The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

0-612-22783-9
Acknowledgements

I would like to acknowledge my advisor, Dr Jean Fugère who got me on track early in the thesis process and provide timely thought provoking ideas and conversation. Thanks are also due to Major Wayne Buck who assisted my first explorations into BGP-MS and Amber Thompson who made sure I made a lot fewer grammatical and typographical errors.
Abstract

This thesis describes the research and development into the intelligent tutoring system called JASMINE (JAVA Adaptive Student Modelling In a NLP Environment). JASMINE is designed to address the ideas of adaptive and responsive, presentation and remediation in tutoring. Remediation is indicated when the confidence level of a given fragment of knowledge falls below a set mastery level. As a result, the student is presented with timely assistance along with presentation adapted to the students preferred style of learning. In order to achieve this, the model of Neuro-Linguistic Programming (NLP) was considered, with the main input channel of the model (visual, auditory or kinetic), identified for each student.

The approach within JASMINE is to forecast the students most highly valued input channel through the use of a number of NLP related questions during the initial session. Then, to use monitoring techniques and to adjust the system's presentation to the student. To aid in modelling and knowledge storage, the User Modelling shell system BGP-MS was investigated and found lacking, particularly in the ability to store student knowledge levels and provide a cross platform environment. Hence, JAVA's role within JASMINE grew from the initial intention of interface integration to the implementation of the complete student model. This allowed the JASMINE application to become a pragmatic solution to the ITS question.
# TABLE OF CONTENTS

## Chapter 1  Thesis Overview

1.1 Introduction ........................................................................................................... 1-1  
1.1.1 Motivation ........................................................................................................... 1-1  
1.1.2 Aim ....................................................................................................................... 1-2  
1.2 Intelligent Tutoring Systems..................................................................................... 1-2  
1.2.1 Definition............................................................................................................. 1-2  
1.2.2 History................................................................................................................... 1-3  
1.3 Adaptation............................................................................................................... 1-3  
1.4 Modelling................................................................................................................. 1-4  
1.4.1 User....................................................................................................................... 1-5  
1.4.2 Student................................................................................................................ 1-6  
1.5 Intelligent User Interface......................................................................................... 1-7  
1.6 An Application......................................................................................................... 1-8  
1.7 Thesis Outline......................................................................................................... 1-9  

## Chapter 2  Intelligent Tutoring Systems

2.1 Introduction............................................................................................................. 2-1  
2.2 Intelligent Tutoring System Structure..................................................................... 2-1  
2.2.1 Pedagogy Module............................................................................................... 2-3  
2.2.2 Psychologist Module.......................................................................................... 2-3  
2.2.3 Interface............................................................................................................. 2-4  
2.2.4 Domain Expert Module..................................................................................... 2-4  
2.2.5 Student Module.................................................................................................. 2-5  
2.3 A Review: ITS Architecture.................................................................................... 2-6  
2.3.1 Actor Based Architecture................................................................................... 2-6  
2.3.2 Student Driven Architecture............................................................................. 2-8  
2.3.3 Co-operative Architecture................................................................................. 2-10  
2.4 A Review: Teaching and Learning Strategies........................................................... 2-12  
2.4.1 Learn-by-doing................................................................................................. 2-13  
2.4.2 Devil's Advocate............................................................................................... 2-15  

iv
Chapter 3  Student Modelling with BGP-MS and JAVA

3.1 Introduction........................................................................................................................................ 3-1

3.2 Student Modelling................................................................................................................................ 3-2
  3.2.1 Overlay Modelling.......................................................................................................................... 3-2
  3.2.2 Stereotypes....................................................................................................................................... 3-6
    3.2.2.1 Learning Style as a Stereotype
      - Neuro-Linguistic Programming........................................................................................................ 3-7

3.3 A Review: Student Modelling................................................................................................................ 3-9
  3.3.1 Using Granularity............................................................................................................................. 3-9
  3.3.2 Belief Net......................................................................................................................................... 3-11

3.4 BGP-MS..................................................................................................................................................... 3-13
  3.4.1 Student Modelling with BGP-MS...................................................................................................... 3-13
    3.4.1.1 Partition Hierarchies.................................................................................................................... 3-14
    3.4.1.2 Amplitude Differential Overlay Model......................................................................................... 3-15
    3.4.1.3 Application Communications (the flow of assumptions)............................................................... 3-16

3.4.2 Failings............................................................................................................................................... 3-18

3.5 Java......................................................................................................................................................... 3-19
  3.5.1 Object Oriented................................................................................................................................. 3-20
  3.5.2 Platform independence..................................................................................................................... 3-24
  3.5.3 Security............................................................................................................................................ 3-25
  3.5.4 Summary ......................................................................................................................................... 3-27
Chapter 4  Adaptability and Intelligent User Interface

4.1 Introduction
4.2 Adaptiveness and Presentation
4.3 The Design Questions
   4.3.1 A Debate: Direct Manipulation vs Interface Agents
      4.3.1.1 Direct Manipulation
      4.3.1.2 Intelligent Agents
      4.3.1.3 Observations and Comments
   4.3.2 A Discussion: How Much Intelligent?
      4.3.2.1 The Discussion
      4.3.2.2 An Answer
4.4 Adaptive and World Wide Web ITS Examples (Influences)
   4.4.1 ELM-ART
      4.4.1.1 Course Structure
      4.4.1.2 Adaptiveness
      4.4.1.3 Summary and Observations
   4.4.2 ATULA-ATS
      4.4.2.1 Clusters / Stereotypes
      4.4.2.2 Presentation
      4.4.2.3 Remediation
      4.4.2.4 Summary and Observations

Chapter 5  An Application

5.1 Introduction
5.2 Domain
   5.2.1 OPDP
5.3 Description of application
   5.3.1 The Tools
      5.3.1.1 Domain Manager
      5.3.1.2 Question Manager
      5.3.1.3 Learning Time: Student Server and Client
   5.3.2 Use of Modelling
Chapter 6 Conclusion

6.1 General........................................................................................................ 6-1
6.2 Testing....................................................................................................... 6-3
6.3 The Aim Revisited.................................................................................. 6-4
6.4 Suggested Enhancements....................................................................... 6-5
6.5 Future Research....................................................................................... 6-6

5.3.3 Adaptation................................................................................................ 5-15
  5.3.3.1 Initial Learning Model................................................................. 5-15
  5.3.3.2 Changes in Presentation............................................................. 5-16
  5.3.3.3 Navigation Links.......................................................................... 5-16
  5.3.3.4 Evaluation.................................................................................... 5-17

5.4 Summary.................................................................................................... 5-18
Chapter One:  
Thesis Overview

1.1 Introduction

As an introduction to the thesis "Adaptive Intelligent Tutoring Systems", this chapter presents the motivation behind the work and an introduction to the challenges addressed during its research. As a start, this chapter summarizes the motivation behind the research, and provides brief overviews of Intelligent Tutoring Systems, Modelling (User and Student), Intelligent User Interfaces and Adaptation, as a basis for the work.

1.1.1 Motivation

Several pieces of work at the Royal Military College of Canada precede this thesis and in doing so, lay its foundations. The exploration of this foundation, and the ideas presented within Knowledge Representation [Rondeau, 1993], User Interfaces [Banks, 1993], Cognitive Modelling [Polliquin, 1993], User Modelling [Buck, 1996a], and the many articles and books noted below, form the basis of this thesis' motivation.
The challenge to be addressed by this thesis is: "How to build on the foundation of ideas and work available and in doing so, how to develop an Intelligent Tutoring System which is adaptive to the learning style and knowledge state of the student."

1.1.2 Aim

The aim of this thesis is to present research which has been completed towards the development of an adaptive student tutoring system.

1.2 Intelligent Tutoring Systems

As mentioned previously, the research and development of an Intelligent Tutoring System (ITS) was the main challenge to be addressed. To understand this challenge, it is important to sift through the many ideas of what an ITS is, and what distinguishes it from earlier computer based systems.

1.2.1 Definition

Many early definitions of ITS are derived from the ITS predecessor Intelligent Computer Aided Instruction. "Any computer program that uses artificial intelligence techniques to help a person learn" [Knezek, 1988]. "A computer program able to assist a student in mastering some subject matter domain" [Sleeman, 1982].

A more recent definition offers a distinction between ITS and its predecessors: "the ability to diagnose student errors and tailor remediation based on the diagnosis represents the critical difference between intelligent and merely clever automated instructional systems" [Shute, 1995].
With these definitions in mind, the following description represents a succinct picture of the ITS developed within this thesis:

*An Intelligent Tutoring System is a computer program which is adaptive to the knowledge needs and learning style/ability of the student.*

1.2.2 History

The introduction of Tutoring Systems began with the first Computer Aided Instruction (CAI) systems. In the beginning these systems were primarily 'linear programs' that would output a frame of text which would move a student one step closer to the desired behaviour or knowledge. Later, the 'branching' of these programs, based on the student's responses, allowed for better corrective feedback. The CAI was brought to a new level of sophistication when the program's ability to generate the required teaching material/problem sets, from information provided by the expert was developed in the next phase.

To that point in ITS history, none of the CAI systems had human-like knowledge of the domain they were teaching, nor could they answer serious questions of the students as to 'why' and 'how' the task was to be performed. However, with the introduction of 'production systems' (organization of facts, rules, and inferences), ITSs are now able to model people's/students' behaviour [Yazdani, 1986] and knowledge. It is with these models that the ITS can begin to intelligently adapt to the requirements of the student.

1.3 Adaptation

For an ITS to be adaptive to the student's needs it must use its knowledge of the student model to determine how information is to be presented and evaluated. Therefore, the system
must be able to adapt its behaviour based on current and subsequent input actions and knowledge of the student. Fischer has outlined some of the major characteristics of adaptive systems [Fischer, 1993] (see chapter 4). In all aspects of adaptation, it is these characteristics on which JASMINE has been based, in comparison to those of adaptable systems, which are directly manipulated by the user.

By using one of the classification models proposed by Dieterich, the techniques within JASMINE can be described as either self-adaptation or user-controlled self-adaptation [Dieterich, 1993]. In particular, movement between presentation techniques is controlled fully by JASMINE and based on the Neuro Linguistic Programmation Test and student inputs, while navigational movement between concepts, although recommended by the system, is at the control of the student. While in both cases JASMINE is responsible for initiating, proposing and executing the changes, in the User-Controlled model the user has some input in the decision to complete execution of the adaptation.

As implied above, the main area within JASMINE where adaptation is encountered, is as a result of the student learning style. Hence, JASMINE uses two of the three main adaptation strategies pointed out by Dieterich. "adaptation before first use of a system" (pre-test) and "adaptation during use" (system observations and stereotype triggers) [Dieterich, 1993]. The third strategy, "adaptation between two sessions" was deemed redundant for JASMINE since the second should quickly cover student differences between sessions based on previous modelling.

1.4 Modelling

Although there are many types of modelling, the focus within this thesis is with user modelling and student modelling. User modelling is presented as a precursor to the more specialized student modelling which is of importance to this thesis and the resulting ITS.
1.4.1 User Modelling

User modelling, as the name implies, is a method of modelling a particular user's, in our case a student's, use of a specific automated system. Normally, the information to be modelled includes the user's desires or goals, beliefs, and the system's beliefs of the user's knowledge [Buck, 1996a]. The information can be derived for the user as an individual, or as a member of a group or stereotype. In addition, the system's determination of the user can be set after some initial analysis (e.g., interview/questionnaire) or modified by some subsequent action of the user.

Wahlster [Wahlster, 1989] provides a basic but useful definition of the workings of a user model:

"a user model is a knowledge source in a ... system which contains explicit assumptions on all aspects of the user that may be relevant to the ... behaviour of the system. These assumptions must be separable by the system from the rest of the system's knowledge."

One such system, which may be of interest to anyone who has had trouble selecting a novel to read, is GRUNDY [Rich, 1979]. GRUNDY uses basic questions to guide the stereotyping of the user toward the types of books which are available at the library and of interest to a particular stereotypical group.

The construction of user modelling shells has begun during the 1990's along the lines of many application shells. Although not used in the final analysis, one shell in particular, BGP-MS, is discussed later in this thesis. The main feature that was to be exploited from BGP-MS was its construction and maintenance of user stereotypes through the use of "initial interviews, observed user actions, and [pre-loaded] knowledge about predefined user subgroups" [Kobs, 1995]. Based on the user's actions, and the rules of retention and retraction preloaded with the stereotypical user groups, BGP-MS would then have modified the user's assigned stereotypical group. In the case of this research, the preloading of three groups, the auditive, visual and kinetic (hands-on) users, is determined through Neuro Linguistic Programmation.
(NLP) testing, which is provided by an initial interview and maintained by observing the remediation requirements of the student.

1.4.2 Student Modelling

Student modelling is a specialized application of user modelling which goes beyond the use of stereotyping and system beliefs of user's goals and wants. Student modelling introduces the idea of recording the user's knowledge state, as perceived by the system. It can be the job of the ITS to model this knowledge, both in the area and level of mastery as related to the expert model (as with this research), and in the monitoring of misconceptions.

The concept of student modelling has been defined as verbosely as, "[a model which] represents student understanding of the material to be taught with the purpose to make hypotheses about student's misconceptions and sub-optimal performance strategies" [Barr, 1981], and as succinctly as, "[an] artificial student simulating the development of student knowledge" [Self, 1986].

In whichever manner a student model is defined, its purpose is to not only allow an ITS to provide information and problems, but to adapt, as necessary, to the students knowledge state. Although not developed completely within this research, the student model can also be used to solve problems, if necessary, as the student would; hence, 'to understand' the misconceptions of the student. An example of early use of this technique is SUMIT, which provides a domain expert which not only follows the traditional teaching progression by offering correct advice to the user, but also diagnoses and remedies the most common bugs [Nicolson, 1992].

The student model used within this research, inspired by the ideas of modelling:

a. records the knowledge of the student in regards to the expert knowledge of concepts and rules regarding application of the concepts
b. records the level of mastery of the student in each of these concepts  
c. maintains the system's beliefs of the student's learning style and abilities  

As a result, student modelling within JASMINE, enables the determination of remediation and evaluation needs and the most appropriate presentation interface.

1.5 Intelligent User Interface  

Although it represents only a small part of most applications, the user interface is the most visible, and therefore, the most important portion of the ITS to the user. Many of us can remember an application we have used which was great in ability but very difficult to operate (interface with). In developing any system, the "iceberg" metaphor [Tyugu, 199] or 10% rule, can be a great 'aide-memoir' to help the developer focus on the user.

Figure 1-1: Iceberg Analogy of User Interfaces
The Intelligent User Interface (IUI) goes beyond user controlled adaption of the available menu and window (e.g. sizing, position, etc.). By introducing user modelling techniques, the system itself is able to adapt to the user's requirements. As Goodman [Goodman, 1993] explains, the use of multimedia goes even further in adapting to the user by incorporating a combination of graphics (e.g. video, speech and sound) to convey information. The implementation of these tools through planned presentations [Goodman, 1993] and communication acts beyond linguistic to graphical and media-independent rhetorical acts [Maybury, 1993], brings a clearer picture of the truly adaptive student environment to ITS. Graphical acts combine such items as highlight, pan, draw, and animate-action, with linguistic speech/audio and tests to improve the explanation tools of an ITS.

To enhance interaction with the student, this research attempts to employ the ideas inherent within a good IUI. Although the input interface remains rather simple (point / click, arrow keys, simple commands), JASMINE provides an interface (particularly output) which allows the system to adapt to the three types of student learning (auditive, visual and hands-on) addressed within this thesis. Navigational output is also adaptive to the student and based on the knowledge state of the student. This navigational output is accomplished through colour coding of navigation paths based on mastery, and pedagogical recommendation (including remediation). In addition, through the use of the HyperText Markup Language, the framework allows for the incorporation of multimedia as desired by the expert within the application.

1.6 An Application

JASMINE (Java Adapted Student Modelling In a NLP Environment) is a research system built to demonstrate the use of student modelling and adaptive techniques (e.g. interface presentation) within the military training domain. However, the tools and framework allow for any domain which can be represented with a hierarchical training structure, to be instantiated by a domain expert or developer. In the case of this thesis, the domain used was a small portion of
the Security domain of the Canadian Force's Officer Professional Development Program. Although the initial intent was to utilize the BGP-MS user modelling shell system to implement the system core of knowledge, as will be discussed later, it was determined that the JAVA development system provided all the tools required. In producing an adaptive student interface, the developed JAVA tool set is weaved with HTML to provide a multimedia capable front-end.

1.7 Thesis Outline

This thesis has been organized into six chapters, moving from the ITS architecture to the specifics of JASMINE, while examining influences on the system's development along the way.

Chapter Two provides the basis of the ITS portions of this thesis. It presents a number of applications through reviews of ITS Architecture and Teaching and Learning Strategies, as well as a more in-depth look at those applications that have a direct influence on JASMINE. In addition, the chapter presents an ITS modular structure which was exploited during JASMINE's development.

Chapter Three provides the basis of modelling (knowledge, user, and student) to be used. A survey of knowledge representation techniques and a discussion of the Beliefs Goals Plans Maintenance System (BGP-MS) shell as related to the initial intention of the research, is provided. Finally, the JAVA development environment is explored with reference to those features exploited within JASMINE.

Chapter Four provides discussion on system adaptability, from the building of the initial learning model and its modification through user actions, to changes in presentation and requirements of remediation. After the initial introduction, the majority of discussion revolves around two IUI design questions, "Direct Manipulation vs Intelligent Agents" and "How Much Intelligence?". This is followed with a presentation and discussion of two IUI systems which
have had a direct influence on JASMINE.

Chapter Five provides a detailed description of JASMINE, including a description of the domain, each of the tools developed, and the areas in which ideas from the previous chapter have been applied.

Finally, Chapter Six discusses the thesis's results and provides some recommendations and suggestions for further study and for further JASMINE development.
Chapter Two: 
Intelligent Tutoring Systems

2.1 Introduction

As described in the introduction to this thesis, the main focus of research has been the development of some portion of an Intelligent Tutoring System (ITS) which will create an adaptive student environment in relation to presentation, remediation and evaluation. However, an understanding of the ITS structure was needed before the required portion, for development of JASMINE, could be selected. Therefore, this chapter presents an integrated, albeit typical, ITS structure and presents its modules as they relate to JASMINE. In addition, two brief surveys related to ITS Architecture and Teaching and Learning Strategies are presented. This is followed by two, more detailed, example systems, SUMIT [Nicolson, 1992] and SMART [Shute, 1995], both of which have had a direct influence on this thesis.

2.2 Intelligent Tutoring System Structure

Several ideas concerning the composition of an ITS system can be found in the literature. The following ITS model (figure 1), modified from Rondeau [Rondeau, 1990], generally follows
Yazdani [Yazdani, 1986], Burns [Burns, 1991], Marcenac [Marcenac, 1992], and others, in its structure. For example, Futtersack [Futtersack, 1992] describes a standard architecture for an ITS as, the expert module, the pedagogical module, the student model and interface which, as a whole, simulate a sole, but multi-function intelligent entity.
2.2.1 Pedagogy Module

The first module, the Pedagogy Module, is generally responsible for interpreting the knowledge and skill level of the student and for providing the best pedagogical plan available. Within this plan, the module will direct the use or production of student exercises based on the expert knowledge [Futtersack, 1992] [Burns, 1991]. The chosen exercise may be either the next topic which has not been discussed within the curriculum, or a topic which the system once believed the student had mastered, that has now fallen below the mastery level. The module then combines this information with the current student attributes to determine the best presentation avenue. As well, within any given topic area, mastery criteria will be determined and the student evaluated through the use of multiple choice questions and other methods as necessary.

Within JASMINE, the pedagogy involves the provision of a recommended course of navigation through the material, with the course of navigation based on student mastery of base concept information. Timing of evaluation is also at the discretion of the student. Evaluation may take place when desired and will be based on the student's location within the material, and the current mastery level of related base concepts.

2.2.2 Psychologist Module

The next module, the Psychologist Module, as illustrated in the above diagram, is responsible for evaluating the results of student responses and for providing for the maintenance of the student model based on observations made by the system.

Within JASMINE, the majority of student responses will be in the form of true or false, multiple choice, and fill-in the blank questions. Therefore, the evaluation will generally be straightforward (ie. correct or incorrect response) resulting in a modification of the base concept mastery level. However, adaption of the system's belief in regards to the student's learning style...
may also occur if student difficulty is detected over a set of evaluation questions.

2.2.3 Interface

One of the crucial modules in any approach to ITS design is the Interface Module. Burns states that the overall goal of the ITS "is to have the computer and the student gradually become partners and learn how to manipulate content and relationships in order to come to an agreement about the knowledge they share" [Burns, 1991]. As such, one of the main problems that must be addressed in the design of an ITS is how to design a system that manages the interaction between student and computer. It is the Interface Module that is responsible for this interaction and all the media necessary for communication.

Within JASMINE, the interface is to be presented through the use of ad-hoc JAVA generated HTML pages. The ad-hoc procedure creates a presentation from the available subject information based on the student's learning style and added navigational links based on the mastery levels of related information. In addition, a JAVA applet, communicating with the Pedagogy and Psychologist portions of the system, is used to generate an evaluation session for the student on request.

2.2.4 Domain Expert Module

The next module, as is indicated in the above model (figure 1), is generally an information repository for the expert knowledge. The main purpose of the ITS is to pass this knowledge to the student. As a point of interest, any questions asked of the student by the Psychologist module should have an answer found within the expert knowledge.

In addition to the base concept knowledge found within the Expert module, JASMINE builds an instructional founded hierarchical domain relationship of this knowledge. It is this
domain relationship which becomes the basis of the recommended course syllabus. To aid in the management of expert knowledge and syllabus, JASMINE provides a Domain Manager which presents the hierarchical structure of the instructional domain to the course developer. The structure can then be managed through the addition, modification, and deletion of concepts and their related expert knowledge. Once constructed, the domain information can be used by both JASMINE’s Question Manager tool, to develop the evaluation question bank, and the Student Server, which maintains and manages the student module and activities during a tutoring session.

2.2.5 Student Module

The final module, the Student Module, like the expert module, is an information repository for system beliefs of the student. Generally, two types of information are stored within the student model. The first type of information required by the system, is any information used to provide adaption based on student attributes (e.g. abilities and learning style). The second type of information is the student's knowledge state in relation to the expert knowledge. This relationship is often based on one of three student overlay models, the basic overlay, differential overlay, and perturbed overlay (chapter 3). Current work in student modelling can be found in several references for this thesis. In addition to the detailed review of papers having a direct influence on JASMINE [Nicolson, 1992] [Shute, 1995] [Brusilovshky, 1996] [Milne, 1996], two papers directly related to student modelling are reviewed in greater detail (chapter 3).

Within JASMINE, the student module is truly straightforward. The student's knowledge and knowledge level, as well as the student's characteristics (of particular interest here is the student's learning style) are managed within Student Server's psychologist and are structured in a manner similar to the expert model. As a result, a modification of the differential overlay model has been used. The "Amplitude Differential" (chapter 3) is a model which not only records that a student knows the concept, but also records the system's belief as to the student's mastery of the
2.3 A Review: ITS Architecture

The first of three brief reviews\(^1\), based on ITS developments presented at the Third International Conference, ITS '96, involves ideas on the changing architecture of the ITS. This section reviews the work presented in three areas of ITS Architecture: "An Actor-Based Architecture for Intelligent Tutoring Systems" [Frasson, 1996]; "A Highly Flexible Student Driven Architecture for Computer Based Instruction" [Gagné, 1996]; and "LearnMedia: A Cooperative Intelligent Tutoring System for Learning Multimedia" [Wang, 1996].

2.3.1 Actor-Based Architecture

The first architecture to be reviewed in this survey involves the use of intelligent agents or learning actors which are reactive, instructible, adaptive and cognitive. In the design of this architecture, the authors were attempting to follow what they called the "evolution of [ITS] towards the use of multiple learning strateg[y] calls on multi-agent architecture" [Frasson, 1996]. Although the end goal of the authors was to meet the multiple learning strategy portion of the design aim, the original tactic had been to focus on the agent architecture.

The actor architecture presented (fig 2-2) by the authors involved the interaction of four modules (Perception, Action, Control and Cognition) across three layers (Reactive, Control and Cognitive). In addition, actors interact with each other through the use of external viewing into the privileged internal view of other actors (eg. previous behaviour).

\(^1\) the second review is found in the next section of this chapter, while the third is located in chapter 3.
Perception module, as the name implies, detects, through the use of external viewing, changes in the system environment. These changes are compared with a set of typical situations and a choice is made as to whether the actor might intervene. Intervention, if it occurs, may initiate activity across four functioning modes: reflex (directly with Action module), control (through control module), reasoning (through control and cognition modules) and learning (directly with cognition).

![Conceptual architecture of an actor](image)

**Figure 2-2: Conceptual architecture of an actor (modified from Frasson, 1996)**

- Action module consists of all those actions / tasks with which the actor may operate on the environment.

- Control module handles situations which require some manner of planning ie. the choice of alternate actions or series of actions.

- Cognition module, when developed by the authors, will have the goal of improving the actors. Each task of this module will be permanently running and assigned the responsibility of improving a specific aspect of the actor.
At the time of publishing, the authors had developed a basic prototype using the Smalltalk object-oriented language and were focusing "work on the implementation of the cognitive layer and on the refinement of the pedagogical expertise of new tutoring strategies" [Frasson, 1996]. Current communication with the authors has indicated that they "are still trying to see what learning mechanism could be implemented [in the cognitive layer]" [Frasson, 1997].

The use of agents² within this architecture was of interest because the authors have modularized the typical ITS function within student, teacher and tutoring aid actors. Some of the same modularizing is seen in JASMINE (eg. the evaluation and navigational guidance can be described as being provided by the activation of primitive agents), although not to the same degree. Work of even greater interest remains to be seen as the cognitive features of the actors are developed: a development which is hypothesized to enable the actors to improve their performance throughout their lifespan.

2.3.2 Student Driven Architecture

The next architecture under review "proposes a domain independent framework for computer based instruction systems capable of generating custom courses" [Gagné, 1996]. Custom course development is based on varying learning objectives / standards and student assessments.

It is the authors' belief that "the student is the best model of himself" and therefore, focuses on a 'learner empowerment approach'. The author's work is presented in contrast to the ITS paradigm where instruction is tailored based on a model of the student, built and updated by the system.

² A further discussion of intelligent agents is found in Chapter 4 of this thesis.
In an effort to enable 'learner empowerment', a framework that separates the pedagogical organization of the content, and the media used to convey the different concepts, is proposed. This framework is built from a corpus of knowledge used to initiate course-based concept grouping (units) and the competency profile set for the student.

The corpus of knowledge consists of a collection of concepts, defining or relating to the subject matter, which is associated with levels of expertise either directly or hierarchically. In addition, precedence relations are specified between concepts.

The competency profile consists of two parts: the learning objectives set by the course expert, and the student assessments of concept knowledge levels. It is interesting to note that the authors felt that the accuracy of the student assessments is not crucial, as they can be easily modified during course delivery.

Once the student's level of expertise has been assessed, the pedagogical planner completes a gap analysis and recommends a series of concepts to be studied; in essence creating a custom course. The pedagogical planner completes the gap analysis by visiting all the ancestors of a given concept $c$ before visiting $c$, with the student providing direction whenever more than one concept can be visited next. Modification is initiated by the student at any time it is felt the material is either too easy or difficult, and is accomplished by changing the student assessment. Hence, the initial knowledge gap analysis is modified and the course re-established.

The basic ideas presented by the authors of this framework are similar to those presented throughout this thesis; however, at the time of publishing the authors had only completed a basic prototype of the framework and as such are not convincing in regards to two claims. The first claim is that a student model is not used within the framework. Although no explicit model is built or maintained, the implicit model can be seen through exclusion. First of all, the student assessment is created and maintained through the life of the course, providing the pedagogical planner with student requirements. Second, other student goals are recorded through objectives.
and expertise levels. Although much of this material is maintained manually by the student, it is the same material found within an automated student model. Hence, the framework would seem to contain a student model even if by direct omission.

The second claim in question states that "the proposed framework can support ... exploratory learning" [Gagné, 1996]. Given that the framework employs three levels of precedence (concept, unit and mediatic learning) and that a demonstration has not been given, this claim is premature.

### 2.3.3 Co-operative Architecture

The architecture of the last paper under review in this survey is based on the premise that "today's state-of-the-art ITSs still lack the collaboration and co-operation aspects in a real classroom" [Wang, 1996]. As a resolution to this perceived lacking, Wang proposes an architecture built on three layers, the educational agent layer, the knowledge server layer, and the repository layer. This architecture, Wang claims, "provides the means for multiple educational agents to conduct co-operative and collaborative interactions between students and teachers and among students."

As laid out in figure 2-3, the proposed architecture is reliant on three types of independent agents: the educational / student, teachers and technical agents (which may include other information providers); each of these agents must be able to inter-operate in order to achieve the "grand vision".

The function of the knowledge server is to facilitate communication between application agents and the repository, and between agents and the repository. For example, "when [students] needs to ask any question, [they] ask the agent to pop-up a Questions form to fill. After the form is filled, an instance of Question [is] created locally and [sent] to the Knowledge Server... [the
teacher in turn] fills in an answer form and send[s] the answer back to the sender [and] all the students who are in the same class as the sender" via the knowledge server [Wang, 1996].

![Diagram of the architecture of LeanMedia](image)

**Figure 2-3: The architecture of the LeanMedia [Wang, 1996]**

The knowledge server also stores such information as the navigation history of a student, the problems that a student had during exercises, and the questions that a student asked in the repository layer. This information then becomes available as required, and is relevant to the management, evolution, operation and maintenance of the architecture (eg. teacher agents can query the repository about student activities and difficulties).

At the time of publishing Wang's article, a prototype tutoring system LearnMedia had been developed which presented information in the form of hypermedia documents. It is interesting to note that Wang's proposal is not concerned with the pedagogical intelligence of a classic ITS; "students are free to browse and navigate through the hypermedia space following the hypermedia links" with no perceived guidance. However, Wang does give the example of "a student agent that serves as a college-level physics student discover[ing that] the student has a particularly hard time solving differential equations". Wang suggests that the agent "may be wise" to communicate this difficulty to another agent serving as a mathematics teacher.
However, Wang does not explain how the 'wise' portion of his ITS is to operate.

In closing, it is agreed that if the ITS is to simulate more closely the actual classroom, then it must "be able to incorporate the ability for such collaborative learning interactions, which might be among students, among teachers, or between teachers and students" [Wang, 1996]. It should be noted, however, that this may not be apparent in the definition of today's 'intelligent tutor'.

2.4 A Review: Teaching and Learning Strategies

The second review area, taken from ITS developments as presented at the Third International Conference ITS '96, involves newer ideas on ITS Teaching and Learning Strategies. This section reviews the work presented in two areas: Learn-by-Doing (or role playing), and argument development. Two papers have been chosen for this review based on their relevance to listed research areas: "Towards the Design of More Effective Advisors for Learning-by-Doing Systems" [Katz, 1996], and "Arguing with the Devil: Teaching in Controversial Domains" [Retalis, 1996]. Before beginning the review of Katz's work, it may be beneficial to review some of the more conventional strategies as defined by Srisethanil [Srisethanil, 1996].

Srisethanil, describes four strategies which are used within the method adaptive ITS called ITS-Engineering:

1) Instructor-Oriented. Promotes 'reproduction' of knowledge via stimulating memory recall or task repetition. The instructor "directly presents the information, controls the pace and rhythm of the instruction, provides feedback, and evaluates the performance of students". This strategy is best used for declarative and procedural knowledge.
2) Guided Discover. Promotes 'production' of new knowledge by inviting the student to think, infer and go beyond the current information to new information whether correct or incorrect. The instructor interacts with the student in such a manner that the "teacher's sequences of information and questions cause corresponding responses that are then discovered by the learner". This strategy is best used for declarative knowledge.

3) User-Initiated. Promotes the student selection of topic, which allows the student to solve problems in relation to the student's emphasized interest. This type of strategy requires little direct interaction from the instructor and is best used for the acquisition of "procedural knowledge requiring the solving/computing [of] mathematical problems".

4) Exploratory. In addition to the freedom of the User-Initiated strategy, this strategy promotes control of the learning path and pace of instruction via student selection of "topics to learn without specific sequence". This strategy is best used for "non-procedural knowledge such as fact and conceptual knowledge". As a point of interest, it is the exploratory strategy which is most employed within JASMINE.

2.4.1 Learn-by-doing

The first paper in the review of learning and teaching strategies is based on the strategy of learning-by-doing and is presented in the context of a system called Sherlock II [Katz, 1996]. In a learning-by-doing system, sometimes referred to as 'role-playing' [Slator, 1996], "learning is produced by extended engagement of learners in complex cognitive activities, often involving peers, experts, teachers, and intelligent learning systems as partners" [Katz, 1996]. As such, the learn-by-doing system is contrasted with a totally exploratory environment which may "[stimulate] learning among peers", but lacks both an "advisor to explain and scaffold learning."

Using the premise that an effective advisor brings the learn-by-doing system beyond the simple exploratory system, the designers of Sherlock II focused on the advising capabilities and,
in particular, attempted to determine which coaching resources the students use more. Attempts to accomplish this focused on the use of Sherlock II\(^3\) and its coach Sherlock.

During study of the student's resource use, the exercising of advice use was left with the learner. Students had control over the type of coaching and the level of detail, with the coach intervening only if the student committed a simulated safety violation. From this learner control it was determined that of the three types of advice available (How to test, 'procedural knowledge'; How it works, 'conceptual knowledge'; and Technical data), use of 'How to test' far outweighed the other two at 93% of all advice sought.

Through observation of student interaction the authors' felt that the 'How it works' and 'How to trace' hints did not satisfy the student's desire to find out quickly what to do and that "understandably, most students took the more direct route to strategy advice offered by the How to test hints" [Katz, 1996]. When used with a student coach, however, students of Sherlock II were directed to their own solution when the coach would "[request] interpretation of readings, check the solver's understanding of how a [component] works, direct the solver to How it works advice on [components] when poor understanding [was noticed] and summarize Sherlock's advice" [Katz, 1996].

As a result of this study, the authors report that an advisor for effective learn-by-doing must "get students engaged in the process of overcoming an impasse, beyond giving them control over the type and amount of information they receive" and "facilitate the integration of procedural and conceptual knowledge". In addition, it is their feeling that more research is needed on how human tutors do these things, and how they encourage learner self-reliance in general. It was suggested that hypertext may be part of the solution by allowing information to be parcelled from general to specific packets in context, making it easier for designers to build a

---

\(^3\) Sherlock II is a computer simulation of the actual job environment where the student runs a set of checkout procedures on a simulation of a test station.
scaffold of advice.

Learn-by-doing is a strategy which has been used since the dawn of creation. Therefore, there is nothing new in this basic strategy; the findings that students will generally take the easy route is intuitive. However, the authors focus on providing a more well-rounded advisor is interesting. An effective advisor which will move the system from simply providing a safe environment for experimentation to one which provides a deeper understanding and true problem solving skills to the student could only be beneficial. This is apparent when the student becomes an actual equipment operator and is faced with the problem of solving a unique situation that the simulation was unable to provide.

2.4.2 Devil's Advocate

The second area of interest visited during this review is concerned with argument development, and details the introduction of the Devil's Advocate strategy into an ITS. The work reviewed here is based on the premise that "teaching in controversial domains can be problematic" [Retalis, 1996]. It can be problematic in the sense that there are multiple points of view which in themselves have not proven to be the correct one.

In such situations of multiple points of view, the Devil's Advocate strategy "encourages students to explore different and often conflicting opinions, consider their justifications and weigh up the various arguments for and against a position" [Retalis, 1996]. Using the Devil's Advocate, the student debates with the system about a topic by choosing one of the conflicting points of view and exploring the expert arguments related to the view chosen. In contrast, it is the authors view that traditional tutoring systems focus on teaching only a single 'expert' view.

An important aspect of the Devil Advocate used within the authors' prototype system called OLIA, is that there is no attempt to win the argument or solve the conflict. The strategy
just enhances the student's experiences with the arguments. The authors assume "the students are not trying to decide which of two conflicting points of view is 'correct', but rather are learning about the nature of the controversy thus enabling them to weigh up the various sides of the argument". In addition to the provision of lower level arguments, OLIA provides the student with the ability to view the argument via 'higher order meta-level relations', (eg. rebuttals, counter arguments, etc) between student and system propositions.

As a result of the initial studies of OLIA, two items are of particular interest in this review. The first is the students' preferences for the Devil Advocate strategy in exploring the domain when given the choice in using OLIA's basic coach. The second is on the authors' acceptance that a facility which would allow the student to "express their own personal arguments and opinions" may be of benefit for widespread use.

As a final point, the authors of OLIA indicate that further analysis as to this approaches affect on improving student learning. However, no mention is made of the pragmatic problems of development in the current system programming language, Prolog (see section 2.5.1.1).

2.5 Examples (Influences)

Now that two general reviews of current work in the ITS field have been presented, this section will take a deeper look at two existing ITS applications which had a direct effect on JASMINE. As will be discussed below, SUMIT's focus is on knowledge of the subject to be taught, the user's current understanding, and the appropriate teaching strategies, in an effort to provide a pragmatic system targeted on learning and the user interface. This focus has had a particular influence in the pragmatic design of JASMINE, while, SMART's use of regression equations, in an attempt to provide more responsive remediation of material, has been used in JASMINE's implementation of knowledge level management.
2.5.1 SUMIT

SUMIT [Nicolson, 1992] was explicitly designed for the domain of arithmetic teaching, and developed to operate as a teacher's assistant (i.e. it was not designed to remove the need for actual teachers). In their bid to grow beyond what had been traditional Computer Aided Learning programs, the creators of SUMIT focused on the idea that an adequate ITS needed (at least) three components: knowledge of the subject to be taught (domain knowledge); knowledge of the user's current understanding (student model); and knowledge of the appropriate teaching strategies for the domain and user state (pedagogic knowledge) [Hartley, 1973]. As discussed earlier in this thesis, these three areas of concern focused upon by SUMIT, along with interest in the psychologist and interface issues of the modern ITS, have been adopted in the development of JASMINE.

2.5.1.1 Ideas

With the desire to meet the need for these three components (domain knowledge, student model, and pedagogic knowledge), the creators of SUMIT analyzed the attempts of Brown and Burton's development of the BUGGY ITS. BUGGY was to focus on the diagnosis of previously found bugs within arithmetic and from this diagnosis, develop an intelligent tutor [Brown. 1978].

Nicolson identified several problems within BUGGY which had an influence on SUMIT. Of those problems some have had a direct influence on JASMINE and are discussed below.

a) **Lack of the appropriate pedagogical knowledge.** As implied above, BUGGY was designed mainly as a diagnostic tool. This choice was made because, with some domains, it is important to know why the student went wrong in order to correct a misconception. However, in other domains, such as those JASMINE was designed for, it is more expedient to determine when remediation of the problem is required.
b) **Lack of immediate feedback.** Within Buggy it was decided to look at groups or sets of mathematical questions. This was to allow easier identification of consistent bugs from the BUGGY bug pool. Although this may be practical where problems are difficult to identify or possibly where the system developers want to be sure of the problem, Nicolson felt that this type of approach failed to utilize a major strength of a computerized system, immediate feedback. As a result, JASMINE has been developed such that feedback is provided based on the previous and current observed beliefs of the student's knowledge state. JASMINE provides immediate feedback, in the case of an incorrect response to evaluation, in the manner of guidance to the correct response and recommended remediation, as required, through the change of navigational links.

c) **Pragmatic problems.** Nicolson felt that as "...BUGGY was written in InterLIST and worked on a PDP-KL10, [it was] highly unlikely that it could be ported to a common educational computer". To avoid this problem JASMINE goes beyond SUMIT's idea of availability on a common educational platform and incorporates the ideas of the World Wide Web (WWW), which of course was not directly available during development of SUMIT (1980s). The use of JAVA programming and HTML interfacing allows the system to be implemented using a JAVA-enabled, platform independent of client (eg. PC, Mac, UNIX w/Web Browser). In doing so, JASMINE has alleviated the pragmatic problems identified.

From the problems discussed above, Nicolson provided a list of what should be expected of an 'arithmetic' ITS. Although originally identified for the 'arithmetic' domain, many of these ideas can be, and should be used in most ITS and are discussed in relation to JASMINE.

a) **Target learning and User Interface.** Students with minimal understanding of the rules of the domain should be able to use the programs and to progress through the stages of learning. JASMINE's operation design was developed, wherever possible, to be easily
navigated with a small amount of standard computer interface manipulation knowledge (eg. turning of pages of information, checking of boxes for evaluation, saving of session for later use). Navigation, hence the interface, becomes domain independent.

b) Feedback. As discussed earlier, feedback should be as helpful and immediate as possible.

c) Pedagogic expertise. The ITS should not only have knowledge of how to diagnose and remedy bugs but how to determine the solution correctly. In the case of JASMINE, this is accomplished by providing the correct answers to the pedagogic function during the establishment of the question bank. It was felt that the determination of a solution was better left for domains which have a large degree of procedural knowledge which may be misinterpreted by the student. Within the domains JASMINE is designed for, a large amount of the knowledge is conceptual; therefore, knowledge is known or inferred by the student. JASMINE however, does retain a system belief model of the student's knowledge and therefore, can predict when the student should know or should not know the solution and hence provide feedback.

d) Motivation. The ITS must of course be motivating and interesting to use. If the student finds a simple reading of the material just as effective as the ITS, then the overhead associated with the system may become a de-motivating factor. Within JASMINE, it is believed that with adaptation to the student's navigational requirements (including remediation) and prominent input channel (chapter three) the student can be motivated along the most direct route of learning and retention.

e) Pragmatic. As discussed earlier, any complete ITS should be available to the common student's platform of choice (PC, MAC, Unix).
2.5.1.2 Results

Early SUMIT research called SUMS "was aimed at a synthesis of the traditional approach to teaching arithmetic with the ITS approach by producing a 'glass box' domain expert which followed precisely the traditional teaching progression but was able not only to offer correct advice to the user but also to diagnose and remedy the most common bugs" [Nicolson, 1992]. Through this diagnosis SUMS provided immediate feedback; and with a focused tutoring style, SUMS forces "the [student] to perform the necessary operations in the 'correct' order, and refuses to accept incorrect input" [Nicolson, 1992]. As a result, several of the bugs earlier identified by Brown and Burton could not and did not occur.

Although JASMINE takes on this focused approach to learning, its availability of evaluation, immediate feedback, and remediation through recommended navigation are the keys to limiting the development of bugs within the student's knowledge.

The design ideas of SUMIT presented in this section are those having the most influence on the development of responsiveness, portability, and adaptability within JASMINE. The remaining example systems, presented throughout this thesis, demonstrate how these ideas can be developed to a pragmatic level and employed within an ITS such as JASMINE.

2.5.2 SMART -- Responsive Tutoring

As mentioned above, the Pedagogy Module is responsible for determining the next recommended topic of action in the course structure. Within JASMINE, the topic may be the next topic which has not been explored or mastered, or a topic which has fallen below the mastery level after the system believed the student already mastered the topic. For the purposes of this approach, the ideas of SMART [Shute, 1995] (Student Modelling Approach of Responsive Tutoring) have been examined. As a point of interest, SMART was initially designed as an
attempt to span a broad range of instructional domains and was first implemented within the statistics tutor called Stat Lady [Shute, 1994]. As described by Shute, SMART was developed to answer the question: "How can Stat Lady be modified to yield more guarantees of successful knowledge and skill acquisition?" [Shute, 1994].

Through the use of the SMART approach, expert concepts to be passed on to the student are assigned a level of mastery and instructed or remediated as necessary. Shute's assessment involved a focus on working-memory capacity, associative learning ability, and impulsivity. As is discussed in chapter 3, JASMINE focuses on the auditory, verbal, and kinetic learning styles of Neuro Linguistic Programming through an initial assessment and later adaptation based on student successes. As with SMART, the focused areas of student ability are used to refine and adapt the learning strategy as necessary throughout the student's session. Stated in a more precise form, the SMART approach combines macroadaptive techniques (assessment of the student's knowledge and skills before tutoring begins) and microadaptive modelling (representation of emerging knowledge and skills of the student). The ideas of microadaptive modelling are one of the more important aspects used within the JASMINE system and are described in more detail in the next section.

2.5.2.1 Representation of Knowledge Level Change

Although the framework of SMART does focus on a different set of learning variables, the resulting system knowledge state's promotion and demotion techniques should be effective across many sets of learning styles. In the development of these techniques, Shute started with heuristics to establish a table (Table 1) as a visual description for the rules of promotion and demotion.

It should be noted that changes on level are based on the student's ability to answer evaluation questions and the degree of assistance (level of help) required from the tutor. For
example, a student at the current state of Low Mastery and requiring Level-2 help would be demoted to High Intermediate, while a student at the current state of Low Intermediate requiring Level-1 help would be promoted to High Intermediate.

The next step taken by Shute was to assign numerical values between 0.0 and 1.0, representing the probability of complete mastery, to the heuristic table. For example, skill levels were given range values, such as 0.33 - 0.50 for low-intermediate, and the resulting new states were given single values within the level value range (eg. changes from high mastery at a level-3 help were assigned a value of 0.42 within the range of low-intermediate).

**New State**

<table>
<thead>
<tr>
<th>Current State</th>
<th>Level of Help</th>
<th>Level - 0</th>
<th>Level - 1</th>
<th>Level - 2</th>
<th>Level - 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remedial</td>
<td>Low</td>
<td>Low Intermediate</td>
<td>Low Intermediate</td>
<td>Low Remedial</td>
<td>Low Remedial</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>High Intermediate</td>
<td>Mid-Low Intermediate</td>
<td>Mid-Low Remedial</td>
<td>Low Remedial</td>
</tr>
<tr>
<td>Intermediate</td>
<td>Low</td>
<td>Low Mastery</td>
<td>High Intermediate</td>
<td>High Remedial</td>
<td>Low Remedial</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>High Mastery</td>
<td>Low Mastery</td>
<td>Low Intermediate</td>
<td>High Remedial</td>
</tr>
<tr>
<td>Mastery</td>
<td>Low</td>
<td>High Mastery</td>
<td>Low Mastery</td>
<td>High Intermediate</td>
<td>Low Remedial</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>High Mastery</td>
<td>High Mastery</td>
<td>Low Intermediate</td>
<td>Mid-Low Intermediate</td>
</tr>
</tbody>
</table>

Table 1-1: Rules for Promotion and Demotion [Shute, 1995].

From the value assignments, Shute plotted the current versus new value for each of the levels of help. Linear, quadratic and cubic curve fits were then performed to determine the best fit curve, resulting in the regression equations (Figure 2) used to represent knowledge level change within the student model.
Figure 2-4: Regression Equations [Shute, 1995].

\[
Y_{(\text{level-0})} = 0.3026 + 1.4377X - 0.7207X^2 \\
Y_{(\text{level-1})} = 0.3316 + 0.2946X + 1.1543X^2 - 0.950X^3 \\
Y_{(\text{level-2})} = -0.0117 + 0.5066X + 0.3518X^2 \\
Y_{(\text{level-3})} = 0.0071 + 0.6001X + 2.5574X^2 - 1.4676X^3
\]

2.5.2.2 Knowledge Level and Remediation

As mentioned, through the use of the regression equations, JASMINE will be able to modify the student's believed knowledge level based on the current knowledge level and the level of success during evaluation. It should be noted that although the general level of master and remediation, for the purposes of equation creation, have been shown at .67 and .33 respectively [Shute, 1995], these figures are not fixed. Through the use of these equations, the levels of master and remediation can be optimized for the learning styles and domain in question.

In addition, the following two rules of remediation, as presented by Shute, can be modified to suit the domain:

a) if a concept has not yet been remediated, and it is < 0.60, and has decreased from the previous value, then invoke remediation,

b) if a concept has been remediated and it is < 0.50, and has decreased across two consecutive trials, then remediate. Shute felt that this "narrowed" the window of
floundering and avoided an endless cycle of solving problem-after-problem.

Within JASMINE, similar rules are used allowing the student to navigate freely with a feeling of some accomplishment while not being marred by immediate and continual remediation due to a small observation window.

As mentioned during the discussion on the ITS modules, student modelling can be represented through the use of one of three overlay models: general overlay, differential, and perturbation [Kass, 1989]. It is due to SMART, and its implied level of knowledge, that the "Amplitude Differential" model has been implemented in JASMINE (chapter three).

The following statement by Shute brings together all the ideas of SMART and is a pattern matching much of what has been designed within JASMINE.

"The SMART approach to intelligent tutoring attempts to facilitate learning by bringing together assessment, diagnosis, remediation, and mastery based learning in a dynamic synergy that will tailor instruction to students' particular profiles of abilities and needs."

2.5.3 Summary

In short, both SUMIT and SMART, through their quest to provide a better training environment, have directly influenced the outcome of JASMINE. In particular, during the development of JASMINE the ideas of a timely, responsive and pragmatic system have been combined with a pedagogic approach which focuses on an understanding of the student's knowledge level, abilities and needs. The following chapters detail how these ideas are incorporated in the design of an ITS.
Chapter Three:
Student Modelling
with BGP-MS and JAVA

3.1 Introduction

This chapter provides the basis for student modelling of both the student / expert relationship and student abilities used within the ITS and JASMINE. Much of the discussion is presented in relation to the features of the Beliefs Goals Plans Maintenance System (BGP-MS) user modelling shell and initially intended for use within the final thesis application.

The first of these chapter topics, the student / expert knowledge relationship, involves a discussion of the knowledge overlay technique as presented by Kass [Kass, 1989]. Further to this technique, and as a result of the remediation technique utilized from SMART, this thesis introduces an adapted overlay model, the "amplitude differential" model, which is used to describe the student model within JASMINE.
This chapter's second point of discussion, on student abilities, introduces the Neuro Linguistic Programming (NLP) approach to learning employed adaptively within JASMINE. The NLP approach allows the system to move away from finding the best 'method of instruction', to discovery of the most appropriate 'learning channel' for the individual student.

Finally, after a discussion of BGP-MS, a look at the beneficial aspects of JAVA is presented. In particular, JAVA's object based approach, platform independence and security features are reviewed.

3.2 Student Modelling

Student Modelling within JASMINE is generally straightforward and involves the use of knowledge overlay and learning stereotypes only. The first of these is what distinguishes student modelling in JASMINE from the more general user modelling. By adapting the presentation of information based on the student's perceived knowledge level in relation to the domain expert, the modelling enables JASMINE to become more adaptive to the individual student. For example, the student model allows for distinction of information required by the student (has not been taught), evaluation of the student's ability, as required, to adjust knowledge levels, and provision of remediation as retention wanes.

3.2.1 Overlay Modelling

One manner of looking at the relationship between the domain knowledges of the expert and student modules (chapter 2) is through the use of overlay modelling. For practical purposes there are three standard views of overlay modelling: the general overlay model, the differential model and the perturbation model [Kass, 1989]. The choice of model applied depends upon that which the student model is overlayed, [Greer, 1996] and on the class of the domain knowledge (eg. conceptual, procedural). In choosing an overlay model it is best to answer the following
questions: Is the overlay model a subset of the expert knowledge or domain knowledge? Is it an overlay of the expected student knowledge? Is it capable of adjusting to knowledge unknown to the domain or expert state?

For example, the designers of QUIZ felt that "an exact modelling of the student's cognitive structure [did] not appear of primary importance", therefore, the overlay model was chosen for the representation of the expert student knowledge relationship [Futtersack, 1992]. QUIZ is a distributed ITS for learning bridge bidding (i.e. the card game) which involves several tutoring strategies and expert solution generation for comparing the student's bidding solution to that of the expert solution generator. QUIZ showed that even a more complex ITS may have need of only a simple knowledge model.

SUMIT on the other hand, concerned with its bug (misconception) analysis and repair, can be seen as an example of the perturbation model, albeit, reference to this model was not found in the paper [Nicolson, 1992].

Kass [Kass, 1989] provides use with a good review and comparison of the three standard views of overlay modelling. The first, the 'Overlay Model' (fig 3-1), is the simplest of the three techniques. It is best used when the knowledge of the student is described as a subset of the domain to teach [Marcenac, 1992]. As such, all actions of the student are explained as skills or lack of skills in relation to the overall expert model. Although simple to model, the overlay technique has a major drawback in that an overlay model has no provision for dealing with knowledge or beliefs of the student that differ from the expert's [Kass, 1989][Rondeau, 1990]. This drawback restricts those systems which due to the nature of their domain, require the model to represent and interpret misconceptions or 'bugs', both known and unknown.

The 'Differential Model' is a modification and enhancement of the previous technique. It is used to compare the performance of the student with the expert. This is often accomplished when the expert solves the same problem as the student using knowledge that the student should
have (i.e. at a particular level). On the other hand, it does not use knowledge from the domain that the student is not expected to know. In this way, the system can compare its solving technique and abilities with that of the student and adapt instruction accordingly. Although stronger than the simple overlay technique (e.g. reduces search space for solution), the differential model still assumes that the knowledge of a student is only a subset of the knowledge of the expert; therefore the third model may be required.

**An Overlay Student Model**

![Diagram](image)

- Domain Knowledge
- Overlay Student Model

Figure 3-1
The strength of the 'Perturbation Model' (figure 3-2) lies in its ability to represent student beliefs beyond the range of the expert model. This technique can not only represent standard or correct student knowledge, but also perturbed knowledge. Perturbed knowledge is knowledge which may be representative of known misconceptions within the domain knowledge, unknown misconception outside the domain knowledge or even correct knowledge (e.g. new solution strategies) unknown to the domain and hence, the expert. A student's actions may therefore, be explained as a missing skill, incorrect application of a skill, or development of an unknown skill. In most instances, "the perturbation model still assumes that a student and the expert are similar enough that a comparison can be made. Thus perturbation modelling seems to be a reasonable and intuitively attractive method for modelling students" [Kass, 1989].

Figure 3-2
As mentioned previously each of these models can be considered to be sufficient for the ITS in question, based on the knowledge requirement and class of domain. Within JASMINE an adaptation of the differential model has been chosen. This new model and its relationship to the differential model are discussed later in this chapter.

A Perturbation Model

![Image of A Perturbation Model]

\[
\text{Domain Knowledge} \\
\text{Overlay Student Model}
\]

Figure 3-3

3.2.2 Stereotypes

The characterization of users into stereotypes has been used in many systems to group knowledge about users into easy-to-use packets. Rich describes stereotypes (clusters of characteristics) as "a major technique people use to build models of other people very quickly" [Rich, 1979].
As a small example of how extensive stereotype use can be, in recent thesis work at the Royal Military College Kingston, Buck implemented a user system using BGP-MS called GADSS (Grouping - Army Decision Support System) [Buck, 1996a]. GADSS uses twenty-one stereotypes, beyond those generic to BGP-MS, to classify the military user based on rank and job classification. Within the ITS environment of JASMINE, stereotype classification will be used to group students by learning style.

3.2.2.1 Learning Style as a Stereotype - Neuro-Linguistic Programming

The first step toward creating a motivation for the students in their particular learning environment is to determine the environment they work best within. The work of Grinder [Grinder, 1976] described what is called the major input channels by which we, as humans beings, receive information about the world around us. In addition, it is pointed out that we often move what we receive in the three modes into the mode best understood by us, described by Grinder as the receiver's most highly valued input channel. It is hypothesised, therefore, that if the tutoring system can address its original message to the student's most highly valued input channel, that success in learning will increase.

To determine the student's most highly valued input channel, this thesis uses a classification test for Neuro-Linguistic Programming (NLP), validated and created by "Groupe de recherche sur l'interactivité personnes / machines (GRIP/M)" [Tremblay, 1993]. Students who are administered this test are classified in one of three groups (stereotypes) : Auditive, Visual or Kinaesthetic.

As an example of communication within these three modes, Grinder provided the following. Note that the mode of communication is based on the predicates used within each sentence.

- **Meaning:** I want to communicate something to you.
- **Auditory:** I want you to **listen** carefully to what I say to you.
Visual:  I want to show you something.
Kinaesthetic:  I want you to be in touch with something.

NLP test was created and designed to classify student learning styles or communication modes between auditive, visual and kinetic. In order to validate the test the following steps were taken:

1. initial questions were created by GRIPM,
2. sample subjects took test to determine test classification,
3. subjects were analyzed by NLP experts in order to determine students classification,
4. results of test and analysis were compared,
5. test was reviewed for deficiencies based on comparison and modified accordingly, and
6. repeated from 2 until a strong correlation was found between test results and analysis.

Final classification of students is accomplished by administering 18 questions related to preferences of the student in every day life. The student grades each of the sub-responses from 1 to 7, as to the importance of each, with 7 being the highest. It should be noted that the students are not provided with the classification of their answers, nor is the order of styles constant throughout the test.

For example:  When I want to remember someone's name:
- I repeat it to myself mentally.  1234567 (A)
- I associate it with images.  1234567 (V)
- I associate it with what I feel towards them.  1234567 (K)

The results of the test are then totalled within each of the three styles and the ratio of A, V, and K is found (eg. 50, 25, 25). From this ratio, an initial communication regime based on the highest result, or a selection of one if two are of the same ratio, is determined and focused upon. At the very least, the lowest of the three is avoided if at all possible.

Once the initial learning style of the student is determined, the system continues with the actual tutoring, while monitoring the student's success and making adjustments to the learning style in use. At the present time, triggers for adjustment of the learning style are based wholly on
student observed actions such as evaluation results.

3.3 A Review: Student Modelling

The third of three reviews\(^1\), based on ITS developments as presented at the *Third International Conference ITS '96*, involves newer ideas on Student Modelling as related to the ITS. This section reviews work in two areas of student modelling: concept grain size and hierarchies, and knowledge level representation. Two papers where chosen to represent these areas: "Adaptive Assessment Using Granularity Hierarchies and Bayesian Nets" [Collins, 1996] and "A Belief Net Backbone for Student Modelling" [Reye, 1996]. The first of these papers was chosen because it addresses similar ideas to those inherent within JASMINE, while the second points out a possible deficiency with the SMART approach.

3.3.1 Using Granularity.

The first paper to be discussed in this review focuses on the problem of adaptive assessment (student evaluation), and is based on the premise that "different course representations along with sound knowledge propagation ... can be merged together and realized in a granularity hierarchy" [Collins, 1996]. That is to say, the granularity structure is managed in such a way as to maintain a view of the student's knowledge level and to allow the appropriate selection of evaluation items related to the current grain size within the hierarchy. Maintenance of the student's knowledge level is accomplished by superimposing a Bayesian belief network on top of the hierarchy. The net is used to update the student's knowledge level as observations (passive or active) are made by the controlling system, hence, the hierarchy has become a portion of the student model.

\(^1\) See chapter 2 for the first two reviews involving ITS architecture, and Teaching and Learning Styles.
In addition to the student characteristics, the authors' assert that other course curricula information can be maintained. Such information as "learning objectives with specific achievement levels, a prerequisite map of the content domain, a set of test items, and varying modes of presentation depending on the needs of the learner" [Collins, 1996] can be provided by the granularity hierarchies.

In particular, the student model can be used to provide adaptive assessment starting at the base concepts. In the case of an evaluation item, the parents of the item are the base concepts or 'primitive learning objectives' that are being measured by that evaluation item. As with most systems, base concepts are not always evaluated independently; therefore, the granularity hierarchy provides the means for assessment at coarser grain sizes. To accomplish this assessment, the authors suggest that evaluation of base concepts may be propagated to the higher coarser grain size through the constant measurement and maintenance of the related Bayesian net. In particular, "conditional probabilities need to be specified only for variables that are directly dependent [at the base level]; all other conditional probabilities can be calculated use Bayesian inference" [Collins, 1996].

For the purposes of the presented hierarchy, the rules governing evaluation are straightforward. Testing begins at the prescribed grain size and completes when "when the knowledge rating of the coarsest-grained concept being assessed (the goal) increases above a certain instructor-specified level (called the mastery level) or below another instructor-specified level (called the non-mastery level)" [Collins, 1996]. Non-mastery is determined after a predefined number of questions has been presented and the student has clearly failed, or is failing to achieve mastery.

At the time of publishing, the authors had not developed these ideas beyond a theoretical stage, with the structure having only been simulated in a limited manner, and not applied directly to any working domain. Although some ideas presented throughout this thesis are very similar to those reviewed here, they have been addressed from a different point of view. Whereas the
authors of the granularity hierarchies have taken the 'Adam' approach to naming their work, the thesis approached this portion of the student model from a very heuristic direction. The domain structure was created from a very typical hierarchical structure with:

a) evaluation based on a straightforward approach to base concept ownership of question objects and occurring, as requested by the student, at any level throughout the hierarchy;

b) propagation based on a simple averaging of child knowledge levels; and

c) testing being complete when a question bank, based on current student knowledge levels (ie. number of questions posed for each base concept), has been exhausted.

It is important to note that the structure of the model for this thesis, other than the management of knowledge levels, has been developed in a very intuitive manner. Therefore, the granularity ideas reviewed lend additional credence to those present in this thesis, furthermore, such ideas as direct prerequisites, OR relationships and testing adaption based on non-mastery, may play a role in the future development of JASMINE.

3.3.2 Belief Net

The goal of the paper under review [Reye, 1996] was to present a "a belief-net-based approach to student modelling which assists an ITS to make determinations as to the extent of the student's knowledge". It is Reye's opinion, as with this thesis, that the association of the student knowledge state with one of three values (known, unknown and undetermined) "limits the modelling power of the system and so limits its decision-making capabilities."

Specifically, Reye's work was chosen for review for two reasons. First, belief nets represent the type of probability information that would be required for a more complete

\[2\text{ In the biblical sense of Adam naming all the animals from scratch.}\]
prerequisite relationship with JASMINE. Second, Reye's comments on the 'undesirable' nature of Shute's work with SMART peaked interest.

In Reye's opinion, when considering student models, the prerequisite relationship is very important. In particular, this relationship allows the system to alter beliefs based on relationships such as 'You can't know B if you don't know A' or 'Most students who know B, also know A'. Therefore, as changes in the student's knowledge state are observed they can be probabilistically propagated through the belief net.

Reye proposes that an appropriate belief network structure for an ITS is based on two categories of nodes:

1. a belief net backbone, which links all the 'student-knows' nodes together in a partial ordering, according to their prerequisite relationships; and

2. a topic cluster for each node in the backbone, which consists of a single "student-knows" node together with a standard set of additional belief nodes either local or global to the student model. Belief nodes include such probabilities as a rule or concept being in the learned state prior to student exposure or making the transition from the unlearned state to the learned state after initial exposure.

With the introduction of topic clusters, Reye believes "the impact of belief updates in a given topic sub-net on its 'student-knows' node can be calculated locally by considering just the nodes in that topic sub-net rather than having to propagate such updates through the entire network in order to determine their net results". From this, it is assumed that the changes to global belief nodes are calculated only as required for a particular 'student-knows' node and not calculated after every possible change; this is perceived as a time saver.

Reye feels that Shute's work with SMART, hence its application within this thesis, is "generally undesirable" due to the focus of updating individual measures in the student model. He does signify, however, that Shute's work is "important for the future development of more sophisticated student models based on belief networks".

3-12
As indicated during the previous review [Collins, 1996], explicit prerequisites are not used within this thesis. Nevertheless, this thesis is in agreement with the importance Reye has placed on prerequisite relationships. Therefore, the implementation of prerequisite relationships has been suggested for future development within JASMINE (chapter 6). It should be noted, however, that implicit prerequisites are employed, and knowledge probabilities propagated with summary concepts as base concepts are updated.

3.4 BGP-MS

The Belief Goal Plan - Maintenance System (BGP-MS) "is a user modeling shell system that can assist interactive software systems in adapting to their current users by taking the users' presumed knowledge, beliefs, and goals into account" [Kobsa, 1995]. BGP-MS attempts to provide user systems and, in the case of this thesis, student systems, with "appropriate modelling mechanisms". These mechanisms can be seen as assumption management tools. Although not all of the following are of direct concern to this thesis, Kobsa lists the mechanisms incorporated within BGP-MS which are able to:

- draw assumptions about the user based on his interaction with the application system;
- represent and store these assumptions;
- draw additional assumptions based on the initial assumptions;
- take appropriate actions when inconsistencies between assumptions are detected; and
- supply the application with current assumptions about the user.

3.4.1 Student Modelling with BGP-MS

Although, BGP-MS was created primarily as a user modelling shell, this thesis' initial
intent was to adapt the shell's use to the student modelling world. As will be discussed throughout this section 3.4, BGP-MS, in its present state was unable to meet the requirements of JASMINE. More precisely, there were difficulties with knowledge levels, multiple users, execution speed, and platform dependence.

3.4.1.1 Partition Hierarchies

The attempted use of the partition Hierarchies of the BGP-MS shell for JASMINE is outlined in Figure 5. The direction of arrows indicate the inheritance factor of the hierarchy (eg. SBUB inherits the information in SBMB and may inherit the knowledge in Visual if the Visual stereotype is activated).

```
SBMB
  └── Audio
    └── SB
    └── SBUB ----> Visual
        └── Kinetic
```

Figure 3-4: Partition Hierarchies.

The SB partition represents what the Systems Believes (eg. domain knowledge or objective facts concerning the user.)
The *SBUB* partition represents what the System Believes the User (student) Believes. These are assumptions of what the user believes; for example, assumptions made from action or evaluation observations of the user. This partition also can be used to record student misconceptions. It was the intent of this thesis to include the degree to which the system believes the student knows a particular concept; however, this was not possible and is one of the main reasons BGP-MS was abandoned.

The *SBMB* block represents what the System Believes is Mutually Believed by the system and user about the domain (e.g. which expert knowledge is shared).

Additional BGP-MS standard partitions are available such as *SBUW* (System Believes User Wants), [Kobsa, 1995] [Buck, 1996]. As the intent was to not employ these partitions they are not covered here.

The *Audio*, *Visual*, and *Kinaesthetic* blocks represent the three stereotypical learning styles that are addressed within this thesis. These stereotypes are initially determined using the NLP test, which was discussed earlier. It should be noted that the purpose of the NLP classification is to demonstrate the concept of student characteristics being including in the student model. Future research in this area should be directed at alternative characteristics (chapter 6).

### 3.4.2 Amplitude Differential Overlay Model

The student model, used in this thesis, is based on a modification of the differential student overlay model. As discussed earlier, the Differential Model (Figure 3-2) itself is a basic enhancement of the general overlay technique and a technique used to compare the knowledge performance of the student with the expert. In this way, the system can compare its available knowledge with that of the student and adjust domain navigation accordingly. Although stronger than the simple overlay technique (e.g. reduces search space for solution), the differential model
still assumes that the knowledge of a student is either available or not (ie. no degree of knowledge).

Figure 3-5: Amplitude Differential Model

The 'Amplitude Differential' Model retains the features of the differential model. It is, however, now able to model the degree of knowledge a student retains about a given knowledge area or concept. This is accomplished through the height or amplitude of the knowledge bar. Shaded areas of the model, including those concepts at a zero level, correspond to the expected student knowledge area of the differential model (figure 3-5), while unshaded areas represent knowledge the student is not expected to have obtained.

3.4.2.1 Application Communications (the flow of assumptions).

In order to facilitate communications between the controlling application and BGP-MS (eg. population of SBUB or notification of stereotype activation), several message types are available (Figure 6) within BGP-MS: bgp-ms-tell, bgp-ms-ask, bgp-ms-answer, alert-from-bgp-ms, interview-next-question-block, and interview-response.
The \textit{bgp-ms-tell} function is used to inform BGP-MS of observed beliefs. Primarily these are observations made by the application through such methods as user actions and system user evaluation of conceptual knowledge. Each instance of a \textit{bgp-ms-tell} is analyzed for additional assumptions and entries made in the appropriate view or hierarchy. During JASMINE development, it was the intent to include the confidence level of concepts. Although inclusion of this feature in BGP-MS was of interest, developers "had to choose between several options as to what to bring into the final version of the system. Confidence values required more manpower than [was] available" [Pohl, 1997].

Figure 6: Message Traffic in BGP-MS

The \textit{bgp-ms-ask} and \textit{bgp-ms-answer} functions are used with BGP-MS to make queries of the user model and, for the purposes of this thesis, the knowledge state of a specific concept. The
response from BGP-MS is provided, synchronously to the application, through bgp-ms-answer.

The `alert-from-bgp-ms` function is used by BGP-MS to inform the application about significant events. For example, this thesis was particularly interested in the activation and retraction of student learning style stereotypes, or the activation of a student remediation rule. The alert is provided asynchronously to the application.

The `interview-next-question-block` and `interview-response` functions are used by BGP-MS to collect primary information about the student. Interview-questions are defined so as to provide immediate input to the appropriate model partition based on the student's response. BGP-MS may then infer based on partition knowledge and stereotype triggers additional student information. Within this thesis, it was the intent to complete initial interview questions of the NLP test, with the goal of triggering one of the three addressed student learning types (audio, visual or kinaesthetic). As previously mentioned, this could be refined using NLP; even if the number of stereotypes increased or the rules were modified the principle of these learning styles remains the same. However, as the NLP is a cumulative test, questioning with the BGP-MS interview was "not possible. However, it is possible to store ... intermediate assumptions in the user model and use these in the conclusion conditions of other questions" [Pohl, 1996].

3.4.3 Failings

As mentioned during the discussion of BGP-MS, there are several reasons why the shell failed to meet the requirements of JASMINE. The most prevalent reasons for failing to meet these requirements were BGP-MS's difficulties in the following four areas:

- **Knowledge levels.** The representation of knowledge levels within JASMINE was a major design requirement. The ability to identify whether a student knows, or does not know, a concept but also to what degree the concept is, or is not known, is inherent in 'amplitude differential' student model. At the present time, it is beyond the capabilities of BGP-MS to measure the certainty of knowledge.
Multiple Users. The initial design of JASMINE has been developed for the single student. However, the ability to move beyond the single student to simultaneous use by many students is easily available. BGP-MS will not be able to make this change effortlessly as only a single knowledge base, initialized to the current user, is available at any given time.

Execution Speed. Although the execution speed of BGP-MS in itself is acceptable, initialization time for the system is not. Long delays can be expected while the Lisp interpreter, BGP-MS shell, and finally, the specific application are initialized. At the present time, there is no avenue for the removal of user specific knowledge which would allow the first of the system initialization delays to be attributed to 'boot-up' only.

Pragmaticality. BGP-MS has been implemented using the specific Lisp interpreter CMU3 LISP, and can executed in the SUN environment only. This makes the pragmatic use of BGP-MS impractical. Although it may be beneficial to the operation of the JASMINE Student Server to execute it on a SUN platform, it is not necessary. In addition, the student end of the application may be executed within any JAVA enabled browser on any platform.

3.5 Java

Java is an object-oriented programming language written as a complete programming language while at the same time it is designed to be small, simple, and portable across platforms and operating systems [Lemay, 1996]. Hence, Java has many advantages (eg. it is easy to learn, platform independent, secure, and robust).

Although Java is often compared to C or C++, its design nature makes it a language which is much more friendly to the newer programmer. Such items as pointers and memory management, which make other languages difficult to manage and debug, are handled directly by

3 Carnegie Mellon University School of Computer Science.
Java. Therefore, programs are easier to compile, easier to debug [Lemay, 1996] and, hence, easier to learn.

By providing automatic memory management [Sun, 1996] and garbage collection, Java has eliminated the requirement for pointer math. Objects and their memory requirements are created as required by the program, and accessed via reference values. These reference values, in fact, point to the memory location of the object, which is then manipulated directly through it's attributes' references. Therefore, to release memory, the programmer has simply to remove all references to an object (e.g. set the references to null). When an object is no longer referenced, it then can be removed from the storage allocation heap by the garbage collector, although its actual removal may be delayed until a beneficial time [Arnold, 1996].

Although Java is a simple language, this does not mean it is rudimentary. Java provides much of what is required to complete more complex programs. Included in Java is a set of class libraries that provides basic data types, system input and output capabilities, and other utility functions. In addition, support for networking, common Internet protocols [Lemay, 1996], and multi-threading are provided.

Although mentioned briefly below, there are several features of the Java programming environment which have made it the language of choice for this thesis: object orientation, platform independence, and security.

3.5.1 Object Oriented

The fundamental structure of the object oriented approach is based on objects and classes. The object class is a template for multiple objects with similar features and is made up of attributes and class behaviours, two components embodying all the required features of a set of similar objects [Lemay, 1996].
The attributes of an object are the items which differentiate one object in the class from another, and are based on such things as the appearance and state of the object. For example, the following describes the Question Class used with this thesis.

```java
public class Question {
    /* *****************************************************/
    /* Question class is the building blocks for the question */
    /* of the Questions is based on a link list structure. */
    /* *****************************************************/

    Concept concept;  /* concept name for which question has direct affect */
    String qType;     /* question type 'T', 'M', or 'F' */
    String question;  /* textual question */
    String answer;    /* correct response */
    String choices[]; /* choices for multiple guess questions */
    int numChoices;   /* number of choices available for above */
    Question nextQuestion; /* next Question in the question bank */
}
```

The behaviours of a class are encapsulated with the attributes and defined as methods or actors which affect the referenced object directly. For the Concept Class, used within this thesis, methods such as nextChild() (provides the value of the next eldest child for a summary concept) and findConcept(String n) (given a starting place attempts to locate the concept named by n) have been provided with navigators within the concept class intra-relationships.

The use of the defined classes and resulting objects within an object oriented approach, is based on four underlying principals: polymorphism, encapsulation, dynamic loading, and inheritance [Sun, 1996]. While Classes and Objects are the building blocks of object technology, these principles define the scaffolding needed to support the overall structure.

*Polymorphism* - the application of a method to objects of different classes achieves the same semantic result. A simple example of this is taken from file management found within the
Domain Manager tool provided with this thesis. In this manner dos.close() and fos.close() provide the same semantic result when the DataOutputStream and FileOutputStream objects, although different, are both closed.

Encapsulation - a single object definition binds the operations and states particular to that object, while the implementation details are hidden. For example the operation of the Concept Class, which provides the foundation JASMINE, has encapsulated all operation on the Class. In doing so the linked tree structure of the domain has been removed from view. In this way, the Concept class provides operations such as:

```
public void addConcept(Concept c) - add a concept to the underling structure
public void deleteConcept(String n) - remove a concept from the underling structure
public void updateParents0 - update the knowledge state of parent concepts
public boolean isBaseConcept(String n) - is this concept a base concept
```

In addition, encapsulation allows what is defined as overloading [Arnold, 1996] - methods having the same name, but different signatures. In this manner the same semantic result, as with polymorphism, is achieved. An example from the Concept Class shows two readDomain methods with different signatures. The first readDomain (List cl) reads the existing linked tree structure and fills a List Object with the required information, while the second readDomain (DataOutputStream dos) reads the existing structure to the provided DataOutputStream object (in essence storing the structure on disk).

Dynamic loading - Objects can come from anywhere, including the network. If your system ever runs a Java application over the network and encounters a piece of the application it does not know how to handle, it can automatically download any missing pieces your system needs to be functional [Sun, 1996]. For example, when using JASMINE's Question Manager, an instance of the Question Viewer is invoked and the dynamic loader will then retrieve the Concept class. This operation is completed when a reference to Concept is encountered for the first time in the import statements of the Question Viewer.
Inheritance - classes can be defined based upon existing class definitions for code reuse and enhancement [Sun, 1996]. Inheritance is a powerful mechanism, when you write a class you only have to specify how that class is different from some other class. To implement this feature, the object language is based on a hierarchy of classes, each class has a super class (the class above it in the hierarchy), and each class can have one or more subclasses. As another measure of simplicity, the Java language has restricted inheritance to a single superclass. At the top of the Java class hierarchy is the class Object [Lemay, 1996].

The main advantage of this inheritance structure is that all subclasses inherit the methods and variables of those in their superclass hierarchy. In essence, the class becomes a combination of all the features of the classes above it in the hierarchy [Lemay, 1996]. Hence, a newly defined class may override the methods of its superclass or, as in the following case, increase the functionality of the superclass under specific conditions.

```java
public boolean action(Event e, Object o) {
    String domainName;  /* name of the Domain Entered by user */
    * 
    * 
    if (e.target instanceof TextField) {
        domainName = domainText.getText();
        if (domainName.length() != 0) {
            qv = new QuestionViewer(domainName);
            qv.start();
        }
    }
}
```
return true;
}
return super.action(e,o); /*activate the super's action method
} // action

It is important to note that method definitions are chosen dynamically when a method is called, activating the first method found which meets the signature of the call is activated. Java first checks the objects's class, then looks in the class's superclass, and so on up the chain until the method definition is found [Lemay, 1996].

Now that the fundamentals of Object technology, Classes and Objects, polymorphism, encapsulation, dynamic loading and inheritance have been discussed, we can move on to the more specific features of Java which make it the language of choice.

3.5.2 Platform independence

The main feature of the Java environment that leads to platform independence is its implementation via a virtual machine. A virtual machine is "an imaginary machine that is implemented by emulating it in software on a real machine"[Sun, 1996]. Thus, whether the Java application is a complete program, an applet, or a combination, it is capable of running on the virtual machine using the current platform. Therefore, as long as the platform supports the Java environment, either through a Java-capable browser or Java interpreter for applications that do not require a browser [Lemay, 1996], then only a single compilation of the Java code is required.

As a result of this environment, platform independence is one of the most significant advantages that Java has over other programming languages [Lemay, 1996]. The fact that JAVA is designed to run on the WWW, which is itself platform independent is also a significant advantage. Java's actual independence is achieved at two levels, the source level and the binary level.
At the source level, "Java's primitive data types have consistent sizes across all development platforms" [Lemay, 1996]. This is significant because in languages which do not provide this implementation of the same program over independent platform computations there maybe varying results. A simple example of this is apparent to any programmer who uses numerics such as the integer. A program which expects the size of the integer to be of a four byte size may have difficulties if a platform representation of the integer is set at two bytes (ie. array out of bounds, integer overflow).

At the binary level, Java code is actually in a form called bytecodes. Bytecodes are "a set of instructions that look a lot like machine code, but are not specific to any one [platform]" [Lemay, 1996]. Bytecodes look a lot like machine code because these are the instructions which are interpreted by the virtual machine. The use of bytecodes may also be considered a disadvantage when comparing execution speed to typical programming languages. However, it is a small disadvantage when considering the benefits of WWW use. As well, tools are available to convert Java Bytecode into native code [Lemay, 1996], a language comparable to compiled C or C++ programs in execution speed.

Although platform independence is a great advantage for programming on the WWW, it does not come without some challenge and risk. This is why the aspects of JAVA security must be addressed.

3.5.3 Security

As with any application design, Java applications need to be secure if people are to feel confident about using them. During the development of Java, security was of great regard because users have to be concerned not only with a known application, but with unknown applications coming from unknown web sites.

To deal with this uncertainty, Sun’s Java Team incorporated several security checks
within the language. Several of these checks have to do with the security of applets, and include checks to make sure applets cannot do nasty things [Lemay, 1996]. Some of the security checks built into the applets are:

- Java applets cannot read or write to the disk on the local system;
- Java applets cannot execute any programs on the local system; and
- Java applets cannot connect to any machines on the Web except for the server from which they are originally downloaded.

It is important to note that some of these restrictions may be disregarded in some browsers or may be turned off in their configuration. However, for the security aware user these feature should be enabled.

To limit security holes in code, the Java compiler and interpreter check both the Java source code and the bytecode looking for such holes as overrun buffers or stack frames [Lemay, 1996], code fragments, code that forges pointers, and code that violates access rights or attempts to change object type or class [Sun, 1996]. To limit Trojan-horse applications from executing as a trusted class, the dynamic class loader separates the name spaces for local and imported classes thereby ensuring that locally defined classes are used first. In addition, "memory layout occurs at runtime, therefore providing protection against illegal addressing of code" [Sun, 1996].

Although Java is as secure as it can be at the present time, precautions could only be taken against foreseen problems. As with any programming language, it is not possible to prove that every possible security hole has been plugged or monitored. For example, some of the precautions taken in the development of Java may even now be superseded through the use of native methods. A native method enables you to implement a method that can be invoked from Java, but which is written in a "native" language, usually C or C++ [Arnold, 1996]. This provides a useful tool if you wish to implement existing code or directly manipulate a piece of hardware. The use of "native" language code, however, is a loss of platform independence (ie. its now native), a security risk (bypassing JAVA security checks) which can result in a loss of
confident networking.

3.5.4 Summary

Overall, Java has proven itself to be a very pragmatic solution in the search for an ITS environment. The features of object oriented technology, platform independence and secure reliability, have made Java the language of choice in JASMINE development. In addition, Java goes beyond these internal features and makes available a highly adaptable and flexible user interface. As will be discussed in the following chapter, it is the interface which is most important to the acceptability of an application, especially where adaptable student presentation is desirable.
Chapter Four:
Adaptability and Intelligent User Interfaces

4.1 Introduction

This chapter provides an introduction to adaptive techniques used within both ITS and Intelligent User Interfaces (IUI). Consequently, several design questions, presentation techniques of IUIs, and other adaptive techniques within tutoring systems are reviewed. With respect to the ITS, ideas presented in this chapter are ordinarily implemented in the pedagogy, psychologist, and interface portions of the system. However, full realization is normally obtained through the interface module via the student presentation.

The first topic to be presented in this chapter is adaptiveness. Although the ideas presented can be implemented in systems other than ITSs and IUIs, it is these systems upon which the thesis focuses.

The second of this chapter's topics covers the design questions or options available to the
interface developer. The design questions presented include items such as media presentation, the degree of intelligence required, intrusiveness, and personification. The use of text, video and sound in recent years has added much to the discussion. Debate goes beyond the question of media use to include issues such as when and how the media should be used. Where some classic intelligent interface language acts were implemented to create a text environment (eg. natural language use, italics, bold, etc...), media-acts now can be used to provide an advanced media environment (eg. zoom, shading, colour, sound bites, etc) for the user.

Finally, this chapter will present two IUI related applications which are applied to adaptive ITS (Adaptive Tutor Using Learner Attributes - Adaptive Tutoring System [Milne, 1996]) and the World Wide Web (ELM-Adaptive Remote Tutor [Brusilovshky, 1996]) and which have had a direct influence on JASMINE.

4.2 Adaptiveness

For an ITS to be truly adaptive, it must use its knowledge within the student model to not only determine the next avenue of learning, but also to determine how this learning is to be presented to the student. In addition, the system must be able to modify its actions based on current and subsequent input actions and perceived knowledge of the student. Fischer [Fischer, 1993] has discussed some of the major characteristics of adaptive systems (Table 4-1) in comparison to adaptable systems. Adaptable systems allow direct manipulation of the environment by the user rather than by (intelligent) system manipulation.

Through the use of user/student modelling and Intelligent User Interfaces techniques JASMINE builds on the strengths of the adaptive system while attempting to reduce or eliminate the concerns regarding weaknesses.
Characteristics of Adaptive Systems

| Definition | - dynamic adaptation by the system itself to current task and current user |
| KNOWLEDGE | - contained in the system  
- projected in different ways |
| STRENGTHS | - little (or no) effort by the user  
- no special knowledge by the user is required |
| WEAKNESSES | - user has difficulty developing a coherent model of the system  
- loss of control  
- few (if any) success models exist (except humans) |
| MECHANISMS | - model of users  
- knowledge base of goals and plans  
- powerful matching capabilities  
- incremental update of models |

Table 4-1

Using one of the classification models proposed by Dieterich (Dieterich, 1993), the techniques within JASMINE can be described as either Self-Adaptation (Table 4-2) or User-Controlled Self-Adaptation (Table 4-3).

<table>
<thead>
<tr>
<th>Initiative</th>
<th>System</th>
<th>User</th>
<th>System</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4-2: Self-Adaptation

<table>
<thead>
<tr>
<th>System</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4-3: User-Controlled Self-Adaptation

While in both cases the system is responsible for initiating, proposing and executing the changes, in the User-Controlled model, the user has some input in the decision to complete execution of the adaptation. In the most basic sense, this may come as an early attempt to switch from visual to audio learning style techniques based on the system's observations of the student.

As implied above, one area within JASMINE where adaptation is encountered occurs in 4-3.
the area of the student's learning style. Therefore, JASMINE uses two of the three main adaptation strategies pointed out by Dieterich: "adaptation before first use of a system" (pre-test), and "adaptation during use" (system observations and stereotype triggers). The third strategy "adaptation between two sessions", was deemed redundant for JASMINE since the second strategy should quickly cover student differences between sessions. As a matter of interest, additional areas of adaptation within JASMINE include the colour coding of navigational links and presentation of interview and evaluation questions.

4.3 The Design Questions

What to say? What media with which to say it? How to say it with the chosen media? These are the questions presented by Maybury and which face the designer of any presentation adaptive system [Maybury, 1993]. Within an ITS, "what to say", is pretty much straightforward because the text of the domain provides this information. The question might better be posed as, "how much to say" (eg. during the first occurrence or a remediation). "What media with which to say it", could also be considered straightforward in JASMINE, as the current learning style can be seen to dictate the proportion of the media being used to a large degree. "How to say it". on the other hand, is not straightforward and consequently, this question had a direct influence on the presentation techniques used within JASMINE.

To answer these questions, whether straightforward or not, one must generally look at the interface of the application. Although each of the questions may be decided, once and for all, by the programmer or answered, intelligently, each time the ITS has a requirement, the issue comes down to the need to present information to the student. Remembering the 'iceberg' rule (Chapter 1), the designer must look deeper than the superficial surface of the application and construct an IUI. This requirement brings to light two discussion area which are of current interest within the research realm of IUI: Direct Manipulation versus Interface Agents, and How much Intelligence?.
4.3.1 A Debate: Direct Manipulation versus Interface Agents.

The first of the IUI discussion areas of interest to ITS, and this thesis, is the question of Direct Manipulation and Intelligent Agents. The debate reviewed below is from an open discussion, which took place at the 1997 International Conference on Intelligent User Interfaces, between Ben Shneiderman and Pattie Maes. As well as the actual debate, their respective position papers have been reviewed. Following the discussion of the debate, observations and points of consideration as related to ITS and JASMINE are made.

4.3.1.1 Direct Manipulation

The first participant in the debate, Ben Shneiderman [Shneiderman, 1997], takes the position that "direct manipulation interfaces are seen as more likely candidates to influence advanced user interfaces than adaptive, autonomous, intelligent agents." In addition, he believes that direct manipulation will provide for the ultimate desire: user control and responsibility.

Shneiderman points out that designers of direct manipulation systems have an 'intuitive grasp' of what users require. Three of the most important principles are:

1. continuous representation of the objects and actions of interest
2. physical actions, such as pressing of labelled buttons, instead of complex syntax
3. rapid incremental reversible operations whose effect on the object of interest is immediately visible

As a result of these 'intuitive' principles, direct manipulation systems can be designed with some very beneficial attributes such as: error messages being rarely needed; users able to immediately see if their actions are furthering their goals; and if the actions are not furthering the effort, the users have the option simply to redirect their activity. As a result, users gain confidence and mastery as the initiators of action, and feel they can exercise a degree of control.
Therefore Shneiderman states, "the task concepts dominate the users' concerns, and the
distraction of dealing with tedious interface concepts is reduced."

One specific use of direct-manipulation which Shneiderman advocates is in the domain of
programming. Such items as robot manipulation by stepping the robot through a sequence of
actions, programming of a car radio by tuning and setting buttons, and creating macros
(currently available in many word processors and spreadsheet programs) by demonstrating the
actions to the program, are all methods of direct manipulation programming. This is
programming which avoids the requirement for a higher programming knowledge or use of a
rigorous syntax.

Shneiderman admits that "macro facilities in graphic user interfaces are more challenging
to design"; however, he does go on to present an example of such a facility. EAGER [Myers.
1992] is a system which monitors user actions. One such example of this ongoing monitoring is
EAGER's ability to recognize two similar sequences, and from such a recognition, to offer help
to the user in completing further iterations. Although Shneiderman feels this approach is
encouraging, he states that "it has been proven to be difficult to generalize this approach". He
suggests a more effective method, which would be to give users the visual tools to specify and
record their intentions (ie. macro manipulation as mentioned earlier). Overall, Shneiderman is
skeptical that user intentions are so easily determined or that vague user requirements are usually
effective. It is here that Shneiderman's argument against agents begins.

Shneiderman describes the desire for intelligent agents as an attraction by many people,
including Myers, "to the idea of a powerful functionary carrying out their tasks and watching out
for their needs". He feels that these designers, "believe that human-human interaction is a good
model for human-computer interaction and seek to create computerized partners, assistants, or
agents" while "promoting their designs as intelligent and adaptive" to the extent that some
"pursue anthropomorphic representations of the computer to the point of having artificial faces
talking to users".
Of interest to Shneiderman is an enhancement on the agent scenario in which the system develops a "user model" to guide the adaptive system. For example, "as users make menu selections more rapidly or indicating proficiency, advanced menu items or a command line interface appears". Although Shneiderman mentions other possible uses of adaptive system, he feels that there is little empirical evidence of their efficacy. Although, a few success stories with adaptive systems are given, including, "a few training and help systems", extensive study and careful refinements are required to give users appropriate feedback. Generalizing of these successes by the advocates of adaptive systems has "proven to be more difficult than ... hoped".

Shneiderman's focus here is on the users' acceptance of the system, as even "occasional unexpected behaviour has serious negative side effects that discourage use"; if sudden changes are made by the adaptive systems, then users may become anxious as they perceive a lack of control. Shneiderman does concede that consultation with the user before changes are made would be helpful; however, he believes that, "such intrusions may still disrupt problem-solving processes and annoy users".

In conclusion, Shneiderman believes that, "users have a strong desire to be in control and to gain mastery over the system" which, in turn, gives them a feeling of accomplishment and responsibility. He asserts that user interfaces must be predictable while masking the underlying complexity of the system; as a result direct manipulation will thrive [Shneiderman, 1997].

4.3.1.2 Intelligent Agents

The second participant in the debate was Pattie Maes [Maes, 1997], who takes the position that, "direct manipulation will have to give way to some form of delegation...instead of exercising complete control ...people will be engaged in a cooperative process in which both human and computer agents initiate communication, monitor events and perform tasks to meet a user's goals".
Maes believes that technology such as interactive television, palmtop diaries and smart credit cards serve to widen the gap between untrained users and sophisticated microprocessors. Therefore, some middle ground will be needed to accommodate the "limited human attention span" within more complex software and data collections. Currently, with direct manipulation, nothing happens unless a user gives commands via some direct input device. Hence, the computer remains a passive entity "waiting to execute specific, highly-detailed instructions" while providing little help for more complex or time consuming tasks.

With intelligent software, users will move beyond the direct input device and will be able to speak or gesture to agents, who will, "appear as 'living' entities on the screen, conveying their current state and behaviour with animated facial expressions or body language". As such, many of these agents will become digital proxies for their owner, performing tasks while hiding the technical details of the tasks, acting as guides or, "even teaching them about certain subjects".

Maes describes the difference between the agents and regular software, "as [the agents] sense of themselves as independent entities". It is Maes' opinion that an, "ideal agent knows what its goal is and will strive to achieve it." Therefore, these agents will become robust, adaptive, and capable of learning and responding to new situations using previous experience and built-in problem solving methods.

Maes admits that currently available agents are not very intelligent and merely follow a set of rules that a user must specify. Therefore, the value of such agents relies on the initiative and ability of its owner. Knowledge engineers, however, attempt to endow programs with information about the tasks to be performed in a specific domain, with the program then inferring the proper response to given situations. Maes gives the example of an intelligent E-Mail agent which:

* people may have as an administrative assistant
* it's user has an assistant named George
* an assistant should know the boss's schedule
* that a message containing the word 'meeting' may contain scheduling information
* therefore, the assistance would infer that a message copy should be forwarded to George.

Beyond the basic intelligent agent is the artificial-life agent. Maes defines the artificial-life agents as agents which are "truly autonomous: in effect, they program themselves, their software is designed to change its behaviour based on experience and on interaction with other agents". Continuing the E-Mail example, an artificial-life agent could learn by observation that it should forward a copy of a message containing the word 'meeting' to an administrative assistant and offer to do so automatically.

In concluding, Maes offers some items for thought. She asks, "should users be held responsible for the actions of their agents?", and "how can we ensure that an agent keeps private all the very personal information it accumulates about its owner?". Maes concedes that people may find it hard to trust an agent which they have not instructed. On the positive side, as intelligent agents proliferate everyone will have access to a support staff which will allow them to, "digest large amounts of information and engage in several [simultaneous] activities". Today this Access is available only to a few fortunate people.

4.3.1.3 Comments

Although the debate between direct manipulation and intelligent agents seems to boil down to one of control, Shneiderman's opinion is that the ultimate desire of the user is control and responsibility. At the same time, Maes leaves the question of responsibility unanswered, simply asking the question "Should users be held responsible for the actions of their agents?". This would seem to point toward direct manipulation providing the answer to responsibility. However, it must be remembered that the technology of user agents is still underdeveloped.

Left to evolve, there is nothing in Shneiderman's argument to suggest user agent
technology could not incorporate the principles, or beneficial attributes, of direct manipulation systems. In fact, examples given by Shneiderman (eg. EAGER, training systems) would appear to suggest agents could be improved.

A common ground between Maes and Shneiderman can be found on the point of detail. Both have a desire to see complex syntax or, highly detailed instruction, hidden from the user. As a result, the interface must be designed to complete detail tasks with little manipulation by the user. To this end, Maes offers a middle ground where control is, "delegated to cooperative processes or agents"; the analogy being with a manager having several task assistants.

The question of control can be addressed, using the analogy of the manager. At first, the manager can be provided with both direct manipulation tools and assistants and can receive advice directly. Then, as trust is developed, the manager can delegate tasks (eg. management of a direct manipulation tool) gradually to the assistant. Control is maintained through the selection of advice and direct delegation of tasks. As a result, as with any managerial-employee relationship, the manager takes responsibility for actions which have a direct link to the advice taken and tasks delegated.

The use of both types of interface can be accomplished intuitively within the ITS. Direct manipulation would provide the student with a free reign approach to learning, while intelligent agents could be programmed to provide a completely authoritative approach. JASMINE's tactic is to apply some of both approaches (ie. find the middle ground suggested by Maes). A 'student model' is employed to adapt the presentation of information within a given domain to the students' characteristics. As well, features of a direct manipulation system, including student access to system information (presentation style and current knowledge state), and free movement (direction is suggested) throughout the expert domain are employed.

Although a solution has been proposed for the mixture of direct and intelligent manipulation, it is important to discuss the degree to which this mixture should include
4.3.2 A Discussion: How Much Intelligence?

The second of the IUI areas to be discussed in this thesis is the question of, "How much AI?". As will be seen from the positions discussed below, much of what is presented has a direct impact on the development of ITS, as well as IUI. The positions reviewed below are from an open panel discussion, which took place at the 1997 International Conference on Intelligent User Interfaces, between the members Larry Birnbaum, Eric Horvitz, David Kurlander, Henry Lieberman and Steve Roth and the panel chair Joe Marks [Marks, 1997].

4.3.2.1 The Discussion

"The most interesting intelligent user interfaces have tended to use minimal or simplistic AI" [Marks, 1997]. This has led to a perceived notion that minimal or simplistic AI is not enough for a compelling intelligent software. However, much of what is presented by the participants of the panel would seem to suggest that minimal and simplistic may be what is required if we focus on the content ("It's the content, silly!" : Birnbaum) ([AI must be] selectively applied as needed as part of more complete product" : Roth), value for effort ("the sophisticated machinery frequently does not deliver great value" : Horvitz), and confidence value ("cost of verifying and correcting inferences, can be prohibitive" : Kurlander).

Although most of the panel members have suggested that simplicity is the first goal when initiating AI in the IUI, there is room for more complex processes and focuses of work, whether intelligent or integrated with intelligence. Several areas of focuses have been suggested and include work in:

a. modelling "work ... is mostly a reflection of the extent to which a good job
has been done in modelling the task and domain" : Birnbaum

"Take advantage of 'free' information implicit in user actions"
: Lieberman

b. UI design "[focus] on better UI design...in parallel, research and development on more sophisticated user models and reasoning machinery for building compelling IUI" : Horvitz

"crafting the interface to support the intelligence" : Kurlander

c. user interaction "casts the software in the role of a helpful assistant rather than an omniscient problem solver" : Lieberman

"better direct manipulation techniques" : Roth

Some areas where more sophisticated AI may be of benefit have also been suggested. Most address complex areas such as:

a. uncertainty "the means for detecting complex sequences of events, for grappling with uncertainty, learning usage patterns" : Horvitz

b. goal detection "for disambiguating the goals of the user and determining the best action to take" : Horvitz

c. user assistance "an 'agent' can help by intelligently making suggestions that help the user choose" : Lieberman.

Roth summarizes the requirement for all AI, both sophisticated and simplistic, "[AI must be] selectively applied as needed as part of more complete product." Many of these concerns and interest focuses are found in the work of the panel participants. Two examples of this work are Kurlander's 'Comic Chat' and Lieberman's 'Letizia'.

4-12
Comic Chat is a graphical Internet chat program which attempts to aid in conversation through the interpretation of natural language and conversation context. The layout of conversations is based on the typical layout of comic books. As the conversation is interpreted, suggested comic character expressions and gestures are provided along with their placement among a temporal positioning of the overall dialog. A focus on user interaction and the desire for a minimalist AI, playing the part of a guide or assistant, while the system attempts to interpret the users goals, has been obtained. In addition, Comic Chat's use of comic characters has added an interesting dimension to the design of User Interfaces. It has explored the requirement for information presentation both to, and from, the user, while at the same time using more conventional interface techniques such as direct manipulation.

Letizia is essentially an advanced scout for a user searching the WWW. The system combines the author's principles for successful integration of AI (don't disturb the user's interaction, suggest rather than act, operate in real time, and watch what the user is doing) into a tool which can be as useful as the user wishes. A focus on modelling, goal detection and user assistance is what makes this system of interest. By watching what the user is doing and inferring information goals, the system is able to 'surf' ahead of the user looking for likely information and preloading web pages as the user pauses on the current page. In a sense, the system is suggesting a navigation route to the user, which may be used or ignored if inferences are incorrect or the user's interest changes.

4.3.2.2 An Answer?

Although an answer to the question 'How much intelligence?' can not be provided directly, indirectly an answer can be found. As presented during the panel discussion, the degree of intelligence can not be measured physically. It can be and should only be, measured by its delta factor of added value to the end product. When the added intelligence begins to burden (eg. lack of control), confuse (eg. inconsistent behaviour), or slow (eg. computational power) the
intended user, then the question of too much intelligence can be answered.

4.4 Adaptive and World Wide Web ITS Examples (Influences)

Within JASMINE, two "browser" techniques of the World Wide Web (WWW) have been employed. The first is the use of the platform independent object-based "JAVA" for student evaluation and system reporting of its observations of the student. The second, inspired by Brusilovshy, involves the ad-hoc production of HTML pages [Brusilovsky, 1995]. This technique allows the system to create learning style based presentations which consider the current learning style and student knowledge level.

In this section, two IUI related applications applied to the adaptive ITS sphere will be examined. Both of these applications have had a direct influence on the design of JASMINE. The first, ELM-Adaptive Remote Tutor (ELM-ART) [Brusilovshky, 1996], as the name hints at, is of particular interest due to its World Wide Web application and ad-hoc presentation based on the student's current knowledge state. The second, Adaptive Tutor Using Learner Attributes - Adaptive Tutoring System (ATULA-ATS) [Milne, 1996], is of particular interest because of the manner in which it creates and modifies its student stereotypes.

4.4.1 ELM-ART

ELM-ART [Brusilowsky, 1996] is a WWW based ITS in the domain of Lisp. The system is considered adaptive in its interaction with the student and remote in its accessability via the WWW. It was the authors' belief that creating, or even porting, existing ITS systems to a WWW platform would join 'flexibility and intelligence' with 'world-wide availability'.

An understanding of ELM-ART's direction of interest can be seen in the authors' classification of ITS functionality into three groups or technologies: curriculum sequencing,
intelligent analysis of student solutions and interactive problem solving support. The first two technologies are considered by the authors to be "the oldest and best-studied technologies in the domain of ITS" and as such become "very usable in the WWW context". Therefore, it is these two technologies on which the authors of ELM-ART focus.

As a matter of interest, it is curriculum sequencing, including timely remediation and the presentation of this curriculum, which was of particular concern in the development of JASMINE. As tutoring systems increasingly become a guide to the student, it is these techniques which will aid his travels through hyperspace and the myriad available information.

4.4.1.1 Course Structure

Course material in ELM-ART is provided in two components: the textbook and the reference manual. It is the textbook which holds the most interest within the curriculum structure. Although basically an on-line version of the earlier hard copy textbook, the on-line textbook is structured hierarchically in such a way that concepts of interest are divided into levels by chapter, section, and subsection (i.e. similar to the instruction domain within JASMINE). This in itself is not unique; however, using this hierarchical structure, each unit of interest can be presented easily via a HTML Web page as required. Therefore, each page can focus on a particular course unit and any related links to other elements of the course. In addition, the authors have designed the system to present examples and problems (evaluation) on, "individual interactive pages which use the possibilities of WWW fill-out forms."

The filling out of forms, as mentioned by the authors, is used for the presentation of examples, and evaluation of students; however, as a matter of interest, any desired interface techniques are limited only by their availability within the WWW tools used. For example, items such as basic fill in the blank, frames, image maps, radio buttons, and check boxes are currently available in most graphical browsers, while JAVA applets can provide even more
detailed interfaces.

One aspect which makes the newer WWW course page more attractive is the types of links available to the student. The first type involves hierarchy navigation throughout the course material. The second type is the content-based link, so titled because the system "knows" what course concept is being presented on each page (i.e. "which concepts are introduced, presented, or summarized in each subsection, which concepts are demonstrated by each example, and which concepts are required to solve each problem") [Brusilowsky, 1996]. As a result, the system is able to provide links from each WWW page to related or interesting reference manual pages for each concept.

As designed, ELM-ART has become more than a basic WWW "hyperbook" in two aspects. First, the system 'knows' the material it presents to the student and directly supports the student in learning and navigating the course material (i.e. built-in intelligence). Second, all supplementary material (eg. examples and problems) in ELM-ART are not just text presentations, as in basic textbooks, but 'live experiences'. These 'live experiences' are a result of direct manipulation by, and interaction with, the student, through the provision of on-page completion of examples and guidance during problem solving attempted by the student. In other words, the system provides some degree of intelligent assistance together with the student-adapted presentation of conceptual material.

Within ELM-ART, course content has remained within the media of text and picture. However, as has already been mentioned during this chapter, course content has the potential to be presented using various additional media. As well, concept links could also be expanded to include additional material via the WWW which may be available in parallel to the actual course material. This would result in a freer learning style, when desirable, to meet the training goals of the domain.
4.4.1.2 Adaptiveness.

The ELM-ART system employs two adaptive techniques which were found to be of particular interest when studied. These were adaptive annotation and adaptive response (generation of HTML pages).

Within the ELM-ART adaptive annotation is implemented with "visual cues (icon, fonts, colors) to show the type and educational state of each link" [Brusilowsky, 1996]. Using the student model, which is represented by an overlay model, the system determines for each page of material whether the content of the page can be known, is ready to be learned, or is not ready to be learned, by the student. The icon and the font of each link (both content and hierarchical) is, therefore, presented to the student dynamically based on the student model.

In addition to the icon and font changes, the authors designed the system so that "when a student enters a page which is not yet ready to be learned, the system warns the student that this material has unlearned prerequisites and shows additional links to textbook and [reference] manual pages where the unlearned prerequisite concepts are presented". Basic help is also provided, when requested, with additional links to all pages where prerequisite concepts can be found.

Figure 4-1: Example from ELM-ART where adaptive annotation takes on the analogy of traffic lights. Red (italic) not ready, Yellow (normal) not recommended, and Green (Bold) ready.
The second technique, adaptive response, is employed whenever the student requests a new page. To generate the pages, the system uses the text of the course which is stored as an annotated HTML file, knowledge about the structure of the course, and the student's current knowledge state within the student model. The general process is very straightforward:

1. the server portion of the ELM-ART system identifies students by Uniform Resource Locators (URL)
2. the server calls an associated function passing enclosed fill-out form values
3. the function generates an HTML page as the adaptive response
4. the system returns the HTML page which was generated

4.4.1.3 Summary and Observations

Much of the design decisions within JASMINE can be wrapped up with the following sentiments from ELM-ART:

"In the WWW context, ITS can be located on HTTP servers in the research laboratories which have powerful equipment to run intelligent systems and ITS professionals to support and update the systems. At the same time, learners from over the world can access these ITS using any WWW browser. These browsers require relatively cheap hardware and can run on almost any platform. It gives ITS [real] world-wide audience[s] and unlimited source data for testing and improving their functionality" [Brusilowsky, 1996].

In addition, until substantially faster media transportation can be provided, much of the course material (especially for video and sound) must be located at the client or student end of the conversation (eg. down loaded earlier or provided via CD ROM). If this is not the case, large lag times within multi-media applications would most certainly cause distraction and tedium for the students who are eager to learn at their own pace.
4.4.2 ATULA - ATS

The second example system used in this thesis to focus on adaptiveness is the, "Adaptive Tutor Using Learner Attributes (ATULA) Adaptive Tutoring System" [Milne, 1996]. ATULA is claimed by the authors to be domain independent, although they have focused on 'Network Theory'. This system also focuses, although from a different angle, on the requirement for adaptive stereotyping of students, and the resulting ad-hoc presentation based on each student's knowledge state and learning attributes (stereotype). As such, each adaptive module in ATULA has variable parameters which can be used to modify its behaviour (eg. starting level of presentation version, maximum number of topic repeats, and variance in attribute change between new stereotype assignments).

As with JASMINE, the authors of ATULA are of the opinion that, "user modelling within tutoring systems [often pay] little attention to the user's individual characteristics in terms of capabilities and preferences". Therefore, within ATULA, domain information is represented by a basic overlay model, rather than a more complicated model. The choice of overlay model allowed the authors to focus on the importance of the attribute learner model. Hence, the main feature of development within ATULA is a model of the student's individual characteristics as related to characteristic stereotypes produced through subject group cluster modelling. As a result, the learner model is broken into two parts, the domain learner model LM-D and the attribute learner model LM-A. It is with the assignment of stereotypes or clusters that the LM-A is concerned.

4.4.2.1 Clusters / Stereotypes

The determination of stereotypes, and the assignment of students to the stereotypes in ATULA, was accomplished through cluster analysis. The cluster analysis was performed using only the data which would be available prior to a learner using the system, which resulted in a
choice of twelve clusters broken into six male clusters and six female clusters. After psychological instruments are used to measure personality traits, reasoning capabilities, and learning style, each student is assigned to one of the twelve clusters.

Once assigned to a cluster, presentation method and possible reassignment are determined by the student's Logic, Arithmetic, Diagram and Problem Size (LADS) attributes, as well as diagnostic question results. For example, within the LM-A, an estimate of the likelihood of the learner's success for each of the four presentation versions (table 4-4) is obtained from former student results with the higher probability being used for the next presentation. Cluster reassignment is achieved by analyzing the number of mismatched attributes, between the student and current cluster attributes for the topic, and the difference between the mean passed and mean failed topic profiles.

Assignment, however, to higher clusters (ie. higher in an attribute requirement) is accomplished only when the learner has succeeded the first time through a new section and shows a mean pass much greater than mean fail (L). Assignment to lower clusters may occur where mean pass is much lower than mean fail (D). This is the point where the remediation organizer is invoked to aid in the review of material by the student.

The authors define the term 'topic' within ATULA as, "a very small learning item such as a single concept or rule, and a group of topics selected for presentation is referred to as the 'presentation group', typically amounting to about four pages on first presentation, but reducing for remediation if some of the topics are judged passed".

1 Details of these clusters is available in "Development of a Model of User Attributes and its Implementation within an Adaptive Tutoring System" [Milne, 1996].

2 The Alice Heim test for verbal, diagrammatic and numerical capability. The Eysenck Personality Inventory and Kolb Learning Style Inventory and questionnaires which provide details of academic background, age, gender and attitudes to aspects of computer aided learning.
Passed and Failed Topic Profiles

![Bar chart showing comparison of passed and failed topic profiles.](image)

**Figure 1** Example adapted from [Milne, 1996] showing the comparison of passed and failed topic profiles.

### 4.4.2.2 Presentation

Presentation versions vary in the amount of support and the type of support given to the student and are based on the student's attributes, last presentation version selected, and the topics learning state (i.e., first time or remediation). The authors of ATULA selected a range of four presentation types based on the student's predicted difficulty with the presentation group, as listed:

1. **presentation type intended for users who are expected to have difficulty in learning the subject matter contained in the presentation group.** The explanations are presented in small stages, avoiding unnecessary use of mathematical terminology. Many working examples are provided, beginning with very simple illustrations and gradually increasing in complexity.

2. **presentation type designed for learners who prefer verbal explanations to**
diagrams or symbolic representations; the explanations are text-based as far as possible, and provide support for learning the necessary diagrammatic material contained in the domain. Some of the most easily worked examples are omitted from this version.

3. presentation type supports learners who prefer diagrammatic explanations, so the explanations depend where possible on visual representations of the material. The worked examples used are the same as for version 2.

4. presentation type intended for users for whom no difficulty in learning the domain is anticipated; explanations contain no additional support material, and there are less worked examples.

Two of the adaptive behaviours mentioned earlier, and incorporated within ATULA, are, the intensity of change between presentation groups, and size of presentation group. Changes between versions 1 and 4 are to be avoided. Instead of possible version changes for each topic or even subsets of the presentation group, it was decided to use the same version throughout a presentation group. In addition, the authors decided to reduce, "interference with the continuity of the presentation" and hence, the distraction for the student. However, between presentation groups and during remediation of presentation topics, the version may change.

4.4.2.3 Remediation

Responsibility for remediation within ATULA is placed within the remediation organiser (RO). It is the RO which decides which topics require further work. The authors of ATULA have placed the RO as a module unto itself; however in the typical ITS, this function would normally be accomplished within the pedagogy module described in Chapter Two of this thesis. Overall operation of the RO, like the other modules of the system, can be modified by the system based on adaptive variables. These variables are factors which effect such behaviours as the number of repeats, and movement between presentation techniques. As designed by the authors, the main function of the RO is to, "review the learner's records for the diagnostic questions and sets up the lists of topics, prerequisites, and diagnostic questions for the next presentation group".

4-22
Actions by the RO are mainly effected by the student's state of learning (ie. results during a diagnostic test). As described by the authors, "[the student] may have passed all the questions on that topic, he may have failed them all or he may have passed some and failed others". The first two results are straight forward; the system either moves on or remediates. The third result was of particular interest to the authors, "[as] there is a possibility that learning has taken place during the diagnostic test as the learner applies the topics to the diagnostic questions". Therefore, the third case results in a reduction in the probability that the student has learned the topic and remediated at a present level, with little consideration of the correct answers. This can be seen as very similar to the adaptive remediation found in SMART [Shute, 1995] and discussed in Chapter two.

When a learner encounters a topic for the first time, an assumption is made that he has succeeded in learning it and the probability of having learned that topic is set to 1. If he fails diagnostic questions on the topic, the probability is reduced by a variable decrement which can be modified by the system adapter up to a maximum value of 1, but if he passes all the diagnostic questions, the probability is set back to 1.

An important aspect of ATULA, which is revealed by the RO, is the system's ability to identify its limitations. At a predefined point, where the student has failed to grasp a presentation through repeated remediation, a message is generated to the student advising him to consult a human tutor regarding the given topic. A similar approach has been recommended in future development of JASMINE. Hence, as with ATULA the ultimate goal is to adapt and reduce the need for additional human assistance. It should be remembered, however, that both ATULA and JASMINE are tutors. Therefore, neither system will advance to the point of eliminating the human requirement of a knowledgable instructor.

4.4.2.4 Summary and Observations

The learner model, broken into the domain learner models and attribute learner models,
enables ATULA to 'computationally' control a larger number of a student's characteristics. Cluster analysis was used to manipulate background and psychological data obtained from the students to create stereotypes of users. Students are then assigned to the clusters most likely to suit their needs.

Although JASMINE uses only one user attribute to assign a new student to a stereotype, additional attributes could be added to the model. The model of the learner's attributes, LM-A, holds with the premise of hierarchical stereotypes within JASMINE, in that additional student attribute blocks could be included with the rules for activation and adaptation implemented within its stereotype framework.

As presented by the authors of ATULA, it is expected that extensive collection of records, including selected psychological and background characteristics, of users, and the abilities of users in these new attributes, would be required, and then analyzed to obtain useful stereotypes. For example, although JASMINE has focused on the stereotypes available within NLP, if the system was to expand its attribute list to include such aptitudes as working-memory capacity and general knowledge as presented by Shute [Shute, 1993], the student most likely would benefit from a more adaptive presentation size and timelier diagnosis.
Chapter 5:  
An Application

5.1 Introduction

Java Adapted Student Modelling in a NLP Environment (JASMINE), is an application which incorporates the ideas presented throughout this thesis. This chapter brings together those previously discussed principles and demonstrates how they can be implemented in a usable application suite. To begin, a brief description of the inherent domain structure and the demonstration domain is provided. This is followed with a discussion of the application suite which makes up JASMINE, along with a discussion of the modelling and adaptation practices used within the system.

5.2 Domain

As mentioned earlier, any domain which can be implemented in an hierarchical instructional manner can be used within JASMINE. In this fashion the domain is, therefore,
described as summary concepts (those within the structure of the domain tree), and base concepts (those concepts at the leaves of the domain tree). Although this structure does not boast a direct prerequisite structure, an indirect structure is inherent. Hence, siblings of the same parent are prerruositely ordered by their child numbers. This structure however, does not preclude the use of a direct prerequisite structure, as will be discussed in chapter six (Application Enhancements).

In addition to the basis hierarchical structure of the domain instruction, the course designer is able, if desired, to move beyond any restrictions that may be inherent, through the design of Base Concept web pages. From these pages the designer can implement any number of additional affects, including sound, imaging, WWW links, and direct manipulation by the student. It is important to note, however, that any direct, or free reign activity, engaged by the student should eventually be directed back to the structured training environment.

5.2.1 OPDP

The purpose of the Officers Professional Development Program (OPDP) is to broaden and deepen Canadian Force Officers’ knowledge and understanding of the military profession [CF, 1996] through self study. The OPDP program is comprised of six topics numbered from two to seven: 2 - General Service Knowledge, 3 - Administration and Training, 4 - Military Law, 5 - Service Support, 6 - National and International Studies and 7 - War and the Military Profession. The specific OPDP chosen for implementation within JASMINE was OPDP-4, Military Law. Success in completing an OPDP is based on a 60 percent exam result, with an honours pass being presented to those with an exam result of at least 80 percent. As a result, the 80 percent rate has been chosen as the mastery level for use within JASMINE.

Of the topics in OPDP-4, subsection 4-5-5, focusing on Physical Security was chosen for a more detailed implementation within JASMINE. It was determined after review of previous exam results that the section on Physical Security showed a reduced success rate, having
consistent results below the 60 percent pass rate.

5.3 Description of Application

A discussion of the three most important design features of the JASMINE application is provided in this section. The first of these features is the application suite that has been developed to support the course development and learning environments of the ITS. The second feature of interest is the implementation of student modelling within JASMINE, while the final feature to be presented describes how the application uses the information within the student model to manage adaptation in the overall application.

5.3.1 The Tools

The application suite provided with JASMINE is comprised of three semi-dependent tools. The first tool, the Domain Manager, is used to build and maintain the hierarchical structure of the chosen domain's instruction. This tool also can assist in the development of the web pages required for use of the stereotypical learning styles determined from the student's primary input channel. The second tool, the Question Manager is used to assist the course developer to provide evaluation questions to the student. The evaluation structure is currently based on true/false, multiple choice, and fill-in questions which are assigned to each of the base concepts within the hierarchical structure. The third, and final, tool is comprised of the actual ITS portion of the application. The Student Server tool implements the student session, manages the student model, answers communication requests from client side applets, and adapts the presentation and evaluation.

5.3.1.1 Domain Manager

As mentioned, the Domain Manager is used to build and maintain the domain structure of
a specific domain and has two specific functions. The first of these functions is the management of the actual hierarchical structure, while the second is management of the concepts' (summary and base concepts) web presentations. The physical structure of the Domain Manager is provided showing the path of course development and maintenance (fig 5-1). Each of the manager's blocks is realized as a separate class within the Java environment and is instantiated by its parent in the path.

![Diagram of Domain Manager Structure]

A typical session use of the Domain Manager begins with the course developer entering the name of the particular domain of interest. Once provided with this information, the Domain Manager displays the current structure of the domain (fig 5-2). The information required to create this hierarchical structure is stored in a file identified by its domain description, \textit{DOMAINNAME}.dbf, within the \textit{DOMAINNAME} directory (eg. the file is found in OPDP/OPDP.dbf).
At this point, the developer can add base concepts (which later may become summary concepts when children are attached to them), delete base concepts, or modify either summary or
base concepts. If any of these actions is taken the 'Apply' option is provided to the developer, who may then chose to save any changes to the course structure. When the modification of a concept is chosen, the tool then provides the developer with the sub tool called the Concept Manager (fig 5-3).

![Concept Manager](image)

Figure 5-3: Concept Manager

The Concept Manager is used to modify the concept description and to manage the primary pages of each concept based on the input channels, A (Audio), V (Visual), and K (Kinetic). These pages are considered primary as they are the first pages which a student will visit on introduction to a concept. From these pages, and depending on the HTML features employed by the course designer, the student may navigate beyond the immediate features of the JASMINE ITS. As mentioned, the primary pages may be managed from this tool, therefore, when a file is selected, it is presented in its current form, or in a basic skeleton form if the file has not been created previously (fig 5-4).

From this point, the designer can modify and save changes to each page individually. It is important for the designer to realize that the <!children> and <!navigation> tags must be present within the primary pages to allow proper operation of JASMINE. Primary pages are stored in the DOMAINNAME directory, based on their concept name and input channel, as conceptname_inputchannel.html (e.g., OPDP/4-5-5_a.html). Once development of the domain has been implemented in this manner, the next tool of interest to the course developer is the Question
Manager.

The National Defence Act and the Government of Canada Security Policy (GSP) assign to the DND the responsibility of all aspects of security for defence works, establishments, facilities, assets, resources and operations in Canada and abroad. The GSP provides that, within the DND and the CF, where standards developed on the basis of the GSP are not effective or practical in the context of military or national emergency operations, commensurate security may be provided.

The objective of this topic is to increase your understanding of:

- the orders pertaining to the physical security of defence establishments;
- the custody of classified and designated matter;
- the security of small arms, ammunition and explosives;
- passes; and
- guarding of defence establishments.

You will confirm that you have achieved the objective of this topic to the desired level through the indicators listed below, any or all of which may be the subject of an examination item or items.

The reference material for this topic is taken from Security Orders, Part 4, Chapters 30, 32 (Annexes G and H), 33 and 35 to 38.
5.3.1.2 Question Manager

As previously mentioned, the Question Manager is used to input, and attach, evaluation questions to each of the domain's base concepts. It is the results of these questions, when presented to the student at the browser end, which are a primary factor in the adaptation of JASMINE. From these results, the student knowledge levels and overall success are monitored. This in turn, affects the choice of presentation style, recommended navigation and even future evaluations. Although important for a complete training experience, the Question Manager is simple in structure (fig 5-5).

![Question Manager](image)

![Question Viewer](image)

Figure 5-5: Question Manager Structure.

From the initial use of the Question Manager, the course developer enters the domain of interest and then is provided with the Question Viewer. Upon access to the Question Viewer (fig 5-6), the developer may proceed with either the entry of new questions or the maintenance of existing questions.

Entry of a new question is accomplished in two steps. The first, is entry of the Base Concept name (number) which causes the description of the concept to appear as confirmation of correct name entry. The developer then proceeds with entering question details: type of question,
question text, alternatives for multiple choice questions, and of course, the answer. Upon pressing the apply button for the question, the next available question number is assigned and the file is stored as \textit{DOMAINNAME}/questions/conceptname.qn (eg. OPDP/questions/4-5-5.q3).

Figure 5-5: Question Manager
Modification of an existing question is accomplished in a manner similar to that used for the new question. After entry of the Concept name, the developer simply enters the corresponding question number and the details are provided for modification. Now that the course structure and evaluation bank have been provided for use by JASMINE, training may begin.

5.3.1.3 Learning Time: Student Server and Client

The final tool available in the JASMINE application suite is the runtime portion of the application. This tool implements the student session, manages the student model, and adapts the presentation and evaluation. As this portion of the suite is designed for implementation via a Java capable browser, it has been created using the Server / Client approach.

The Server side of the JASMINE application handles the majority of the application's responsibilities, while the client side, downloaded via HTML web pages from the server's location, makes requests as required of the server portion. Hence, the Java components which comprise the Server and Client have been appropriately written as a combination of a Java application and applets (fig 5-7).

Use of the ITS portion of JASMINE begins with the instantiation of the Student Server. Then, once the domain of study has been entered, the Student Server builds and populates the student model based on the expert model (created by the Domain Manager), and previous student sessions. If the primary input channel of the student has not yet been determined, then the Student Server implements the NLP testing frame and the student is tested. The introduction to the test and a sample question are provided below in the adaptation section of this chapter.

In addition to the testing, the function of the Student Server, at this point, is to instantiate the Request Thread for applet communications and the Link Thread, so called because its
primary role is the interpretation of the student model and the colour coding of navigational links within the primary pages of the domain. The job of the Link Thread, at this point, is to build the student's starting page of instruction (fig 5-8).

Server / Application

- Main
- Request Thread
- Question Thread
- Link Thread

Client / Applets

- Concept Page
  - <HTML>
  - <Applet ....>

- Evaluation Page
  - <HTML>
  - <Applet ....>

Figure 5-7: Client / Server Structure of JASMINE

From the start page created by the Student Server and imported by the Browser, the student can begin navigation through the domain. Communication with the Student Server continues on two occasions. The first of these occasions is on the activation of a primary page by the Browser. When this activation occurs, the imbedded applet initiates communications with the Student Server. Ad-hoc HTML pages are then created by the server for each of the links (eg. each child, next sibling, evaluation, etc ...) which was previously included by the Link Thread of the Server and the communicating page is marked as the current page of instruction.
Indicators

You will confirm that you have achieved the objective of this topic to the desired level through the indicators listed below, any or all of which may be the subject of an examination item or items.

Link Applet will be inserted here

- Physical Security of Defence Establishments
- Physical Security Measures
- Custody of Classified and Designated Matter
- Security of Small Arms Ammunition and Explosives
- Use and Issue of Passes
- Issue of Small Arms

Evaluation

Reference Material

The reference material for this topic is taken from Security Orders, Part 4, Chapters 30, 32 (Annexes G and H), 33 and 35 to 38.

Previous: Personnel Security

Next: Information Security

Up: Security

Figure 5-8: Typical Browser Page
The second occasion for communication occurs whenever the students choose to be evaluated on their knowledge of the current summary or base concept. When this situation occurs the embedded evaluation applet initiates communication with the Server Request Thread which activates the Question Thread. The Question Thread then establishes a current question bank. The question bank is based on the student's knowledge level for each of the 'legacy' base concept of the current concept. It should be noted that if the current concept is a base concept, then it is the only 'legacy' base concept. The Question Thread and Evaluation applet then continue to communicate, passing each question and its result (updating the knowledge level of each corresponding concept) until the question bank is exhausted. On completion of the evaluation period, the current HTML page of instruction is updated via the Link Thread and the student returns to the domain¹.

On completion of the session, the browser is closed and the Student Server is deactivated. This operation causes the current student model to be written to disk as:

/student/DOMAINNAME.dbf (eg. /student/OPDP.dbf)

5.3.2 Use of Modelling

As described earlier in Chapter Three, student modelling within JASMINE is based on the amplitude differential model. This model is used to manage the knowledge state of the student. However, the structure of the overall student model contains much more information than can be attributed to a single overlay model. Accordingly, JASMINE records additional information such as NLP test results, preferred input channel, and an inherent prerequisite vector for concepts within the area of study. In this way, the student model is constructed of a single student object which contains, by way of reference, a tree structure of concept objects (fig 5-9).

¹ It is important to note that the browser must be configured to update each page as loaded to ensure changes to navigation links are provided to the student.
The concept portion, or knowledge tree of the student model, is originally constructed from existing domain or expert structure during a student's initial session. Throughout the history of a student's study within the domain, the student model is maintained continually and interpreted by the Student Server. It is through this interpretation that the server is able to adapt to individual student requirements.
5.3.3 Adaptation

In an effort to present information in the best manner and to provide evaluation during each student session, four particular areas of JASMINE which adapt to the student model have been implemented. These areas range from the establishment of the learning model to evaluation of the student when requested.

5.3.3.1 Initial Learning Model

The initial learning model is established during the first student session within a new domain of study. Upon initialization of the new domain, via the Student Server, the student is interviewed with the intention of establishing the primary input channel (Audio, Visual or Kinetic). The student responds to each question asked by ranking each of the situations provided (eg. fig 5-10).

![Figure 5-10: Typical Interview Question](image)

The interview, in itself, can be considered adaptive. Although there are 18 questions in total, if at any time the student's primary input channel can be established with little uncertainty then the interview is complete. This can occur in two ways: if a single trait's values have
established it far beyond the values of the other traits (ie. twice in value); or if based on the remaining questions neither of the lower two traits could exceed the greatest of the three. At these points the input channel is established.

5.3.3.2 Changes in Presentation

Changes in presentation are a direct result of the current student input channel. Although JASMINE does not affect presentation based on the input channel directly, the framework for such changes has been provided. JASMINE's function is to select HTML files for the current summary or base concept, based on the input channel. These HTML files, although managed by JASMINE (Domain Manager), are created by the course designer. In this way, the course designer is able to control the use of all media, as well as to control introduction of new material, or navigational routes through Internet links.

5.3.3.3 Navigation Links

Adaptation of JASMINE controlled navigational links is based on the current knowledge level of the student in relation to each of the links. Hence, each link is colour coded according to its recommended use by the student (fig 5-11).

In this way, four colours are used within JASMINE:

- **White** - concept has been mastered
- **Green** - concept study or evaluation is recommended
- **Yellow** - concept is not recommended, as it may have some prerequisite knowledge
- **Red** - evaluation is not recommended at this time

---

2 Colour codes for JASMINE are adapted from the colour scheme of ELM-ART [Brusilovsky, 1996].

5-16
Use and Issue of Passes

Figure 5-11: Link Colour Coding

As described earlier, each of these links is updated constantly as knowledge levels are updated due to evaluation of the student.

5.3.3.4 Evaluation

Evaluation of the student is the primary way in which adaptation within JASMINE is
triggered. Apart from the fact that the evaluation itself is adaptive (ie. basing the number of questions on current knowledge levels) it is the student's success and failures which direct other system adaptation.

The first evaluation effect in on presentation style. As can be see from the attributes of the student object (fig 5-9), JASMINE counts the number of question posed to the student and the number of correct responses. From these counts, a ratio of the overall success of the student while using the current input channel, is established. If this ratio begins to fall below a predetermined point\(^3\), then presentation is adapted by reassigning the primary input channel to the next likely value.

The second effect evaluation has on system adaptation is in its straight forward modification of the concept knowledge levels. To this end, knowledge levels are modified, based on the equations provided by Shute [Shute, 1995], at the level-0 for correct responses, and level-2 for incorrect responses. Further integration of the remaining equation levels has been recommended in Chapter Six. Changes in knowledge level, in turn, affect link colour coding as discussed.

5.4 Summary

JASMINE is a good development suite for courses where domains may be structured for instruction in a hierarchical nature. It demonstrates all the functions of an adaptive intelligent tutoring system. In addition, the object oriented nature of the applications allows for easy integration of new tools and enhancements. The application is viable, and shows potential for use in the development of a complete distributed course of self study.

\(^3\) As the expected, master level for JASMINE has been set at .80 the success ratio has been selected at .70. This value is tested after an initial 25 question have been presented and may be modified by the system designer.
6.1 General

This thesis has described the work completed in the research and development of an ITS framework. The framework presented may be used for any domain where the instructional information can be structured in a hierarchical manner. With the introduction of prerequisite vectors, the structure will not be limited to one that is hierarchical. An introduction was given which covered a brief history of the ITS and the essential concepts which support the development of an ITS. From these basic concepts, the first chapters of the thesis reviewed current work in the field, and presented those ideas adapted for use in JASMINE.

The first idea to be covered in detail was the conceptual structure of the typical ITS. The structure involved five functions arranged in five conceptual modules: pedagogy, psychologist, interface, domain expert and student model. As these were conceptual modules, it was important to cover their implementation in several existing systems. Therefore, the first and second of three reviews presented research involving current ITS architectures and pedagogical strategies.
In addition, two systems, SUMIT and SMART, which had a direct influence on the development of JASMINE, were presented in more detail. SUMIT provided guidance which directed JASMINE towards a more pragmatic system targeted on learning and the user interface, whereas SMART provided direction towards a responsive system aimed at a more probable understanding of the student's current knowledge state.

Once the ITS conceptual structure was described, it was important to develop two concepts which have the greatest effect on the final ITS. These concepts are student modelling and adaptiveness, both of which have had a direct influence on the student interface. Student modelling, a direct specialization of User modelling, is used within the ITS to monitor the system's beliefs of the student's characteristics (including stereotype groupings), and knowledge levels. The model is then used and maintained in an effort to adapt student presentation with the goal of achieving maximum student performance. Stereotyping of the student within this thesis was provided using NLP testing, where, the students is grouped in one of three stereotypes (auditive, verbal or kinetic) based on the student's primary input channel. Student modelling can be viewed in a several ways; therefore, a review of two papers involving current work in student modelling was also provided.

In an effort to best model the student within JASMINE, two specific avenues where explored. The first avenue was the User modelling shell BGP-MS, which at first pass, was believed to possess the tools necessary to maintain the more specific student model. After additional review of BGP-MS was completed, it was determined that the shell was lacking in several key areas, the most pronounced of which was its lack of a pragmatic environment. BGP-MS is very restricted in its software use and platform availability. The second avenue of approach was the use of JAVA, a language which was initially intended to be used as an interface to the ITS's browser interface. After the difficulties with BGP-MS were determined, it was resolved that the JAVA language, being very flexible and pragmatic, could be used independently of BGP-MS, producing the whole of the intended ITS.
The final point of interest in the development of JASMINE was research in the area of adaptation and Intelligent User Interfaces. Work on the IUI, being a close relative of the ITS, directed much of the importance placed on the interface within JASMINE. It is the intelligence of any system which allows it to maintain and monitor the Student model, and from the model, to provide the best possible presentation or interface. Hence, the IUI portion of the thesis provided a review of recent discussion regarding the design issues of the IUI. In particular, the questions of interface agents and degree of intelligence were addressed, with the main response being 'value added'. The intelligence of the system must assist the user in some measurable degree while leaving a sense of user control. This section also provided a more in-depth review of two existing systems, ELM-ART and ATULA-ATS. ELM-ART had a particular influence on JASMINE, due to its application to the World Wide Web, and ad-hoc presentation based on the student’s current knowledge state. ATULA-ATS's influence came in its use of student stereotypes.

The final result of the above research and review of current work is the ITS JASMINE (Java Adaptive Student Modelling In a NLP Environment). As the name implies this system has employed all of the imperative ideas presented throughout this thesis and can be used for any hierarchically based study domain.

6.2 Testing

As each function and component of JASMINE was implemented, scenario testing was used to detect any abnormalities within the component, as well as among existing components. Scenario testing involved the use of the system tools and learning environment in every conceived manner, including attempted misuse, and simultaneous tool operation. After such testing, the system, including its development and student learning tools, was deemed operational for its intended purpose as an adaptive training environment.
Additional empirical testing to prove the training efficiency of JASMINE could not be completed in the time frame of this thesis. OPDP testing is completed in the months of December, March and June of each year. This time frame does not allow for complete course transfer to the JASMINE domain, student training and evaluation of examination results. However, it is proposed that JASMINE be enhanced beyond the single user system to a multi-user system; this will be discussed later in this chapter. Once enhancement has been completed, the system is to be used to present a course at the Royal Military College Canada (RMC). It has been proposed that the 'Data Base' course set in the SQL environment would be a candidate for JASMINE. This event would provide complete conceptual testing of JASMINE, hence opening the door to future enhancements, including student characteristic environments of particular interest at RMC.

6.3 The Aim Revisited

The original aim of the thesis was:

"to present research which has been completed towards the development of an adaptive student tutoring system."

From the information presented during this thesis it can be clearly seen that the aim of an adaptive student tutoring system has been achieved in JASMINE. In particular, JASMINE is intelligent in its awareness of the student's characteristics and knowledge state. As a result, JASMINE is adaptive in its recommended course of study and presentation, based on the student knowledge state and characteristics, respectively. Adaptation also takes place on a minor scale in the determination of student interview questions and evaluation questions. The research needed to complete this task has also been presented. Information detailing the conceptual areas of development, review of related works, and detailed analysis of work having a direct influence on development have been presented. Therefore, the aim of this thesis has been met.
6.4 Suggested Enhancements

Several enhancements to the JASMINE application are possible. The following is a list of those enhancements that are considered the most beneficial to course development.

a. *Mastery Indicator.* This indicator would be located in each concept object and would be used to indicate current or previous mastery of a subject. In this way, the student, as well as the presentation, can be directed towards a more realistic remediation environment. This could be accomplished as simply as reminding the student that they have reviewed the remediated information previously.

b. *File Editing.* The integration of a file management environment, or enhancement of the file viewer embedded within the Domain Manger, would enhance the capabilities of the tool by allowing the complete editing of the base HTML files.

c. *Student Model Viewer.* This viewer would add to the JASMINE tool set and would allow easy review of the student knowledge set by the instructor. In environments where the student has more control, this tool could be of use to the student as well, if it were adapted to the browser environment (eg. colour coding of the entire domain or a subset of the domain).

d. *Current State.* Inclusion of the last concept visited or current node of interaction in the student object would allow the student to begin a new session where the last left off.

e. *Prerequisite Vectors.* This enhancement would allow the system to colour code navigation links based on direct prerequisite vectors as well as the inherent prerequisite already built into JASMINE. The use of prerequisite vectors would
involve additions to the Domain Manager which would allow definition of the vectors. The Concept class would be modified to include the storage and retrieval of prerequisite vectors and the Student Server modified to interpret the significance of the vectors.

f. **Multiple Students.** This enhancement would allow the Student Server to manage and maintain several student models simultaneously. It is suggested that the students would have to log into the server where a student object and knowledge base would be established. Using the JAVA Vector object, individual student objects could be managed in a list structure allowing for integration and removal of additional student objects. Two additional features of JASMINE would also have to be adapted to the multiple student environment. First, NLP testing would have to be integrated into the browser environment, possibly as an individual thread. Second, some mechanism or algorithm would have to be established to handle the storage of the student object when a student ceases to use the system or times out.

6.5 Future Research

The field of ITS offers several areas of research, some of which have been covered in this thesis (ie. architecture, teaching and learning strategies, and student modelling). Other areas include help and advising systems, knowledge representation, learning environments, and collaborative learning. In particular, there are two areas of research which may increase the usefulness and effectiveness of JASMINE, natural language and cognitive modelling.

Question management in JASMINE is currently at the complete discretion of the course designer. It is felt that research in the natural language area would be of benefit in the intelligent generation of evaluation questions. In this manner, the course designer would set parameters for
questions centred on base concepts within the domain. These parameters would include primary focal points within the concept principles and allow natural language processors to generate ad-hoc questions of the true/false, multiple choice, fill-in, and possibly, short answer, varieties. It is believed that this type of question generation would allow for a wider variety of questions and would reduce the possibilities of evaluation repeats.

Basic student characteristics in JASMINE are founded on the Neuro Linguistic Programming principles. NLP was chosen for JASMINE because it offered a fundamental foundation for the addition of student characteristics or cognitive functioning in the student model. However, within the realm of cognitive modelling, these are additional features which might be considered for inclusion in the JASMINE student model.

One such cognitive environment involves presentation and evaluation based on three student preferences; simply put: general vs specific; procedural vs optional; and internal vs external [Angers, 1997].

- **General vs Specific.** A general personality learns better when a topic is first presented in a general manner moving from fundamental principles to more-specific applications. The specific personality enjoys learning more when case-by-case examples of specific principles are presented and then generalized for future use.

- **Procedural vs Optional.** A procedural student desires an environment which is very structured, moving through the course of study in a patterned manner and directing the student through every learning situation. The optional student enjoys the freedom of learning in a more exploratory manner; choices of study path and which examples and learning situations to explore, are opportunities this student enjoys.
Internal vs External. Internal individuals make their own determination of success; therefore, they must be guided to domain mastery through evaluation which provides self-affirmation, particularly when system and student determination are in conflict. External individuals make determination of success based on external stimulation; hence, they require constant and immediate feedback on successes and failures during evaluation and will absorb specific direction well.

It is believed that these type of cognitive environments, combined or not combined with the NLP environment, will serve to enhance the abilities of the student and the tutoring effectiveness of JASMINE. With the integration of the suggested enhancements, and involvement of the natural language and cognitive sciences, the complete intelligent tutoring system becomes a multi-discipline project. This thesis has focused on the conceptual features of the ITS and some specific features as related to the field of computer science.
List of References


Reference-1
List of References


List of References


Reference-3
List of References


List of References


### List of References

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
</table>

Reference-6