	I	43	M	M	L	M	I	59	M	M	L	H	B
12	H	M	L	VL	I	28	H	M	L	L	I	44	H	M	L	M	I	60	H	M	L	H	I
13	VL	L	L	VL	B	29	VL	L	L	L	B	45	VL	L	L	M	B	61	VL	L	L	H	N
14	L	L	L	VL	B	30	L	L	L	L	B	46	L	L	L	M	B	62	L	L	L	H	B
15	M	L	L	VL	I	31	M	L	L	L	I	47	M	L	L	M	I	63	M	L	L	H	B
16	H	L	L	VL	I	32	H	L	L	L	B	48	H	L	L	M	B	64	H	L	L	H	B

R	$p$	$t$	$a$	$h$	$u$	R	$p$	$t$	$a$	$h$	$u$	R	$p$	$t$	$a$	$h$	$u$	R	$p$	$t$	$a$	$h$	$u$
65	VL	VS	M	VL	B	81	VL	VS	M	L	I	97	VL	VS	VL	M	B	113	VL	VS	M	H	B
66	L	VS	M	VL	B	82	L	VS	M	L	I	98	L	VS	VL	M	B	114	L	VS	M	H	B
67	M	VS	M	VL	I	83	M	VS	M	L	I	99	M	VS	VL	M	I	115	M	VS	M	H	B
68	H	VS	M	VL	A	84	H	VS	M	L	A	100	H	VS	VL	M	I	116	H	VS	M	H	I
69	VL	S	M	VL	B	85	VL	S	M	L	I	101	VL	S	VL	M	B	117	VL	S	M	H	B
70	L	S	M	VL	B	86	L	S	M	L	I	102	L	S	VL	M	B	118	L	S	M	H	B
71	M	S	M	VL	I	87	M	S	M	L	I	103	M	S	VL	M	I	119	M	S	M	H	I
72	H	S	M	VL	A	88	H	S	M	L	A	104	H	S	VL	M	I	120	H	S	M	H	I
73	VL	M	H	VL	B	89	VL	M	H	L	B	105	VL	M	L	M	I	121	VL	M	H	H	B
74	L	M	H	VL	B	90	L	M	H	L	B	106	L	M	L	M	I	122	L	M	H	H	B
75	M	M	H	VL	I	91	M	M	H	L	I	107	M	M	L	M	I	123	M	M	H	H	I
76	H	M	H	VL	I	92	H	M	H	L	I	108	H	M	L	M	I	124	H	M	H	H	I
77	VL	L	H	VL	B	93	VL	L	H	L	B	109	VL	L	L	M	B	125	VL	L	H	H	N
78	L	L	H	VL	B	94	L	L	H	L	B	110	L	L	L	M	B	126	L	L	H	H	N
79	M	L	H	VL	B	95	M	L	H	L	I	111	M	L	L	M	I	127	M	L	H	H	B
80	H	L	H	VL	I	96	H	L	H	L	I	112	H	L	L	M	I	128	H	L	H	H	I

## Appendix K – build.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
- <project basedir="." default="help" name="ITS">
- <!--
Static Properties
-->
<property name="file.its.ear" value="its.ear" />
<property name="file.resources.jar" value="resources.jar" />
- <!--
End of Static Properties
-->
- <!--
Development Properties
-->
<property name="build" location="build" />
<property name="build.classes" location="${build}/classes" />
<property name="conf" location="conf" />
<property name="conf.ear" location="${conf}/ear" />
<property name="conf.ejb" location="${conf}/ejb" />
<property name="conf.hibernate" location="${conf}/hibernate" />
<property name="conf.properties" location="${conf}/properties" />
<property name="conf.spring" location="${conf}/spring" />
<property name="conf.jboss" location="${conf}/jboss" />
<property name="conf.webwork" location="${conf}/webwork" />
<property name="conf.xml" location="${conf}/xml" />
<property name="dist" location="dist" />
<property name="docs" location="docs" />
<property name="lib" location="lib" />
<property name="setup" location="setup" />
<property name="setup.mysql" location="${setup}/mysql" />
<property name="src" location="src" />
<property name="test" location="test" />
<property name="test.report" location="${test}/report" />
<property name="webapp" location="webapp" />
<property name="webapp.web-inf" location="${webapp}/WEB-INF" />
<property name="webapp.src" location="${webapp.web-inf}/src" />
<property name="webapp.classes" location="${webapp.web-inf}/classes" />
- <!--
End of Development Properties
-->
- <!--
Deployment Properties
-->
<property file="build.properties" />
- <!--
End of Deployment Properties
-->
- <!--
```

#### Classpath Definition

```
-->
<path id="ccpath">
  <pathelement path="${build.classes}" />
  <pathelement path="${conf.dev.properties}" />
  <pathelement path="${conf.hibernate}" />
  <pathelement path="${conf.spring}" />
</path>
<fileset dir="${lib}">
  <include name="**/*.jar" />
</fileset>
<fileset dir="${jboss.default.deploy}/jbossweb-tomcat55.sar">
  <include name="**/*.jar" />
</fileset>
</path>
```

- <!--

#### End of Classpath Definition

-->

- <!--

#### Task Definition

--> 

- <!--

```
<taskdef name="junit" classname="org.apache.tools.ant.taskdefs.optional.junit.JUnitTask"
classpathref="dev.ccpath"/>
```

--> 

```
<taskdef name="jasper2" classname="org.apache.jasper.JspC" classpathref="ccpath" />
```

- <!--

#### End of Task Definition

--> 

- <!--


#### Initialize Directories

--> 

```
<target name="init">
  <mkdir dir="${build.classes}" />
  <mkdir dir="${dist}" />
  <mkdir dir="${test.report}" />
  <mkdir dir="${webapp.classes}" />
  <mkdir dir="${webapp.src}" />
</target>
```

- <!--

#### Compile the Source Codes

--> 

```
<target name="compile" depends="init">
  <javac debug="true" srcdir="${src}" destdir="${build.classes}" classpathref="ccpath" optimize="on" />
  <javac debug="true" srcdir="${test}" destdir="${build.classes}" classpathref="ccpath" optimize="on" />
</target>
```

- <!--

#### Pre-Compile JSP

--> 

```
<target name="jsp-compile" depends="compile">
  <delete includeEmptyDirs="true" dir="${webapp.src}" />
  <delete includeEmptyDirs="true">
    <fileset dir="${webapp.classes}">
      <include name="**/*.jsp" />
    </fileset>
  </delete>
  <delete file="${webapp}/generated_web.xml" />
```

```


<jasper2 validateXml="false" uriroot="${webapp}" webXmlFragment="${webapp}/WEB-
INF/generated_web.xml" outputDir="${webapp}/WEB-INF/src" />
<javac debug="true" srcdir="${webapp.src}" destdir="${webapp.classes}" classpathref="ccpath"
optimize="on" />
</target>
- <!--
Setup JBoss
--> 
<target name="setup-jboss">
<delete dir="${jboss.its}" includeemptydirs="true" />
<mkdir dir="${jboss.its}" />
<copy todir="${jboss.its}" overwrite="true">
<fileset dir="${jboss.default}">
<include name="**/*" />
</fileset>
</copy>
<copy todir="${jboss.its.deploy}" overwrite="true">
<fileset dir="${conf.jboss}">
<include name="mysql-ds.xml" />
</fileset>
</copy>
<copy todir="${jboss.its.conf}" overwrite="true">
<fileset dir="${conf.jboss}">
<include name="log4j.xml" />
</fileset>
</copy>
<copy todir="${jboss.its.lib}" overwrite="true">
<fileset dir="${lib}">
<exclude name="j2ee.jar" />
<exclude name="mail.jar" />
<exclude name="ant*.jar" />
<exclude name="log4j*.jar" />
</fileset>
</copy>
</target>
- <!--
Build the source codes only (clean)
--> 
<target name="build" depends="clean, compile, jsp-compile" />
- <!--
Clean up the build directory
--> 
<target name="clean">
<delete includeEmptyDirs="true" failonerror="false">
<fileset dir="${build}">
<include name="**/*" />
</fileset>
<fileset dir="${dist}">
<include name="**/*" />
</fileset>
<fileset dir="${test.report}">
<include name="**/*" />
</fileset>
</delete>
<delete dir="${build.classes}" failonerror="false" />
<mkdir dir="${build.classes}" />
</target>
- <!--

```







Distribute the Enterprise Application

```
-->   
: <target name="dist" depends="build, resource-compile, lib-compile, ejb-compile, web-compile">  
: <ear destfile="${dist}/${file.its.ear}" appxml="${conf.ear}/application.xml" update="true">  
: <fileset dir="${build}">  
: <exclude name="**/sb/**" />  
: <exclude name="**/classes/**/*" />  
: </fileset>  
: </ear>  
: </target>  
- <!--
```

Deploy the web-app

```
-->   
: <target name="web-compile" depends="jsp-compile">  
: <copy todir="${webapp}">  
: <fileset dir="${conf.xml}">  
: <include name="itssyntax.dtd" />  
: </fileset>  
: </copy>  
: <jar destfile="${build}/its.war" update="true">  
: <fileset dir="${webapp}">  
: <exclude name="sub-web.xml" />  
: <exclude name="**/generated_web.xml" />  
: <exclude name="**/classes/**" />  
: <exclude name="**/src/**" />  
: </fileset>  
: </jar>  
: </target>  
- <!--
```


Compile the necessary libraries

```
-->   
: <target name="lib-compile">  
- <!--  
copy other external libraries into the deploy folder  
-->   
- <!--  
  <copy todir="${build.lib}">  
  <fileset dir="${lib}">  
  <exclude name="j2ee.jar"/>  
  <exclude name="log4j*.jar"/>  
  <exclude name="junit*.jar"/>  
  <exclude name="ant*.jar"/>  
  </fileset>  
  </copy>
```



```
-->   
</target>
```

```
- <!--
```

Compile all the resources (e.g. Hibernate Config, Spring Config and other property files)

```
-->   
: <target name="resource-compile">  
: <jar destfile="${build}/${file.resources.jar}" update="true">  
: <fileset dir="${conf.hibernate}">  
: <include name="*.hbm.xml" />  
: </fileset>  
: <fileset dir="${conf.properties}">  
: <include name="*.properties" />
```

```

</fileset>
<fileset dir="${conf.spring}">
  <include name="*.xml" />
</fileset>
<fileset dir="${conf.webwork}">
  <include name="*.xml" />
</fileset>
<fileset dir="${build.classes}">
  <exclude name="**/sb/**" />
</fileset>
</jar>
</target>
- <!--
Invoke the JBoss EJB Compilation
--> 
<target name="ejb-compile" depends="compile">
  <ant antfile="${conf.ejb}/admin/ejb-build.xml" target="ejb-compile" inheritAll="true"
inheritRefs="true" />
  <ant antfile="${conf.ejb}/authentication/ejb-build.xml" target="ejb-compile" inheritAll="true"
inheritRefs="true" />
  <ant antfile="${conf.ejb}/estudent/ejb-build.xml" target="ejb-compile" inheritAll="true"
inheritRefs="true" />
  <ant antfile="${conf.ejb}/pretest/ejb-build.xml" target="ejb-compile" inheritAll="true"
inheritRefs="true" />
  <ant antfile="${conf.ejb}/posttest/ejb-build.xml" target="ejb-compile" inheritAll="true"
inheritRefs="true" />
  <ant antfile="${conf.ejb}/student/ejb-build.xml" target="ejb-compile" inheritAll="true"
inheritRefs="true" />
  <ant antfile="${conf.ejb}/topic/ejb-build.xml" target="ejb-compile" inheritAll="true" inheritRefs="true"
/>
  <ant antfile="${conf.ejb}/tutor/ejb-build.xml" target="ejb-compile" inheritAll="true" inheritRefs="true"
/>
  <ant antfile="${conf.ejb}/tutorial/ejb-build.xml" target="ejb-compile" inheritAll="true"
inheritRefs="true" />
</target>
- <!--
Perform unit test
--> 
<target name="test" depends="compile">
  <junit printsummary="yes" haltonfailure="no" fork="yes" showoutput="yes">
    <classpath refid="ccpath" />
    <formatter type="xml" />
  </junit>
  <batchtest todir="${test.report}">
    <fileset dir="${build.classes}">
      <include name="test/**/*Test.class" />
    </fileset>
  </batchtest>
  <junitreport todir="${test.report}">
    <fileset dir="${test.report}">
      <include name="*.xml" />
    </fileset>
    <report format="frames" todir="${test.report}" />
  </junitreport>
</target>
- <!--

```

Development Deploy (Only for Developent)

--> 

```
<target name="deploy" depends="dist">
<copy todir="${jboss.its.deploy}">
<fileset dir="${dist}">
<include name="${file.its.ear}" />
</fileset>
</copy>
</target>
<target name="cancel" />
<target name="redploy" depends="dist" />
<target name="undeploy" />
<!--
```

Print the HELP menu

--> 

```
<target name="help">
<!--
```

to be implemented

--> 

```
</target>
</project>
```

## Appendix L – List of C++ STL vector Topics and Sub-Topics

Topic: C++ STL vector

Sub-Topics:

1	Populate using iterator
2	Output using subscript
3	Output using iterator
4	Accessing member function
5	Using operator <<
6	Using operator >>
7	Iterator - Forward RW
8	Iterator - Reverse RW
9	Iterator - Forward R
10	Iterator - Reverse R
11	Populate using member function
12	Description