A METHODOLOGY FOR EVALUATING INTELLIGENT TUTORING SYSTEMS

by

INDIRA PADAYACHIE

submitted in part fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in the subject

INFORMATION SYSTEMS

at the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: Mrs M.R. DE VILLIERS

JUNE 2000
Abstract

This dissertation proposes a generic methodology for evaluating intelligent tutoring systems (ITSs), and applies it to the evaluation of the SQL-Tutor, an ITS for the database language SQL.

An examination of the historical development, theory and architecture of intelligent tutoring systems, as well as the theory, architecture and behaviour of the SQL-Tutor sets the context for this study. The characteristics and criteria for evaluating computer-aided instruction (CAI) systems are considered as a background to an in-depth investigation of the characteristics and criteria appropriate for evaluating ITSs. These criteria are categorised along internal and external dimensions with the internal dimension focusing on the intrinsic features and behavioural aspects of ITSs, and the external dimension focusing on its educational impact. Several issues surrounding the evaluation of ITSs namely, approaches, methods, techniques and principles are examined, and integrated within a framework for assessing the added value of ITS technology for instructional purposes.

Key terms (in alphabetic order):

Artificial intelligence; Computer-assisted instruction; Constraint-based modelling; Design principles for computer tutors; Evaluation criteria; Evaluation framework; Evaluation techniques; Generic evaluation methodology; Intelligent tutoring systems; Internal and external approaches to evaluation; Learning and instructional theories; and Usability.
MY SINCERE APPRECIATION TO...

Ruth De Villiers, my supervisor, for the valuable advice, continuous support and encouragement, prompt return of chapter submissions, as well as the constructive criticism provided.

The Head of Department, Mr T. Nepal and colleagues in the Department of Computer Studies at the M L Sultan Technikon, who provided support and technical assistance in the production of the document.

In particular, I would like to thank Prof. Don Petkov for serving in the capacity of a mentor, by providing relevant literature sources, reading through my thesis and offering advice and suggestions for improvement.

The Computer Science librarian at UNISA and the library staff at the M L Sultan Technikon for providing an excellent quality of service.

The National Research Foundation for providing financial assistance for this research.
# TABLE OF CONTENTS

## CHAPTER ONE

### Introduction

1. Context and subject matter of the research
   1.1 The context: automated intelligent tutoring for computer programming
   1.1.2 The specific subject matter: SQL database language

2. The main goal and sub-goals of this dissertation

3. Scope and delimitation of the research

4. Research design

5. Importance of the research

6. Structure of the dissertation

## CHAPTER TWO

### Intelligent Tutoring Systems: History, Theory and Architecture

1. Introduction

2. History and development of ICAI/ITS
   2.2 Evolution in the development of ITSs
      2.2.1 ITS development issues in the 1970s
      2.2.2 ITS development issues in the 1980's
      2.2.3 ITS development issues in the 1990's

2.2.2 The shifting development focus of ITSs
   2.2.2.1 Stage 1: early 1970's - middle 1980's
   2.2.2.2 Stage 2: mid and late 1980's
   2.2.2.3 Stage 3: early 1990's and subsequent
CHAPTER THREE

Theory, Architecture and Behaviour of the SQL-Tutor

3.1 Introduction 48
3.2 Constraint-based modelling 49
   3.2.1 A formal notation for constraints 49
   3.2.2 Use of constraint-based modelling for educational applications 50
   3.2.3 State constraint theory 51
3.3 Application domain - learning SQL 51
3.4 The SQL-Tutor: implementation & architecture 53
   3.4.1 The interface 54
   3.4.2 The knowledge base 57
      3.4.2.1 Syntactic constraints 58
      3.4.2.2 Semantic constraints 60
   3.4.3 Student modeller 61
3.4.4 Pedagogical Module 62
  3.4.4.1 Feedback 62
  3.4.4.2 Examples of interactions with the SQL-Tutor 63
  3.4.4.3 Problem selection 66
3.5 The SQL-Tutor's role in instruction and learning 66
3.6 Conclusion 67

CHAPTER FOUR

Evaluation of Computer-Assisted Instruction and Intelligent Tutoring Systems: Characteristics and Criteria

4.1 Introduction 70
  4.1.1 What is an evaluation? 70
  4.1.2 Why are evaluations necessary? 70
  4.1.3 What is the role of evaluation in ITS applications? 71
  4.1.4 Overview of Chapter Four 72
4.2 Characteristics of and evaluation criteria for conventional CAI 73
  4.2.1 Characteristics of CAI 74
  4.2.2 Evaluation criteria for CAI 75
    4.2.2.1 An evaluation model 75
    4.2.2.2 Focus of evaluation forms and checklists 77
    4.2.2.3 Evaluation criteria for educational technology 79
    4.2.2.4 CAI evaluation framework 80
    4.2.2.5 Hexa-C Metamodel 82
4.3 Characteristics of and evaluation criteria for ICAI/ITSs 82
  4.3.1 ITSs compared and constrained with conventional CAI 83
  4.3.2 Characteristics of ITSs 85
CHAPTER FIVE

Evaluation of Intelligent Tutoring Systems: Principles, Approaches, Methods and Framework

5.1 Introduction 103

5.2 Principles for evaluating ITSs 104

5.2.1 Principle 1: Delineate the goals of the ITS 104

5.2.1.1 What instructional approach underlies the tutor? 104

5.2.1.2 What learning theory does it assume? 105

5.2.1.3 What exactly does it teach? 105

5.2.1.4 What other impacts is it expected to have? 105

5.2.1.5 In what context is it supposed to operate? 105

5.2.2 Principle 2: Define the goals of the evaluation study 106

5.2.2.1 What should be known after the study is completed? 106

5.2.2.2 By what standard will success be measured? 107

5.2.2.3 What are potential confounds, and which of these can be controlled? 107

5.2.2.4 Will quantitative indices, protocols, or observational data be used? 108

5.2.3 Principle 3: Select an appropriate design to meet defined goals 108
5.2.4 Principle 4: Instantiate design with appropriate measures, number and type of subjects, and control groups  
5.2.4.1 Learning outcomes (or dependent measures)  
5.2.4.2 Independent measures (individual differences)  
5.2.4.3 Control conditions  
5.2.4.4 Subjects  

5.2.5 Principle 5: Make necessary logistical preparations for conducting the study  

5.2.6 Principle 6: Pilot test the ITS and the study  

5.2.7 Principle 7: Plan the primary data analysis when the study is planned  

5.2.8 Summary of principles  

5.3 Approaches to evaluation of educational technology  
5.3.1 Formative and summative evaluation  
5.3.1.1 Formative evaluation  
5.3.1.2 Summative evaluation  
5.3.2 Internal and external evaluation  
5.3.2.1 External evaluation  
5.3.2.2 Internal evaluation  

5.4 Techniques and methods for evaluating ITSs  
5.4.1 Techniques for formative and summative evaluation of ITSs  
5.4.1.1 Proofs of Correctness  
5.4.1.2 Criterion-based evaluation  
5.4.1.3 Expert inspection and Turing test  
5.4.1.4 Certification  
5.4.1.5 Sensitivity Analysis  
5.4.1.6 Pilot Testing  
5.4.1.7 Experimental Research  
5.4.1.8 Objections to the use of different techniques for formative and summative evaluation
5.4.2 Methods for external evaluation of ITSs
5.4.2.1 External evaluation method: the cognitive perspective 120
5.4.2.2 External evaluation method: the learning achievement and learning affect perspective 122
5.4.3 Methods for internal evaluation of ITSs
5.4.3.1 Internal evaluation method: knowledge engineering techniques 123
5.4.3.2 Internal evaluation method: formative evaluation techniques 126
5.4.3.3 Comparison of techniques for internal evaluation method 127
5.5 The Brown Wallnau framework for evaluating software technology 128
5.5.1 Descriptive modelling phase 128
5.5.2 Experiment design phase 129
5.5.3 Experiment evaluation phase 130
5.6 A proposed generic methodology for evaluating ITSs
5.6.1 Techniques for external evaluation of ITSs 135
5.6.2 Techniques for internal evaluation of ITSs 135
5.6.3 Proposed framework for evaluating ITSs 136
5.7 Conclusion 138

CHAPTER SIX

Evaluation of the SQL-Tutor: An Intelligent Tutoring System for the database query language SQL

6.1 Introduction 140
6.2 Descriptive modelling phase 141
6.2.1 The ITS/SQL-Tutor genealogy 142
6.2.2 SQL-Tutor Habitat 144
6.3 Evaluation design phase 144
6.3.1 External evaluation 146
6.3.1.1 External evaluation criteria 146
6.3.1.2 Experimental design

6.3.2 Internal evaluation

6.4 Evaluation implementation phase

6.4.1 External evaluation

6.4.1.1 Learning achievement

6.4.1.2 Learning affect

6.4.2 Internal evaluation of SQL-Tutor

6.4.2.1 Analysis of domain model

6.4.2.2 Analysis of tutoring model

6.4.2.3 Analysis of student model

6.4.2.4 Analysis of overall system control

6.4.2.5 Analysis of design principles

6.4.2.6 Analysis of usability

6.4.2.7 Analysis in respect of current directions in learning and instructional theories

6.4.2.8 General

6.5 Conclusion

CHAPTER SEVEN

Conclusion

7.1 What has been accomplished?

7.2 Generic methodology for evaluating ITSs

7.3 Results and recommendations from evaluation of SQL-Tutor

7.4 Conclusion
LIST OF TABLES

1.1 Aspects investigated & their relevance to ITSs 5
2.1 Important issues related to ITS development 11
4.1 Criteria for internal evaluation of ITSs 100-101
4.2 Criteria for external evaluation of ITSs 101
5.1 Suitability of evaluation techniques 117
5.2 Proposed methodology for evaluating ITSs 132-133
6.1 Links between evaluation issues, theory and application 140-141
6.2 Evaluation criteria for internal evaluation of ITSs 150-151
6.3 Pre-test and Post-test Mean Analysis 153
6.4 Statistical analysis of Pre-test 154
6.5 Statistical analysis of Post-test 154
6.6 T-test results 155
6.7 Application of evaluation framework 195
6.8 Domain model 195
6.9 Tutoring model 196
6.10 Student model 197
6.11 Overall system control 197
6.12 Design principles 198
6.13 Usability properties 198
6.14 Learning and instructional theories 199
6.15 General 199
LIST OF FIGURES

1.1 Structure of dissertation 7
2.1 General Framework for ITSs 29
2.2 Siemer's & Angelides's general intelligent tutoring system architecture 34
2.3 McCalla's & Greer's SCENT-3 Architecture for Intelligent Advising Systems 40
2.4 Costa's & Perskuchisk's Architecture of MATHEMA for IT Learning Environment 43
2.5 Self's ITS components 44
3.1 Architecture of the SQL-Tutor 54
3.2 Log-in interface 54
3.3 Getting-started window 55
3.4 Main interface of SQL-Tutor 56
3.5 Help Screen 57
3.6 Open Menu 57
3.7 LISP code for constraint 2 58
3.8 LISP code for constraint 146 59
3.9 LISP code for constraint 22 60
3.10 LISP code for constraint 361 60
4.1 ICA/ITS Domain 72
4.2 Shute's & Psotka's CAI Model 73
4.3 CAI Evaluation Framework 80
4.4 Internal evaluation of intelligent tutoring systems 98
5.1 External evaluation of an ITS 122
5.2 Internal evaluation of an ITS 124
5.3 Technology delta evaluation framework 131
5.3 Proposed evaluation framework for ITSs 137
6.1 ITS/SQL-Tutor genealogy 143
6.2 SQL-Tutor Habitat 144
6.3 Experiment: Pre-test & Pre-test data 153
F.1 Use of computed columns 235
F.2 Use of SQL word LIKE 236
F.3 Use of NOT operator 237
F.4 Use of nulls 238
F.5 Sorting with single key 239
F.6 Use of sub-query 240
F.7 Use of aliases 241
F.8 Use of SQL word ALL 242
CHAPTER ONE

Introduction

This chapter introduces the study by reviewing the background to the development of intelligent tutoring systems. A brief discourse on the context and specific subject matter of the study is provided, together with an outline of the study's goals and sub-goals, scope and delimitation, and research design. Finally, the importance of the research is discussed, and a brief outline of the remaining chapters follows.

Instructional systems resulting from the use of artificial intelligence (AI) techniques in computer programs to facilitate learning are referred to as intelligent computer-assisted instruction (ICAI) or intelligent tutoring systems (ITSs). The design and development of such programs lie at the intersection of computer science, cognitive psychology and educational research, comprising a discipline referred to as cognitive science [Kearsley, 1987].

From the 1970's to the present day, ITSs are heralded as one of the most appropriate approaches for delivering individualised instruction. Before launching into a discussion of intelligent tutoring systems per se, it would be prudent to briefly review the background to the development of intelligent computer-assisted instruction and intelligent tutoring systems.

In the 1960's research commenced into the use of computers in education. The end products of such research efforts were computer-assisted instruction (CAI) and computer-based training (CBT) systems. There was much optimism about potential applications of computers in the fields of education and training. The development of CAI systems was largely influenced by theories of behaviourism, which aimed at reducing every psychological process, including learning, to a stimulus-response causal model. This provided a one-way teaching interaction whereby the system controlled the interaction using rigid pre-defined dialogues that could not adjust to the needs of individual students. The system's evaluation of the student's learning was based primarily on a comparison of the student's answers with a limited number of pre-defined stored answers [Garito, 1991].

Researchers in AI and cognitive psychology in the 1970's pursued the same goal as CAI, namely the promotion of single teaching interaction between a teacher and a student. They,
however, chose cognitive psychology as the basis of a theory of learning and used methodologies and techniques from the field of AI to design a new breed of computer-assisted instruction systems that could perform intelligent activities (e.g. solving a mathematics problem, understanding natural language, programming a computer, etc.). These systems were initially known as ICAI systems, later as ITSs and of late, intelligent educational systems (IESs). These systems abandoned the stimulus-response model of traditional CAI in favour of a mixed-initiative dialogue, tailored to the needs of the individual student. Garito [1991] outlines the distinguishing features of the architecture of ITSs as follows:

• Expertise on the subject to be taught, implemented by knowledge representation techniques;
• Expertise on the teaching methodology (teaching strategies and diagnostic capabilities); and
• Expertise on how to individualise the teaching dialogue usually guided by a student model for each learner, which stores information about the student’s level of knowledge and skill.

1.1 Context and subject matter of the research

1.1.1 The context: automated intelligent tutoring for computer programming

Learning to program computers is an appropriate domain for research on intelligent tutoring, since computer programming is constrained, yet rich enough to present a real modelling challenge. The domain is well structured, so that it is possible to determine whether an answer is correct or incorrect. A functional model of computer programming - encompassing goals, problem-states and problem-solving operators - can be specified using computer simulations that mirror the reasoning of human subjects. Programming is richer than many mathematical domains in that there is frequently a variety of correct answers for a programming problem. Students often choose the type of algorithm, data structure and implementation to achieve the goals of the programming problem. In an intelligent tutor for teaching programming, the problem-solver must know about the variety of algorithms and the goals that each algorithm achieves. The problem-solving rules must encode the constraints that govern the use of the algorithms, the data structures required for an algorithm, and the
techniques for storing and manipulating different types of data. The functional model should solve the problem as the students solve them, using a psychologically accurate simulation of the knowledge representations and procedures actually used by students [Reiser et al, 1992].

Learners face a number of challenges in the field of programming. Firstly, they must learn the syntax and semantics for the available programming constructs. Secondly, they have to handle the difficulties experienced with natural language usage of programming terms (for example: 'UNTIL', 'WHEN', 'IF'), and thirdly, they must avoid assuming too much intelligence from the computer and expecting that it will figure out what was intended by the programmer. In order to provide helpful guidance and feedback to learners writing computer programs, ITSs will need to understand and be able to explain each of these aspects of the problem-solving process [Reiser et al, 1992].

1.1.2 The specific subject-matter: SQL database language

Despite the simplicity and highly structured nature of the database language SQL, students experience many difficulties learning it. Some errors in student queries may be attributable to the fact that students are required to memorise database schemas, with the result that incorrect solutions may contain incorrect table and/or field names. Other errors may be rooted in students' misconceptions of the elements of SQL and the relational data model in general. Some of the concepts that students find complex are grouping, restricting grouping, join conditions and the difference between aggregate and scalar functions [Mitrovic, 1998]. In addition, students experience difficulties with the use of SQL words such as 'ANY', 'ALL', and 'EXISTS', as well as the correct use of set operations. Their learning problems are compounded by the fact that it is difficult to learn SQL directly by working with a DBMS, as the error messages are limited to syntax only.

1.2 The main goal and sub-goals of this dissertation

This MSc half-dissertation has a dual aim:
1. Firstly, the study aims to develop a comprehensive generic methodology for the evaluation of ITSs.
2. Secondly, it aims to apply this methodology to evaluating an existing ITS, the SQL-Tutor developed by Mitrovic [1998], as a guided-learning environment to help tertiary-level students overcome difficulties experienced with learning the SQL database language.

The sub-goals of the study are to:

- Provide a historical perspective and to study the state of the art in ITSs, in terms of underlying theory and architecture;
- Discuss the theory, architecture and behaviour of the SQL-Tutor;
- Perform an extensive literature study on the evaluation of instructional software, in particular ITSs, in terms of 'what' to evaluate and 'how' to evaluate;
- Devise sound evaluation criteria on behavioural aspects, learning and instructional theories, design principles, and human-computer interface aspects; and
- Propose principles, approaches, methods, and a framework for the evaluation of ITSs.

1.3 Scope and delimitation of the research

Table 1.1 illustrates a number of disciplines/aspects, their specific features, and the respective relationships they bear to intelligent tutoring systems. This table represents the scope and delimitation of the research with respect to the evaluation of ITSs. In addition, this research includes an in-depth study of the history, architecture, and theory of ITSs, as well as the theory, architecture and behaviour of the SQL-Tutor.

The research does not involve the development of an educational tool but rather provides a generic methodology integrating evaluation criteria, principles, approaches, and methods within a framework for evaluation. The application of the evaluation methodology would provide valuable insights about the strengths and shortcomings of an ITS which would help developers to effect improvements to subsequent versions of their software products. Moreover, this methodology would help educators to decide whether this educational technology provides added value for students.
### Table 1.1 Aspects investigated and their relevance to ITSs

<table>
<thead>
<tr>
<th>Discipline/Aspect</th>
<th>Encompassing</th>
<th>Relationship to ITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructional Aspects</td>
<td>Learning and instructional theories; instructional strategies; learning achievement; learning affect</td>
<td>Relates to instructional evaluation criteria of ITS</td>
</tr>
<tr>
<td>Behavioural Aspects:</td>
<td>Domain Model: Domain knowledge and expertise</td>
<td>Internal evaluation criteria for this component of ITS</td>
</tr>
<tr>
<td></td>
<td>Tutoring Model: Teaching strategies; teaching goals; teaching knowledge and didactics</td>
<td>Internal evaluation criteria for this component of ITS</td>
</tr>
<tr>
<td></td>
<td>Student Model: Student knowledge and diagnosis</td>
<td>Internal evaluation criteria for this component of ITS</td>
</tr>
<tr>
<td></td>
<td>System control: System feedback; treatment of detected errors; system response and intervention; adapting to students needs</td>
<td>Internal evaluation criteria for overall system control of ITS</td>
</tr>
<tr>
<td>Design Aspects</td>
<td>Design principles for ITS construction</td>
<td>Internal evaluation criteria of ITS</td>
</tr>
<tr>
<td>Usability and user interface factors</td>
<td>Human-computer interaction</td>
<td>Internal evaluation criteria relating to control, interaction, individualization and ease of use</td>
</tr>
<tr>
<td>Principles for evaluation</td>
<td>Goals of ITS and evaluation study, design, instantiation of design, logistical preparations, pilot testing and planning of data analysis</td>
<td>Principles for ITS evaluations</td>
</tr>
<tr>
<td>Approaches to evaluation</td>
<td>Internal and external evaluation;</td>
<td>Approaches for evaluation of ITSs</td>
</tr>
<tr>
<td>Methods for evaluation</td>
<td>Criterion-based evaluation and experimental research</td>
<td>Methods for internal &amp; external evaluation of ITSs</td>
</tr>
<tr>
<td>Framework for evaluation</td>
<td>Descriptive modelling, evaluation design, and evaluation implementation</td>
<td>Framework for evaluation of ITSs</td>
</tr>
</tbody>
</table>
1.4 Research design

The research methods to be employed include literature searches, internal evaluation using criterion-based techniques and external evaluation using experimental research. The modus operandi for the experiment is to use three groups of students from the target population composed of third-year students enrolled for the IS302 Database Management module at the M L Sultan Technikon in Durban. All three groups would receive the same classroom instruction and instructional materials. Group 1 subjects would practice SQL problem solving on paper, while group 2 subjects would solve practice-problems using a DBMS, and group 3 subjects would solve problems by interacting with the SQL-Tutor. Pre-tests and Post-tests would be administered to all three groups involved in the experiment and averages will be computed and compared. Random sampling would be used to select ten students from the group practising with the SQL-Tutor. The evaluation instruments to be used to generate empirical data for analysis, are tests, questionnaires, observation and in-depth interviews.

1.5 Importance of the research

CAI and CBT systems are in common use as educational tools in schools, tertiary institutions, and in the workplace. There is extensive research into the evaluation and practice of CAI systems, but the same is not true of ITSs, which were developed primarily for research purposes. The problem that gave rise to this study is the limited amount of theory and literature on the evaluation of ITSs within authentic instructional and learning settings.

A conceptual framework for evaluation can serve as a tool for sustaining research and development, by providing suggestions for the overall improvement of the architecture and the behaviour of ITSs. Evaluation will assume even greater significance in the future when ITSs become commonplace in educational institutions, where questions pertaining to the usefulness of ITSs, namely their ability to foster learning, will have to be answered. Evaluations influence what and how students learn. Evaluations help to determine the extent to which a particular system meets certain requirements and reveals its research value, such as its strengths and shortcomings. Evaluations ultimately influence the choice as to whether or not one should use a particular intelligent tutoring system [Siemer & Angelides, 1998].
In view of the specific need for evaluating ITSs within real instructional and learning settings, this study undertakes to develop a generic evaluation methodology integrating all the aspects outlined in table 1.1, and to apply this methodology to the evaluation of an ITS for the database query language SQL.

1.6 Structure of the dissertation

Figure 1.1 illustrates the structure of the dissertation by depicting the chapters, their internal relationships, and their relevance to intelligent tutoring systems (ITSs) and the SQL-Tutor.
An overview of each of the chapters depicted in figure 1.1 follows:

Chapter One provides background information on ITSs, and briefly discusses the context of automated intelligent tutoring for computer programming, as well as the difficulties experienced in learning the specific subject matter of the SQL database language. The study's goals, scope, and research design are outlined, together with a brief statement of the importance of this research undertaking. Chapters Two through to Five comprise extensive literature studies into the various topics and the intelligent tutoring system investigated to support this research. Chapter Two deals with the history, theory and architecture of ITSs. Chapter Three focuses on the theory, architecture and behaviour of the selected ITS, namely the SQL-Tutor. Chapter Four addresses the aspect of required characteristics and evaluation criteria for CAI and ITS technologies, and proposes a set of generic criteria for the evaluation of ITSs. Chapter Five proposes a generic methodology integrating principles, approaches, methods and a framework for the evaluation of ITSs, followed up in Chapter Six by the practical application of the methodology to the internal and external evaluation of the SQL-Tutor. The methodology is applied within a three-phase framework characterised by a set of predefined activities and specific outputs for each phase. The proposed criteria for evaluating ITSs, identified in Chapter Four, are integrated with principles and disparate techniques (planned experiment and criterion-based techniques) for evaluating the SQL-Tutor in Chapter Six. This penultimate chapter also provides detailed analyses of the results of student feedback to evaluation instruments used, and discloses the findings of the experiment. Chapter Seven concludes the dissertation by reviewing the main and subsidiary contributions of the study, the generic methodology for evaluating ITSs, and the results and recommendations emanating from the evaluation of the SQL-Tutor.
CHAPTER TWO

Intelligent Tutoring Systems: History, Theory and Architecture

2.1 Introduction

An intelligent tutoring system (ITS), like any other form of educational technology, has its own characteristic history, is based on educational and computational theories and comprises a number of architectural components. This chapter provides a close examination of these issues, namely the historical perspective of intelligent tutoring systems, the underlying theories of ITS development, and several architectures proposed for ITS development in various application domains.

2.2 History and development of ICAI/ITS

ICAIIITS systems are based on cognitive psychology as an underlying theory of learning, and are implemented using methodologies and techniques from the field of AI. These systems adopt a mixed-initiative teaching dialogue, tailored to the needs of the individual student [Garito, 1991].

One of the main objectives of AI was the design and development of computer systems that could perform intelligent activities. AI techniques particularly relevant to ITSs are those dealing with efficient representation, storage, and retrieval of knowledge (i.e., an aggregation of facts and skills - correct and incorrect/buggy versions) as well as effective communication of that knowledge. Other pertinent AI techniques involve inductive and deductive reasoning processes and the access of a system to its own database to derive particular answers to student queries.

Researchers in the field of cognitive psychology contributed by examining issues relating to the representation and organisation of various types of knowledge within the human memory and providing detailed structural specifications for implementation in intelligent computer
programs. Cognitive psychology also addressed the nature of errors, a key feature in the design of intelligent tutoring systems [Shute & Psotka, 1996].

An inherent feature of both CAI and ICAI/ITS systems is branching, which demonstrates the many ways in which knowledge may be related, and provides the mechanism for a student to progress through the curriculum via a number of equally appropriate alternative paths. The complexity of the branching differentiates CAI from ICAI/ITSs. Branching in ICAI/ITSs is complex and algorithmic, and not enumerated or pre-defined as in CAI. Flexible interaction and a greater potential for communication accompany this increase in complexity. Another aspect of computer intelligence deals with the ability of the system to identify and remediate errors (bugs) in a student’s knowledge structure or performance. The issue of computer intelligence and the fundamental differences between the older CAI and ICAI/ITS systems are discussed at great length in Chapter Four, sections 4.3.1 and 4.3.2.

Intelligent tutoring systems are designed according to a number of paradigms, namely problem-solving monitors, coaches, diagnostic tutors, microworlds, laboratory instructors, consultants, and articulate expert systems. Each paradigm deals with a set of cognitive science issues and ignores others. A paradigm does not cover all the concerns of ICAI, nor does any existing ICAI program span more than one paradigm [Sleeman & Brown, 1982; Kearsley, 1987].

The following two sub-sections examine important milestones and issues pertaining to the development of ITSs, as well as the shift in development focus spanning their 25+ years of existence.

**2.2.1 Evolution in the development of ITSs**

Shute & Psotka [1996] tabulate and discuss several important issues related to ITS development since its inception more than 20 years ago. The table shows how ITSs have evolved from the 1970’s through to the 1990’s with a current focus on learner control, collaborative learning, constructivism and virtual reality issues.
Table 2.1 Important issues related to ITS development

<table>
<thead>
<tr>
<th>1970's</th>
<th>1980's</th>
<th>1990's</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem generation</td>
<td>Model-tracing technique</td>
<td>Degree of learner control in systems</td>
</tr>
<tr>
<td>Simple student modelling</td>
<td>More buggy-based systems</td>
<td>Individual vs. collaborative learning</td>
</tr>
<tr>
<td>Knowledge representation</td>
<td>Case-based reasoning systems</td>
<td>Situated learning vs. information processing</td>
</tr>
<tr>
<td>Socratic tutoring</td>
<td>Discovery worlds</td>
<td>Virtual reality</td>
</tr>
<tr>
<td>Skills &amp; strategic knowledge</td>
<td>Progression of mental models of subject matter</td>
<td></td>
</tr>
<tr>
<td>Reactive learning environments</td>
<td>Simulation-based instruction</td>
<td></td>
</tr>
<tr>
<td>Buggy library</td>
<td>Natural language processing</td>
<td></td>
</tr>
<tr>
<td>Expert systems &amp; tutors</td>
<td>Authoring systems</td>
<td></td>
</tr>
<tr>
<td>Overlay models / genetic graph</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Shute & Psotka [1996:23]

Shute & Psotka’s structure will be used as a basis for discussing each aspect.

2.2.1.1 ITS development issues in the 1970’s

Real-time problem generation

The earliest systems incorporating ITS features/elements were ones that generated problems and learning tasks. This development, where a series of systems were implemented that generated problems in arithmetic and vocabulary-recall, was a departure from the canned problems stored in CAI databases during the mid-60’s [Sleeman & Brown, 1982]. These generative systems could answer some of the learners’ questions, as well as incorporate some sort of measure of difficulty of the task [Yazdani, 1987]. The next major advance was the design of systems to provide drill and practice in arithmetic, and to select problems at a level of difficulty appropriate to the student’s overall performance. The latter systems were termed ‘adaptive’ and their sophistication was attributed to task selection algorithms. These systems incorporated models of the student that were based largely on parametric summaries of behaviour rather than explicit representation of student knowledge. In effect, this meant that
the student model built by the system was based on mere statistical data concerning the number of correct answers, the number of mistakes, and possibly on the response times. These systems provided the foundation for student modelling in ITSs. The inherent simplicity of the task domain made these systems functionally effective for real instructional uses. The Stanford and Leeds systems were prime examples of such adaptive instructional systems [Sleeman & Brown, 1982].

There was no sharp boundary between adaptive instruction systems and intelligent tutoring systems, since one of the original goals of intelligent tutoring systems was, in fact, to extend the domain of applicability, power, and accuracy of adaptive systems [Sleeman & Brown, 1982].

Simple student modelling

A student model is a component (data structure) of an ITS that represents the student's current state of knowledge (mastery) of the domain, that is, a detailed model of student cognition [Van Lehn, 1988]. The instructional program developed to teach the BASIC programming language, employed simple student modelling techniques [Barr, Beard & Atkinson, 1976, cited in Shute & Psotka, 1996]. Problems were selected on the basis of a student's past performance and on which skills the system believed should be taught next.

Knowledge representation

There are various methods of representing knowledge, but the two most generally used are production rules and semantic networks. Production rules are more appropriate for representing procedural tasks and semantic networks are more appropriate for representing conceptual knowledge [Park & Seidel, 1989]. The first true ITS, the SCHOLAR program [Carbonell, 1970] used semantic nets to represent both domain knowledge (South American geography) and the student model. Nodes in the network had tags to mark concepts known by the student. This type of knowledge representation allowed for mixed-initiative dialogues, where the student and the computer could ask questions of each other.
Socratic tutoring

Socratic dialogues (mixed-initiative dialogues) are appropriate for the tutoring of declarative knowledge (knowledge about basic principles and facts of a domain). Socratic dialogues give control to the tutor, who asks a series of questions, to which the learner responds. In a true Socratic dialogue, both instruction and knowledge assessment are handled by asking questions [Anderson, 1988]. A set of tutorial rules for Socratic tutoring was incorporated in the intelligent tutoring system WHY [Steven & Collins, 1977, cited in Shute & Psotka, 1996], where a structure called a 'script hierarchy' was used to store domain knowledge.

Skills and strategic knowledge

A group of researchers at Xerox PARC in the mid-to-late 1970's worked on getting students to think for themselves instead of being passive recipients of information. The intelligent tutoring system WEST [Burton & Brown, 1976, cited in Shute & Psotka, 1996] was developed towards this end, by assisting students learn/practice skills needed in the manipulation of arithmetic expressions. This system offers advice to the player of a computerised arithmetic game. It characterises the game in terms of issues or strategies that may be of use to a player on certain moves, and it tutors by reminding the learners of these issues at appropriate occasions, throughout the game [Halff, 1988].

Reactive learning environments

Reactive learning environments allow the system to respond to the learner's actions, in a number of ways that extend understanding and change fixed beliefs, by providing examples that challenge the learner's current hypothesis [Shute & Psotka, 1996]. An example of this type of environment is SOPHIE (Sophisticated Instructional Environment), where learners are assisted in developing troubleshooting skills [Brown, Burton & De Kleer, 1982]. SOPHIE 1's intelligence resides in a collection of procedural specialists, that select, set up and run experiments on a general-purpose circuit simulator. This permits SOPHIE 1 to evaluate a learner's hypothesis, in relate to his/her measurements and answer any question presented in the context of electronic troubleshooting.
Buggy library

A bug library technique is a student-expert difference model that represents misconceptions. It augments an expert model with a list of bugs [Polson & Richardson, 1988]. Diagnostic tutors 'debug' a student's work by incorporating a 'bug catalogue', that identifies the misconceptions that learners make when solving problems. Although diagnostic tutors are suitable for any type of problem-solving situation, they are easiest to implement for problems with closed solutions [Kearsley, 1987]. BUGGY [Brown & Burton, 1978, cited in Shute & Psotka, 1996] is an example of a system that used the 'buggy' library approach for diagnosing student errors and served as a framework for modelling misconceptions underlying procedural errors in arithmetic.

Expert systems and tutors

An expert system is a computer program that uses knowledge base and inference procedures to act as an expert in a specific domain. Articulate expert systems are expert systems that can explain their decisions. These expert systems can be used as job aids and provide practice in problem-solving and decision-making skills [Kearsley, 1987]. MYCIN [Shortliffe, 1976, cited in Shute & Psotka, 1996; Clancey, 1987] was a classic rule-based medical expert system for diagnosing bacterial infections. GUIDON [Clancey, 1979, cited in Shute & Psotka, 1996; Clancey, 1982] was built to interface with MYCIN for teaching purposes.

Overlay models / genetic graphs

An overlay model may be defined as a novice-expert difference model representing missing conceptions. The ITS, WUSOR incorporated an overlay model representing the expertise as rules and the student's knowledge state as a subset of the expert's knowledge. [Stansfield, Carr, & Goldstein, 1976, cited in Shute & Psotka, 1996]. This system was designed as an online-coach for the electronic game WUMPUS [Yob, 1975, cited in Shute & Psotka, 1996]. The objective of the game was to locate and destroy the WUMPUS (the beast) without being entrapped by the dangers that prowl in the maze of caves surrounding the hidden lair [Polson & Richardson, 1988].
WUSOR (III) [Goldstein, 1979, cited in Shute & Psotka, 1996] used a genetic graph combining overlay modelling (rule-based representation), with a learner-oriented set of links between curricular elements. The term ‘genetic’ is a reference to the evolutionary nature of knowledge, while the word ‘graph’ is a depiction of relationships between parts of knowledge, as links in a network.

2.2.1.2 ITS development issues in the 1980's

The 1980’s were a period of enormous growth and momentum in the field of ITSs. Shute & Psotka [1996] point out four main problems with ITSs at the time:

1. Instructional feedback did not contain the degree of detail needed by a learner;
2. Systems forced learners into their own conceptual framework, rather than adapting to the conceptual framework of the particular learner;
3. Tutoring strategies employed by the systems lacked a theoretical cognitive foundation; and
4. User interaction and exploration were too restricted.

These criticisms were addressed during the 1980’s, as attempts were made to solve the problems.

Model-tracing technique

Model tracing is a diagnostic technique used to construct a student model. It uses the student’s surface behaviour to infer the sequences of rules fired in a rule-based model of performance; that is, the student’s actions are traced as a path through the rule base [Polson & Richardson, 1988].

Anderson, Boyle & Reiser [1985] describe a model-tracing approach to tutoring based on the use of production systems to model student behaviour, which are employed in the LISP Tutor [Farrell, Anderson & Reiser, 1984] and the Geometry Tutor [Anderson, Boyle & Yost, 1985]. Model tracing allows cognitive theories to be validated and low-level personalised remediation to be delivered. The approach specifies/encodes hundreds of production rules that model curricular ‘chunks’ of cognitive skill. A student’s acquisition of these chunks is
monitored (i.e., the student model is traced), and deviation from the optimal path is immediately remediated.

Model tracing addresses criticism 1, by providing detailed, specific feedback. It falsifies criticism 3, since it is based on the advanced computer tutoring (ACT) theory. It does not, however, overcome the restricted environment criticism, as it does not afford the student the freedom to learn from his/her mistakes. Another shortcoming of this approach is that while it works well for modelling procedural skill acquisition, it is not well suited to other domains such as economics, history, creative writing, etc.

**More buggy-based systems**

Shute & Psotka [1996] describe the increase of tutors based on the 'buggy library approach'. Such systems succeed in providing specific feedback to the learner about the nature of the error(s) committed, but their response is dependent on the program being able to match the learner's error against a stored bug. Since the systems only recognise stored bugs and ignore novel bugs, it is not possible to update the buggy library or adapt to the learner's current conceptualisation (i.e., falling foul of criticism 2 - non-adaptability). This approach is theoretically founded on the notion of cognitive errors in specific procedures, impasse learning and repair theory, thereby overcoming criticism 3 (lack of a theoretical foundation). These systems do not constrain the learner as much as the model-tracing approach (addressing criticism 4 - restrictive environment). A good example of a system based on the buggy approach is PROUST [Johnson & Soloway, 1987], which is designed to diagnose non-syntactic student errors in Pascal programs.

**Case-based reasoning systems**

Another category of intelligent tutoring systems developed in the 1980's employed case-based reasoning. Proponents of this approach believed that the goal of ITSs should be to teach cases, and to have an indexing system to facilitate retrieval of cases. Given this goal, it is the learner, and not the program that performs the indexing. This, in turn, allows the learner greater freedom, providing a more adaptive learning environment (addressing criticisms 4 and 2 respectively). Case-based reasoning (CBR) systems work well in
structured as well as in poorly structured domains, but they do not provide detailed feedback to students (failing to address criticism 1). Furthermore, CBR systems are suited to domains where there are too many rules or too many ways, in which rules can be applied, as is the case in programming and game playing. CBR suggests approximate answers to complex problems, thereby restricting the number of rule combinations that could be applied. While CBR systems serve as a model of cognition and learning, these systems cannot claim to have a solid theoretical foundation (failing to address criticism 3). A major shortcoming of this approach is the problem of anticipating and representing a sufficient number of cases to be catalogued [Shute & Psotka, 1996].

**Discovery worlds**

In the late 1970's and early 1980's, research was carried out on supportive learning environments to facilitate *learning-by-doing*. Systems evolving from this research attempted to combine problem-solving experience and the motivation of *discovery-learning* with effective guidance of tutorial interactions. In order to overcome the potential for conflict posed by such a combination of objectives, these tutoring systems needed its own problem-solving expertise, diagnostic or student modelling capabilities, explanatory capabilities and reasoning capabilities. By extending open-ended, problem-solving environments with the above kind of tutorial intelligence, it was believed that a student's conceptual floundering and misconceptions could be transformed into profound and efficient learning experiences [Sleeman & Brown, 1982].

One of the main strengths of these systems was their ability to adapt to a range of different learners (addressing criticism 2). Students were free to explore and act within the microworld in any manner they desired with the consequences of their actions being discernible, thus countering criticism 4 (restrictive environment). This movement was based on the theoretical premise that *discovery-learning* can dramatically change the student's perception of the relation between themselves and the knowledge or skills to be acquired, thus addressing criticism 3 (a theoretical foundation). A shortcoming of these systems is that they do not suit all learners, in the sense that not all learners demonstrate the required inquiry behaviours to enable them to achieve success in these environments [Shute, 1990].
Progression of mental models of the subject matter

White et al. [cited in Shute & Psotka, 1996] developed QUEST (Qualitative Understanding of Electrical System Troubleshooting) and 'Thinker Tools', from AI research on mental models and qualitative reasoning. These systems initially point out to learners the errors and inconsistencies in their current beliefs. Learners are next guided through a series of micro-worlds, each more complex than the preceding one, with the objective of evolving more precise models of the subject matter. Finally learners are given an opportunity to formalise their mental models by evaluating a set of laws describing phenomena in the micro-world, and applying a selected law to see how well it predicts real-world events. This approach to learning has a theoretical foundation and systems using such an approach are able to adapt to a wide range of learner misconceptions (addressing criticism 2 - non-adaptability).

These systems provide a type of learning that lies midway between true discovery environments and model-tracing environments (relating to criticism 4). A programmed series of mental models produces progressively higher-levels of feedback (addressing criticism 1).

Simulation-based instruction

Simulation is a process of determining a likely course of future behaviour of a physical system, starting with a structural description of some device or system and some initial conditions. Graphical simulations have been increasingly used in intelligent tutoring systems wherever real objects are involved in learning or training tasks. Simulations may be used to display combined behaviour or may be decomposed into constituents that mimic novice or expert mental models.

The power of graphical simulations was demonstrated with the development of Steamer [Hollan, Hutchins & Weitzman, 1987] and the use of personal LISP machines. The purpose of Steamer was to provide simulation-based instruction on the workings of a basic steam propulsion system. The system was interactive in that learners could manipulate controls and see their effect. Steamer included a graphics interface that allowed learners to view the workings of the steam system in an animated form [Rickel, 1989]. Soon after, the graphical
power became available on personal computers that could be used in industrial and educational settings.

Feedback in these systems could achieve various levels of detail (criticism 1, feedback specificity). As simulations are very reactive to learner actions, they address the second criticism of non-adaptability. Simulations, like discovery worlds, provide freedom to explore and manipulate simulated objects and devices (countering criticism 4). However, these systems lack a solid theoretical basis.

**Natural language processing (NLP)**

Few intelligent tutoring systems have allowed true natural language input. SCHOLAR incorporated rich natural language facilities that allowed it to understand most of the learners' questions and answers. SOPHIE used a technique called semantic grammar that searched the input for understandable fragments. Most other systems have bypassed natural language processing by using graphical or menu-based input [Rickel, 1989]. NLP technologies have been used in several ITSs for discourse networks, and especially for language instruction. The development of powerful Prolog compilers and languages on PCs have led to the implementation of instructional grammars that can handle discourse in English or foreign languages, and provide multimedia instruction in advanced language concepts and grammar, as well as simple vocabulary and verb declension [Shute & Psotka, 1996].

**Authoring systems**

Authoring refers to the development of courseware. An authoring system is a domain-independent component of an ITS that allows the developer to enter specific domain knowledge into the tutor's knowledge base. The goal of authoring toolkits is to provide a software toolkit so that a relative computer novice could design intelligent tutoring systems. During the 1980's powerful authoring systems were developed that incorporated graphics and graphical user interfaces [Shute & Psotka, 1996].
2.2.1.3 ITS development issues in the 1990's

Shute & Psotka [1996] summarise four topics debated in the ITS field in the 1990's as follows:

1. How much learner control should be allowed in systems?
2. Should learners interact with ITS individually or collaboratively?
3. Is learning situated, unique, and ongoing, or symbolic and following an information processing model?
4. Does virtual reality (VR) uniquely contribute to learning beyond CAI, ITS, or multimedia?

Degree of learner control in systems

Shute & Psotka [1996] present two different perspectives to the issue of the optimal learning environment. One approach is to develop a computerised environment containing a number of tools and allowing students the freedom to explore and learn independently. Proponents of the opposing position believe that it is more effective to develop straightforward learning environments with no digressions. The issue, as to which is the better learning environment for what type(s) of students, a classic aptitude-treatment interaction question, complicates this debate.

There is also the view that it might be more efficient to learn a new cognitive skill initially by direct instruction, then later by greater exploration. The issue of learner control is also related to other variables, such as the subject matter being instructed, the desired knowledge or skill outcome, incoming aptitudes, and so on.

‘Coached practice environments’ (i.e. Sherlock I and II) provide control during learning by combining apprenticeship training with intelligent instructional systems [Lajoie & Lesgold, 1989; Lesgold et al, 1992, cited in Shute & Psotka, 1996]. These systems provide greater learner initiative because the apprentice learns by doing, knowledge is anchored in experience and the coach can provide information within the context of application.
In summary, an optimal ITS learning environment should offer a reasonable degree of student control, be flexible in response to learners' changing needs, and where possible allow them some input in the design of the environment.

**Individual versus collaborative learning**

ITSs have been traditionally designed as single-learner enterprises. Collaborative learning represents an alternative approach to individualised instruction. Collaboration is defined as a process by which 'individuals negotiate and share meanings relevant to the problem-solving task at hand' [Teasley & Roshelle, 1993:229, cited in Shute & Psotka, 1996].

Collaborative learning environments using computers may be implemented in two ways: (1) a small group of learners interact with a single intelligent computer system, or (2) the computer system itself acts as a 'partner' in the collaboration.

The system must provide the following functionality: (a) introduce and accept knowledge into a joint problem-solving space, (b) monitor ongoing activities for evidence of divergences in meaning, and (c) repair divergences that impede the progress of the collaboration. The student model built is based on a joint, rather than single, problem-solving space.

More research and controlled studies are needed in order to test the efficacy of collaborative versus individualised instruction [Shute & Psotka, 1996].

**Situated learning controversy**

The position one takes with respect to situated cognition or the traditional information-processing model has implications for ITS design. These two positions present different views on how learning or knowledge acquisition occurs. The situated cognition model views learning as a process of creating representations, inventing languages, and formulating models for the first time. Learning is perceived as an ongoing activity, taking place with every thought, perception, and action and is situated in each unique circumstance. This model calls for an instructional system with explicit tools that can support and extend the learners' discovery processes. The information-processing model views learning as
progressing from declarative knowledge to procedural skills to automatic skills. This model depends on what one knows and can transfer to new situations, as well as cognitive processes, such as working-memory capacity and information processing speed. Systems based on the latter models must be able to analyse the initial state of knowledge and skill, describe the desired state of knowledge and learning outcome, and present material and problems that will move the student from the initial to desired state. This system is based on a well-defined curriculum that has been arranged so as to promote knowledge/skill acquisition [Shute & Psotka, 1996].

Virtual reality and learning

The virtual reality (VR) technology is a combined reference to hardware, software and interface technologies that allow users to experience certain aspects of a simulated 3-dimensional environment. VR has the potential to change relationships between learning and experience, highlighting the role of perception (in particular visual), in learning. Experience involves both social and perceptual aspects, and VR epitomises the notion of experiential learning. Virtual reality can play an important role in the construction of micro-worlds for physics and other science subjects [Shute & Psotka, 1996].

2.2.2 The shifting development focus of ITSs

Supplementing the work of Shute & Psotka [1996], Seidel & Park [1994] offer a historical perspective based on the slowly shifting development focus of ITSs. They perceive this shift as taking place over three stages:

2.2.2.1 Stage 1: early 1970's – middle 1980's

During this initial period AI researchers were exploring the use of knowledge representation methods (e.g. semantic networks, production rules, etc.) to generate CBI lesson contents without incorporating pre-programmed algorithms. Techniques for implementing natural language dialogues were used in an attempt to develop flexible interaction between the computer and the student. Techniques for making inferences were applied in order to diagnose students’ misconceptions and select/administer the best instructional approach from the data available.
The systems developed were evaluated in terms of successful functional running of the system and not from the viewpoint of instructional effectiveness. There was little or no evidence that systems were developed on the basis of planned design specifications incorporating a pedagogical approach and tutorial strategies. To facilitate development, special computers made for AI (e.g. Symbolics and Xerox Lisp Machine) and special programming languages appropriate for symbolic processes (e.g. LISP and PROLOG) were used.

2.2.2.2 Stage 2: mid and late 1980's

In response to growing criticism that instructional issues were largely being ignored in ITSs, there was a shift from demonstration of the technical functions to the practical application in education and training. Hence, the pedagogical approach was taken into consideration and some systems were developed on the basis of specific instructional strategies (e.g. QUEST, mentioned in section 2.2.1.2, used a qualitative modelling evolution approach). Various mathematics tutors were used in schools, and programming language tutors in universities, as well as certain task-oriented tutors in industry. ITS systems were beginning to be evaluated from an instructional effectiveness perspective. This shift in development focus was accompanied by a growing expertise in instructional psychology. Development teams in ITS were being characterised by expertise in multidisciplinary areas, namely AI, instructional psychology/design and targeted domains. ITSs were being developed on microcomputers and general-purpose languages like C began to be used for ITSs, as much as AI languages of LISP and Prolog.

2.2.2.3 Stage 3: early 1990's and subsequent

Several important trends were observed during this stage:

- **Effects of specific instructional strategies in ITSs** - Since evaluation studies for assessing the overall instructional value of ITSs produced inconsistent results, there was a move to examine what variables make a difference and what kinds of advantages ITS has for manipulating the variables. For example Lajoie and Lesgold [1989] identified nine candidate instructional strategies that might have contributed to the success of SHERLOCK,
including variables that range from functionally relevant embedded learning to metacognitive skills and heuristics governing the domain knowledge.

- *Flexibility of instructional strategy approach* - Research on ITSs focused on applying intelligent features to the manipulation of specific instructional strategies that were difficult to implement in CAI. For example, Reiser *et al.* [1992] used the GIL tutor to investigate the instructional advantages of dynamically constructed graphical representations in guiding the student's reasoning process.

- *Modelling the human tutor* - An alternative approach to the development of ITSs was to model a human expert tutor, by storing a representation of his/her domain knowledge and reasoning process, communication process, the diagnosis process reflecting student's understanding and cognition, and the tutoring process. Intelligent tutoring systems such as the LISP tutor [Farrell, Anderson & Reiser, 1984] and Geometry Tutor [Anderson, Boyle & Yost, 1985] were built on Anderson's ACT theory that modelled the cognitive process of human learning and cognitive skills.

- *Simulating human learning and cognition* - Attempts have been made to develop learning theories and instructional strategies by simulating human learning processes. For example, Ohlsson [1991] simulated different learning processes on the same topic using the intelligent features of ITS with the intention of developing learning and instructional design theories. A computer simulation model was used to ascertain the computational effort expended in different forms of instruction and the one with the least computational effort was predicted to be the most favoured.

- *Creating multimedia environments* - The rapid development of computer technology has led to many current ITS and CBI systems being able to incorporate presentation of multimedia types of instruction, including text, audio, high-resolution graphics, interactive video and animation of virtual reality.
2.3 Learning and instructional theories of ITSs

With this background on the history and development of ITSs, the discussion moves onto theories underlying the development of ITSs.

2.3.1 Impasse-success-problem-solving-driven-learning (ISPDL) theory

Mandl & Lesgold [1988:iii] stated that 'the only theory available to guide instructional development was behaviour theory, which poorly matched the cognitive goals of education'. This situation changed as Mobus, Pitschke & Schröder [1992] proposed a theoretical framework for problem solving and learning. This theory called ISPDL, which stands for impasse-success-problem-solving-driven learning, is an attempt to integrate the theoretical concepts of impasse-driven learning, success-driven learning and problem-solving phases or action phases. ISPDL has three aspects:

1. The distinction of different problem-solving phases - In the deliberate phase, the problem solver considers many goals and selects one. In the plan phase, a solution plan is developed to obtain the goal. Sub-goals are created and arranged in sequence. The plan is next executed or implemented. Lastly, the problem solver evaluates the result.

2. Impasse-driven acquisition of new knowledge - In response to an impasse, the problem solver applies weak heuristics, like asking questions and seeking help. In so doing, the learner overcomes the impasse and obtains new information. Hence impasses trigger the acquisition of knowledge.

3. Success-driven improvement of existing knowledge – The successful application of existing knowledge serves as an improvement over the mere acquisition of knowledge.

The ISPDL theoretical framework leads to several design principles pertaining to issues of when and how to supply help information to a learner who is engaged in solving given problems:

1. The help system should not interrupt the learner but offer information, since according to the theory, information is only helpful at impasse time. This implies that the tutor should only supply information to the learner on request.

2. The learner must have the opportunity to obtain detailed feedback and information each time that an impasse occurs during problem solving. Since different impasses can occur at
different phases of problem solving, the system must offer support in the problem-solving phases of planning, implementation and evaluation.

3. The learner should be enabled to make use of his/her prior knowledge as much as possible when asking for help.

4. The information provided should be tailored to the actual knowledge-state of the learner. A state model is needed to maintain the actual hypothetical domain knowledge-state of the learner. It is used to analyse solution proposals of the learner and to determine which help information to offer in case of several possibilities.

5. The state model should be embedded in a process model. The latter models the processes of knowledge acquisition and modification, the application of weak heuristics and control processes. The process model serves as a support for the more restricted state model.

6. The learner should be free in the choice and sequencing of his/her interactions with the system.

2.3.2 Advanced computer tutoring (ACT) theory of learning and cognition

Cognitivism plays a crucial role in the development of ITSs, since they apply AI methodologies, as well as psychological and didactic theories, in order to design a teaching-learning relationship. Cognitivism is a theory that defines the competence level achieved by learning, and describes how to promote and facilitate the achievement of such a level in a given knowledge or skill area [Garito, 1991].

Anderson et al [1995] discuss the basic tenets of the ACT theory of cognition and learning as follows:

Procedural-declarative distinction - The theory distinguishes between declarative knowledge and procedural knowledge, assuming that goal-independent declarative knowledge is encoded into the system more or less directly from observation and instruction. Cognitive skill requires converting this knowledge into production rules, which represent the procedural knowledge.

Knowledge compilation - It is assumed that students use various interpretative procedures, such as instruction following and analogy, to generate problem-solving behaviour by relating declarative knowledge to task goals. A learning process called knowledge compilation converts
this interpretative problem solving into production rules. Thus the theory assumes that production rules can only be learned by employing declarative knowledge in the context of a problem-solving activity.

**Strengthening** - It is assumed that both declarative and procedural knowledge acquire strength with practice. Application of weak knowledge can result in slips and errors. Thus even when knowledge has been successfully encoded, further practice produces smoother, more rapid and less error-prone execution.

Richardson [1988] believes that further research is needed in building a meta-theory of expert knowledge that shows how declarative, procedural and causal knowledge relates.

Farrell and Reiser [1987, cited in Anderson *et al.*, 1995] examined the ACT theory and extracted eight instructional principles for the design of tutors:

1. **Represent student competence as a production set** - An accurate cognitive model of the target skill should inform the tutoring system. The cognitive model allows appropriate curriculum objectives to be set and the actions of the student to be interpreted. The production rules define a more abstract and accurate representation of the target skill than the behavioural objectives of typical behaviourist analysis. This approach however, shares with the behaviourist approach the idea of decomposing a skill into components and organising instructions according to the componential analysis. The difference lies in what the components are.

2. **Communicate the goal structure underlying the problem solving** - One of the assumptions of the ACT theory has been that solving a problem involves decomposing that problem into a set of goals and sub-goals. An approach called reification was adopted where interfaces were developed to make explicit the goal structures that were only implicit in the instruction.

3. **Provide instruction in the problem-solving context.** This principle was based on research showing the context-specificity of learning.
4. **Promote an abstract understanding of the problem-solving knowledge** - This principle could be applied by reinforcing correct abstractions in the language of help and error messages.

5. **Minimise working memory load** - This principle was motivated by the fact that learning a new production rule in ACT requires that all relevant information be simultaneously active in memory. However, keeping other information active could interfere with learning the target information. This also implies that one should provide instruction on specific components only where other components of the skill have been mastered. This leads to a curriculum design in which only a few things are taught at a time.

6. **Provide immediate feedback on errors** - Immediate feedback can be beneficial in reducing time spent in error states and making it easier to interpret the student's problem solving.

7. **Adjust the grain size of instruction with learning** - It seems reasonable to design the interface so that one could process the student's problem solving in larger units of analysis.

8. **Facilitate successive approximations of the target skill** - More often than not when students are initially trying to learn a skill, they cannot perform all the steps. The expectation was that with repeated practice the division of labour between student and tutor would change with the student providing more and more of the work.

While the ISPDL theory (section 2.3.1), focuses on problem solving and learning, the ACT theory distinguishes between declarative knowledge and procedural knowledge, and demonstrates how the two fuse to produce learning. Both theories contribute in the form of principles for the design of tutors.

### 2.4 Architecture of intelligent tutoring systems

This section examines a variety of architectures for intelligent tutoring systems, i.e. three-tier, four-tier and novel. The systems covered span the period 1986-1999.
2.4.1 Three-tier architectures for ITSs

2.4.1.1 Ikeda's architecture

Ikeda [1988] views an ITS as comprising three major functional modules:

- **Student model module**, which contains inferences about student's knowledge;
- **Tutoring module**, which designs and monitors adaptive instruction for each student; and
- **Expertise module**, which represents deep understanding of the teaching material.

![General Framework for ITSs](Source: Ikeda et al [1988: 84])

1. **Student model module**

This module is responsible for constructing a student model, which represents the student's knowledge. It is in fact an inductive inference, which induces a model from observed data. The student model should be an executable program described in a knowledge representation language, with the data being the student's answers to the problems.

The student model should be capable of identifying knowledge of students who have a consistent logical belief structure, and of detecting students who have an illogical one. The students with illogical belief structures can be detected by the truth maintenance system. The
A student model is constructed according to the synthesis method, called SMIS (Student Model Inference System). SMIS is extended to include synthesis of SMDL (Student Model Description Language) programs, and supports a truth maintenance mechanism. SMDL is a modelling language that can simulate the problem-solving behaviour of the student. The truth maintenance mechanism in SMIS is referred to as OTMS (Oracle’s TMS). An oracle is defined as a pair comprising a ground atom and a truth attribute that is used for inductive inference, as data to be covered by the model. The problems presented to students and the student's answers to them can be translated into the oracle form.

SMIS repeatedly applies two operations to the model: removal of an incorrect rule and addition of a new one, until the model is able to explain all the given oracles. Incorrect rules, as well as uncovered oracles are identified by SPDS (Student Program Diagnosis System).

2. Tutoring module

The tutoring module is made up of four building blocks, namely bug identification, bug causal analysis, plan generator, and tutoring strategy.

*Bug identification* - The Student Program Diagnosis System (SPDS) is used as a general problem-solver for identifying the bugs.

*Bug causality analysis* - The tutor should reason about where the student's bugs originate, and why they occur. This building block should produce a correspondence between rules in the student model and the expertise knowledge.

*Plan generator* - An ITS conducts tutoring in a bottom-up manner, according to the student model. The bottom-up order is generated by this building block. The plan generated is executed by the other building blocks. For example, the bug currently being investigated demonstrates what a student does not understand, which in turn, invokes the tutoring building block to provide a direct relevant explanation.
Tutoring strategy - Many building blocks should be developed to embody tutoring strategies, which cover all the subject matter.

3. Expertise module

The expertise module consists of only one building block, that is, a Prolog interpreter, which interprets the teaching material described in Prolog.

This architecture implicitly incorporates the knowledge base(s) without identifying them as separate components. Each of the building blocks constituting the modules identified in figure 2.1 is designed as a general problem-solver for the tasks identified. The tutoring module of this architecture is based on the buggy library approach discussed in sections 2.2.1.1 and 2.2.1.2.

2.4.1.2 Derry et al's architecture

Derry, Hawkes & Ziegler [1988] discuss a plan-based opportunistic architecture for intelligent tutoring. A planning architecture uses global models of domain knowledge to structure the student’s learning experience. Opportunistic systems base didactic decisions on operational models of performance, and are activated by observations made while monitoring the student’s learning activity. A plan-based opportunistic architecture combines the advantages of planning and opportunistic architectures. This architecture comprises a:

- **Tutoring model** that makes use of heuristically-guided routines;
- **Expert domain model**, which provides information to guide routines in the tutoring model; and
- **Student knowledge model**, which also informs the routines of the tutoring model.

The **student knowledge model** and the **expert domain model** are not discussed in detail as part of this architecture since they are similar to their counterparts in other architectures. The **tutoring model**, on the other hand, will form the basis for the discussion to follow, since it provides a framework for performing three levels of instructional activity, which are:

- Planning an individual’s route through a curriculum (the agenda);
- Planning lessons (using action schemata); and
- On-line tutorial intervention.
At each of these instructional levels, student performance data can be compiled and made available to other levels.

1. At the global level, a curriculum planner examines the current student model and sets current instructional goals in the student record, with the ultimate aim of moving the student closer to an expert model of knowledge. This is achieved by testing the student on skills in the knowledge model, starting with the lowest prerequisite level. The student's level of ability is classified on each skill by using concepts such as non-master (cannot perform skill), novice master (performs accurately but slowly), and expert (performs rapidly and accurately). If the desired level of expertise is obtained for all the skills on one level in the hierarchy, testing continues at the next horizontal level and the process continues until a level is reached where skill deficiencies are found.

Goals are set for all eligible skills within a horizontal level. A planner should also set all possible instructional goals for moving vertically from one horizontal level to another. Current achievement goals set by the curriculum planner are recorded in the student's permanent record and hold for that student until amended by the plan. Goals are updated after each instructional session. If a lengthy time lag expires between instructional sessions, or if the student does not progress through the curriculum, it may be necessary for the planner to reassess the student and reformulate current achievement goals.

2. The lesson planner takes eligible instructional goals as input and uses this information in the selection of specific lesson goals and the planning of lessons. Lessons are formulated by selecting appropriate problem-solving routines, and by selecting and sequencing problems, and variations on basic routines. The lesson planner also makes modifications to lessons when they do not meet with success.

The set of problems selected is based on knowledge of the target level of skill achievement. Lessons that aim to promote expertise will employ high difficulty problems with great structural variety, while novice problem sets are characterised by less structural variety and low difficulty problems. The routine's available variations range from the more concrete graphic representations to the more numerical and symbolic representations.
The lesson planner advises the intervening monitor, which will actually coach the student as to which routine, and which routine variations are present in the plan. The monitor is also instructed as to the student's level of expertise, so that an appropriate level of intervention frequency can be set (high frequency for novices, lower frequency for experts).

3. Problem-solving performance is 'overseen' by an intervening monitor, which attempts to detect familiar error patterns associated with intervention routines. When each lesson is completed, the student's performance profile, including level of goal achievement, is developed and posted in the student's record. The intervening monitor tracks and judges students' performance so that corrective feedback and interventions can be chosen. The monitor also evaluates the difficulty of the on-going lesson, and notifies the lesson planner when problems occur. Monitoring and intervention continue until the user arrives at a correct answer.

This architecture incorporates expert domain knowledge, student knowledge and a sophisticated tutorial component embodying curriculum planning, lesson planning, and tutorial monitoring and intervention with the ultimate goal of moving the student toward the expert model of knowledge.

2.4.1.3 Siemer & Angelides architecture

The next architecture presented by Siemer & Angelides [1998] also supports a basic three-model structure with the identification of additional processes. This architecture comprises the following three models:

- Domain model, which contains the knowledge about the domain to be taught;
- Student model, which represents the emerging knowledge and skills of the student; and
- Tutoring model, which designs and regulates instructional interactions with the student.

Each of the models identified above incorporates specific processes that manipulate the information needed for the tutoring interaction. For example, the domain model includes domain knowledge processes referred to as the expertise; the student model incorporates student knowledge processes, referred to as diagnostics; and the tutoring model incorporates
tutoring knowledge processes, known as the didactics. Other processes are needed in order to maintain overall system control or management so as to co-ordinate the interaction between the system’s three knowledge and process models. Siemer’s & Angelides’s general intelligent tutoring system architecture is illustrated in figure 2.2.

![Figure 2.2: Siemer’s & Angelides’s general intelligent tutoring system architecture](image)

The three major components of the intelligent tutoring system architecture and the overall system control are described in greater detail below.

1. **The domain model**

The domain model contains knowledge relating to the subject matter. The intelligent tutoring system utilises its domain knowledge to reason with and solve problems, or to answer questions posed by students. To this end, domain knowledge representation should support reasoning mechanisms that are employed by human teachers. Different knowledge representations of the same domain knowledge may be incorporated to support alternative teaching strategies.

Domain knowledge containing learner errors may also be stored in order to correct misconceptions and missing concepts displayed by learners.

Domain expertise includes processes that provide for the content of tutorial interactions.
These processes include the retrieval of teaching material upon system request, the selection of problems/exercises, and the recording of student errors for subsequent diagnosis and correction. In addition, a knowledge level process may manipulate or retrieve existing domain knowledge, in response to a system request for a different presentation of the same knowledge.

2. The tutoring model

The tutoring model provides the knowledge needed to attain teaching goals. It should have:

- Control over the sequence and selection of subject material to be presented to the student;
- Response mechanisms to answer learners’ questions with appropriate answers; and
- Knowledge of when learners need help, in the course of solving a problem or practising a skill, and what type of help to offer.

To achieve this, the tutoring model needs to embrace different teaching strategies. A teaching strategy is a style of material delivery adopted to lead the student through subject material, and to provide assistance when deemed necessary. The subject matter and the instructional objectives of an intelligent tutoring system largely determine its teaching strategies. Different teaching strategies may be applied in different teaching situations, to provide the intelligent tutoring system with more flexibility and better adaptation capabilities.

The tutoring knowledge processes, namely the didactic aspect is responsible for selecting teaching goals, and for determining appropriate teaching strategies for learners on the basis of their student models. Other factors that influence the choice of teaching strategy are the learner’s needs and/or preferences, his/her experience and the domain of discourse.

3. The student model

The student model represents the learner’s emerging knowledge and skills. A more sophisticated student model may contain more learner details than a less sophisticated one. Information such as learning preferences, past learning experiences and advancement may also be relevant in adapting the teaching process. It may also record the learner’s errors and misconceptions.
The student model reflects the relative strengths (the subject knowledge attained) and weaknesses of the learner (e.g. misconceptions and missing concepts). The student knowledge processes, known as diagnostics, analyse the behaviour of the student.

**Overall system control**

The overall system control co-ordinates the three-knowledge models to provide student-centred tutoring. There are a number of tasks and features that require overall system co-ordination. For example, an intelligent tutoring system has to select teaching strategies and presentations for each subject area, in accordance with an individual learner’s needs and preferences stored in the student model. The system has to also account for learner errors made during the course of a tutorial in order to effect the required remedial process. Additional flexibility may also be provided with a student or system initiated help system.

The Siemer & Angelides architecture provides a simple structure that is comprehensive and easily understandable. Its specific contribution is that each of the models identified incorporates a knowledge base as well as processes to manipulate that knowledge base.

In summary, the three-tier architecture represents the basic architecture of ITSs comprising three main components that are commonly referred to as domain model, student model, and tutoring model. The three-tier architecture of ITSs made way for the classical four-tier architecture, which accommodates added functionality.

### 2.4.2 Dede’s four-tier architecture for ITSs

Arising from development work on stand-alone intelligent computer-assisted instruction (ICAI) systems, Dede [1986] believed that ICAI tutors should have four major components, namely a:

- Knowledge base;
- Student model;
- Pedagogical module; and
- User interface.
This architecture, including the user interface component, has become the classical four-tier standard architecture for ITSs. A brief discussion on each of these components follows:

1. **Knowledge base**

A tutor or coach requires a mix of declarative (what), procedural (how), and metacognitive (thinking about what and how) knowledge. Since no single method of encoding content can optimise all of these capabilities, multiple strategies for incorporating information are needed in a single program. Traditionally semantic nets and schema-based coding strategies have been used for descriptive representation, while production systems, rule-based methods and simulation approaches have modelled process information. Tools such as object-oriented languages (e.g. SMALLTALK) and truth maintenance (constraint-based) systems are being used to combine the strengths of both representation types in an overall metacognitive structure.

2. **Student model**

The goals of student modelling are prediction of the learning behaviour of individual users and diagnosis of the causes of errors. These goals require an internal model of the learner, which represents cognitive processes (such as information retrieval, calculation and problem solving), metacognitive strategies (for example, learning from errors) and psychological attributes (developmental level, learning style, and interests). In constructing this learner representation, the intelligent coach or tutor uses four types of evidence:

- Implicit (from learner behaviour in problem-solving situations);
- Explicit (based on dialogue between ICAI device and learner);
- Structural (from intrinsic complexity relations that exist among knowledge representation skills); and
- Background (based on estimates of average learner proficiency).

Psychological capabilities and learning styles must be added to cognitive and metacognitive skills, and the full spectrum of mental configurations based on these three dimensions, over the learner population, should be incorporated.
3. Pedagogical module

Given the subject domain expertise and a model of the learner's present comprehension, the intelligent tutor or coach selects an efficient path through its knowledge representation to generate expert behaviour by the learner. Initial teaching strategies based on a prototype or a learner's previous performance, are modified as the student model evolves. The pedagogical strategies used may include the presentation of increasingly complex concepts or problems, simulation of phenomena, socratic tutoring with correction of pupil misconceptions, and the modelling of expert problem solving via coaching.

The ICAI system must have a discourse-oriented theory of explanation to co-ordinate these teaching strategies. The underlying instructional theory would have rules relating to which pedagogical means are most efficient to accomplish a given end, alternative approaches to dialogue management (adjusting to different learning styles), and domain-dependent teaching heuristics. Some of these skills can be derived from protocol analysis of expert teachers, while others have no human pedagogy counterpart since they are steeped in the practices of computer-based instruction (such as simulation of complex phenomena).

A teaching module may be used to facilitate integration and co-ordination of the functions of the other components. Intelligent tutors and coaches need a means of structuring learning to maximise efficiency and effectiveness.

4. User interface

Communication issues in ICAI parallel research themes in the field of natural language comprehension and generation, which is a major area in artificial intelligence.

A general analysis of discourse indicates the need for three types of information in carrying out a dialogue:
1. Knowledge about patterns of interpretation (to understand a speaker) and action (to generate utterances) within dialogues;
2. Domain knowledge needed for communicating content; and
The state of attention of the learners is also important, which is used to link the user interface to the ICAI system's expertise, student model, and instructional strategy.

The four-tier architecture gave due recognition to the important role of the user interface in intelligent tutoring systems by explicitly incorporating a separate component to fulfil that role.

2.4.3 Novel architectures for ITSs

Novel architectures represent a departure from the traditional three-tier and classical four-tier architectures.

2.4.3.1 SCENT-3 architecture

McCalla & Greer [1988] developed a new-generation architecture called SCENT-3 for a full-scale student advising system. The SCENT-3 architecture was designed to create an intelligent advising system for higher order problem-solving activities with minimal domain dependence. The basic organising principle underlying the architecture is that distributed, communicating processes called 'entities' co-operate to perform various aspects of the program-advising task. These processes include monitoring learner input; analysing learner knowledge, strategies and misconceptions; and designing instruction for the learner. The entities are grouped into six relatively independent components with control mediated by a blackboard control and data structure. Various knowledge bases are associated with individual components: some are shared among components, while others are shared globally through the blackboard.

The six components comprising the SCENT-3 architecture (illustrated in figure 2.3) include: the interface, the instructional planner, the student model manager, the cognitive analyst, the domain knowledge analysts, and the student response analyst.
1. Interface

The interface includes mechanisms for accurate communication with the student.

2. Instructional planner

The instructional planner must decide on what to present to the student and how to present it. The SCENT-3 architecture caters for this by separating the instructional planning process into two phases. The first phase involves planning the content of instruction by the content planner and selector sub-component of the instructional planner. The second phase involves planning the delivery of the content, a task for which the instructional delivery planner is responsible. The instructional goal occupies a central role in the content planner and selector. There may be many instructional goals and they can relate to each other in a variety of ways. The task of the
content planner and selector is to use the information in the 1-goal knowledge base, and that provided by the student knowledge analyst, to formulate a plan of 1-goals that are individually tailored for a given student.

The plan is then executed by selecting an appropriate 1-goal(s) based on the parts that have already been delivered, and the desired level of instruction. After the learner has responded, the student knowledge analyst updates the student model and the instructional planner assesses how changes in the student model affect its plan. Some student problems will only have minor effects, thus allowing the plan to be continued. Others will have major effects, which may require patching the plan or even re-planning the instruction.

The important point about the content planner and selector is that maintenance of an individualised student model and use of dynamic planning techniques allows the content of instruction to be tailored to the needs of the student.

The delivery planner translates an instructional goal(s) into appropriate instructions. This requires it to choose an overall form of instruction (e.g. assign problems, embark on question-answer sessions, provide direct help, give the student a hint), and then to plan out a specific sequence of actions that will achieve the instructional goal(s).

3. Student model manager

The student model manager maintains a student model and student history for each learner.

4. Cognitive concepts knowledge analyst

This component focuses on general cognitive strategies (problem-solving strategies) used by learners. Examples of such cognitive strategies are problem decomposition, means-ends analysis, reasoning by analogy, deductive reasoning, inductive reasoning, etc.
5. **Domain knowledge analysts**

The purpose of domain knowledge analysts is to extract task-specific knowledge (programming concepts and language concepts) and to generalise it into task-independent domain knowledge bases.

6. **Student response analyst**

The goal of the student response analyst is to examine the LISP program a student has created in response to a programming task, and to produce a description of the strategies used and the bugs made in that program.

The main contributions of the SCENT-3 architecture are:

- The integration of many kinds of knowledge into one model;
- A modularised approach allowing incremental development and change;
- The use of flexible control and information-sharing capabilities; and
- The use of instructional planning and the incorporation of cognitive knowledge.

### 2.4.3.2 Multi-agent architecture

A multi-agent architecture called MATHEMA [Costa & Perkusich, 1996] was used as a basis for the design of a computer-based intelligent learning environment, which comprised the following six components:

1. **An external motivator** (representing human external entities that motivate the learner to work in MATHEMA, for example, a teacher);
2. **A human learner**;
3. **A micro-society of artificial tutoring agents (MARTA)**, that may co-operate among themselves to achieve problem solving activities;
4. **A human experts society (HES)**, working as sources of knowledge to MARTA;
5. **An interface agent** between a human learner and MARTA, responsible for communication and which includes a mechanism of selection for tutor agent (supervisor); and
6. **A communication agent** providing interaction between MARTA and HES, and responsible for the communication and maintenance of MARTA.
MATHEMA functions as follows: A **human learner**, motivated by an **external motivator**, commences working with the system, by engaging in a dialogue with the interface agent about the tutoring environment. The **interface agent** helps the learner to choose his/her supervisor. Thereafter a co-operative interactive process between the learner and the supervisor ensues. During this process, situations with varying complexities may be encountered. Demands on the learner may, in some instances, involve more than the supervisor and may involve the **HES**.

The **micro-society of artificial tutoring agents** (MARTA) is made up of tutoring agents that work in a specific, formal and well-structured knowledge domain such as the classical logic domain. This knowledge domain is divided into different micro-domains, each one covering micro-specialities. Each agent is an intelligent entity in some micro-domain, having the necessary knowledge to solve problems in this micro-domain.

Each **tutoring agent** (TA) is capable of:

1. **Problem-solving**: the system should not only present the problem to the learner, but should also be capable of solving or helping to solve it;
2. **Learning**: it is required that an agent should learn about other agents, the human learners, and the HES;
3. **Disagreements-resolution**: this concerns the resolution of disagreements that may arise between the agent knowledge and the learner; and
4. **Communication and co-operation**: with other agents, with the learner, and with the HES.

---

**Figure 2.4: Costa's & Perschik's Architecture of MATHEMA for IT Learning Environment**

Source: Costa & Perkusich[1996:170]
The main idea underlying this architecture is to integrate human learners in a micro-society of artificial agents with the objective of promoting their learning. The distinctive feature of this architecture is the co-operative interaction that exists among agents in the micro-society.

Grandbastien [1999] is of the opinion that agent-based architectures are more flexible and enhance modular developments as well as promote module reusability.

### 2.4.3.3 Self's architecture

Self [1999] revisits the conventional tripartite division of ITSs into the domain, student and tutoring model from a constructivist learning perspective. A three-model architecture for intelligent tutoring systems is proposed, comprising the:

- **Situation model**;
- **Interaction model**; and
- **Affordance model**.

The new tripartite model of the architecture of computer-based learning environments includes the standard ITS architecture as a subset, and is displayed in figure 2.5.

![Figure 2.5 Self's ITS components](Source: Self [1999:15])
1. **Situation model**

An ITS designer assumes that knowledge can be conceptualised as facts, principles, etc., which can be represented symbolically and hierarchically and learnt in an incremental fashion. Much effort is invested in developing complex representations of such knowledge. Constructivists, on the other hand, emphasise that learners construct their own knowledge through interpreting their experiences in interaction contexts. The designer of a constructivist-oriented learning environment does not focus on knowledge representation, but rather on the nature of situations, contexts and interactions. This leads to a consideration of contexts and the dynamics of the learning process. A *situation model* contains descriptions of resources (although it may also contain representations of aspects of the domain of knowledge) which are available in a learning situation as opposed to a pure domain model, which contain descriptions of target knowledge. A model of domain knowledge may thus be perceived as a subset of the broader notion of a situation model.

2. **Interaction model**

The student model of an ITS is determined by analysing the learner’s interactions with reference to the domain of domain knowledge, in order to determine gaps or errors which may serve as a basis for instructional interventions. However, constructivists argue that the learner’s individual constructive process is more important than the product of any learning process. Thus the model of the learner should focus more on the interactive process, extended in time, taking into account the learner’s actions, the contexts in which they occurred, and the learner’s cognitive structures at the time. Such an ‘interaction process model’ enables one to consider the kinds of regularities of interaction sequences, which lead to properties that benefit or hinder learning. The learner’s cognitive structures may form part of the descriptions of the time-extended interactive process, since the significance of interactive events depends on individual cognition. Here again, the notion of an interaction process model is perceived as a superset of an ITS-style student model.
3. Affordance model

Designers of ITSs believe that their systems should be able to determine instructional plans by interpreting student models, with respect to a curriculum structure based on a model of domain knowledge. Constructivists argue that the learning process is too unpredictable to be analysed by pre-specified structures, and that learning sequences emerge from interactions between the learner and the environment, and are influenced by the opportunities that become available. Accordingly, the pedagogical role of the learner is based on some model of the affordances of potential situations. An affordance model could be developed in terms of ‘items of knowledge’ which may be learned through particular events (for example, an event such as the presentation of remedial feedback affords the learning of the item of knowledge being remediated). The affordance model is thus broader than the model of teaching as curriculum-based planning.

Novel architectures for ITSs emerged from the need to build specific functionality for specialised application domains, to embrace important trends in software development, namely modular and incremental development, global sharing of knowledge, etc., and to accommodate a current trend in learning and instructional theories, namely constructivism.

2.5 Conclusion

The historical perspective presented in this chapter demonstrated the evolution of ITSs from the early 1970’s to the mid-1990’s. Salient characteristics of ITSs spanning that period were discussed and a number of research issues relating to each decade were identified and examined. The 1970’s were characterised by the exploration of various research themes and the development of research prototypes. The 1980’s were a period of growth, consolidation, and momentum in the ITS field. The 1990’s, however, were dominated by the emergence of newer issues in terms of learning theories and technological developments, such as learner initiative, collaborative and constructivist learning, and virtual reality. The shift in the development focus of ITSs, spanning the period 1970 to 1990 and beyond was also reviewed. The initial concentration on knowledge representation methods and technical functionality was replaced by the need to address instructional concerns and issues surrounding the assessment of the effects of instructional strategies, as well as modelling of the human tutor, simulating human learning and cognition, and creating multimedia environments.
Certain theories underlying the development of ITSs, namely the ISPDL and ACT theories were discussed. Universal principles were extrapolated from these theories, to support the design of help subsystems and computer-based tutors in general.

The basic architecture of ITSs was established some 25 years ago, with the tripartite division comprising 'what' (domain knowledge), 'who' (student model) and 'how' (tutoring strategy) components. Since then a number of architectures have been proposed in keeping with the scope and application domains of intelligent tutoring systems. ITSs built for more complex domains required more functionality than those developed for static domains. Hence the tripartite architecture was extended to the four-component classical structure (including the user interface component). The architectures examined in this chapter, while not exhaustive, represent a sample of major research efforts in this field. They include among others the classical, plan-based opportunistic, new-generation, multi-agent, and the new tripartite architecture proposed by Self. In particular, Self's architecture may be regarded as a watershed for the development of future intelligent tutoring systems investigating constructivist processes involved in learning.

Architectures of ITSs in general should promote and support a component-based architecture, which is the norm for large-scale software systems. There should be global sharing of certain information and co-operation among components to fulfil learner needs. Components should be developed independently, and assembled for the purposes of instruction. Furthermore, protocols should be developed to ensure optimal interaction among components, and between the tutoring system and the learners. Communication and co-operation between software components at the knowledge level should be promoted in computer-based educational software design.

A close examination of ITSs is relevant to this study in that it forms the cornerstone of the internal evaluation process discussed in Chapters Four and Five, and applied in Chapter Six. The behavioural aspects inherent in the architecture of intelligent tutoring systems are to be used as criteria for the evaluation of ITSs. Hence the architecture of a complete ITS has to be defined in order to provide baseline criteria for the evaluation process. Furthermore the SQL-Tutor discussed in Chapter Three supports a particular type of architecture that is discussed in detail in this chapter.
CHAPTER THREE

Theory, Architecture and Behaviour of the SQL-Tutor

This chapter focuses on three major aspects of the SQL-Tutor - its theory, architecture and behaviour. In addition, the SQL-Tutor is mapped to the theories, principles, and architectures of intelligent tutoring systems that are discussed in Chapters Two and Four.

3.1 Introduction

The SQL-Tutor is an intelligent tutoring system developed by Antonija Mitrovic [Mitrovic, 1998] to assist students in the formulation of database queries in the Structured Query Language (SQL). This tutoring system is designed as a guided discovery-learning environment to help students overcome difficulties experienced in learning SQL. It provides individualisation of instruction by developing a model of each student’s knowledge, and tailoring of instruction to meet the learning needs of the individual student. The SQL-Tutor does not explicitly teach the concepts of SQL, but rather incorporates implicit teaching by way of its rich feedback mechanisms. The scope of the current system is limited to the SELECT statement, but this approach may be applied to other SQL statements and, in so doing, the current system can be extended. The SQL-Tutor is an implementation of a constraint-based modelling (CBM) technique [Greer & McCalla, 1991].

The object-oriented development methodology as outlined by Coad & Yourdon [1990] was used in producing the tutor. The object-oriented analysis (OOA) method comprises 5 major steps, namely identifying objects, identifying structure, defining attributes, defining connections, and defining services. The SQL-Tutor is implemented in the artificial intelligence programming language, LISP.

This chapter discusses the theoretical principles of constraint-based modelling; the application domain covered by the tutor, namely SQL; as well as a description of the tutor’s implementation details, the components underlying its architecture, and its observable behaviour.
3.2 Constraint-based modelling

A constraint-based model represents domain knowledge as a set of constraints on correct solutions. Constraints are used to partition the universe of possible solutions into correct and incorrect ones [Mitrovic & Ohlsson, 1999].

3.2.1 A formal notation for constraints

Ohlsson & Rees [1991] have developed a formal notation for constraints. A state constraint is a reference to a unit of knowledge. Each state constraint consists of an ordered pair \(<Cr,Cs>\) where \(Cr\), the relevance condition, identifies the class of problem states for which the constraint is relevant and \(Cs\), the satisfaction condition, identifies the class of (relevant) states in which the constraint is satisfied. Each member of the ordered pair may be perceived as a set of properties of a problem state. A constraint may thus be explained as follows: ‘if the properties \(Cr\) hold, then the properties \(Cs\) have to hold also’ or else something is amiss [Mitrovic & Ohlsson, 1999:2]. Mitrovic & Ohlsson provide an illustration of state constraints as follows:

*If the code for a LISP function has \(N\) left parentheses, there have to be \(N\) right parentheses as well (or else there is an error).*

In this example, \(Cr\) the relevance criterion is \(there are \(N\) left parentheses\) in a function and \(Cs\), the satisfaction criterion is \(there have to be \(N\) right parentheses\) as well.

A second example considers the following constraint on the addition of fractions:

*If \((x + y)/d\) is given as the answer to \(x/d1 + y/d2\), then it has to be the case that \(d = d1 = d2\) (or else there is an error).*

This constraint is merely stating that you cannot add fractions by adding their numerators unless they have the same denominator. In this example, \(Cr\), the relevance criterion, is the complex predicate \((x + y)/d\) that is given as the answer to \(x/d1 + y/d2\), and \(Cs\), the satisfaction criterion, is the predicate \(d = d1 = d2\).
A state constraint may be implemented as a pair of patterns where each pattern is a list of elementary propositions. In this implementation scheme, each part of a constraint is similar to the condition part of a production rule. State constraints may also be implemented as pairs of LISP predicates. It should be noted that each state constraint is a pair of tests on problem states.

The test for consistency of a problem state with a set of constraints involves the following two steps:

- Test all relevance patterns against a problem state to identify those constraints relevant in that state; and
- Test the satisfaction patterns of the relevance constraints against a problem state.

If the satisfaction pattern of a relevance constraint matches the current problem-state, then that constraint is satisfied. If not, that constraint is violated [Mitrovic & Ohlsson, 1999].

3.2.2 Use of constraint-based modelling for educational applications

Mitrovic & Ohlsson [1999] discuss the use of constraint-based modelling for educational applications and its implications:

1. Intelligent tutors with constraint-based modelling have a set of constraints for a specific target domain, and students are given on-line information about their constraint violations.

2. The state constraint approach overcomes the overspecificity problem (where formal knowledge representations of an ITS require detailed and specific models of students' knowledge) by providing two forms of abstraction:
   - The first is that the state constraint approach permits selective evaluation of problem-solving steps, based on the reasoning that not all problem-solving steps are equally important in diagnosing students' knowledge. If a step does not evoke a constraint, then it is ignored.
   - The second abstraction is achieved by allowing an instructional system to operate with pedagogically equivalent solution paths. This means that the system is required to group student solutions into classes of solutions that require the same instructional response from the tutoring system.

3. It is not imperative for a system to have a complete set of constraints. To the extent that a system is able to track and remedy common student errors, it is useful, even though it may not be able to detect rare errors.
3.2.3 State constraint theory

The concept of state constraints may be traced to a theory that the acquisition of a new cognitive skill consists, in part, of the transfer of knowledge from the evaluative to the generative component. The function of generative knowledge (for example, a rule set) is to produce actions based on the current problem and the function of evaluative knowledge (a set of constraints) is to evaluate action outcomes as appropriate or inappropriate [Ohlsson, 1993, 1996a, 1996b].

The state constraint theory submits that the knowledge base of a constraint-based tutoring system should contain the constraints the student should have mastered as part of his/her repertoire having attained the desired level of mastery of some target skill. The underlying belief is that a tutoring system with constraint-based knowledge will speed up and facilitate the transfer of information from the evaluative to the generative component [Ohlsson, 1991].

3.3 Application domain – learning SQL

SQL is currently the dominant database language in use [Elmasri & Navathe, 1994]. It is a complete database language incorporating data definition, single-table queries, multiple table queries, database updates and data administration. It may be used for interactive or programmed access to databases. Its future use as a database language is secure with the arrival of SQL3, the latest standard, supporting knowledge-based and OO applications and distributed databases.

The SQL database language supports single- and multiple-table queries via the SELECT statement, which is used to access data from a database. The basic form of an SQL expression is simply SELECT FROM WHERE. This form is elaborated below:

- After the word SELECT the columns that are to be displayed are listed;
- After the word FROM the tables that are involved in answering the query are listed; and
- Finally, after the word WHERE the conditions that apply to the data being retrieved, are listed.
Learners are required to learn the syntax and semantics of the language, as well as demonstrate competence in constructing, testing and correcting queries. In particular, they should know, understand, and apply the following aspects, with respect to the SELECT statement [Pratt, 1995]:

- Compound conditions;
- The use of computed columns (involving arithmetic operators);
- The use of the SQL word, LIKE;
- The use of the SQL word, IN;
- Sorting with single and multiple keys in ascending and descending order, using the SQL words, ORDER BY;
- SQL built-in functions;
- Sub-queries;
- Grouping rows having matching values in some column, using the SQL words, GROUP BY;
- The use of the SQL word HAVING to select individual groups;
- The test for nulls;
- Joining of tables;
- The use of the SQL words, IN and EXISTS with sub-queries;
- Nesting of sub-queries;
- The use of aliases to join a table to itself;
- The use of set operations (union, intersection, and difference); and
- The use of the SQL words, ANY and ALL.

For examples of SQL queries, refer to section 3.4.4.2 and Appendix F.

SQL is a relatively simple and well-structured language, yet it presents difficulties for learners. Some of the difficulties experienced by learners may be attributable to high memory load when formulating database queries. Learners must remember database schemas (i.e. table and attribute names), and as a result many queries contain incorrect table and/or attribute names. Other errors in learner solutions may be traced to misconceptions about SQL elements and to the relational data model. Many learners experience problems with grouping and restricted grouping while the join conditions and the difference between aggregate and scalar functions are additional sources of confusion [Mitrovic, 1998].
Learners generally receive group instruction in SQL via the lecture method. They are given problems for practice, and the lecturer revises solutions to these problems on the screen or board. Lectures are normally complemented by laboratory practical sessions where students use an appropriate relational database management system (RDBMS) hands-on to implement SQL queries for a chosen database schema(s). When students work directly with a RDBMS, there is a high probability that learner-generated semantic errors slip through the net, that is, they go unnoticed by the student since the RDBMS produces output which the student does not scrutinise adequately. Furthermore, syntax error messages produced by RDBMSs are generally difficult for learners to comprehend, with the result that they are unable to remedy their errors. The SQL-Tutor was born out of an attempt to overcome the limitations of commercial DBMSs, and to provide useful and informative feedback based on learners’ solution [Mitrovic, 1998].

3.4 The SQL-Tutor: implementation & architecture

The SQL-Tutor is implemented in Allegro Common LISP on SUN workstations and PC compatibles. The system contains definitions of several databases implemented on the relational database management system (RDBMS) used in the laboratory where the system is employed. New databases can be added to the SQL-Tutor by supplying the same SQL files used to create the database in the RDBMS. The SQL-Tutor contains a set of problems for specified databases and the optimal solutions for them. Solutions are incorporated, since the system does not have a domain module. The rationale for this departure from typical intelligent tutoring systems is attributable to the application domain. Database queries are expressed in a natural language. The current state of natural language processing is not sufficiently advanced to handle various subtleties in the form of references and synonyms found in queries. Furthermore, it is difficult to develop a problem-solver that can generate SQL, since the knowledge required to write SQL queries is very fuzzy [Mitrovic, 1998].

The SQL-Tutor has a simple architecture consisting of an interface, a knowledge base, a student modeller and a pedagogical module [Mitrovic & Ohlsson, 1999]. The architecture is illustrated in figure 3.1, and the components are elaborated in the following subsections:
The main window of the SQL-Tutor, as shown in figure 3.4, is partitioned into three sections that are on permanent display. The upper section displays the text of the problem being solved, thus providing an easy and convenient student reference to the elements requested in the query. The middle section provides the underlying goal structure containing the clauses of the SQL SELECT statement. The tutor supports a design principle, which aims 'to communicate the goal structure underlying the problem-solving' [Anderson et al, 1995:179] and is based on one of the assumptions of the ACT theory, namely, that problem solving involves a number of goals and sub-goals. Another design principle on which the tutor is based is to 'minimise working-memory load' [Anderson et al, 1995:180], where students are not required to memorise the exact keywords used and the relative order of the clauses. The lowest section displays the schema of the current database. The name and description of the schema is provided, together with the names, definitions and descriptive information for tables and attributes. The primary key of each table is underlined. Hence, the interface of the SQL-Tutor, makes explicit, the goal structure that is only implicit in the instruction, an approach called
reification. In addition, the SQL-Tutor reduces the working-memory load of students by displaying the text of the problem, the database schema and the goal structure of the query.

The main window also supports a help menu that can provide explanations on the elements of SQL namely queries, clauses, and expressions, as displayed in figure 3.5. Students can also obtain descriptions of the various clauses listed in the goal structure by selecting the appropriate clause. Students have easy access to descriptions of databases, tables or attributes by either directly selecting relevant names or waiting for the tool tip to appear. Table and attributes can be incorporated into the student's solution by means of a point and click interface, thereby reducing the amount of typing required, to produce a solution. Learners are thus freed to focus their energies on high-level definition problems, without also having to commit to active memory the low-level syntax details.
The open menu in the main window provides for the selection of a database and problem to work on, as is illustrated in figure 3.6, where the respective databases, Company, Movies and Registration are displayed.

3.4.2 The knowledge base

Domain knowledge in the SQL-Tutor is represented by constraints, using constraint-based modelling (CBM) techniques. The SQL-Tutor has, at present, 406 constraints in its knowledge base. Mitrovic compiled these constraints from an analysis of the target domain knowledge, as well as from an analysis of correct and incorrect student-solutions noted while lecturing a computer science course to university students. These constraints are general in that their
conditions can be tested against any problem. The expressive notation used for constraint representation gives the system the capability of testing specific features of students' solutions and comparing them to stored correct/ideal solutions. The relevance and satisfaction patterns can be implemented as arbitrary logical formulas containing atomic predicates. Some conditions are patterns that match parts of the student's solution or the ideal solution, while others are LISP predicates. Each constraint has, in addition to relevance and satisfaction conditions, an associated number, natural language description, and name of the clause to which the constraint applies.

Mitrovic & Ohlsson [1999] distinguish between two types of constraints. The first type represents syntactic properties of queries, which refers only to the student's solution, while the second type represents semantic properties of queries and compares the student's solution to the ideal solution.

3.4.2.1 Syntactic constraints

Syntactic constraints deal specifically with the syntax of the SQL language. This type of constraint is relatively simple to design. For example, constraint 2 specifies that the SELECT clause of a SQL query cannot be empty; this is demonstrated in figure 3.7. This particular constraint, unlike other constraints, is always relevant, hence its relevance condition reduces to 't', the LISP symbol for a condition that is always satisfied. The current knowledge base contains 76 constraints (19% of all the constraints) that are always relevant.

```
(p 2
 "The SELECT clause is a mandatory one. Specify the attributes/expressions to retrieve from the database."
 t
 (not (null (select-clause ss)))
 "SELECT")
```

*Figure 3.7 LISP code for constraint 2*
Source: Mitrovic & Ohlsson [1999:8]

The satisfaction condition of constraint 2 checks that the SELECT clause of the student solution (denoted by the variable ss) is not empty. This constraint is implemented by a combination of atomic LISP predicates. The first part of the constraint describes the instructional feedback message that the SQL-Tutor displays upon violation of the constraint, while the last part of the
constraint provides the name of the clause to which the constraint makes reference, which is also included in the instructional feedback message.

Another more complex example of a syntactic constraint is constraint 146 which checks that the names used in the FROM clause of a query are valid names of tables in the current database chosen (see figure 3.8). In this case, the relevance condition checks that the student has used some names in the FROM clause. If the student has not used any names, then this particular constraint does not apply, but others will. The satisfaction condition then tests whether each of those names is to be found among either the relevant attributes or the valid tables for the current database. These tests do not contain pattern elements; instead, they contain complex LISP predicates.

```
"You have used some names in FROM that are not from this database!"
(bind-all '?n (fmd-names ss 'from) bindings)
(or (attribute-in-db (fmd-schema (current-database *student*)) '?n)
(valid-table (fmd-schema (current-database *student*)) '?n))
"FROM")
```

Figure 3.8 LISP code for constraint 146
Source: Mitrovic & Ohlsson [1999:9]

A further example of a syntactic constraint is constraint 22 which verifies that the student is using the BETWEEN predicate correctly (see figure 3.9). The relevance condition checks whether the WHERE clause has, in fact, been specified and then finds all parts of the BETWEEN clause. The next set of checks is performed by the satisfaction part of the constraint. These checks confirm that each condition is specified on a valid attribute, that the AND keyword is used to separate the lower and upper value of the interval, that the constants used are of the appropriate type, and determines whether the NOT operator has been used within the condition. For this particular constraint, several conditions are of the pattern matching type, denoted by the use of the 'match' function.
3.4.2.2 Semantic constraints

Some constraints are concerned with the meaning of the symbols and commands in a query. These constraints are typically more complex than syntactic constraints. The distinction between the two types of constraints is, however, not altogether rigid, since some constraints deal with both the syntax and semantics of the student's solution. For example, constraint 361 checks that if the results of a query are sorted in decreasing order in the ideal solution, then the student's solution should comply with respect to the order (see figure 3.10).

In general, the constraints have a modular design, with each constraint focusing on one aspect of the solution. This has led to an increase in the total number of constraints, but it has the added benefit of appending a single feedback message to each constraint, which in turn results in a simplified student model and delivery of instruction.
3.4.3 Student modeller

The SQL-Tutor maintains a student model for each student, which contains general information about the student-name and knowledge level, a history of previously solved problems, and information obtained from student's solutions about the usage of constraints [Mitrovic, 1998].

Upon initialisation of the SQL-Tutor, the constraints are compiled into two structures called the relevance and satisfaction networks. Three types of nodes characterise these structures, namely input, test and output nodes. The test nodes have a single input each, which is similar to a tree structure. The term 'network' will, however, be used since it is consistent with terminology used to describe pattern-matching techniques.

Constraint violations are identified in a two-step process. The first step propagates the student's solution and the ideal solution through the relevance network. The result is a list of constraints containing the relevance conditions, which match the current situation. The number of relevance constraints per student solution varies from 86 for simple queries to more than 100 for more complex ones. In the second step, a comparison is made between the satisfaction component of constraints whose relevance conditions match the current situation, and the current problem-state. This means that the student's solution and the ideal solution are propagated through the satisfaction network. If a satisfaction condition matches the state, the corresponding constraint is satisfied and the system does not take any instructional action. If, on the other hand, this constraint is violated, the outcome is recorded and appropriate instructional action, in the form of system feedback, is initiated. For examples of system-generated feedback, refer to section 3.4.4.2. Hence the student model consists of a list of violated constraints [Mitrovic & Ohlsson, 1999].

The student modeller maintains a history of each constraint recording how often the constraint was relevant for the ideal solutions to the practice problems the student attempted, how often it was relevant for the student's solutions and how often it was satisfied or violated. This information is stored in three indicators called relevant, used, and correct. The pedagogical module uses this information to manage tutorial instruction.
3.4.4 Pedagogical Module

The pedagogical module (PM) of the SQL-Tutor:

- Selects problems for students; and
- Generates appropriate instructional actions according to the student model.

Tutorial instruction is tailored to the student, since both types of action (problem selection and instructional action) are based on the student model. The SQL-Tutor evaluates students' solutions by matching them to constraints. Constraints deal with the syntax of the language as well as with semantics of problems [Mitrovic & Ohlsson, 1999].

3.4.4.1 Feedback

The SQL-Tutor does not trace the student step-by-step through his/her problem-solving process, which means that it does not give feedback after individual problem-solving steps. Feedback is delayed until the student attempts a solution and communicates this to the system by selecting the 'submit' button. The justification for such an approach is that the sequence or individual steps taken cannot in themselves constitute a correct or successful query.

A particular student solution to a problem can violate several constraints. The SQL-Tutor examines and notes all constraints but targets one of them for instruction. The one targeted is the one with the highest number of violations, computed as the difference between the used and correct indicators in the student model. The rationale for this approach is obvious, since having violated the constraint so often, the student is viewed as being in urgent need of instruction on the correct use of that constraint.

The student is given an indication of the total number of errors in his/her solution but receives feedback on only one of them. The underlying assumptions are that students find it easier to work on one error at a time and that multiple error messages related to multiple errors or misconceptions might be confusing.

Students receive feedback on incomplete solutions as well. Constraints have been designed to check that mandatory parts of a SELECT statement are specified and to compare the student's
solution to the ideal one. These constraints equip the tutor to provide feedback, even in the event of an empty solution being submitted.

The SQL-Tutor provides support for five levels of feedback, namely:

- Positive/negative feedback;
- Error flag;
- Hint;
- Partial solution; and
- Complete solution.

Feedback messages vary in the amount of detail they provide. At the least detailed level, a positive or negative feedback message informs the student whether his/her solution is correct or incorrect. If there are errors in a solution, the student is informed of the number of errors committed. The system uses an error flag to inform the student as to the clause in which the error occurred. A hint message provides more information about the nature of the error. The student is given a general description of the error as contained in the definition of the constraint. A partial solution displays the correct format of the clause in question, while a complete solution shows the correct content of each clause in the query.

The feedback mechanism operates by providing students with increasing detail, only if required. When a student submits a solution to a new problem, he/she receives only positive or negative feedback. If the student subsequently makes several unsuccessful attempts at solving the problem, the feedback is upgraded to the error flag level and then to the hint level. The system is designed so as not to offer more than a hint, but students have the option of requesting partial and complete solutions by clicking on the appropriate feedback level.

3.4.4.2 Examples of interactions with the SQL-Tutor

A number of examples follow to illustrate typical interactions. The student specifies SQL code for the required query and the system provides appropriate feedback.
Example 1

Example 1: List the names of directors born in or after 1920.

Correct solution:
SELECT LNAME, FNAME
FROM DIRECTOR
WHERE BORN >= 1920

Student’s solution:
SELECT LNAME, FNAME
FROM DIRECTOR
WHERE DIED >= 1920

SQL-Tutor:
Check that you are comparing the numerical constant to the right attribute in the WHERE clause.

In this example the student is informed of a semantic error committed. Instead of using the BORN attribute the student mistakenly supplied the DIED attribute, an error which would not have been picked up by a commercial RDBMS, since the syntax is acceptable although it would generate the wrong data.

Example 2

Example 2: List the titles of all comedies or dramas.

Correct solution:
SELECT TITLE
FROM MOVIE
WHERE TYPE = 'comedy' or TYPE = 'drama'

Student’s solution:
SELECT TITLE
FROM MOVIE
WHERE TYPE = 'comedy' OR TYPE = 'drama'

SQL-Tutor:
Well done!
You’ve solved this problem correctly! Select a new problem to work on.

This is an example where positive system feedback is provided, and the student is encouraged to attempt a new problem.
**Example 3**

Example 3: For each director, list the director's number and the total number of awards won by comedies he/she directed if that number is greater than 1.

Correct solution:

```sql
SELECT DIRECTOR, SUM(AA WON)
FROM MOVIE
WHERE TYPE = 'comedy'
GROUP BY DIRECTOR
HAVING SUM(AA WON) > 1
```

Student's solution:

```sql
SELECT NUMBER, SUM(AA WON)
FROM DIRECTOR, MOVIE
WHERE DIRECTOR.NUMBER =
AND TYPE = 'comedy'
```

SQL-Tutor:
Feedback:
Almost there – you made 4 mistakes.
You can correct your query and press 'submit' again or try getting some more feedback.

Error Flag:
Almost there – a few mistakes though.
One of them is in the FROM clause.
You can correct your query and press 'submit' again or try getting some more feedback.

Would you like to have another go?

Amended student's solution 1:

```sql
SELECT DIRECTOR, SUM(AA WON)
FROM MOVIE
WHERE TYPE = 'comedy'
```

SQL-Tutor:
Error-Flag: One of the errors is in the GROUP BY clause.

Amended student solution 2:

```sql
SELECT DIRECTOR, SUM(AA WON)
FROM MOVIE
WHERE TYPE = 'comedy'
GROUP BY DIRECTOR
```

SQL-Tutor:
Hint:
You have made only one mistake. You need to restrict grouping. Specify the HAVING clause as well!

This example demonstrates several unsuccessful attempts made by a student at solving the problem. It also shows how the system upgrades its feedback from the initial feedback level to the error flag level and finally to the hint level. This example reveals its instructional strategy of selecting one error at a time for remediation purposes.
3.4.4.3 Problem selection

Students have the option of selecting problems themselves from a pre-defined sequence of problems by using the next problem option. A student can peruse and select a particular problem directly without having to follow the pre-defined sequence. This feature allows students to return to problems unsolved on a previous attempt. From a pedagogical perspective, however, problem selection by the student has the disadvantage that constraint coverage is unpredictable.

Alternatively, students can use the system’s choice facility for problem selection. The system selects problems on the basis of information in the student model. The problem-selection strategy is based on two rules. The first rule is concerned with identifying the highest number of constraint violations, which gives the tutor an indication of the constraint(s) in which the student needs instruction. The second rule’s function is to identify constraints that have not been relevant for any problem attempted up to that point, in order to select a problem for which the constraint(s) is relevant.

It is important to note that the SQL-Tutor has not been designed to replace classroom instruction, but rather to supplement it by providing an environment in which students can practice their SQL programming skills, obtain direct and immediate feedback on their solutions, as well as receive instruction on constraints violated. The SQL-Tutor is primarily concerned with helping students to correctly formulate/define SELECT statements for a variety of real-world problems. While some may argue that the focus is limited, the current approach and techniques of the tutor can be expanded to teach other SQL statements.

3.5 The SQL-Tutor’s role in instruction and learning

The SQL-Tutor supports a guided discovery and learning-by-doing instructional strategy. In particular, three kinds of learning are promoted: conceptual learning, problem solving, and meta-learning. Firstly, the student can learn concepts, elements and query-formulation rules by using interface controls, menu options, and studying the system-generated explanations to student solutions via the feedback mechanisms employed. Secondly, the SQL-Tutor is characterised by a problem-solving environment where students typically acquire knowledge of
the application domain in a declarative form via constraints, and strengthen this knowledge in practice. In this sense the SQL-Tutor serves as an *implicit teacher* of the rules constituting the SQL database language. The SQL-Tutor provides guidance in problem solving by a system of feedback where students can learn from their mistakes and remedy them. Finally, the system supports meta-learning in the form of self-explanations as typified by error-messages and correct solutions [Mitrovic, 1998].

The SQL-Tutor makes two important contributions to the field of educational technologies for teaching SQL. Firstly it provides syntactic as well as semantic error messages because it understands the semantics of the problems held in its database. Furthermore, since the feedback error messages are rooted in constraints defined for the language, they are more valuable and informative to the student. Secondly, the tutor can adapt its instruction to a particular student by providing instruction on constraints the student violated and selecting problems containing constraints not yet encountered by that student.

### 3.6 Conclusion

The SQL-Tutor does not explicitly teach the concepts and elements of SQL, but rather provides an intelligent practice environment, incorporating implicit teaching by way of diagnosis and focussed feedback. Learning is also guided by judicious system-selection of appropriate problems, although students can opt for learner-control and select problems themselves. A discussion follows on the SQL-Tutor’s conformance to certain theories and principles outlined in Chapter Two.

It embraces many of the principles emanating from the ISPDL (impasse-success-problem-solving-driven learning) theory discussed in section 2.3.1. Specifically, it is characterised by the following principles:

1. **The system does not interrupt the learner** while he/she is engaged in the problem-solving activity, but instead offers information in the form of feedback when the learner has completed the process;
2. **Learners are encouraged to discover and correct their own errors**;
3. **System feedback is tailored to the actual knowledge-state of the learner** by the use of learner models; and
4. The system offers learner-initiative in the choice and sequencing of interactions with the system, in the sense that a learner can choose which problems to solve and select the level of feedback required.

The SQL-Tutor also embraces aspects of the ACT theory of cognition and learning, discussed in section 2.3.2. In particular, the tutor:

1. Distinguishes between declarative knowledge and procedural knowledge by encoding declarative knowledge into the system via a set of constraints or rules. Furthermore it provides a practice environment for solving problems, thereby allowing learners to practice their cognitive skills and procedural knowledge;

2. Assumes knowledge compilation, that is, learning by employing declarative knowledge in the context of a problem-solving activity; and

3. Allows for strengthening, of both declarative and procedural knowledge, through sustained practice.

The principles for the design of tutors, extracted from ACT theory, which have been implemented in the SQL-Tutor are:

1. Communicate the goal structure underlying the problem solving - this is done by providing a solution template;

2. Promote an abstract understanding of the problem-solving knowledge - this is achieved by reinforcing correct application of SQL knowledge in a guided problem-solving environment;

3. Minimise working memory load - implemented by provision of database schemas and presentation of the text of the problem on the same screen as the solution template; and

4. Provide immediate feedback on errors - done by the provision of feedback mechanisms, which permits students to select the desired level of feedback.

The architecture of the SQL-Tutor is generic, in that it resembles the three-tier architecture, containing domain, student, and tutoring models, which has been adopted by many intelligent tutoring systems. However, this architecture deviates from the norm, since its domain model does not include an expertise module or problem-solver. The reasons for this deviation are explained in section 3.4.
The SQL-Tutor uses constraint-based modelling (CBM) since this type of modelling does not require an executable expert model. The suitability of CBM for student-modelling and intelligent tutoring is demonstrated by the successful implementation of the SQL-Tutor.

In comparing the SQL-Tutor to general characteristics of intelligent tutoring systems, as discussed in Chapter Four, section 4.3.2, the following observations are true of the SQL-Tutor:

1. It allows both the system and learner to initiate instructional activities by its problem selection strategies and feedback mechanisms;
2. It can make inferences, by interpreting the learner’s inputs, diagnosing misconceptions and learning needs, and generating instructional actions;
3. It models the learner, by maintaining records of the learner’s knowledge and errors committed;
4. It acquires knowledge of the progress of the learner, and constructively uses this knowledge in guiding interaction;
5. It makes qualitative decisions with respect to instruction, in the problems it presents, and the feedback it generates to the learner; and
6. By its system of feedback, learners are able to reflect on their activities, progress and results.

The SQL-Tutor’s role in teaching and learning is in line with current directions set out in 4.2.2.5 and 4.3.2, in that higher-order thinking skills and independent real-life learning skills are being inculcated in the learner. Furthermore, individual learners can adapt aspects of the tutor to their personal requirements.

This version of the SQL-Tutor has scope for improvement in both the usability and functionality aspects. In particular the user-interface, student modelling and pedagogical components, as well as the knowledge base can be improved to provide a more effective learning tool for students. The SQL-Tutor is comprehensively evaluated in Chapter Six, using the generic methodology, framework, and criteria developed in Chapters Four and Five.
CHAPTER FOUR

Evaluation of Computer-Assisted Instruction and Intelligent Tutoring Systems: Characteristics and Criteria

The main theme of this chapter is to investigate the required characteristics of, and general evaluation criteria for, both CAI and ITS technologies, and to establish appropriate criteria for the specific evaluation of ITSs.

4.1 Introduction

This introduction attempts to answer three key questions, and gives an overview of the remaining sections in the chapter, which address the main theme of characteristics and evaluation criteria for computer-assisted instruction (CAI) and intelligent tutoring systems (ITSs). Chapter Four investigates ‘what’ aspects should be evaluated, laying a foundation for Chapter Five, which addresses the issue of ‘how to evaluate’.

4.1.1 What is an evaluation?

According to O’Neil & Baker [1987], evaluation is an activity intended to provide an improved basis for decision making. Winne [1993] defines an evaluation study as a systematic effort aimed at gathering and interpreting, in a principled way, information with which to gauge the worth or value of an instructional enterprise.

4.1.2 Why are evaluations necessary?

Evaluations are undertaken to ascertain strengths and weaknesses; they provide guidance for improvement, documentation for accountability, and information to increase understanding of the processes under study. The process of evaluation is complex, and involves defining the assessment criteria, identifying key questions, obtaining and analysing relevant information, judging the merit and/or worth of the program, reporting results, and promoting effective use of the findings, to improve products or take decisions regarding their usage [Center for Evaluation, 1994].
Cohen & Howe [1988] believe that evaluation should be a mechanism of progress both within and across artificial intelligence research projects, as it:

- Provides impetus to the research cycle;
- Opens new avenues of research, because experiments often raise new questions while answering some, and also identify deficiencies which become problems for further research;
- Provides a basis for accumulation of knowledge; and
- Convinces the community that the researcher’s ideas are worthwhile, that they work and how.

According to Winne [1993], evaluations of ITSs are important in that they:

- Command personal and fiscal capital;
- Influence interest in, and support for, future work in the field of intelligent tutoring systems;
- Sway public and professional opinions about the field as a whole and portray progress in education, training, practice and science; and
- Shape what and how people learn.

These motivations for, and purposes of, evaluation are in the context of AI, that is, looking at evaluation for research and development purposes, whereas the educational perspective views evaluation as a means to improve instructional products that lead to better, and more effective learning.

### 4.1.3 What is the role of evaluation in ITS applications?

O’Neil & Baker [1987] suggest that evaluation can assist ICAI/ITS applications in the following ways:

1. Encouraging social acceptance of the practice of ITS evaluations accompanied by expectations of repeatability, verifiability and public reporting;
2. Eliciting designer participation and co-operation in the evaluation of their products;
3. Inviting AI expert participation in ICAI evaluations from a professional responsibility perspective;
4. Giving due consideration to the evaluation of specific features of ICAI/ITS development;
5. Subdividing the process into componential evaluation where appropriate, focusing on the utility of the portion of software under development; and
6. Ensuring objectivity by responsible and responsive evaluation.

4.1.4 Overview of Chapter Four

This chapter focuses on 'what to evaluate' as opposed to 'how to evaluate', which forms the basis for discussion in Chapter Five. In accordance with this theme, a major section within the chapter reviews quality standards and characteristics/criteria that are frequently cited in the literature as forming the basis for the evaluation of conventional CAI. The next major section discusses ITSs – the technology under investigation – to learn about the special features and characteristics of ITSs and to ascertain the focus of evaluation endeavours in that area of instructional science. Finally, a taxonomy bearing criteria appropriate for the evaluation of ITSs is presented.

Figure 4.1 illustrates the key disciplines involved in the domain of ICAI/ITS [Kearsley, 1987]. The arrows represent requirements and evaluation criteria. This figure serves as a justification for the approach adopted in this study, in that one cannot consider the evaluation requirements of ICAI/ITSs without a concomitant consideration of evaluation criteria for other key disciplines related to the ICAI/ITS technology.
4.2 Characteristics of and evaluation criteria for conventional CAI

In the 1960's research commenced into the use of computers in education (CAI: Computer-Assisted Instruction). This led to the development of CAI systems, which were largely influenced by theories of behaviourism. These systems provided a one-way teaching interaction, whereby the system controlled the interaction using rigid pre-defined dialogues. The system evaluated the student's learning by comparing the student's answers with a limited number of pre-defined stored answers [Garito, 1991]. Figure 4.2 illustrates a typical flow of events in CAI [Shute & Psotka, 1996].

![Figure 4.2 Shute & Psotka's CAI Model](Source: Shute & Psotka [1996: 6])

This simple model could be modified to include various mastery criteria whereby students have to answer a certain proportion of questions correctly before moving ahead in the curriculum. Failure to meet with pre-set criteria would force the student back into remediation mode.

The characteristics of CAI software systems are overviewed, followed by an investigation into their evaluation criteria.
4.2.1 Characteristics of CAI

A ranked ordered list was compiled [Bitter & Wighton, 1987:8-9], delineating the most important characteristics of educational software from the viewpoint of major software evaluation agencies.

1. Correctness of content presentation - The program should be free from content-informational-computational-grammatical and syntactical errors.
2. Presentation of content - The pedagogical content should be presented in a clear, concise, logical and manageable fashion, and should have sufficient depth of instruction and/or practice, so that learning will occur.
3. Use of technology - There should be an appropriate use of computer technology, such that the program takes full advantage of the computer's capabilities and provides students with a learning experience that cannot be presented better in other media.
4. Integration into classroom use - The program should be effectively and easily integrated into classroom use. The software should lend itself to use within a classroom time frame. Effective and appropriate teacher-support materials should be available. A teacher should be able to use the program easily.
5. Ease of use - The program should be user friendly.
6. Curriculum congruence - The content should directly support the curriculum.
7. Interaction - The interaction should be effectively achieved for the target audience. There should be a sufficient amount and a sufficiently high quality of interaction to promote learning.
8. Content sequence/levels - There should be multiple levels of difficulty with appropriate incremental steps between the levels, so that the development sequence and the difficulty of the levels are appropriate for the target audience.
9. Reliability - The program should be free from programming and technical errors.
10. User control of program - The user (student or teacher) should be able to control the rate, amount and sequence of presentation.
11. Feedback (general) - The program should correctly assess student input and provide appropriate and effective feedback messages.
12. Objectives - The objectives should be clearly stated, and met.
13. Motivation - The program should be motivational.
14. Branching - There should be branches to provide individualised instruction according to each student's needs.
15. **Negative feedback/help** - Corrective feedback messages or help screens should be provided where needed.

16. **Content modification** - The teacher should be able to modify the content.

17. **Content bias** - The content should be free from bias (race, sex, cultural, ethnic, stereotyping, and violence).

18. **Teacher documentation** - The documentation should be comprehensive, easy to understand, and well organised.

19. **User support materials** - User support materials should be present; they should be appropriate and effective.

20. **Sound, video, graphics, animation** - These features, if present, should be effectively used to enhance the program, and should not distract.

21. **Screen displays** - Screen displays should be effectively and appropriately formatted.

22. **Management system** - There should be a management system, which provides an effective means for record keeping and/or assignment control.

From the above ranked order, one can gather that the emphasis is on pedagogy, integration and content, and that aspects such as ease of use and machine presentation have shifted from top priorities to assumed priorities.

### 4.2.2 Evaluation criteria for CAI

Criteria proposed by various authors for the evaluation of CAI programs are described.

#### 4.2.2.1 An evaluation model

Lauterbach & Frey [1987] suggest that three standards should be incorporated into an evaluation model, namely technical standards, didactical standards, and interactive standards. Deviations from the standards should be recorded as either bonuses or deficiencies.

A sample, extracted from their complete list of evaluation questions associated with each of the three types of standards, is given below [Lauterbach & Frey, 1987:389-391]:
1. **Technical standards**

*Service for running the program*

(Starting, parameter input, manipulation)

- Does the program operate reliably?
- Can a demonstration run be made with specification of good (and possibly also of unsuitable) parameters?
- Can the individual program sections be repeated (analogous to turning back the pages of a book) or omitted?

*Service for recognition, start and manipulation of program*

- Is a clear overall representation of the program (menu levels) provided?
- Can any program function (e.g. a particular menu) be started at any time?
- Does the program have help functions? For example, can a summary of edit instructions or of all input instructions be given?

*Graphic quality of screen display*

- Are the texts and graphics clearly arranged?
- Is the screen display appropriate and related to the contents?
- Are the graphic techniques such as colours, framing, underlining used to elucidate the contents?

*Connection of peripheral equipment*

- Can screen contents be printed out?
- Is there a provision to print out a record of the results?

2. **Didactical standards**

*Objectives, contents, methods*

- Is there a rationale for the selection of educational objectives and contents?
- Is there a proven/visible educational value to the program?
- Is there a justified/recognisable relationship between the objectives, contents and methods?
- Does the didactic approach represent the present state of knowledge with regard to scientific content and didactic deduction?
Representational form (e.g. graphics, tables, text, animation)
- Is there a recognisable relationship between representational form, program run, content and didactical methods?

Effects
- Does the program facilitate new learning experiences and teaching forms, which would not be feasible or possible without a computer?

3. Interactive standards

Scope for intervention
- Can the teacher or student select different levels of difficulty/complexity?
- Are there options on content selection (by teacher and student)?
- Is there a provision for input and processing of real data?
- Does the program software permit modification of data or of the program?

Scope for activity-prompting feedback
- Does the program accept and give variable answers in order to minimise routine behaviour on the part of the user?
- Does the program include an error analysis function to support the learner?
- Is the program embedded in other teaching and learning methods?
- Does the program stimulate additional, computer-free activities?
- Does the program foster co-operative work between students?

Lauterbach’s & Frey’s [1987] set of standards may be regarded as a watershed in establishing the quality of computer-assisted instruction programs.

4.2.2.2 Focus of evaluation forms and checklists

Heller [1991] reviewed a number of evaluation forms and checklists proposed/designed by several researchers/evaluation agencies. Some of these are outlined below:
• The **National Council of Teachers** nine-section checklist, including among others:
  - Demographic information about the grade and the ability level of the software;
  - The software's intended instructional grouping and usage;
  - Likert-like scale sections representing instructor and student viewpoints on the content and motivational aspects of the program; and
  - Social characteristics such as the presence or absence of competition, co-operation, humanisation of the computer, and moral issues or value judgements.

• The **Northwest Regional Educational Laboratory** early review form, developed in 1980 featuring a 5-point Likert-scale portion with three sections, namely:
  - Content characteristics;
  - Instructional (pedagogic presentation) characteristics; and
  - Technical characteristics.

• **EDUCOM's Software Initiative** [1989] project guidelines for software review including:
  - Ease of use;
  - Adequacy of content;
  - Accuracy of instructions; and
  - Appropriateness of response to student.

• The evaluation form presented in **Power On!** [1988], with major categories of:
  - Instructional quality;
  - Teacher modifiability;
  - Evaluation and record keeping; and
  - Technical quality.

• The **York Software Educational Evaluation Scales** [1987] representing a criterion-based evaluation with four dimensions, to be evaluated as exemplary, desirable, minimally acceptable or deficient:
  - Pedagogical content;
  - Instructional presentation;
  - Documentation; and
  - Technical adequacy.
From the evaluation forms and checklists examined, Heller [1991] concluded that evaluations vary in detail, focus and length, and that they generally focus on one or all of the following:

- Technology;
- Content; and
- Pedagogical presentation.

### 4.2.2.3 Evaluation criteria for educational technology

Castellan [1993] proposes a set of criteria for the evaluation of instructional software and educational technology in general. The criteria cover aspects such as technical accuracy, pedagogical soundness, substantive fidelity, integrative flexibility, and cyclic improvement. Some typical evaluation questions [Castellan, 1993:234-236] that fall under each of these aspects are outlined below:

1. **Technical Accuracy**
   - Does the software execute correctly?
   - Will it work on the platforms that students will use?

2. **Pedagogical Soundness**
   - Does the software use appropriate pedagogy?
   - Are instructional goals clearly articulated for the student?
   - Is it clear where technology ends and substance begins?
   - Is the use of technology appropriate to the concepts to be learned?
   - Is the technology used at the appropriate time in the course?
   - Can the student use the technology to get an overview, do a detailed study, and subsequently review the material?
   - Does the technology facilitate appropriate collaborative learning?
   - Does the technology encourage exploration, testing, and application of ideas and concepts?
   - Does the technology permit appropriate self-assessment of comprehension?
   - Can the skills and concepts learned be transferred beyond the context in which learning takes place?
3. **Substantive Fidelity**
   - Is the material accurate?
   - Is the material worth learning?

4. **Integrative Flexibility**
   - Can the instructional software or technology be integrated into the course easily?

5. **Cyclic Improvement**
   - Evaluations of instructional technology should be made before, during, and after use.
   - If problems surface one must be able to trace the cause of the problem either to the technology or to the way in which the students use the technology or the way in which the technology is integrated into the course.

All the above mentioned evaluation aspects/characteristics/criteria presented by various researchers can be conveniently classified under the categories of technical, didactic, and interactive standards as proposed by Lauterbach & Frey [1987] – see section 4.2.2.1. These characteristics, while qualitative in nature, can be used to yield quantitative measures for evaluation by the use of appropriate scales.

### 4.2.2.4 CAI evaluation framework

Gros & Spector [1994] identify three different components of evaluation for CAI, which are illustrated in figure 4.3.

![Figure 4.3 CAI Evaluation Framework](source: groespector[1994:41])
A brief description follows for each of these components:

1. **Product-oriented evaluation**

A product-oriented evaluation focuses on an object or a product. A critical appraisal of courseware could be conducted using an evaluation checklist. One possible list of criteria for evaluating courseware is the one presented by Bitter & Wighton [1987] - see section 4.2.1.

2. **User-oriented evaluation**

User-oriented evaluation focuses on the user of the product. This level is aimed at evaluating the effects of the program on the user in terms of:
- The interaction between the program and the learner;
- The levels of adaptation, if any;
- The means used to ensure motivation;
- Learning effectiveness; and
- User responsiveness to the courseware.

3. **Context-oriented evaluation**

This level relates to the cultural and social factors involved in the use of computers.

Gros & Spector [1994] combine most of the technological, didactic and interactive concerns from the earlier evaluation checklists/ forms into a single level, namely the product. The user-oriented evaluation level is designed to measure the effect of the program on the user/learner. Evaluation of interactive standards falls partially into this level, but evaluated, from the perspective of the user. The factors included in the third level, context-oriented evaluation are not altogether new, since the social factor is included in some of the earlier checklists, but more recognition is given to the role of cultural and social factors in learning.

Gros and Spector indicate that many evaluation studies focus on only one of the levels mentioned above. They believe that a complete evaluative undertaking should address the product, the users, the context, and their inter-dependencies.
4.2.2.5 Hexa-C Metamodel

De Villiers [1999b] proposes an investigation of instructional programs from a different perspective. The Hexa-C Metamodel focuses on the extent to which learning and instructional environments conform to current theories and principles of instructional design.

Learning and instructional theories are related to the underlying philosophy of educational technology products. Current directions that are emerging in learning theories and instructional design are cognitive science, constructivism, customisation, creativity, component-based instruction, and collaboration, compositely termed the Hexa-C Metamodel [De Villiers, 1998; 1999a; 1999b]:

1. **Cognitive science** refers to the reasoning and thinking processes a learner engages in when acquiring new knowledge and skills.
2. **Constructivism** is a branch of cognitive learning theory, based on the premise that learners actively construct their own knowledge.
3. **Customisation** deals with the tailoring of instruction to meet the needs of individual learners.
4. **Creativity** entails providing instruction that is engaging, motivating, satisfying, and which gains the attention of, and instils confidence in, the learner.
5. **Component display theory** examines the relationship between instructional goals and instructional strategies, and between the type of performance required of the learner and the content to be taught.
6. **Collaboration** is concerned with promoting learning by the creation of group-related learning activities, where learners participate by co-operating with one another.

4.3 Characteristics of and evaluation criteria for ICAI/ITSs

This section highlights the differences between CAI and ICAI/ITSs in order to determine what should be the focus of ITS evaluations. This is followed by an examination of the characteristics of, and evaluation criteria, for ITSs.
4.3.1 ITSs compared and contrasted with conventional CAI

A comprehensive study of evaluation requirements/criteria for CAI systems has been undertaken. Before investigating the specific evaluation criteria for ITSs, it is necessary to understand the principal differences between CAI systems and ICAI/ITS systems, as well as the unique features of ITSs.

Park, Perez & Seidel [1987] highlight several differences between conventional computer-based instruction (CBI) and intelligent computer-assisted instruction (ICAI), summarised as follows:

1. There is a fundamental difference in the philosophies underlying the structure and development of CBI and ICAI, since ICAI has evolved from the field of computer science (in particular, artificial intelligence) while CBI has its roots in instructional psychology or technology.

2. CBI was developed by educational researchers with the intent of addressing practical problems in the fields of education and instruction by applying computer technology. ICAI, on the other hand, was developed primarily by computer scientists to explore the potential of AI in the fields of teaching and learning. The focus of ICAI is thus on technical and theoretical aspects of the systems, namely, knowledge representation techniques, natural language dialogues, inferencing mechanisms, etc.

3. Most CBI systems follow a systems approach to incorporate instructional principles, while ICAI systems are designed and built on the theoretical notions of cognitive science that have emerged from the theory of human information processing, rooted in cognitive psychology.

4. The style of CBI is typically frame-oriented where the student has little or no initiative in the learning process. In contrast, ICAI is generative, in that it processes knowledge, which it stores in the system in its process of interacting with students.
5. CBI systems cater for many different types of instructional strategies depending on the purpose of instruction, student characteristics, and the subject matter. Instructional strategies range from a teacher-centred expository approach to the adoption of cognitive principles and strategies. ICAI researchers, on the other hand, adopt ‘learning–by-doing’ as the basic instructional approach. Most ICAI systems use a student-centred discovery form of instructional method, with tutorial dialogues guided by the student’s conceptual understanding and learning behaviours.

6. Task analysis is the systematic method used by CBI developers to identify all subtasks and content elements constituting the learning task. As opposed to task analysis, AI knowledge representation techniques are used in ICAI systems to organise knowledge (including subtasks and content elements) into a data structure for subsequent computer manipulation.

7. In CBI, a number of quantitative procedures are applied to model the student’s learning and to select instructional strategies on the basis of this quantitative information. By contrast, the student modelling method in ICAI systems is qualitative, where students’ learning is assessed by analysing their responses (or response patterns).

8. Common CBI instructional formats includes tutorial, drill and practice, games, and simulations. The CBI tutorial is typified by the system’s expository representation of instruction followed by questions to reinforce the student’s understanding of the presentation. Most ICAI systems are classified into two types of instructional formats: tutorial and games. The ICAI tutorial is typified by a series of question and response exchanges, whereby inferences are made about the student’s conceptual understanding of the given problem in order to determine which instructional approach to apply.

9. CBI has been, and continues to be, used in a wide variety of subject matter areas from mathematics to the arts. ICAI, however, is applied predominantly to well-structured subject areas, such as electronic troubleshooting, mathematics, medical diagnosis, informal gaming environments, and program-plan debugging as well as consultancy systems.
10. CBI programs adopt a systems approach following the phases of analysis, design, development, formative evaluation, implementation, summative evaluation and maintenance. In contrast, few ICAI systems follow a generalised development methodology. The development process and procedures varies among projects, depending on the researcher’s goals, his/her instructional development skills, and the characteristics of the domain knowledge.

11. The success of a CBI program is, in large measure, determined by the degree of its instructional effectiveness and efficiency. Different evaluation methods are used to monitor the CBI development process, namely subject-matter expert review, one-on-one trials, pilot tests, etc. The success of ICAI programs, on the other hand, is primarily determined by a system’s ability to handle specific features or processes in instruction (for example inference mechanisms, bug analysis procedure and natural language dialogue capability).

12. CBI programs use general-purpose languages, authoring languages and authoring aids/systems. ICAI systems use mainly LISP and Prolog programming languages.

Having reviewed the principal differences between CAI and ICAI/ITS systems, a closer examination of special features and characteristics of ITS systems follows, before an attempt is made to establish specific evaluation criteria for ITS systems.

4.3.2 Characteristics of ITSs

Dede [1986], Ohlsson [1987], Seidel, Park & Perez [1988], MacKenzie [1990], and Shute & Psotka [1996] describe a number of intelligent or powerful features inherent or desirable in intelligent tutoring systems. These characteristics, outlined below, are intended to facilitate the development of robust adaptive instructional and learning environments:

1. ITSs have a computational, inspectable model of expertise [Dede, 1986] that can generate knowledge, as opposed to selecting pre-programmed frames of instructional content to present to the student. This dynamic or spontaneous generation of knowledge, in response to students’ needs, is attributable to the utilisation of AI techniques for handling

2. ITSs apply AI techniques to allow both the system and student to initiate instructional activities in a subset of natural language. Although limited, this *natural language dialogue capability* makes it possible to simulate one-on-one tutoring using natural language interaction [Seidel, Park & Perez, 1988; Dede, 1986].

3. ITSs have epistemological distinctions of inference and process structure [Dede, 1986]; they can *make inferences* in interpreting the student's inputs, diagnosing misconceptions and learning needs, and generating instructional actions. The system's capability to draw the same inferences from student intentions and meanings as contained in the different syntactic and semantic representations of the students, is one of the unique attributes of an ITS [Seidel, Park & Perez, 1988].

4. ITSs have *generic teaching knowledge* [Dede, 1986], whereby an ITS can monitor, evaluate, and improve its own tutoring performance by applying AI techniques commonly used in machine learning. The system can make explicit experimental changes in its tutorial strategies and evaluate such changes (after teaching more students or teaching the same student more material). The change may involve the selection of another existing rule as an alternative, a modification of the condition and action parameters in the same rule, or the generation of a new rule. Shute & Psotka [1996] refer to this trait as *adaptive remediation*.

5. ITSs *model the student learning process* by maintaining records of the student's knowledge and reasoning [Dede, 1986 & Seidel, Park & Perez, 1988].

MacKenzie [1990] refers to this trait as *knowledge acquisition*, whereby an intelligent system acquires knowledge of the progress of the learner and constructively uses this knowledge in guiding interaction. This knowledge should further enable ITSs to adapt to a learner’s style of interaction. The key component that allows system acquisition of such knowledge is learner evaluation (usually based on test items). More sophisticated systems may create a knowledge base of the learner without using test items, by recording errors, enumerating constraint violations, monitoring actions and choices, etc.
Shute & Psotka [1996] reaffirm that student modelling is the most critical element of intelligent tutoring systems.

Ohlsson [1987] adds that an intelligent system must have certain knowledge about the cognitive state of the students, in terms of what they know, how they think, and preferably, how they learn.

6. ITSs can make qualitative decisions with respect to instruction [Seidel, Park & Perez, 1988]. Ohlsson [1987] proposes that an intelligent tutor should be a problem-solver, capable of generating teaching plans, executing such plans and revising plans when the plan does not fit the student.

7. ITSs have generic problem-solving procedures [Dede, 1986].

8. Mackenzie [1990] believes that learner control is an important feature of intelligent tutoring environments, whereby learners can control and guide the learning activities offered by the system.

9. MacKenzie [1990] advocates fault tolerance as an important characteristic of ICAI/ITS packages. These systems should provide an intelligent user interface that is functional, consistent, and flexible to minimise errors, and to allow a high degree of latitude for acceptable user responses.

10. MacKenzie [1990] believes that feedback is a key feature of intelligent tutoring systems, allowing learners to reflect on their activities, progress, and results. Shute & Psotka [1996] also cite system-generated feedback as an important characteristic of intelligent tutoring systems.
11. Ohlsson [1987] advocates the principle of *generative interfaces* for intelligent tutoring systems. This principle states that a tutor must distinguish between the subject matter and the formats in which it can be presented, and be able to generate different presentations of each subject matter unit as and when needed.

From the comparative study undertaken, and the ensuing description of the intelligent features of ICAI/ITS systems, it is evident that ICAI/ITS systems are substantially different from their earlier counterparts, namely CAI, but they share several important characteristics. This implies that evaluation criteria employed for ITS systems would, to some extent, differ from the evaluation criteria used for conventional CAI systems.

### 4.3.3 Evaluation criteria for ITSs

This section reviews the practical and technical problems associated with evaluating ITSs, before discussing criteria for evaluating the internal and external dimensions of ITSs. Internal evaluation analyses the relationship between the architecture of an ITS and its actual behaviour and is concerned with the *inner workings* of a tutoring system, while external evaluation measures or assesses an *effect external* to a tutoring system, namely the student’s learning [Littman & Soloway, 1988; Siemer & Angelides, 1998].

#### 4.3.3.1 Problems associated with evaluating ICAI/ITSs

O’Neil & Baker [1987] discuss the technical and practical problems associated with evaluating emerging technologies such as the fields of intelligent computer aided instruction or intelligent tutoring systems.

As ICAI/ITS endeavours were largely steeped in a research rather than a development context, they had the following characteristics:

- Research goals contributing to knowledge and theory building appeared to be of foremost importance;
- Efforts were selected on the basis of research leanings rather than project development requirements; and
• There being no real 'off-the-shelf-item' components available for easy substitution into the project, more effort was invested into one component (e.g. knowledge representation) at the expense of other components, and the final product was not a fully operational instructional system because of the emphasis placed on the one component.

The problem being highlighted is that the boundary between research and development is not clearly defined. This presents evaluation problems in that the 'what' to be evaluated is less concrete and identifiable.

4.3.3.2 External evaluation criteria for ITSs

The external dimension of an intelligent tutoring system focuses on its educational impact.

Mark & Greer [1993] cite two major criteria, namely achievement and affect for judging the educational effects of ITSs. Achievement and achievement-related measures are concerned with the acquisition, understanding, performance, retention, and transfer of a learner's knowledge and skills. Affective measures are concerned with attitudes and emotions, which may impact upon the students' use of and learning from ITSs.

4.3.3.3 External evaluation criteria used in ITS case studies

The works of a number of researchers will be discussed with specific reference to the external evaluation requirements underpinning their investigations.

1. External evaluation criteria for FITS study

The first case discusses the evaluation of FITS, an intelligent tutoring system that teaches fractions to pupils in the age range 12-15. Nwana [1990] addresses the external dimension of FITS by evaluating the system with real students, where pre-test and post-test scores were obtained to measure improvement in fraction skills. The emotional affective factors emanating from system usage, such as enjoyment, confidence-boosting potential, motivation, etc. were also investigated.
2. **External evaluation criteria for PROUST study**

Two evaluative studies were undertaken for PROUST, an intelligent tutoring system designed to teach introductory programming in Pascal.

O’Neil & Baker [1987:17] address the external dimension of PROUST with the following evaluation questions:

- What are the learning outcomes for students?
- To what extent do learners achieve system goals?
- Do students with different backgrounds profit differentially from exposure to the system?
- To what extent does the program create unanticipated outcomes, either positive or negative?

Littman & Soloway [1988] address the external dimension of PROUST by examining its educational impact on students. Their goal was to identify properties of the ITS that affect students’ problem-solving processes in positive and negative ways. Their evaluation was based on a process model that explained novice buggy programming. The model was used to identify the management of boundary cases, a task in programming that students have difficulty with. They used students’ ability to find and repair condition bugs as measures of PROUST’s impact.

3. **External evaluation criteria for Intuition study**

This case study evaluates INTUITION, an intelligent tutoring system for business gaming-simulation, which teaches principles and skills in areas such as marketing, production, stock control and labour relations.

Siemer & Angelides [1998] address the external dimension of INTUITION by assessing the ITS, in terms of learning achievement and learning affect. The evaluation of learning achievement involved the determination of how well the system taught underlying knowledge and skills. Learning achievement included aspects such as the acquisition and understanding of, and performance with, the student’s knowledge.
Due consideration was afforded to user-centred evaluation, which addressed the more general issue of users' satisfaction with the system. Learning affect was concerned with aspects such as attitudes and emotions caused by the intelligent tutoring system. Motivation in the learning context was used as an indication of the student's willingness to be active and involved in the learning process.

4. **External evaluation criteria used in Mitrovic's SQL-Tutor study**

The SQL-Tutor described in Chapter Three, is an intelligent tutoring system that guides students as they practice the formulation of database queries in SQL. The developer, Mitrovic used empirical evaluation to assess learning achievement and the usability aspect of the SQL-Tutor [Mitrovic & Ohlsson, 1999].

4.3.3.4 **Internal evaluation criteria for ITSs**

The internal evaluation dimension of an ITS focuses primarily on the relationship between the architecture of an intelligent tutoring system and its behaviour [Littman & Soloway, 1988]. The scope of internal evaluation is extended beyond system architecture and associated behaviour, to incorporate generic principles of computer tutor construction, usability principles and paradigms, and learning and instructional theories, which are in keeping with the concept of 'inner workings' or 'internal dimension' of an ITS.

Mark & Greer [1993] identify six architectural components of an ITS and explore possible evaluation requirements for each:

1. **Domain Knowledge**

The domain knowledge component is concerned with storing, manipulating, and reasoning with knowledge of some subject domain. The domain knowledge base may be assessed in terms of *facts and reasoning mechanisms, breadth or depth or standards of coverage*, if clearly defined for the system.
2. Teaching-knowledge component

The standards by which teaching knowledge can be evaluated are *instructional theory* and the *expert human teacher*. Possible criteria that could be used for assessing ITS teaching knowledge are:

- The range of instructional methods offered by a program;
- The degree to which a program can adapt its behaviour to individual student differences; and
- The degree to which instruction is based on educational and psychological research.

3. Student Knowledge

The student knowledge component of an ITS can carry out both diagnosis and modelling. Diagnosis examines a student's behaviour to gain meaningful information from it. Modelling relates past and present information about the student in meaningful ways, and maintains that information for use by the system.

A student knowledge component is similar to an educational or psychological test instrument in that it measures student characteristics. The criteria of test instruments, namely *validity* and *reliability* are most relevant to evaluation of the *student diagnostic component*. The *criterion of validity* is most pertinent for the evaluation of the *student-modelling component*.

4. Communications Component

The communications component, that is the user interface, presents information to the student and receives responses from the student. An intelligent tutoring system will have limited usefulness if students cannot understand the information that it presents, or if they misinterpret directions for responding, or make errors because of an awkward interface. The interface features most commonly employed in instructional computer systems are graphics and prepared text while the most common means of entering information are mouse-driven menus and graphics and the entry of domain specific notation (numbers, formulae, limited text, etc.) via the keyboard. The criteria used to evaluate user interface features may be drawn from *usability principles and paradigms* [Dix et al, 1993; Hix & Schulman, 1991].
5. Learning Component

An intelligent tutoring system may also be able to monitor and adjust its own behaviour with a learning component. In such a scenario, it is necessary to assess changes in system behaviour over time, to determine whether or not the learning component actually improved system behaviour. This assessment must be done in terms of the criteria that the learning component uses as a basis for learning, as well as in terms of other criteria related to other system components.

6. System control

The control component manages overall operation of an ITS. Interactions between components must be mediated. The control component is concerned with the underlying features of system behaviour. Evaluation requirements for assessing underlying features of system behaviour, such as scheduling of processes for different components, are used.

4.3.3.5 Internal evaluation criteria used in ITS case studies

The works of a number of researchers are discussed, with specific reference to the internal evaluation requirements underpinning their investigations.

The same evaluative studies, introduced in section 4.3.3.3, are cited in this section, this time mentioning internal evaluations.

1. Internal evaluation criteria for FITS study

The internal dimension of FITS was evaluated by Nwana [1990], in terms of:

- The principles upon which it was built to determine how well it achieved its goals;
- Its behaviour, by using a set of subject-independent questions to determine how it measures up to its prefix 'intelligent' [Self, 1985];
- ITS design, using O'Shea et al.'s [1984] thirteen pillars of design as a standard; and
- Real students to assess its usability potential.
The evaluation requirements for each of these categories are elaborated below:

- **Evaluation against principles**

  FITS was evaluated against the following underlying principles upon which it was based, namely:
  - Student pre-modelling;
  - Teaching of pre-requisite skills;
  - Provision of examples to demonstrate when and where the skills are to be employed;
  - Student monitoring;
  - Diagnosis in a problem-solving context;
  - Student communication of his/her intentions (plans);
  - Remediation in a problem-solving context;
  - Student modelling;
  - Testing of the student’s understanding;
  - Support for various idiosyncratic ways of student problem solving;
  - Explicit representation of knowledge;
  - Motivation and support for a more flexible style of tutoring;
  - Maintaining control over the whole tutoring endeavour; and
  - The mapping of real interactions between a student and an instructor and the system’s environment.

- **Evaluation of behaviour**

  Self’s [1985] set of questions regarding the behaviour of an ITS, were classified into four categories, and used for the evaluation of FITS behaviour:

  **Subject Knowledge**
  - Can the system answer arbitrary questions from the student about the subject?
  - Can the system give an explanation of a problem solution (including those posed by the student)?
  - Can the system give alternative explanations, using perhaps analogy?
  - Can the system answer hypothetical questions, that is questions not about the present situation but about some imagined situation relating to it?
Student Knowledge
- Could the system give a report on the student's level of understanding?
- Are the system explanations tailored to the student?
- Does the system provide informative feedback?
- Are the problems presented by the system adapted to the student's needs, and level of understanding?

Student control
- Does the system actively engage the student?
- Can the student initiate some new area of investigation?
- Does the system monitor changes initiated by student, and comment on them if they seem to be unwise?
- Does the system intervene when the student appears to be having difficulty?

Mode of communication
- Can the student express his/her inputs to the system in whatever way is most natural?
- Does the system help if the student's input is not understandable to the system?
- Are the system's outputs natural?

Evaluation of ITS design

The following checklist of O'Shea et al. [1984], representing the 'pillars' of design, were used to evaluate FITS's design:
- Robustness;
- Helpfulness;
- Simplicity;
- Perspicuity;
- Power;
- Navigability;
- Consistency;
- Transparency;
- Flexibility;
- Redundancy;
Sensitivity;
- Omniscience; and
- Docility.

- **Evaluation of usability potential**

This entailed an investigation of the user interface features of the system.

2. **Internal evaluation criteria for PROUST study**

O’Neil & Baker [1987:17] evaluated the internal dimension of PROUST, in terms of the adequacy of the AI components, namely knowledge representation, instructional strategy, and student model. The following evaluation questions fell into the internal evaluation category:

- What is the underlying theoretical orientation of the system under evaluation?
- To what extent does the program serve as a model for ICAI?
- What instructional strategies and principles are incorporated into the program?
- To what extent does the system exhibit instructional content and features that are useful to the application for which it was designed?

Littman & Soloway [1988] conducted a second evaluation study of PROUST. They addressed the internal dimension by the following key evaluation questions, which examined the relationship between the architecture and behaviour of PROUST.

- **What does the intelligent tutoring system know?**
  
  This question is addressed by an analysis of what an ITS can possibly do, based on what it knows.

- **How does the intelligent tutoring system do what it does?**
  
  It calls for analysing the ITS to determine how the algorithms use available knowledge to produce the observed behaviour of the ITS.
• **What should the intelligent tutoring system do?**

  This question is relevant when a revision of the ITS is proposed in order to increase its teaching ability in one area or reduce it another. The question is answered by clarifying the areas of the tutoring domain for which the ITS is responsible.

3. **Internal evaluation criteria for Intuition study**

Siemer & Angelides [1998] like Littman & Soloway [1988], evaluate the internal dimension of INTUITION by examining the relationship between its architecture and behaviour. The same 3 key evaluation questions, as originally proposed by Littman & Soloway, are used in this evaluation, but are addressed by Siemer & Angelides in the following manner:

- **What does the intelligent tutoring system know?**

  This question is specifically addressed by an analysis of the system's domain, student and tutoring knowledge with respect to what the intelligent tutoring system can do based on the knowledge of its three available knowledge bases and process models.

- **How does the intelligent tutoring system do what it does?**

  This question is answered by analysing the intelligent tutoring system to determine how its processes generate the system's observed behaviour. The processes to be investigated include the system's expertise, diagnostics, and didactics, as well as the overall system control that directs the co-operation of the three knowledge and process models.

- **What should the intelligent tutoring system do?**

  This question is addressed by examining the overall capabilities of the system's teaching processes.

Figure 4.3 displayed on the following page illustrates the complete Siemer & Angelides model encompassing all the criteria used to examine the behaviour of the ITS.
Evaluation Requirements

Knowledge Level Analysis
What does the ITS know?

Program Process Analysis
How does the ITS do what it does?

Tutorial Domain Analysis
What should the ITS do?

Behavior to be examined

**Domain Model:**
What area is covered by the system’s domain knowledge? Can the system give an explanation of a problem solution?
Can the system give alternative explanations of the same concept?
Can the system answer arbitrary questions from the student?
Does the domain model include knowledge about common misconceptions and missing concepts?

**Tutoring Model:**
What are the system’s teaching goals?
Does the system provide alternative teaching strategies?
Are the system’s teaching strategies closely tailored to the student’s needs?
Can the student initiate some new area of investigation?
Does the system intervene if the user appears to be having difficulty?
Does the system actively engage the student?
Can the tutoring model relate a diagnosed error to a misconception or a missing concept?
Does the tutoring model incorporate remedial strategies in order to provide alternative remedial teaching styles?

**Student Model**
What information about the student’s knowledge and skills is stored in the student model?
What information about the student’s learning preferences is stored in the student model?
What information about the student’s past learning experiences is stored in the student model?
Does the student model store information about the student’s advancement stage?
Does the system monitor changes proposed by the student and comment on them if they seem to be unwise?

**Overall System Control**
Does the system provide helpful feedback on student input?
Does the system treat all detected errors?
How does the system respond if it cannot diagnose an error?
When does the system intervene to remediate a misconception or a missing concept?
Does the system attempt the remediation of a misconception or a missing concept when it recognises a student’s need for it?
Does the system adapt to the student’s advancement stage?
Does the system adapt to the needs and preferences of the student?

Figure 4.4 Internal evaluation of intelligent tutoring systems
Source: Siemer & Angelides [1998:91]
4. Internal evaluation criteria used in Mitrovic's SQL-Tutor study

The developer, Mitrovic investigated two variables of interest from the transformation of the system's computerised records, using data from the students' models. The first studied the relationship between the degree of mastery of a given constraint and the amount of practice on that constraint. The second investigated the rate of mastery of the target skill as a whole [Mitrovic & Ohlsson, 1999].

4.3.3.6 Evaluation criteria specific to ITSs

In comparing evaluation criteria of CAI and ITSs, it can be seen that both incorporate aspects such as didactics, interaction, tutoring capabilities, and subject-matter/domain expertise. They differ in that the additional 'intelligent' features of an ITS such as direct knowledge representation, diagnostic and modelling capability, as well as problem-solving expertise, are also subject to evaluation in the case of ITSs. The ability of an ITS to diagnose causes of student errors and successfully guide tutorial instruction represents a further important area of ITS investigation.

4.4 A proposed set of generic criteria for evaluating ITSs

A comprehensive set of generic criteria for the evaluation of ITSs is proposed and presented in tables 4.1 and 4.2. This set of criteria will be applied in Chapter Six for the evaluation of the SQL-Tutor.
Table 4.1 Criteria for internal evaluation of ITSs

<table>
<thead>
<tr>
<th>ITS internal features</th>
<th>Evaluation Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain Model</strong></td>
<td>What area is covered by the system’s domain knowledge? Can the system give an explanation of a problem solution? Can the system give alternative explanations of the same concept? Can the system answer arbitrary questions from the student? Does the domain model include knowledge about common misconceptions and missing concepts?</td>
</tr>
<tr>
<td><strong>Tutoring Model</strong></td>
<td>What are the system’s teaching goals? Does it provide alternative teaching strategies? Are the system’s teaching strategies closely tailored to the student’s needs? Can the student initiate some new area of investigation? Does the system intervene if the user appears to be having difficulty? Does the system actively engage the student? Can the tutoring model relate a diagnosed error to a misconception or a missing concept? Does the tutoring model incorporate remedial strategies in order to provide alternative remedial teaching styles?</td>
</tr>
<tr>
<td><strong>Student Model</strong></td>
<td>What information about the student’s knowledge and skills is stored in the student model? What information about the student’s learning preferences is stored in the student model? What information about the student’s past learning experiences is stored in the student model? Does the student model store information about the student’s level of advancement? Does the system monitor changes proposed by the student and comment on them if they seem to be unwise?</td>
</tr>
<tr>
<td><strong>Overall System Control</strong></td>
<td>Does the system provide helpful feedback on student input? Does the system treat all detected errors? How does the system respond if it cannot diagnose an error? When does the system intervene to remediate a misconception or a missing concept? Does the system attempt the remediation of a misconception or a missing concept when it recognises a student’s need for it? Does the system adapt to the student’s level of advancement? Does the system adapt to the needs and preferences of the student?</td>
</tr>
<tr>
<td><strong>Design Principles</strong></td>
<td>Robustness; helpfulness; simplicity; perspicuity; power; navigability; consistency; transparency; flexibility; redundancy; sensitivity; omniscience; docility.</td>
</tr>
</tbody>
</table>
Evaluation of Computer-Assisted Instruction and Intelligent Tutoring Systems: Characteristics and Criteria

<table>
<thead>
<tr>
<th>ITS internal features</th>
<th>Evaluation Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning and Instructional theories</td>
<td>Cognition; Constructivism; Creativity; Customisation; Component Display Theory; Collaboration.</td>
</tr>
<tr>
<td>Usability Principles &amp; Paradigms</td>
<td>User's memory should not be overloaded; System should be easy to use; Interaction dialogue should be natural, Dialogue should be task-oriented and adaptive; Effectiveness of screen design; Interaction styles (menus, windows, icons, typed input strings).</td>
</tr>
</tbody>
</table>


Table 4.2 Criteria for external evaluation of ITSs

<table>
<thead>
<tr>
<th>ITS educational impact</th>
<th>Evaluation criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Achievement</td>
<td>Improved academic performance;</td>
</tr>
<tr>
<td>Learning Affect</td>
<td>Satisfaction and motivation</td>
</tr>
</tbody>
</table>

Sources: Littman & Soloway [1988]; Siemer & Angelides [1998] and Mark & Greer [1993]

4.5 Conclusion

Evaluation criteria for computer-assisted instruction, as proposed by various researchers, are enumerated, and classified. Characteristics of, and evaluation criteria for, intelligent computer-assisted instruction and intelligent tutoring systems are then explored in an attempt to identify the distinctive features of ICAI/ITS and to compare the foci of the respective CAI and ICAI/ITS evaluation endeavours. The rationale for using such an approach is that CAI and ITSs are two different types of educational technology, the latter representing more sophisticated technology than the former. Hence the criteria for evaluating the two respective technologies should be compared to establish similarities and differences, and to demonstrate how advances in educational technology have contributed to and shaped the current state of ITS evaluations.

While the didactic, interactive, subject matter, and user-oriented aspects of evaluating CAI technology are incorporated within ITS evaluations under the behavioural properties of the various
architectural components (i.e. domain knowledge, student, tutoring, and overall system control) and the educational impact assessment, ITS evaluations also investigate the underlying ‘intelligent’ features. Hence, the architecture and distinctive features of ITSs largely determine the nature of ITS evaluations. As with CAI evaluations, the technical and usability aspects of ITSs are no longer at the forefront of evaluation studies. These aspects have been downplayed, since minimum acceptable standards are already assumed to be place for these technologies. The user-oriented evaluation perspective evident in CAI evaluation studies is maintained in ITS evaluations, since the educational impact of any learning and instructional technology must be assessed against the community for which it was designed and developed. In particular, the diagnostic and modelling aspects of the student model component have come under scrutiny accompanied by an investigation of its role in guiding instruction and remediation.

It is the researcher’s view that a complete evaluation of an intelligent tutoring system requires a close examination of the relationship between the architecture and behaviour of the ITS and its educational impact on learners. Moreover, since ITSs are embedded in other disciplines, namely cognitive psychology, computer science, and education and training, ITSs can, and should be, assessed with respect to their user-interface aspects, learning and instructional theories, and generic design principles.

The chapter concludes by proposing a set of generic criteria for the internal and external evaluation of ITSs and ICAI. These evaluation criteria are integrated within the proposed generic methodology for evaluating ITSs in Chapter Five, and are used in the actual evaluation of the SQL-Tutor described in Chapter Six.
CHAPTER FIVE

Evaluation of Intelligent Tutoring Systems:
Principles, Approaches, Methods and Framework

ITS research takes place in an environment that is characterised by great promise, complexity and uncertainty. Principled evaluations of ITSs are important, yet rare [Murray, 1993]. This chapter discusses principles, approaches, methods and a framework for the evaluation process.

5.1 Introduction

Although ITSs holds great potential for education, there have been few attempts to develop a comprehensive methodology for evaluating such systems that would result in greater diffusion of this educational technology. Furthermore, a conceptual framework for evaluating ITSs is even more rare despite its importance in helping educators and training agencies to make informed decisions about the added value of this technology. Siemer & Angelides [1998] emphasise the need for a conceptual framework to:

• Propel research developments;
• Answer questions pertaining to the utility of ITSs;
• Influence what, and how, students learn;
• Reveal the research value of an ITS; and
• Influence the choice as to whether or not one should use a particular ITS.

The previous chapter focussed on criteria for evaluating CAI and ITSs. This chapter provides an in-depth discussion on principles, approaches, methods, and techniques for the evaluation of ITSs. In addition, a conceptual framework for evaluating software technology is described with a view to adapting and applying this framework to the domain of ITSs. The ultimate goal is to develop an appropriate methodology encompassing evaluation requirements, principles, methods, and techniques within a framework appropriate for the evaluation of ITS applications. This methodology will be applied to the evaluation of an ITS for the database query language SQL, in Chapter Six.
5.2 Principles for evaluating ITSs

This section identifies and addresses seven principles that are relevant to the design of ITS evaluations, namely:

- Delineate the goals of the ITS;
- Define the goals of the evaluation study;
- Select an appropriate design to meet defined goals;
- Instantiate design with appropriate measures, number & type of subjects, and control groups;
- Make necessary logistical preparations for conducting the study;
- Pilot test the tutor and the study
- Make necessary logistical preparations for conducting the study

Principled evaluations of intelligent tutoring systems are necessary, in view of the failure of many evaluation studies due to poor experimental design, inadequately operationalised constructs and measures, or deficient logistical planning and implementation [Shute & Regian, 1993]. A theoretical context for each of the principles follows:

5.2.1 Principle 1: Delineate the goals of the ITS

According to Shute & Regian [1993] the designer of an evaluation should consider the following key issues relating to the goal of the tutor:

5.2.1.1 What instructional approach underlies the tutor?

This question covers the general and specific instructional approach of the tutor, and entails specific questions, such as:

- Is it a tutoring system with pedagogical intelligence, or is it a guided-practice environment, or a free-form discovery micro-world?
- Is the goal of the system to guide learning, to provide an environment for the induction of principles, or to permit students to practice skills?
- How does the tutor diagnose student performance and select instructional interventions?
- What information about the student is modelled and how is it employed?
5.2.1.2 What learning theory does it assume?

This asks the question: Is the instructional approach of the tutor based on one of the clearly defined and established instructional, didactic or learning theories?

5.2.1.3 What exactly does it teach?

This raises questions of the form: What are the desired learning outcomes in terms of specific, measurable knowledge and skills?

5.2.1.4 What other impacts is it expected to have?

The evaluation should also investigate other ways that the tutor may affect the student. In addition to the primary effects of treatment or intervention, are there any secondary effects such as:

- Transfer-of-skills to another domain;
- Improved self-esteem; and
- Modifying attitudes about computers.

5.2.1.5 In what context is it supposed to operate?

Relevant questions are:

- Is the system intended to supplement a lecture, or provide stand-alone instruction?
- Does it teach individuals or small groups?
- What prerequisite knowledge is assumed of the students?
- Will the tutor be used in an academic environment to support acquisition of declarative knowledge, or in a vocational training environment to facilitate acquisition of procedural skills?

Winne [1993] cautions the researcher to be aware of the larger context in which an ITS operates, where students learn by studying with an ITS and peers, a teacher/lecturer, and other resources (reference books). All of these together comprise a complete instructional system.
5.2.2 Principle 2: Define the goals of the evaluation study

The goal of assessing any technology is to provide a broader view of the utility and potential impact of a class of functionality for a specific set of potential users [Baker, 1991]. The goals of an evaluation study form the basis for selecting an experimental design that addresses the appropriate research goals for that study. The following questions are relevant:

5.2.2.1 What should be known after the study is completed?

Winne [1993] proposes the following criteria:

- **Qualities of the ITS as an engineered computing system**;
- **Capability of the ITS to promote students' achievements**, for example task transfer and learning transfer;
- **Power of the ITS** to adapt its instruction to individual students, based on data that it collects during tutorials, that is, its adaptivity;
- **Flexibility of the ITS** to be customized by those who manage students studying with an ITS, that is its adaptability; and
- **Effort required to change the subject matter** of an ITS to revise material or incorporate new information.

Shute & Regian [1993] suggest that student-achievement can be addressed by the questions:

- To what extent does the tutor affect students' capabilities to perform some task?
- In what way does the tutor influence learning in relation to classroom instruction on the same material?

Littman & Soloway [1988], Mark & Greer [1993] and Siemer & Anglides [1998] all agree that two major questions should form the focus of ITS evaluation, as outlined in Chapter Four, sections 4.3.3.2 and 4.3.3.4. These are:

- What is the educational impact of an ITS on students?
- What is the relationship between the architecture of an ITS and its behaviour?
5.2.2.2 By what standard will success be measured?

Shute & Regian [1993] suggest that the question ‘What exactly does the ITS teach?’ should be considered before ways can be identified to measure whatever is being taught, for example:

- What indices are to be used to assess the accuracy of knowledge, or the successful application of problem-solving skills?
- To whom would the students be compared on these measures?

The use of a set of subject-independent questions suggested by Self [1985] may be used as a standard to assess how the ITS lives up to its prefix of intelligent, as outlined in Chapter Four, section 4.3.3.5. O'Shea et al's [1984] thirteen pillars of ITS design may be used as a standard for assessing the design of the tutor, identified in section 4.3.3.5. Usability principles and paradigms can serve as standards to judge the usability properties of the ITS. Siemer & Angelides [1998] define the behavioural characteristics of a complete ITS (comprising a number of architectural components), in section 4.3.3.5, and suggest the use of such characteristics/properties as a standard by which the behaviour of an ITS can be assessed. The successful solution of problems by students would serve as a standard to assess learning achievement. The use of reaction assessment tools, like questionnaires and interviews, containing evaluation questions, can serve as a means to judge whether an ITS can behave in ways suggested by those questions, and whether it motivates and satisfies the students.

5.2.2.3 What are potential confounds, and which of these can be controlled?

Potential confounds are unwanted influences that can contaminate the results of a study. Potential confounds can be controlled if they are identified before conducting the study. An example of a confound cited by Duncan [1993] is that students become aware of different teaching methods and either become demoralised, engage in compensatory rivalry, or insist on equal treatment. Another confound is that the lecturer-cum-researcher is obliged to offer tutoring assistance to students having difficulty in learning a subject, irrespective of which group (experimental or control) they belong to.
5.2.2.4 Will quantitative indices, protocols, or observational data be used?

Data should be collected, so as to test severely or to negate expectations about:
- How a system functions; and
- The kinds or levels of outcomes that the system was designed to promote.

Since an evaluation study must lead to judgements of worth or quality, rating scales that describe order (first, second, clearer, more vague), dominance (like, dislike, prerequisite), or prevalence (more, less; usual) are commonly used in questionnaires to obtain qualitative measures. Such descriptors have an underlying rank or interval scale. A five point scale, like (strongly disagree, disagree, maybe, agree, strongly agree) allows a neutral answer [Melville, 1996]. While these measures are not precise, they are quantitative as well as qualitative. An evaluator may use observation to assess physical reactions, attitudes, rate of progress, requests for help, etc.

The use of on-line quantitative measures of performance may be used, such as a count of the number of times a constraint (correct use of a rule) relevant for a problem was satisfied in students’ solutions [Mirovic & Ohlsson, 1999]. Pre-tests and Post-tests serve as off-line quantitative measures. Qualitative measures may be obtained through the use of open-ended questions as used in interviews. All of these data types provide a means of discovering what a student is learning from the ITS.

5.2.3 Principle 3: Select an appropriate design to meet defined goals

An appropriate research design must be chosen to test the selected research questions. The question to be asked, is whether the study is undertaking a:
- A formative or summative type of evaluation, as defined in traditional educational evaluations. Formative evaluations, elaborated in section 5.3.1.1, are used during development of a system for the purposes of revision or improvement. Summative evaluations are conducted on existing systems for the purpose of a decision about acquisition of appropriate systems, and asks the question, ‘Does the intervention work, in terms of does the tutoring system produce desirable results on the part of the learners?’ They are discussed in more detail in section 5.3.1.2; or
• An internal and external type of evaluation as proposed by Littman and Soloway [1988] and Siemer & Angelides [1998] - see sections 4.3.3 and 5.3.2.

The design selected depends on the answers to questions 5.2.1.1 - 5.2.2.4 that underpin principles of 5.2.1 and 5.2.2 respectively. Shute & Regian [1993] define the following five different designs that are suitable for summative-type evaluation studies:

• Within-system design compares two or more alternative versions of a single tutor;
• Between-system design compares the tutor in relation to another one teaching the same subject matter;
• Benchmark design compares the tutor in relation to some standard instructional approach;
• Hybrid design represents a combination of the options; and
• Quasi-experimental design examines how your system operates in a real-world setting.

It should be noted that experimental research involves comparing something new (the experiment) with something standard (the control), and as such, involves experimental group(s) and a control group [Melville & Goddard, 1996].

Shute & Regian [1993] state that theory-driven laboratory evaluations with true experimental designs followed by data-constrained field evaluations are preferable. The laboratory offers a high level of experimental control (internal validity) difficult to achieve in a field study, while the field study offers a high level of external validity not possible in a laboratory.

Duncan [1993] is of the opinion that between-group design with the same course content serves as a better choice for efficacy studies than within-group designs, where it is difficult to equate subject matter and performance measures over different parts of the course.
5.2.4 Principle 4: Instantiate design with appropriate measures, number and type of subjects, and control groups

5.2.4.1 Learning outcomes (or dependent measures)

Shute & Regian [1993] outline typical measures used to assess understanding of course material as follows:

- Open-ended questions (example: ‘Why is the SQL language so popular?’)
- Survey (example: ‘How well do you feel you have learned SQL?’, followed by specific options)
- Concepts post-test
- Skills post-test
- Final exam

These measures collectively reflect the degree to which students learned the course material. The use of multiple dependent measures – investigating the same issues in multiple ways, called triangulation, is recommended since a researcher has the opportunity to capture on-line as much data as needed and hence is in a better position to measure indicators of performance. For example, achievement outcomes might be composed of a number of measures such as student problem solving, problem identification, efficiency, attitude, etc.

Siemer & Angelides [1998] and Mark & Greer [1993], suggest that both learning achievement and learning affect measures should be obtained in evaluative studies (see sections 4.3.3.2 and 4.3.3.3). Learning achievement includes aspects such as the acquisition and understanding of, and performance with, the student’s knowledge in a specific domain. A possible measure of learning achievement is the design of problems for students to solve.

The affect of the teaching process is concerned with aspects such as attitudes and emotions caused by the intelligent tutoring system. Motivation and active involvement in the learning process are important factors of learning. Asking the student to simply rate his/her agreement with specific issues, such as attitudes and activities, is a means of assessing motivation.
5.2.4.2 Independent measures (individual differences)

Individual learners come with differing profiles of knowledge, skills and traits (termed individual difference dimensions). A common individual difference measure is the standard aptitude test (SAT) scores. In addition, a researcher may wish to collect data on cognitive process measures (e.g. working memory capacity, information processing speed), personality measures (e.g. impulsiveness, aggregation, introversion) and demographic information (e.g. gender, age, years of school, experience with computers, etc.) [Shute & Regian, 1993].

Individual difference measures may be collected to investigate aptitude-treatment interactions in an experimental study.

O'Neil & Baker [1987] further motivate that the measurement of individual differences is relevant for the analysis and implementation of alternative student models and tutoring strategies.

5.2.4.3 Control conditions

Shute & Regian [1993] believe that the choice of treatment condition, in addition to proper control condition(s), must be principled, based on a theoretical approach to performance.

A potential problem that needs to be avoided is the creation of Hawthorne effects. These effects are treatment differences resulting from special attention and consideration being paid to only one group, namely, the group receiving instruction on the tutor.

Due consideration should also be given to the time factor. The two options that are available are:

- Allow time to vary across subjects, as is the case with self-paced and mastery learning;
- Hold time constant, allowing achievement to vary.
5.2.4.4 Subjects

Shute & Regian [1993] also present guidelines for the selection of subjects:

- Firstly, obtain the right type of subjects for the experiment. The researcher needs to identify the target population for whom the tutor is intended. The sample being tested should match the target population. If the control condition takes place in a group setting, then the treatment condition should similarly take place within a group setting.

- Secondly, obtain the right number of subjects for the study. As a rule-of-thumb, ITS evaluations should have at least 30 subjects per condition for simple treatment comparisons.

- Finally, randomly select subjects to conditions. If the subjects are not randomly assigned to treatment or control conditions, then any ensuing treatment effects may be attributed to a number of confounds, namely, self selection, site differences, experimenter bias, and so forth.

5.2.5 Principle 5: Make necessary logistical preparations for conducting the study

Careful planning, training and general preparation is needed for an evaluation study [Shute & Regian, 1993]. Qualified trained personnel are required to implement such a study and should be provided with clear instructions and procedural checklists. It would also be prudent to consider certain issues in advance, namely, possible worst-case scenarios, rescheduling of subjects in the event of computer-related problems, the use of computer analysts or technicians at data collection sites, and the availability of extra computers to serve as backup systems.

5.2.6 Principle 6: Pilot test the ITS and the study

Further aspects that need checking during a pilot test of an ITS include:

- Ensuring that the tutor is running bug-free;

- Knowing what the subjects should be doing at all times;

- Checking that the subjects are in fact learning something;

- Ascertaining whether the subjects like the system;

- Checking whether estimates of learning time are accurate; and

- Checking whether all subjects were able to complete the program.
5.2.7 Principle 7: Plan the primary data analysis when the study is planned

According to Shute & Regian [1993], each ITS evaluation study will have its own unique requirements for data analysis. They offer some general guidelines for common analyses as follows:

**Confirmatory data analyses**
When a specific hypothesis is to be tested in a study, a confirmatory data analysis technique is employed. Data collected during evaluation design could be analysed by t-tests, correlation, Chi-square, confirmatory factor analysis, or analysis of variance depending on the focus of the research question.

**Exploratory data analyses**
Exploratory data analysis is interactive and iterative with no fixed procedure to analyse the data. Exploratory analyses tend to suggest, rather than confirm, hypotheses. For studies that are exploratory in nature, as with pilot studies and formative within-system designs, exploratory techniques are appropriate. Relevant data analysis methods include exploratory factor analysis, cluster analysis, multiple regression analysis, and structural equation modelling.

**Cost-benefit analyses**
A researcher may be interested in ascertaining whether the development costs associated with implementing a tutoring system are justified in terms of increased learning outcomes. When performing cost-benefit analysis, the cost of the hardware and software should be included into the cost-benefit calculations.

5.2.8 Summary of principles

These principles are appropriate to apply to small, medium or large-scale evaluation studies, since they provide a comprehensive framework for planning, operationalising, and implementing experimental design. All of these principles, with the exception of the sixth principle, as outlined in section 5.2.6 ‘Pilot-test the ITS and the study’, will be practically applied in the evaluation of
in section 5.2.6 ‘Pilot-test the ITS and the study’, will be practically applied in the evaluation of the SQL-Tutor, an intelligent tutoring system for the database query language SQL, described in Chapter Six, section 6.3.1.2.

5.3 Approaches to evaluation of educational technology

A brief discussion follows on the major approaches to the evaluation of educational technologies.

5.3.1 Formative and summative evaluation

Traditional educational evaluation comprises two major categories, namely, formative evaluation and summative evaluation [Littman & Soloway, 1988].

5.3.1.1 Formative evaluation

Formative evaluation occurs during the design and development of instructional and learning products, and seeks information that focuses on the revision and improvement of the innovation, in order to help the developer. The formative evaluation approach is designed so that its principal outputs are identification of success and failure of segments, components, and details of programs, rather than an overall estimate of system success.

Data is systematically collected to permit the isolation of elements for improvement and ideally generate remedial options to assure that subsequent revisions make the system instructionally and motivationally stronger. Formative evaluation is therefore characterised as an iterative process of test-modify-retest-modify cycles [O’Neil & Baker, 1987; Kandaswamy, 1980].

5.3.1.2 Summative evaluation

As opposed to formative evaluation, summative evaluation occurs after design and development, and places emphasis on overall choices among systems or programs, based upon performance levels, time, and cost. This model of evaluation is essentially comparative and contrasts the innovation against other options [O’Neil & Baker, 1987].
Summative evaluation is usually carried out by organisations considering the acquisition of appropriate systems, and asks the question, ‘Does the intervention work, in terms of whether the tutoring system produces desirable results on the part of the learners?’ Implicit in that question is comparison to other alternatives, either current practice, or hypothetically, in terms of other ways resources could be used. A second summative evaluation question asks, ‘Is the system more cost effective than alternative methods of instruction?’ and collects data to decide whether or not to adopt an instructional package [Kandaswamy, 1980; O’Neil & Baker, 1987]. A weakness of summative evaluation is failing to identify what to do if a system or intervention is not an immediate, unqualified success.

5.3.2 Internal and external evaluation

A new approach to evaluation - namely introducing internal and external dimensions, was proposed as a way of evaluating emerging technologies such as ITSs. These approaches offer a somewhat different perspective from that of traditional educational evaluation, which uses either formative or summative evaluation. Littman & Soloway [1988] believe that ITS design and construction is still very much an art, and since there are few ITSs that may be deemed ‘finished products’, the pre-dominant concern should be with usefully guiding the development of systems rather than with determining whether they are effective educational end-products. Hence the two evaluation approaches are focused on the development of ITSs.

Mark & Greer [1993], hold a different viewpoint from Littman & Soloway[1988] and Siemer & Angelides[1998], in that they perceive formative evaluation as corresponding with internal evaluation and summative with external evaluation.

The internal and external dimensions of ITS evaluations are introduced in Chapter Four, section 4.3.3. The methods for conducting internal and external evaluations of ITSs follow in sections 5.4.2 and 5.4.3.

5.3.2.1 External evaluation

An evaluation approach that attempts to measure or assess an effect external to a tutoring system, namely the student’s learning, is appropriately termed external evaluation and addresses the
following evaluation question [Littman & Soloway, 1988; Siemer & Angelides, 1998]: ‘What is the educational impact of an ITS on students?’

5.3.2.2 Internal evaluation

An evaluation approach that analyses the relationship between the architecture of an ITS and its actual behaviour and is concerned with the inner workings of a tutoring system is termed internal evaluation. This approach addresses the following evaluation question [Littman & Soloway, 1988; Siemer & Angelides, 1998]: ‘What is the relationship between the architecture of an ITS and its behaviour?’.

5.4 Techniques and methods for evaluating ITSs

The general discussion of section 5.3 sets the background for a close examination of the various methods proposed by researchers for the specific evaluation of intelligent tutoring systems.

Mark & Greer [1993] draw upon the areas of intelligent tutoring systems research, expert systems design, computer-based instruction, education, and psychology to identify techniques for the formative and summative evaluation of ITSs.

5.4.1 Techniques for formative and summative evaluation of ITSs

Mark & Greer [1993] suggest that intelligent tutoring systems, because of their inherent complexity, may be evaluated in terms of a complete and total system, in terms of system components or in terms of specific features. The idea propounded is that techniques which are suited to evaluating complete ITSs, may not be suitable for the evaluation of ITS components or features (and vice versa). Hence the application of different evaluation techniques/methods are suggested for different purposes, at different times, with different components, or with overall systems.

The suitability of techniques for the formative and summative evaluation of an ITS, as well as for the evaluation of a complete ITS or components thereof, is presented in table 5.1
A brief discussion of these techniques/methods and their suitability for the formative and summative evaluation of ITSs follows.

### 5.4.1.1 Proofs of Correctness

Conventional computer programs may be verified and validated through formal proofs of correctness. However, this technique is not applicable to AI programs (including ITSs) that involve analytically intractable problems.
5.4.1.2 Criterion-based evaluation

Mark & Greer [1993] suggest that criterion-based evaluation is appropriate for formative development, where developers are concerned with general characteristics, or for evaluation of specific aspects of a system, like interface design where criteria can be specified and measured precisely.

5.4.1.3 Expert inspection and Turing test

Mark & Greer [1993] believe that while an expert's knowledge is often used as an explicit standard for judging a program, it has limited applicability for ITS evaluations, since ITS behaviour is complex and dynamic, and underlying representations may not be inspected or intuitively understood. Inspection is sometimes recommended for formative evaluation of ITS components.

The behaviour of human experts can also be used to evaluate knowledge-based systems by the use of the Turing test. This test is a technique for comparing human and computer behaviour, and a system is regarded as successful if its behaviour is indistinguishable from or superior to, the behaviour of human experts. A Turing test, however, is unlikely to offer conclusive proof of a system's overall merits or failings to the extent desirable for summative evaluation. It may be used in examining the behaviours of specific ITS components, as part of formative or summative evaluation.

5.4.1.4 Certification

Mark & Greer [1993] suggest the use of independent human teachers to appraise/certify ITSs, give feedback on their strengths and weaknesses (formative evaluation), and rate their adequacy (summative evaluation). Certification is recommended provided that questions about standards for judging programs, criteria for evaluating systems and components, and the accuracy with which humans can assess educational programs, are addressed.
5.4.1.5 Sensitivity Analysis

Mark & Greer [1993] recommend the use of sensitivity analysis to examine a component (for example, diagnostic and teaching) or system to see how responsive its behaviour is to differences in the information given to it. The sensitivity of an ITS to different learner characteristics, for example, may indicate whether additional teaching expertise is needed in the system. A system displaying similar responses to significantly different input might be regarded as less suitable than one displaying various responses.

5.4.1.6 Pilot Testing

Mark & Greer [1993], advise that pilot testing be carried out in conjunction with summative testing of completed systems, to determine whether systems are used as expected and that there are no unexpected outcomes.

Three types of pilot testing are available, namely one-to-one testing, small-group testing, and field-testing. In one-to-one testing, detailed observations of how a student interacts with the developed instructional materials are conducted to identify indistinct directions, questions and information, and note unexpected features of the instructional situation. This type of testing is usually carried out early in the development life cycle to minimise inappropriate development. Small-group testing is generally carried out later in the development process, once the format of the program and its content are stable. A small sample of students, representative of the target population, is questioned before and after system use to assess whether learning has occurred, and to ensure students understand how to use the system. Field-testing examines system use in actual instructional settings with real instructors and students to determine whether students' display expected behaviour and learning outcomes, and to identify possible unanticipated outcomes.

5.4.1.7 Experimental Research

Mark & Greer [1993] propose that experimental research is suited to ITSs because it enables researchers to examine relationships between teaching interventions and student-related teaching
outcomes, and to obtain quantitative measures of the significance of such relationships. Experimental techniques are often used for summative research, where formal power is desired and where overall conclusions, rather than acquisition of information, are sought.

5.4.1.8 Objections to the use of different techniques for formative and summative evaluation

Other researchers have opposing views to that of Mark & Greer [1993] in respect of the use of different techniques for formative and summative evaluation. Kandaswamy [1980] is of the opinion that the same techniques can, and should be, used in both evaluation processes, since the major difference between formative and summative evaluation is in the use of the evaluative data. O’Neil & Baker [1987] propound the view that two approaches should share some common procedures and criterion measures, since they differ only in purpose and client as well as in the types of data (cost for summative, componential analysis for formative) appropriate for evaluation. Winne [1993] reiterates that formative and summative evaluation differ in their function. The former serves improvement and the latter serves decision-makers.

5.4.2 Methods for external evaluation of ITSs

Methods for external evaluation of ITSs are discussed from a cognitive perspective and from the perspective of learning achievement and learning affect. Cognition refers to the strategies, processes, and techniques used by the student while engaged in learning and problem-solving activities.

5.4.2.1 External evaluation method: the cognitive perspective

Littman & Soloway [1988] define a class of evaluation methods based on progress in student modelling, to allow researchers to determine how the behaviour of an ITS affects students and transforms their knowledge and problem-solving skills. Methods in this category are used for external evaluation and are presented from a cognitive perspective. The goal of external evaluation from a cognitive perspective is to ascertain how well an ITS teaches students the knowledge and skills that support the cognitive processes required for problem solving.
Through interaction with a student and the application of diagnostic procedures, an ITS builds up an understanding of the student’s knowledge and skills, which it uses to interpret the student’s behaviour and to guide the system’s teaching processes. A common term used to depict an ITS’s understanding of the student is ‘student model’, as defined in Chapters Two and Four, sections 2.2.1.1 and 4.3.2. The designer must provide the ITS with methods for reasoning about students’ problem-solving strategies in the intelligent tutoring system’s domain of instruction.

The two major types of student modelling are those that are based on process models of problem solving and those that are not. Student modelling techniques based on process models solve problems in a human-like way. However, student models, irrespective of technique, can be used to assess how well an ITS teaches students skills and knowledge for solving problems, thereby serving as a valuable tool in the evaluation of the tutor itself. Student modelling techniques can guide the construction/selection of new problems for testing, as well as identify problems that the student should be able to solve, since they capture how students solve problems, and not merely that they can solve problems. Process-based techniques, in particular, can be used to predict the actual process the student will go through to solve problems. From the foregoing discussion, it can be seen that the external evaluation of ITSs can be substantially different from the evaluation of CAI, in that the latter focuses mainly on correct and incorrect answers, while the former assesses the reasons why students give correct and incorrect answers.

Littman & Soloway [1988] applied this method to the evaluation of PROUST (an ITS for Pascal) and concluded that techniques using student-modelling alone cannot determine decisively whether or not an ITS is effective. They suggest that external evaluation should be performed, by using the student-modelling capability of intelligent tutoring systems to construct/select a set of problems to test the student. The success rate of the student can be used as a measure of the student’s learning achievement. This success, in turn, serves as an indicator that the underlying processes or knowledge and skills have been successfully taught by the system. An added benefit is that the result of such a test would indicate not only whether the student is able to solve certain problems, but could also furnish reasons why the student was able or unable to solve the problems.

Possible objections to the cognitive perspective as raised by Littman & Soloway [1988] are:
1. It is a difficult process involving detailed evaluations of students’ cognitive processes;
2. The use of a single student model in evaluating student knowledge may be insufficient and may lead to incorrect conclusions being drawn; and
3. Students' internal representations are not relevant to education, in the view of micro-world proponents, who argue that an educator should not so much teach as provide tools that make it possible to learn. This view implies that student models are superfluous, if not counterproductive.

5.4.2.2 External evaluation method: the learning achievement and learning affect perspective

The goal of external evaluation, from another perspective, is to reach an overall estimate about the system's ability to foster learning, referred to as learning achievement, and to motivate and satisfy the student, described as the learning affect [Mark & Greer, 1993; Siemer & Angelides, 1998]. This perspective relates to the educational impact of the ITS by measuring actual learning of the subject matter and its affective value, using some form of assessment mechanism, rather than examining the students' cognitive processes used in problem-solving.

Mark & Greer [1993] recommend pilot testing for obtaining qualitative or quantitative assessments of educational effects during formative evaluation, and experimental research for making summative claims about the educational effects of an ITS.

Siemer & Angelides [1998] approach external evaluation of ITSs from the perspectives of learning achievement and learning effect, which address the educational impact of ITSs on students. This approach is illustrated in Figure 5.1.

Figure 5.1 External evaluation of an ITS
Source: Siemer & Angelides [1998:90]
Siemer & Angelides [1998] suggest that the external evaluation of an intelligent tutoring system should involve a planned experiment. This experiment should be designed to assess whether the implementation of a system or part thereof has been successful. It must be determined whether the students accept the system and perceive its behaviour in the intended way. Reaction assessment tools namely, interviews and questionnaires, are suggested for measuring the learning affect of intelligent tutoring systems. For example, experimental evaluation may involve interviewing students in the experimental group or requiring them to fill in a questionnaire. The assessment of learning achievement would require the design of tasks/problems for the students to perform/solve. Successful completion of tasks and/or problem solutions would be an indication of learning achievement.

5.4.3 Methods for internal evaluation of ITSs

Methods for internal evaluation of ITSs are discussed from an architectural perspective. Sections 5.4.3.1 and 5.4.3.2 demonstrate the respective use of knowledge engineering and formative evaluation techniques for the internal evaluation of ITSs.

5.4.3.1 Internal evaluation method: knowledge engineering techniques

Littman & Soloway[1988] suggest a set of evaluation methods, adapted from knowledge engineering techniques developed for AI. These methods identify and analyse the relationship between the architecture of an ITS and its actual behaviour, and form the basis of internal evaluation. They can help to characterise all components of an ITS, including the student model, the curriculum content, the instructional component, the expert problem-solver, and the interface.

In addition, the methods suggested are useful for addressing the three principal questions central to internal evaluation of ITSs, as discussed in section 4.3.3.5. A description of each follows:

1. **Knowledge level analysis** attempts to characterise the knowledge in the ITS and thus answers the first question, *What does the ITS know?* In essence, knowledge level analysis focuses on whether the program has enough of the right kinds of knowledge to meet the requirements that were set for it.
2. **Program process analysis** uses focused simulations of the ITS to answer the second question, *How does the ITS do what it does?* This method throws light on whether the ITS does what it does in the way the designers intended. Program process analysis examines how a program uses its knowledge in the process of going from input to output.

3. **Tutorial domain analysis** answers the question, *What should the intelligent tutoring system do?* This method represents a reasoned approach to adding new tutorial capabilities to an ITS. Initial descriptions of the domain to be tutored serve as part of the design specification for an ITS. Aspects of the domain that the designers wish to tutor may shift as the program takes shape and as the process of tutoring in the domain becomes better understood. The ongoing evaluation of the appropriate domain of tutoring can help maintain a clear view of the goals of the ITS.

Siemer & Angelides [1998] approach internal evaluation of ITSs from an architectural perspective, which examines the relationship between the system’s architecture and the system’s behaviour. This approach is illustrated in figure 5.2. It should be noted that figure 5.2 is related to figure 5.1, in that the output of internal evaluation namely, system’s behaviour serves as input to external evaluation.

![Figure 5.2 Internal evaluation of an ITS](source: Siemer & Angelides [1998:90])

To clarify the relationship between the three main components of the architecture as discussed in Chapter Two, section 2.4.1.3, and the behaviour of an ITS, the same three questions as suggested by Littman & Soloway [1988] are supported by Siemer & Angelides [1998] for the internal evaluation of an ITS. They elaborate on the three respective knowledge engineering techniques as follows:

1. **Knowledge level analysis** has to address issues such as the scope of the system’s domain, student and tutoring knowledge and whether the knowledge representation is appropriate.
2. **Program process analysis** investigates the following:

- *Expertise*, i.e. the way domain knowledge is used and manipulated;
- *Diagnostics*, i.e. the procedures used by the system to analyse the input of the student to maintain the student model; and
- *Didactics*, i.e. the way teaching goals are determined and teaching strategies are used to guide instruction.

3. **Tutorial domain analysis** investigates any lack of tutorial abilities in any of the three standard knowledge components of the intelligent tutoring system.

The results of these three analyses explain whether, and how, all the knowledge and process models (domain model, student model, tutoring model, and overall system control) of an ITS architecture, account for the system’s desirable behaviour. In order to undertake such an investigation, the behaviour provided by the architecture of an ITS would have to be defined. Siemer & Angelides [1998] define the desirable behavioural properties of an intelligent tutoring system by establishing a set of evaluation questions, where each evaluation question addresses a criterion associated with a particular behavioural property. The evaluation questions are founded on Self’s [1985] proposed set of subject-independent questions outlined in section 4.3.3.5, and are relevant to the notion of a complete intelligent tutoring system, as defined in section 2.4.1.3. They make general inquiries about the properties of an ITS, so that all positive and negative features addressed by the questions may be unveiled. The use of evaluation questions is referred to as *criterion-based evaluation*. A complete model of the internal evaluation as proposed by Siemer & Angelides [1998] is presented in Chapter Four, section 4.3.3.5, Figure 4.4.

In short, internal evaluation is conducted to determine whether the system can behave in the way suggested by the evaluation questions within the three categories. It should be noted that the same question could pertain to knowledge level analysis and program process analysis.
5.4.3.2 Internal evaluation method: formative evaluation techniques

Mark & Greer [1993], on the other hand, take the view that different techniques are appropriate for the evaluation of the different components of an ITS. The evaluation criteria for these components are discussed in Chapter Four, section 4.3.3.4. The suitability of formative evaluation techniques (identified in section 5.4.1) for the different architectural components are presented as follows:

1. Domain knowledge

Appropriate formative techniques for evaluation of domain knowledge are expert inspection and the Turing test.

2. Teaching knowledge component

Formative evaluation techniques of sensitivity analysis and certification are recommended for investigating the teaching knowledge component. The behaviour of the teaching knowledge component is dependent upon the student and the domain knowledge maintained in the system.

3. Student knowledge

The use of experimental research is suggested for assessing the validity and reliability aspects of student modelling. Assessing the validity of the system's diagnostic capability may be achieved by comparing diagnostic information obtained by the system to diagnostic information obtained independently and checking for consistency of results.

4. Communications component

Guidelines, checklists, and reviews may be used in guiding development of the systems interface. Pilot testing with members of the target population is more likely to unveil problems and concerns for the formative evaluation of an interface. Another approach recommended for formative or summative evaluation of interfaces is to compare different versions of an interface experimentally to see which is more effective.
5. Learning component

Criterion-based assessment may be used to assess whether the learning component improved systems behaviour.

6. System control

Conventional techniques for computer performance evaluation may be appropriate for the evaluation of the control component, since it is concerned with the underlying features of system behaviour.

5.4.3.3 Comparison of techniques for internal evaluation method

Both Littman & Soloway [1988] and Siemer & Angelides [1998] use knowledge engineering techniques, namely, knowledge level analysis, program process analysis and tutorial domain analysis for performing internal evaluation. However, Littman & Soloway conduct knowledge level analysis by examining the knowledge contained in the ITS, deriving explicit descriptions of the types of knowledge, and discovering weaknesses in its representation. Furthermore, they undertake extensive process tracing to ascertain whether the tutor can completely understand and correctly diagnose student errors. Finally, they undertake tutorial domain analysis by observing the ITS in action and discovering how they could extend the existing capabilities of the ITS. Siemer & Angelides, on the other hand, propose the use of criterion-based techniques for all three levels of analyses for internal evaluation.

Mark's & Greer's [1993] approaches do not use knowledge engineering techniques, but rather use different formative and summative evaluation techniques for evaluating the various architectural components and the system as a whole.

All six researchers agree that learning achievement and learning affect should be the focus for external evaluations of ITSs, irrespective of the techniques or methods (planned experiment, successful problem-solving or task-completion) used for measuring their impact on students.
5.5 The Brown and Wallnau framework for evaluating software technology

This section discusses a framework for assessing software technology in a particular application domain.

Brown & Wallnau [1996] developed an experimental framework for the evaluation of component integration technologies. It incorporates concepts that can be transferred to other domains, and for the purposes of this study, it will be adapted in section 5.6.3, and applied to the domain of ITSs, in particular to the evaluation of the SQL-Tutor.

A brief description follows on the framework's principles and techniques in their original form:
1. It is based on the premise that a technology's potential impact is best understood in terms of its feature delta, that is, features that differentiate it from other technologies.
2. A technology's distinctive features must be described, then evaluated in a context of usage.
   With the framework, an evaluator of the technology forms hypotheses about how a feature delta supports a defined context of use and employs rigorous experimental techniques to confirm or refute these hypotheses.
3. It makes use of descriptive modelling for analysing the interdependencies between technologies, as well as between technologies and their contexts of use.
4. Finally, information is sought on the added value of the technology in hand.

5.5.1 Descriptive modelling phase

This phase relates to the first principle of the framework that a technology's potential impact is best understood in terms of its feature deltas. During this phase, assumptions are made concerning features of interest and their relationship to contexts of usage. The descriptive models provide an approach for describing technologies, identifying feature deltas, that is, key features needed to distinguish technologies, and documenting the evaluation process itself. The descriptive modelling phase addresses discovery of the technology's features and predicts their impact through the development of technology genealogies (ancestry of the technology) and problem habitats (uses of a technology, and its competitors). The technology genealogy reflects that new technologies are often minor improvements of existing technologies. In order to
understand a technology's features, it is necessary to understand that technology's historical and technological predecessors. The added value of a technology requires an understanding of how the features identified will be used and the benefits of these features.

Genealogies and habitats may be modelled as semantic networks where the nodes represent concepts, and the links represent relationships among concepts.

The output of the descriptive modelling phase is a situated technology where models describe how a technology is related to other technologies, in terms of its feature deltas, and the context of usage in which it can be evaluated.

5.5.2 Experiment design phase

This phase relates to the second and third principles of the framework pertaining to the use of hypothesis formulation, experimental techniques, and descriptive modelling for analysing the interdependencies between technologies and between technologies and their contexts of use. This phase is a planning activity comprising three activities, namely comparative feature analysis (referred to as 'comparative anatomy' in framework), hypothesis formulation and experiment design.

Comparative feature analysis involves a more detailed investigation of feature deltas using empirical techniques rather than the descriptive techniques of the earlier phase. The techniques that may be used for evaluation are:

- Reference model benchmarking for qualitative feature descriptions using an internal structure or feature lists for the technology; and
- Feature benchmarking for quantitative feature descriptions.

A reference model is an annotated feature list. Feature benchmarks quantitatively measure features in terms that make sense for the feature and the technology.

Hypothesis formulation on how a feature delta supports a defined context of usage requires that:

- The hypothesis is refutable from experimental evidence;
- Suitable experimental techniques exist to test it; and
• The set of hypotheses is sufficient for evaluating added value.

The output of the experiment design phase is:
• A set of hypotheses about the added value of a technology that can be substantiated or refuted through experimentally acquired evidence; and
• A set of defined experiments that can generate this evidence and that are comprehensive to support conclusions in respect of added value.

5.5.3 Experiment evaluation phase

This phase relates to the fourth principle of the framework. During this phase evaluators carry out experiments, gather and analyse experimental evidence and confirm or refute hypotheses of added value.

The technology delta framework is illustrated in figure 5.3.
5.6 **A proposed generic methodology for evaluating ITSs**

This section addresses the main goal of the study (as outlined in Chapter One, section 1.2), which is the development of a generic methodology for the evaluation of ITSs. The proposed methodology will be applied to the evaluation of the SQL-Tutor in Chapter Six. The proposed set of generic criteria for evaluating ITSs, identified in section 4.4, is integrated within this methodology.

Table 5.2, which appears below, identifies the underlying theory, and the main function(s) and implementation technique(s) associated with each of the issues pertaining to the proposed methodology.
### Table 5.2 Proposed methodology for evaluating ITSs

<table>
<thead>
<tr>
<th>Evaluation issues</th>
<th>Underlying theory</th>
<th>Function (s) and/ or implementation technique(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principles for evaluating ITSs</td>
<td>Section 5.2</td>
<td>Planning, operationalisation and implementation of experimental design.</td>
</tr>
<tr>
<td>Evaluation approaches:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal evaluation</td>
<td>Section 4.4</td>
<td>Ways of evaluating educational technologies.</td>
</tr>
</tbody>
</table>
|                                         |                   | * Analyses relationship between architecture of ITS & its behaviour;*  
|                                         |                   | * Analyse other internal aspects:*  
|                                         |                   | * Design principles*  
|                                         |                   | * Learning & instructional theories*  
|                                         |                   | * Usability principles & paradigms.*  
<p>|                                         |                   | Assesses an effect external to a tutoring system, viz. students' learning.                                                                                                                                                                |
| External evaluation                     | Section 4.4       |                                                                                                                                                                                                                                               |
| Techniques &amp; methods for evaluating ITSs| Section 5.4       | Examine existing techniques &amp; methods for evaluating ITSs.                                                                                                                                                                                      |
| Methods for external evaluation:        | Section 5.4.2     | Examine existing methods for external evaluation of ITSs.                                                                                                                                                                                       |
| The perspective of learning achievement &amp; learning affect | Section 5.4.2.2   | Estimate system's ability to foster learning (learning achievement) and to motivate and satisfy the student (learning affect).                                                                                                                                 |
|                                         |                   | Implementation technique to be used for learning achievement: experimental design.                                                                                                                                                               |
|                                         |                   | Implementation technique to be used for learning affect: reaction assessment tools such as observation, interviews and questionnaires.                                                                                                                  |
| Methods for internal evaluation:        | Section 5.4.3     | Examine techniques &amp; methods for internal evaluation of ITSs                                                                                                                                                                                    |
| Knowledge engineering techniques        | Section 5.4.3.1   | Knowledge level analysis: Analyse scope of domain, student &amp; tutoring knowledge of system.                                                                                                                                                      |</p>
<table>
<thead>
<tr>
<th>Evaluation issues</th>
<th>Underlying theory</th>
<th>Function(s) and/or implementation technique(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation issues</td>
<td>Underlying theory</td>
<td>Program process analysis: Investigate expertise, diagnostics and didactics of system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tutorial domain analysis: Investigate lack of tutorial abilities in any of the 3 knowledge components of the ITS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Criterion-based evaluation technique to be used for all 3 levels of analyses to determine whether the system can behave in ways suggested by evaluation requirements and criteria.</td>
</tr>
</tbody>
</table>

**Adapted Framework for evaluating ITSs**

Three phases of adapted framework:

1. Descriptive modelling
   - Section 5.6.3
   - Identify features of ITS technology that differentiate it from other technologies;
   - Identify context of usage in which it will be evaluated.

2. Evaluation design phase
   - Section 5.6.3
   - Planning activities comprising reference modelling, hypothesis formulation and experimental design.
   - Reference models provide evaluation questions and criteria for detailed feature analysis in context of usage.
   - Experimental design planned in line with principles for evaluating ITSSs in section 5.2.

3. Evaluation implementation phase
   - Section 5.6.3
   - Carry out experiment, gather and analyse experimental evidence and confirm/refute hypotheses.
   - Experimental techniques to be used for experiment are:
     1. Hybrid design incorporating elements of between-system design and benchmark design.
     2. Query techniques (observation, interviews & questionnaires).
A discussion follows on the various evaluation issues outlined in table 5.2, with a view to justifying choices made in respect of approaches, techniques and methods, and framework for the proposed methodology.

All seven principles for the evaluation of ITSs as discussed in section 5.2 are relevant in terms of the proposed methodology, and are recommended for practical application to the evaluation of ITSs.

The approach of internal and external evaluation is adopted for the generic evaluation methodology, for the following reasons:

- Conventional formative and summative evaluations while useful and appropriate for traditional educational evaluation (for example, instructional materials, CAI/CAL packages), present problems in fields such as ICAI or ITSs;
- Many of the intelligent tutoring systems were, and continue to be, developed as research projects testing the limits of computer science and exploring the use of AI processes, and as such they cannot be regarded as finished products;
- Intelligent tutoring systems remain embedded in research and development, where new techniques for student modelling are continually being employed and tested to assess their effectiveness in providing individualised instruction; and
- Evaluation of ITSs should inform developers on constructive ways to improve these systems, so as to serve the purpose of providing more effective personalised instruction.

The internal evaluation of an ITS can, and should be, extended beyond an examination of the relationship between the its architecture and its behaviour, in order to investigate other internal aspects, such as the design principles, learning and instructional theories, and usability principles and paradigms of the tutor.

The evaluation method recommended for adoption within the methodology is both generic and comprehensive. The reasons substantiating this choice are:

1. The method addresses all aspects of an ITS, in that it incorporates and goes beyond Self’s set of subject-independent questions by examining the system’s ability to diagnose student errors and provide remedial tutoring.
2. It assesses a system’s ability to apply other tutoring strategies during a tutoring session.
3. This evaluation method, unlike others, is not closely tailored to assess the specific features of the system being evaluated. Instead it is generic and may be applied to the evaluation of any intelligent tutoring system.

4. The method is comprehensive in that it encompasses both internal and external evaluation.

5.6.1 Techniques for external evaluation of ITSs

The use of experimental research is recommended for the external evaluation of ITSs to assess learning achievement. Certain ITSs have a problem-generation capability that offers a testing mechanism based on their student models whereby a student’s success rate or lack thereof is used as a measure of learning achievement. However, since not all ITSs are characterised by such a feature, experimental research is a practical and viable alternative to investigate learning achievement.

Observational techniques and query techniques (questionnaires and interviews) are recommended for assessing learning affect, since they are well-recognised and commonly used for such measurements.

5.6.2 Techniques for internal evaluation of ITSs

The recommendation of a criterion-based technique for the internal evaluation of ITSs is justifiable for the following reasons:

1. Independent third party evaluators cannot derive an explicit description of the knowledge base other than the high-level description and examples provided by the developer.

2. Extensive model tracing is not desirable for all types of ITSs, since not all student models are process-oriented.

3. Third-party evaluators would not have ready access to the code of the system, thereby making it difficult to undertake certain types of evaluation involving, for example, the transformation of computerised records to ascertain the learning of students via their interactions with the tutoring system.

4. The use of interviews and questionnaires covering evaluation criteria, combined with a user-centred approach to evaluation, as well as observation feedback mechanisms, represents a simple, yet powerful, way to learn about the behaviour of the system and its learning affect.
5. An important spin-off of criterion-based internal evaluation is that it provides valuable suggestions for the improvement of specific internal aspects, as well as for the overall improvement of the behaviour and architecture of the ITS under investigation.

5.6.3 Proposed framework for evaluating ITSs

Brown & Wallnau’s [1996] framework for evaluating software technology introduced in section 5.5, may be adapted and applied to the domain of ITSs. The three phases, namely, descriptive modelling, experiment design and experiment evaluation accommodate the approaches of internal and external evaluation, and can be respectively renamed:

1. Descriptive modelling phase
2. Evaluation design phase
3. Evaluation implementation phase

Methods and techniques that are suitable for the evaluation of ITSs may be used within the framework.

The adapted framework is displayed in figure 5.4, and is used for evaluating Mitrovic’s SQL-Tutor [1998] in Chapter Six. Figure 5.4 is an adaptation of figure 5.3.
It is recommended that the reference models created during the evaluation design phase contain a set of evaluation questions and evaluation criteria, to provide a detailed 'feature analysis' that can be evaluated in its context of usage. The use of empirical techniques to perform a qualitative feature analysis is not recommended, as there are few, if any, competing technologies for a particular application domain. Furthermore, feature benchmarks, using quantitative measurements are not appropriate for the ITS technology and as such, are not recommended.

The experimental techniques suggested by Brown & Wallnau [1996] for the experiment evaluation phase, namely the modelling of problems, compatibility studies, demonstrator studies and synthetic benchmarks are not appropriate for the domain of intelligent tutoring systems, nor are they in keeping with the goals of evaluating ITSs. Experimental designs such as within-system design, between-system design, benchmark design, hybrid design and quasi-experimental design as suggested by Shute & Regian [1993] are directly relevant to the evaluation of intelligent tutoring systems and are recommended.
This framework was chosen because it is compatible with internal and external approaches to evaluation, and permits the use of criterion-based and experimental research techniques that are appropriate for the evaluation of ITSs. It is based on sound principles that are applicable to the evaluation of ITSs, allowing:

- The potential impact of the ITS technology to be assessed in a real world usage context;
- The ITS technology to be understood and evaluated in terms of its feature delta (internal evaluation);
- Hypotheses to be formulated and experiments to be conducted for external evaluation of ITSs; and
- Educators and professionals to make decisions about the added value of the ITS technology.

5.7 Conclusion

This chapter identified principles relevant to the design of ITS evaluations, before addressing general approaches to the evaluation of educational technologies, namely: formative and summative evaluation, then internal and external evaluation. With this background, various techniques and methods were examined for undertaking formative and summative as well as internal and external evaluations, specific to the field of intelligent tutoring systems. A conceptual framework for evaluating software technology was described, with a view to adapting it for the domain of ITSs.

The chapter culminates by proposing a comprehensive generic methodology for the evaluation of intelligent tutoring systems.

The conventional use of formative and summative evaluation is not considered appropriate for ITSs. Instead the categorisation of evaluation into internal and external strategies appears to be applicable to many of the evaluations reported by designers of ITSs.

The methodology chosen for the current evaluation study is comprehensive, in that it examines behavioural properties, design principles, learning and instructional theories, usability properties and the educational impact of the ITSs.
The framework adopted for ITSs proposes distinct phases within an evaluation process and is integrated with selected evaluation methods. The methodology offers a two-prong approach, aimed at evaluating both the features and behavioural properties of the tutor, as well as the impact and added value of the technology. While the framework allows for direct comparison of alternative technologies and for compatibility studies between technologies, this is generally not the focus of ITS evaluation studies.

The proposed methodology is applied in Chapter Six to an actual evaluation – the evaluation of the SQL-Tutor introduced in Chapter Three.
CHAPTER SIX

Evaluation of the SQL-Tutor: An Intelligent Tutoring System for the database query language SQL

This chapter describes a comprehensive practical application of the proposed generic methodology for the evaluation of intelligent tutoring systems discussed in Chapter Five, to the evaluation of the SQL-Tutor. The chapter also provides detailed analyses of the results of student feedback to the evaluation instruments used, and discloses the findings of the experiment conducted.

6.1 Introduction

This chapter applies the generic methodology for the evaluation of ITSs (as discussed in Chapter Five, section 5.6) to the evaluation of the SQL-Tutor. The proposed methodology incorporates a set of generic evaluation criteria (outlined in Chapter Four, section 4.4). The chapter is structured into three major sections corresponding to the phases of the evaluation framework, namely descriptive modelling, evaluation design and evaluation implementation. Each phase is distinguished by a set of pre-defined activities and associated outputs. The outputs from previous phases serve as input into successive phases.

Table 6.1 indicates the structure of the evaluation. Each issue is practically applied in the evaluation of the SQL-Tutor within a particular phase of the framework and documented in a particular section of this chapter.

Table 6.1 Links between evaluation issues, theory and application

<table>
<thead>
<tr>
<th>Evaluation Issues</th>
<th>Underlying Theory</th>
<th>Practical Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>External evaluation criteria</td>
<td>Sections 4.3.3.2, 4.3.3.3 &amp; 4.4</td>
<td>To be applied within: Section 6.2.2 (Descriptive Modelling phase) Learning Achievement &amp; Learning Affect. Section 6.3.1 (Evaluation design phase: external evaluation criteria) Experimental design.</td>
</tr>
<tr>
<td>Evaluation Issues</td>
<td>Underlying Theory</td>
<td>Practical Application</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Internal evaluation criteria</td>
<td>Sections 4.3.3.4, 4.3.3.5 &amp; 4.4</td>
<td>To be applied within: Sections 6.2 (Descriptive Modelling phase)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 6.3.2 (Evaluation design phase : internal evaluation criteria)</td>
</tr>
<tr>
<td>Evaluation Approaches: internal</td>
<td>Section 5.3.2</td>
<td>Section 6.3 (Evaluation design phase) &amp; Section 6.4 (Evaluation implementation phase)</td>
</tr>
<tr>
<td>&amp; external</td>
<td></td>
<td></td>
</tr>
<tr>
<td>External evaluation methods</td>
<td>Sections 5.4.2 &amp; 5.6.1</td>
<td>Section 6.4.1 (Evaluation implementation phase: External evaluation methods)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Analysis of experimental evidence to confirm/refute hypotheses identified in section 6.3.1.</td>
</tr>
<tr>
<td>Internal evaluation methods</td>
<td>Sections 5.4.3 &amp; 5.6.2</td>
<td>Section 6.4.2 (Evaluation implementation phase: Internal evaluation methods)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Analysis of evaluation criteria identified in section 6.3.2.</td>
</tr>
<tr>
<td>Principles for evaluating ITSs</td>
<td>Section 5.2</td>
<td>Section 6.3.1.2 (Evaluation design phase)</td>
</tr>
<tr>
<td>Adapted evaluation framework:</td>
<td>Section 5.6</td>
<td></td>
</tr>
<tr>
<td>Descriptive modelling phase</td>
<td>Section 5.6.3</td>
<td>Section 6.2</td>
</tr>
<tr>
<td>Evaluation design phase</td>
<td></td>
<td>Section 6.3</td>
</tr>
<tr>
<td>Evaluation implementation phase</td>
<td></td>
<td>Section 6.4</td>
</tr>
</tbody>
</table>

### 6.2 Descriptive modelling phase

The descriptive modelling phase of the evaluation framework, as applied to the SQL-Tutor, focuses on its distinctive features and internal structure (outlined in Chapter Five, sections 5.5 & 5.6), as well as its context of usage. The output of this phase is a situated technology model that describes how the SQL-Tutor is related to other educational technologies and the specific context of usage in which it can be evaluated. It should be noted that these models are guides for structuring evaluation experiments and interpreting the results of experiments [Brown & Wallnau, 1996]. Two descriptive models, namely the ITS/SQL-Tutor genealogy and the elements of the ITS/SQL-Tutor habitat follow.
6.2.1 The ITS/SQL-Tutor genealogy

The reader is referred to Chapter Three, section 3.4, figure 3.1, for an illustration of the architecture of the SQL-Tutor. Figure 6.1 in this chapter depicts part of the SQL-Tutor's genealogy, focusing on its architectural components.

The node at the top of the class hierarchy, *Learning System* represents a specification of an educational technology. On the left and right of the genealogy are nodes, which are architectural components, i.e. *Part-of a Learning System*. They represent the underlying learning and instructional theories, pedagogical strategies, instructional goals, etc. that collectively form a *Learning System* and are vital parts of its specification.

Below *Learning System* are sub-classes - nodes that are related to it via *Is-a* links. A first-level sub-class is *Intelligent Tutoring System* (ITS), a specific class of functionality, in other words, it is also a *Learning System*, but of a specialised kind. The SQL-Tutor is an actual instantiated object of the *Intelligent Tutoring System* class, related to it via an *Is-a* link.

The nodes relating to *Intelligent Tutoring Systems* and the *SQL-Tutor* via the *Part-of* links represent, respectively, the internal structure of the class of functionality, and the internal structure of the *SQL-Tutor* instantiation.

Peer relationships are also depicted in figure 6.1, *Intelligent Computer Aided Instruction* (ICAI) and *Adaptive Learning Environment* (ALE) are peers to *Intelligent Tutoring Systems*, which implies that the three technologies share some common features.

Similarly, intelligent tutoring systems *Proust* (Pascal tutor), *Intuition* (gaming simulation) and *Smihtown* (economics knowledge) are depicted as peers to the *SQL-Tutor*. 
Figure 6.1 ITS/SQL Tutor genealogy
6.2.2 SQL-Tutor Habitat

This section is a high-level depiction of the distinctive features and internal structure of the SQL-Tutor, as well as its learning achievement and learning affect that will be evaluated in a real-world context of student usage. Figure 6.2 depicts the elements of the SQL-Tutor's habitat. The evaluative work on the SQL-Tutor focuses on its compliance with general design principles, learning and instructional theories and usability properties as well as its use to:

- Enable students to acquire domain knowledge;
- Provide individualised tutoring;
- Maintain student models;
- Draw on information from student models to inform its pedagogical strategy;
- Diagnose students' errors, provide helpful feedback and remediation; and
- Promote learning achievement and a positive learning affect.

![Figure 6.2 SQL-Tutor Habitat](image)

Each of the models and the overall system control depicted in figure 6.2, is evaluated in the problem context of the following three levels of analysis (identified in section 5.4.3.1):
1. **Knowledge level analysis** which addresses issues such as the scope of the system’s domain, student and tutoring knowledge and whether the knowledge representation is appropriate.

2. **Program process analysis,** which investigates the following:
   - *Expertise*, i.e. the way domain knowledge is used and manipulated;
   - *Diagnostics*, i.e. the procedures used by the system to analyse the input of the student to maintain the student model; and
   - *Didactics*, i.e. the way teaching goals are determined and teaching strategies are used to guide instruction.

3. **Tutorial domain analysis** investigates any lack of tutorial abilities in any of the three standard knowledge components of the intelligent tutoring system.

### 6.3 Evaluation design phase

The previous section described the distinctive features of the SQL-Tutor in relation to other technologies (technology domain genealogy) and to its usage context (problem domain habitat). The outcome of the descriptive modelling phase, namely a situated technology is fed into the evaluation design phase.

This phase, the evaluation design phase, is in essence a planning phase involving three principal activities associated with internal and external evaluation of the SQL-Tutor: a more detailed investigation of features to be evaluated in a focused usage context, hypotheses formulation, and experimental design.

The outputs of the evaluation design phase are:
- Sets of specific evaluation criteria – both internal and external criteria (emanating from the technology’s distinctive features and internal structure);
- A set of hypotheses about the added value of the technology that can be substantiated or refuted through experimentally acquired evidence; and
- A planned experiment that generates this evidence to support conclusions of added value.
A user-centred evaluation approach is adopted for this study.

6.3.1 External evaluation

6.3.1.1 External evaluation criteria

External evaluation (as described in sections 5.4.2.2 & 5.6.1) assesses the impact of the SQL-Tutor on the students in terms of its ability to:

1. Foster learning, referred to as learning achievement; and
2. Motivate and satisfy the student, referred to as the learning affect.

Empirical methods (a controlled experiment), as well as observational methods and query techniques (questionnaires and interviews) are to be employed for external evaluation. These techniques provide both quantitative and qualitative results. Observation methods are to be used in order to ascertain students' attitudes, rate of progress, concentration and perseverance, and requests for help/elaboration.

6.3.1.2 Experimental design

The seven principles underlying a good ITS evaluation study [Shute & Regian, 1993], discussed in section 5.2, are applied to the evaluation of the SQL-Tutor. A discussion of the experimental design follows:

1. Delineate the goals of the ITS

The goals and teaching strategies of the SQL-Tutor are discussed in Chapter Three, section 3.5 and evaluated in section 6.4.2.2.

2. Define the goals of the evaluation study

The goal of the evaluation study (as outlined in Chapter One, section 1.2) is to perform a systematic controlled evaluative study of the SQL-Tutor and to draw conclusions from the
Evaluation of the SQL-Tutor: An Intelligent Tutoring System for the database query language SQL

study as to the efficacy of the ITS for teaching programming in general and SQL in particular.

The null hypotheses for the experiment are defined as:
- The use of the SQL-Tutor has no effect on student performance in formulating SQL queries
- The SQL-Tutor does not motivate and satisfy the student

The alternative hypotheses are as follows:
- The use of the SQL-Tutor improves student performance in formulating SQL queries
- The SQL-Tutor motivates and satisfies the student.

The standard by which learning achievement is to be measured is the improved ability to formulate SQL single and multiple table queries. The standards used to measure learning affect are student motivation and satisfaction.

The potential confounds (unwanted influences) identified for this case study are:
- Student awareness of different teaching methods, which leads to them engaging in compensatory behaviour; and
- Lecturer's obligation to offer tutoring assistance to students having difficulty in learning SQL.

Quantitative indices, namely, Pre-test and Post-test scores and observational data (body language, concentration and perseverance, requests for elaboration and help) as well as usability, learning affect and instructional value questionnaires are to be used. These data types provide a means of discovering what a student is learning from the ITS, and how effective and satisfactory is the learning.

3. Select an appropriate design to meet defined goals

A hybrid design (incorporating a between-system design and benchmark design) is to be used to compare the following approaches to learning SQL with three groups of subjects.
Group 1: Classroom instruction supplemented by problem solving on paper.
Group 2: Classroom instruction supplemented by problem solving with a DBMS.
Group 3: Classroom instruction supplemented by problem solving with the SQL-Tutor.

4. Instantiate design, with appropriate measures, numbers and types of subjects, and control groups

The dependent measures (variables) are categorised as input measures, namely skills Pre-tests, skills Post-tests, questionnaires, interviews, observations and attitude scales.

The independent measures (variables) are the use of different mediums, namely the SQL-Tutor, paper and pencil, and dBASE version 5 for practising problem solving in SQL.

The statistical measures include sample means, sample standard deviations, standard error of difference between the means, the z-test for equality of mean for independent samples, and the t-test for equivalence of related samples. The t-test for equivalence of related samples is to be used to compare two sets of measurements (in this instance, Pre-test and Post-test means) for each of the three groups.

The subjects projected for participation in this study are 90 undergraduate students enrolled for a third-year Database Management course. Approximately 30 subjects are to be placed into each of the three groups through a process of random selection. Group 3 subjects will solve problems with the SQL-Tutor, group 2 subjects will solve problems using dBASE version 5, and group 1 subjects will solve problems on paper.

Group 3 subjects are to be given approximately 3 hours to solve 25 problems using the SQL-Tutor. Group 2 subjects are to be given approximately 4 hours to complete the same set of problems using dBASE version 5, and group 1 subjects are to be given approximately 2 hours to complete the said problems on paper.

All three groups would receive the same lectures and have access to the same instructional material.
Since group 3 subjects would have the facility to view the solutions to the problems, this privilege would be extended to group 1 and group 2 subjects after they have attempted to solve the problems, and before they are scheduled to write the Post-test.

Potential confounds, such as students not writing the tests or attending practical sessions would be controlled. Another round of practical sessions would be scheduled for absentee subjects belonging to the respective three groups. Subjects who do not attend the rescheduled practical sessions, or who do not write both Pre-test and Post-test would be dropped from the experiment. A small sample method would be used for statistical estimation for group sizes that are below 30.

5. Make logistical preparations for conducting the study

Two separate computer laboratories are to be used for the experiment. The SQL-Tutor would be installed into one computer laboratory, for the exclusive use of group 3 subjects. Group 2 subjects would be allowed to practice problems using DBASE version 5 in another computer laboratory.

6. Pilot-test the ITS and the study

The SQL-Tutor was pilot-tested by the software developer, Tanja Mitrovic using students from the University of Canterbury [Mitrovic & Ohlsson, 1999]. The author’s study was not pilot-tested.

7. Data Analysis

The experimental data arising from the design are to be analysed by:

- A comparison of Pre-test and Post-test group means to ascertain whether the three groups differed significantly on their Pre-test and Post-test performance; and
- A comparison of Pre-test and Post-test group means for each of the three groups to ascertain differences on Pre-test to Post-test performance.
Furthermore, student feedback from questionnaires and interviews are to be analysed and the findings of the student evaluation reported.

6.3.2 Internal evaluation

The purpose of internal evaluation (as discussed in sections 5.3.2.1 and 5.3.2.2) is to investigate the relationship between the behaviour of an intelligent tutoring system and its architecture. In view of this, internal evaluation involves a closer analysis of the SQL-Tutor's features in terms of:

- Internal structure accounting for system's behaviour – where evaluation questions, serving as evaluation criteria are used to analyse its behavioural properties;
- Design principles - using the thirteen pillars of design [O'Shea, 1984] as a reference model for assessing principles of computer tutor construction;
- Usability properties – where the SQL-Tutor's conformance to certain usability principles and paradigms is investigated to assess its usability properties; and
- Theories of learning and instruction – using the Hexa-C Metamodel as a basis for evaluating the extent of compliance with learning and instructional theories.

The evaluation criteria that are to be used for internal evaluation of ITSs, as discussed in section 4.4, are reproduced here.
Table 6.2 Evaluation criteria for internal evaluation of ITSs

<table>
<thead>
<tr>
<th>ITS Features &amp; Internal Structure</th>
<th>Evaluation Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behaviour</td>
<td>What area is covered by the system’s domain knowledge? Can the system give an explanation of a problem solution? Can the system give alternative explanations of the same concept? Can the system answer arbitrary questions from the student? Does the domain model include knowledge about common misconceptions and missing concepts?</td>
</tr>
<tr>
<td>Domain Model</td>
<td></td>
</tr>
<tr>
<td>Tutoring Model</td>
<td>What are the teaching goals of the system? Does the system provide alternative teaching strategies? Are the system’s teaching strategies closely tailored to the students’ needs? Can the student initiate some new area of investigation? Does the system intervene if the user appears to be having difficulty? Does the system actively engage the student? Can the tutoring model relate a diagnosed error to a misconception or a missing concept? Does the tutoring model incorporate remedial strategies in order to provide alternative remedial teaching styles?</td>
</tr>
<tr>
<td>Student Model</td>
<td>What information about the student’s knowledge and skills is stored in the student model? What information about the student’s learning preferences is stored in the student model? What information about the student’s past learning experiences is stored in the student model? Does the student model store information about the student’s advancement stage? Does the system monitor changes proposed by the student and comment on them if they seem to be unwise?</td>
</tr>
<tr>
<td>Overall System Control</td>
<td>Does the system provide helpful feedback on student input? Does the system treat all detected errors? How does the system respond if it cannot diagnose an error? When does the system intervene to remediate a misconception or a missing concept? Does the system attempt the remediation of a misconception or a missing concept when it recognises a student’s need for it? Does the system adapt to the student’s advancement stage? Does the system adapt to the needs and preferences of the student?</td>
</tr>
<tr>
<td>Design Principles</td>
<td>Robustness; helpfulness; simplicity; perspicuity; power; navigability; consistency; transparency; flexibility; redundancy; sensitivity; omniscience; docility.</td>
</tr>
</tbody>
</table>
### ITS Features & Internal Structure

<table>
<thead>
<tr>
<th>Learning and Instructional theories</th>
<th>Evaluation Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree to which system conforms to principles of: Cognition; Constructivism; Creativity; Customisation; Component Display Theory; Collaboration.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usability Principles &amp; Paradigms</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User's memory should not be overloaded; System should be easy to use; Interaction dialogue should be natural, Dialogue should be task-oriented and adaptive; Effectiveness of screen design; Interaction styles (menus, windows, icons, typed input strings).</td>
<td></td>
</tr>
</tbody>
</table>

### 6.4 Evaluation implementation phase

This phase used the planned experiment and evaluation criteria output from the previous phase as a basis for implementation of the evaluation. The evaluation criteria were discussed from a theoretical perspective in sections 4.3.3.2- 4.3.3.3, and 4.4. The use of a planned experiment (as described in sections 5.4.2, and 5.6.1) was selected as a technique for performing external evaluation of the SQL-Tutor. Interviews and questionnaires covering evaluation criteria (discussed in 4.3.3.4, 4.3.3.5 and 4.4) were used as principal evaluation instruments to acquire the required data in the internal evaluation of the SQL-Tutor.

### 6.4.1 External evaluation

External evaluation was carried out in order to assess the educational impact of the SQL-Tutor in terms of learning achievement and learning affect. The results of the experiment and the statistical measures used in the interpretation of findings are presented below.
6.4.1.1 Learning achievement

The mean percentages obtained by students in the Pre-tests and Post-tests, as attained by the various groups, are presented in table 6.3 and displayed in figure 6.3.

Table 6.3: Pre-test & Post-test Mean Analysis

<table>
<thead>
<tr>
<th>Group No.</th>
<th>Description</th>
<th>Pre-Test</th>
<th>Post-Test</th>
<th>% Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Group 1 (Control Group)</td>
<td>64.4</td>
<td>66.4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Group 2 (Experimental Group 2)</td>
<td>61.6</td>
<td>63.5</td>
<td>1.9</td>
</tr>
<tr>
<td>3</td>
<td>Group 3 (Experimental Group 1)</td>
<td>64.2</td>
<td>66.5</td>
<td>2.3</td>
</tr>
</tbody>
</table>

The three groups (Group 1, Group 2 and Group 3) did not demonstrate major differences on their pre-test means, which assessed student ability to apply their knowledge of the SQL database language to formulate single and multiple table queries for a chosen database schema. The Pre-test was administered after covering the relevant concepts in class, illustrating with examples, and reviewing solutions to practice exercises completed by students, but before applying the respective treatments.

The average Pre-test score, representing incoming knowledge for the three groups, was 63.4 percent. A two-tailed z-test was used to compare equality of Pre-test means for the three independent groups. Other measures used were the actual difference between the means and the standard error of difference between the means. These statistical Pre-test measures were computed and are tabulated below.
Table 6.4 Statistical analysis of Pre-test

<table>
<thead>
<tr>
<th></th>
<th>Group 1 &amp; Group 2</th>
<th>Group 1 &amp; Group 3</th>
<th>Group 2 &amp; Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference between group means:</td>
<td>2.8</td>
<td>0.1</td>
<td>2.7</td>
</tr>
<tr>
<td>Standard error of difference:</td>
<td>3.23</td>
<td>3.62</td>
<td>4.24</td>
</tr>
<tr>
<td>Z score:</td>
<td>0.87</td>
<td>0.37</td>
<td>0.63</td>
</tr>
</tbody>
</table>

The standard error of difference (s.e.d.) is a measure of the expected variability of the difference between the means. The actual difference is well within this random variation. The critical values of the z-distribution, to which the respective z-scores were compared, range between −1.96 and 1.96 at the 5% level of significance, and between −2.58 and 2.58 at the 1% level of significance. Testing the statistic, \( z = 0.87 \) against critical values of the normal z-distribution, shows that there is no significant difference between the means of the group 1 and group 2. Testing the statistic, \( z = 0.37 \) against critical values of the z-distribution shows that the difference between the means of the group 1 and group 3 is not significant either. The final comparison between the means of group 2 and group 3 shows a similar result of insignificant difference when testing the statistic, \( z = 0.63 \) against critical values of the z-distribution.

A similar trend emerged with the Post-test means. There was no significant statistical difference between the Post-test means of the three groups. The following test statistics confirm these findings:

Table 6.5 Statistical analysis of Post-test

<table>
<thead>
<tr>
<th></th>
<th>Group 1 &amp; Group 2</th>
<th>Group 1 &amp; Group 3</th>
<th>Group 2 &amp; Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference between group means:</td>
<td>2.9</td>
<td>-0.1</td>
<td>3.0</td>
</tr>
<tr>
<td>Standard error of difference:</td>
<td>3.44</td>
<td>2.67</td>
<td>3.10</td>
</tr>
<tr>
<td>Z score:</td>
<td>0.83</td>
<td>-0.05</td>
<td>0.96</td>
</tr>
</tbody>
</table>
The *t-test for equivalence of related samples* was employed to ascertain whether there was a significant increase in Pre-test to Post-test performance. Related sample pairs were formed where two sets of measurements, namely Pre-test and Post-test scores were compared. To do the test, difference data was computed and this data constituted a sample for the computation performed. The t-test statistics for each of the three groups follow:

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>t-test statistic</td>
<td>1.14</td>
<td>1.02</td>
</tr>
</tbody>
</table>

Comparing the t-score of 1.14 for group 1, with 27 degrees of freedom to critical values of 1.70 at the 5% level of significance, and 2.47 at the 1% level of significance indicates there was no significant increase in performance from Pre-test to Post-test at either level.

One sees a similar pattern emerging when comparing the t-score of 1.02 for group 2, with 22 degrees of freedom, against critical values of 1.72 and 2.51 at the 5% and 1% level of significance. This pattern is maintained in the comparison of the t-score of 1.10 for group 3, with 29 degrees of freedom, against critical values of 1.70 and 2.46.

One can thus accept the null hypothesis, 'The use of the SQL-Tutor has no effect on SQL query formulation results', as defined in section 6.3.1.2, in view of the fact that there was no significant increase in mean performance from Pre-test to Post-test for each of the individual groups. Furthermore, the two-tailed z-test used to compare equality of Pre-test and Post-test means, revealed that there did not exist any significant difference between the Pre-test and the Post-test means of the three groups used in the experiment.

In this particular case study, the result of 'no significant difference' may be attributed to the fact that all three groups involved in the experiment were given the same lectures, illustrative examples, and practice exercises prior to the experiment. The only major difference between the three groups occurred in the method used for solving database problems, with group 1 solving problems on paper, group 2 solving problems using dBASE version 5 and group 3 solving problems with the aid of the SQL-Tutor. The solutions to the database-practice
Evaluation of the SQL-Tutor: An Intelligent Tutoring System for the database query language SQL 156

problems were made available to all groups prior to the Post-test. Another contributing factor for this result is that the SQL-Tutor is designed as a guided discovery-learning practice environment, and this type of learning does not suit all students [Shute, 1990]. This no-significant difference phenomenon is in line with many evaluative studies conducted where the use of different media for learning purposes have not produced any significant difference in academic performance [Kozma, 1994; Clark, 1994, Russell, 1999].

6.4.1.2 Learning affect

Observational techniques and query techniques (questionnaires and interviews) were used to assess learning affect.

From the observations conducted during the experiment, group 3 subjects (practising with the SQL-Tutor) appeared to be relaxed, attentive, focused and in control of their learning. They concentrated on the problem-solving tasks, and progressed through the problems at a reasonable rate, with some students progressing faster than others. There were instances when students were confused by the system’s feedback to their solutions, and they called upon the researcher for assistance. Students also appreciated the value of collaborative learning; on occasion, two students chose to work together in order to detect and correct error(s) in solutions to problems.

Most students persevered in the task of solving the database problems provided by the practice environment. There were a few, however, who became frustrated after a few attempts, and resorted to consulting the system’s ideal solutions. There was no evidence of a rushed, apathetic attitude on the part of the students. Some showed signs of slight weariness after working with the tutor for over an hour, and requested a ten-minute break, which was granted. The overall impression gained was one of positive student attitude and constructive and active engagement in learning.

The actual questions in the questionnaire, which pertain to learning affect, are tabulated below together with their respective average and standard deviation ratings.
Evaluation of the SQL-Tutor: An Intelligent Tutoring System for the database query language SQL

Answers were rated on the following five-point scale:

<table>
<thead>
<tr>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly agree</td>
<td>Agree</td>
<td>Maybe</td>
<td>Disagree</td>
<td>Strongly disagree</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question</th>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  I like practising examples on the computer.</td>
<td>4.33</td>
<td>1.03</td>
</tr>
<tr>
<td>11 The positive feedback gives me a feeling of achievement and motivates me to go on solving all the problems in the database.</td>
<td>4.13</td>
<td>0.78</td>
</tr>
<tr>
<td>22 I prefer doing exercises on paper.</td>
<td>1.83</td>
<td>1.02</td>
</tr>
<tr>
<td>23 It is a waste of time doing interactive practise with the tutor when I can read worked examples in the textbook.</td>
<td>1.90</td>
<td>0.80</td>
</tr>
<tr>
<td>58 I would like similar practice environments for other courses in my diploma.</td>
<td>4.20</td>
<td>0.76</td>
</tr>
</tbody>
</table>

The average student rating of 4+ for questions 1, 11 and 58 is an endorsement of the positive learning affect of the SQL-Tutor in terms of satisfaction and motivation. This result is confirmed by interview findings that reveal complete student consensus on the attention, concentration, and relevance and satisfaction properties of the SQL-Tutor.

Question 58 required students to justify why they would have liked more opportunities for practice on the SQL-Tutor. A sample of the reasons given are provided below:

- Would help them learn;
- Would improve their skills in SQL programming;
- Would be easier to understand their errors;
- More practice would improve their knowledge;
- Would improve their ability to solve problems quickly and enhance their knowledge of SQL;
- Time allocated for mastering SQL was insufficient, more time was needed to practice with problems;
- Would have helped them with the more advanced SQL commands;
- Working with the tutor was interesting because it was interactive and solutions could be verified immediately;
- More time and practice opportunities were needed to fully understand solutions to some of the problems;
- Working with the tutor was fun; and
- More practice would help for the exams.
Judging from the results of the student feedback, one can irrevocably refute the second null hypothesis, that the 'The SQL-Tutor does not motivate and satisfy the student'.

The students found that the tutor provided them with a positive and rewarding learning environment. They wanted more opportunities for practice in order to fully exploit its potential and gain the desired level of mastery or expertise.

6.4.2 Internal evaluation of SQL-Tutor

The reader is advised to refer back to Chapter Three, which describes the features, components, and capabilities of the SQL-Tutor.

The concept of complete intelligent tutoring as outlined in sections 4.3.3.5, 4.4, 5.4.3 and 5.6.2, was used to support the design of the questionnaire, since this concept portrays tutoring in the way it should be perceived by the student. The internal evaluation of the SQL-Tutor was carried out by examining how its architecture supports tutoring. This involved a close inspection of the knowledge and processes underlying its domain, student, and tutoring models as well as its overall system control to unveil its overt behaviour. The scope of internal evaluation was extended beyond the architecture of the intelligent tutoring system to incorporate design principles, usability criteria, and also new directions in learning and instructional theories.

It should be noted that the an evaluation question appearing in the questionnaire may examine both knowledge and processes, as part of knowledge level analysis, and program process analysis as discussed in section 5.4.3.1.

As described in section 6.3.1.2, three groups were used for the experiment. Subjects in Group 3 practised with the SQL-Tutor, subjects in Group 2 practised with dBASE version 5, and subjects in Group 1 practised on paper.

A total of thirty questionnaires completed by Group 3 subjects (practising with the SQL-Tutor), were used for the internal evaluation. In addition, interviews were conducted with a random sample of 10 students drawn from Group 3, and observations made during the laboratory sessions were recorded.
For the actual questions, see Appendix C.

The following four scales were used in the questionnaire:

**Category A:**
1. Strongly disagree
2. Disagree
3. Maybe
4. Agree
5. Strongly agree

**Category B:**
1. Complete Solution
2. List all errors
3. Partial Solution
4. Hint
5. Error Flag

**Category C:**
1. Certainly not enough
2. Needed some more
3. About the right amount
4. Needed a bit less
5. Needed a lot less

**Category D:**
1. Could not use it at all
2. Slowly
3. Gradually
4. Fairly quickly
5. Very quickly

Category A was the most common category used, while categories B, C and D were categories employed once-off because they were more appropriate for specific questions. In addition, students were required to respond with either a ‘yes/no’ or ‘agree/disagree’ type of
answer for certain questions. In many instances, students were required to elaborate by substantiating their answers with reasons or examples.

An analysis of each aspect of internal evaluation follows with a set of pre-determined questions forming the focal point for the discussion. The evaluation questions posed in sections 6.4.2.1 – 6.4.2.7 and summarised in tables 4.1 and 6.2, were initially introduced in figure 4.4 in the context of the Siemer & Angelides [1998] model. The reader is again reminded that the goal is to ascertain whether the SQL-Tutor can behave in the way suggested by the questions.

**6.4.2.1 Analysis of domain model**

There are five main evaluation questions addressed in this section. An analysis of each follows:

*What area is covered by the system’s domain knowledge?*

The area covered by the system’s domain knowledge is the SELECT statement of the SQL database language. The scope does not include table creation nor table maintenance.

*Can the system give an explanation of a problem solution?*

The question of whether or not the system provides an explanation of a problem solution is relevant to an analysis of both the domain and tutoring models. The system does not, at present, provide explanations of its ideal solutions. The architecture of the SQL-Tutor does not include an expertise module for deriving solutions. Instead, the system supports a set of stored ideal solutions that have been produced externally. The system does, however, provide explanations of students’ solutions via its feedback mechanisms informing students of the number of errors present in their solutions, the clause(s) in error, as well as a description of the error(s) committed.
Can the system give alternative explanations of the same concept?

The SQL-Tutor does not currently offer alternative explanations of concepts. This was corroborated by student opinion elicited in question 29, where 86% of the students disagreed, while 14% agreed that the SQL-Tutor could give alternative explanations.

Two typical examples, provided as substantiation for agreement, follow:

- The system accepts your answer as correct even though it is different from the system's ideal solution; and
- The system recognises the use of the word 'between' in a solution and its equally acceptable alternative using relational operators.

From the reasons cited above, it is clear that these students are providing examples of alternative answers instead of alternative explanations, and as such they cannot be regarded as valid responses.

Can the system answer arbitrary questions from the student?

Ninety-one percent of the students who answered the question stated that the SQL-Tutor is not able to answer arbitrary questions. Of the remaining 9% who agreed, a few mentioned the help and explanation capabilities of the tutor, aspects not relevant to the question. A number of students did not answer this question which seems to imply that these students, together with the 9% that agreed, do not understand the word 'arbitrary'.

Does the domain model include knowledge about common misconceptions and missing concepts?

The domain model does not explicitly incorporate knowledge about common misconceptions and missing concepts, but instead covers them implicitly by its use of constraint-based modelling techniques to represent domain knowledge. Since domain knowledge in the SQL-Tutor is represented as a set of constraints on correct solutions, violation of constraints in students' solutions represent typical misconceptions in the domain of SQL programming.
The actual questions pertaining to the domain model are tabulated below, together with respective averages and standard deviations calculated using the ratings assigned by students.

<table>
<thead>
<tr>
<th>Question</th>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2  It is easy to access HELP information such as description of databases, tables and/or attributes by directly selecting database, table/attribute names.</td>
<td>4.23</td>
<td>0.57</td>
</tr>
<tr>
<td>3  The explanations pertaining to the various clauses (SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY) are a useful form of help.</td>
<td>4.13</td>
<td>0.63</td>
</tr>
<tr>
<td>4  I can learn about the elements of SQL such as functions, expressions, predicates and operators by selecting appropriate options in the HELP menu.</td>
<td>3.70</td>
<td>0.60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question</th>
<th>% No</th>
<th>% Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>25  The SQL-Tutor contains the definition of more than one database.</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>26  The SQL-Tutor contains a set of problems and ideal solutions for specified databases</td>
<td>13%</td>
<td>87%</td>
</tr>
<tr>
<td>27  Do you agree that the SQL-Tutor is able to answer arbitrary questions about SQL programming</td>
<td>91%</td>
<td>9%</td>
</tr>
<tr>
<td>29  The SQL-Tutor can give alternative explanations</td>
<td>86%</td>
<td>14%</td>
</tr>
</tbody>
</table>

6.4.2.2 Analysis of tutoring model

Eight principal questions are addressed in this section. A discussion on each follows:

What are the teaching goals of the system?

The teaching goals of the tutor are to promote three kinds of learning, namely conceptual learning, problem solving, and meta-learning as discussed in section 3.5. The student can learn about the concepts, elements, and query formulation rules by using interface controls, menu options, and feedback in the form of system-generated explanations to student solutions. Secondly, the SQL-Tutor provides a problem-solving environment in which students acquire knowledge of SQL in a declarative form via constraints and strengthen their knowledge by solving practice problems. It provides guidance in solving problems by a system of feedback where students can learn from their mistakes and remedy them. Finally, the system supports meta-learning in the form of self-explanations as typified by error-messages and correct solutions [Mitrovic, 1998].
Does the system provide alternative teaching strategies?

The two principal instructional strategies employed are learning-by-doing, where students are actively engaged in solving database problems, and discovery-learning, where students are encouraged to discover and correct their own errors. The system does not provide alternative teaching strategies.

Are the system's teaching strategies used closely tailored to students' needs?

Yes, the teaching strategies are closely tailored to student needs. The system provides individualisation of instruction by developing student models for individual learners and is thus able to tailor its instruction to meet individual learning needs via these models. Questions 13, 24, 28, 30, 31, 32, 37, 42 and 56 address the extent to which the tutor is capable of meeting a wide range of student needs. The averages of the responses to these questions range from 2.13 to 3.97.

Certain questions required students to elaborate, give reasons or substantiate their answers. A discussion of these responses follows:

24. The SQL-Tutor helped me to solve more advanced database problems. If you disagree, state why.

The students' ratings averaged 3.83 for this question. A fair number of students chose the option 'maybe', showing their neutrality. One student boldly stated that the tutor did not help her to solve more advanced database problems that she could not have solved on paper, but this is a representation of an extreme viewpoint.

28. The SQL-Tutor gives a satisfactory explanation of a problem solution (including a problem solved by me). Qualify your answer.

The average rating for this question was 2.82. Students qualified their answers as follows:

- Disagree – 'Simply gives the system's solution but no explanation.'
- Disagree – 'Should show other approaches to solving the same problem.'
- Disagree – 'Does not adequately explain why my solution is wrong. Does not accept other solutions as correct even though I think that they were correct.'
- Maybe – 'Does not explain its own solution, but shows errors in your solution.'
- Agree – 'Gives you a list of errors that are present in your solution.'
- Agree – 'Able to understand the solutions given.'
- Agree – 'Able to guide the students towards solving the problem.'

Student opinion was clearly divided on this matter; this is attributable to a number of factors, namely student interpretation of the question, individual learning experiences with the tutor, and the fact that the system does not offer an explanation of its own solutions. There is no doubt that the tutor does provide explanations of violations within students' solutions by its capability of identifying and indicating syntactic and semantic errors. Whether the explanations provided are satisfactory or not would, however, depend on individual learning experiences with the tutor and whether the student elected to use more detailed levels of system feedback.

37. The practice environment provided by the SQL-Tutor is sensitive to my needs and preferences. Give a reason for your answer.

The average rating for this question was 3.23 with a standard deviation of 0.95. The following reasons were given:

<table>
<thead>
<tr>
<th>Response</th>
<th>Reason</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disagree</td>
<td>Tutor merely contains pre-defined databases and related problems.</td>
<td>3</td>
</tr>
<tr>
<td>Disagree</td>
<td>Does not incorporate general questions on SQL.</td>
<td></td>
</tr>
<tr>
<td>Disagree</td>
<td>You can't choose problems.</td>
<td>1</td>
</tr>
<tr>
<td>Disagree</td>
<td>Same problems presented to students in the same order.</td>
<td>1</td>
</tr>
<tr>
<td>Disagree</td>
<td>Tutor caters for my needs not my preferences.</td>
<td>2</td>
</tr>
<tr>
<td>Disagree</td>
<td>Tutor does not cater for my needs since I have not used the system's</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>choice option but rather the student's choice.</td>
<td></td>
</tr>
<tr>
<td>Maybe</td>
<td>Provides very different examples, some more difficult than others.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Simple problems building up to the more complex ones.</td>
<td></td>
</tr>
<tr>
<td>Maybe</td>
<td>The tutor can pinpoint weaknesses or errors but does not cater for</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>all types of errors.</td>
<td></td>
</tr>
<tr>
<td>Agree</td>
<td>Gives me adequate practice in solving different problems.</td>
<td>1</td>
</tr>
<tr>
<td>Agree</td>
<td>Allows students to choose the database to work on and the problems to</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>solve. Provides structure for making queries.</td>
<td></td>
</tr>
<tr>
<td>Agree</td>
<td>Databases used in Tutor are general. One can identify easily with</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>them.</td>
<td></td>
</tr>
<tr>
<td>Agree</td>
<td>The tutor presents me with a problem similar to the one I had</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>difficulty with.</td>
<td></td>
</tr>
<tr>
<td>Agree</td>
<td>Can select the level of expertise (novel, familiar, experienced)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>before working with the tutor.</td>
<td></td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>Makes it easy to trace my errors.</td>
<td>2</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>No need to type solutions</td>
<td>1</td>
</tr>
</tbody>
</table>
The answers given were directly related to students' individual learning needs. For example, some students cited the mere provision of specific pre-defined databases with related practice problems as a reason for disagreeing. They would have appreciated a facility that permitted general questions on SQL. Others used the very same databases as a basis for agreement.

Certain students felt that the tutor was sensitive to their learning needs, in that it helped them to grasp concepts and successfully solve problems. Others felt that the tutor did not present them with additional problems covering concepts with which they had difficulty.

42. My solution (with/without errors) is understandable to the SQL-Tutor. Substantiate your answer.

<table>
<thead>
<tr>
<th>Response</th>
<th>Reason</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disagree</td>
<td>I believed some of my solutions were correct but the tutor would not accept it.</td>
<td>5</td>
</tr>
<tr>
<td>Disagree</td>
<td>Tutor does not pick up spelling errors.</td>
<td>2</td>
</tr>
<tr>
<td>Disagree</td>
<td>You have to select more detailed levels of feedback for more information on your errors.</td>
<td>1</td>
</tr>
<tr>
<td>Disagree</td>
<td>The tutor has one set of answers to which students' solutions are compared.</td>
<td>1</td>
</tr>
<tr>
<td>Maybe</td>
<td>Can determine correct answers and errors of students. In some cases, tutor seems to accept only system answers.</td>
<td>4</td>
</tr>
<tr>
<td>Maybe</td>
<td>Tutor is case sensitive. If you use lower case letters in your solution it will reject your solution even though your solution is otherwise correct.</td>
<td>1</td>
</tr>
<tr>
<td>Maybe</td>
<td>Tutor tells you which clause is in error but does not give you anything more specific.</td>
<td>1</td>
</tr>
<tr>
<td>Maybe</td>
<td>I used another method to solve a problem. The tutor picked up the errors in my solution but did not give me the correct solution using my method.</td>
<td>2</td>
</tr>
<tr>
<td>Agree</td>
<td>Lets me know I have errors. Lists all errors.</td>
<td>6</td>
</tr>
<tr>
<td>Agree</td>
<td>The tutor accepted most of my solutions.</td>
<td>1</td>
</tr>
<tr>
<td>Strongly agree</td>
<td>Motivates you if you have errors in your solution and gives you the correct solution.</td>
<td>1</td>
</tr>
</tbody>
</table>

Some of the students did not substantiate their answers. It emerges clearly from their responses that the tutor does not identify all errors. Furthermore, there are different ways to solve a problem, and the students would like to correct their solutions using their chosen method and not the method employed in the ideal solution. In addition, they would like to be presented with alternative correct answers that are equally acceptable. Other student comments reveal that they do not understand the rationale behind the instructional strategy that requires students to analyse their solutions and discover their errors before soliciting additional help from the tutor.
Can the student initiate some new area of investigation?

Yes, students can initiate a new area of investigation within the scope of the system, in the sense that they can elect to solve problems of their own choice, as well as select the desired level of system feedback, thereby bypassing system control. Furthermore, students in need of clarity can get explanations on the elements of SQL by selecting options in the HELP menu without having to leave the problem-solving environment. An average of 4.03 was obtained for question 36, which relates to this property of the tutor.

Does the system intervene if the user appears to be having difficulty?

The system does not intervene if the student is having difficulty during the problem-solving task, but rather, after the task is completed and the solution has been submitted to the system. The justification for this, which is provided in section 3.4.4.1, is that the sequence of the individual steps cannot, of itself, constitute a successful query. The student average rating of 2.39 for question 34, covering the timing of system intervention, indicates that opinions diverge on this issue.

Does the system actively engage the student?

The average rating for question 35 was 3.97, which endorses the tutor’s ability to, actively engage the student.

Can the tutoring model relate a diagnosed error to a misconception or a missing concept?

The tutoring model definitely relates a diagnosed error to a misconception or a missing concept, by virtue of the fact that domain knowledge is represented as a set of constraints on correct solutions. Violation of a constraint represents an incorrect application of a rule or a logical error or a missing concept. Question 14 of the questionnaire is relevant to this characteristic of the tutor. The majority of students (namely 63 %) agreed that the tutor could diagnose and communicate syntactic and semantic errors. Comments follow from students who disagreed:

- ‘Tutor does not identify an error as syntactic or semantic.’
• 'Tutor provides vague error messages.'
• 'Tutor does not pick up all the syntax errors.'
• 'Tutor does not diagnose case-sensitive errors.'
• 'Tutor does not recognise alternative correct solutions.'

From the reasons given, one can see that some of the students misinterpreted the question. They cited one or more instances of syntax errors that were not diagnosed correctly by the tutor, or alternative solutions that were not accepted by the tutor, as justification for their dissent. These errors may be traced to bugs that exist in the program or to an incomplete constraint base, but they alone do not constitute proof that the SQL-Tutor is incapable of diagnosing or communicating syntactic or semantic errors.

**Does the tutoring model incorporate remedial strategies in order to provide alternative remedial teaching styles?**

The tutoring model does not incorporate remedial strategies in order to provide alternative remedial teaching styles. It does not support alternative teaching styles.

The actual questions pertaining to the tutoring model are tabulated below, together with their respective averages and standard deviations.

<table>
<thead>
<tr>
<th>Question</th>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 Which of the following feedback mechanisms did you most frequently use when the tutor indicated that your solution was incorrect? (1. Correct solution 2. List all errors 3. Partial solution 4. Hint 5. Error Flag)</td>
<td>2.87</td>
<td>1.20</td>
</tr>
<tr>
<td>13 Do you think that the feedback mechanisms provided by the tutor gave you adequate help and advice on the error(s) made in order for you to correct your solution?</td>
<td>2.13</td>
<td>0.57</td>
</tr>
<tr>
<td>24 The SQL-Tutor helped me to solve more advanced database problems.</td>
<td>3.83</td>
<td>0.83</td>
</tr>
<tr>
<td>28 The SQL-Tutor gives a satisfactory explanation of a problem solution (including a problem solved by me.)</td>
<td>2.82</td>
<td>1.02</td>
</tr>
<tr>
<td>30 The SQL-Tutor stores each of my actions in the session history so that I can inspect it to see how well I am doing.</td>
<td>3.97</td>
<td>0.67</td>
</tr>
<tr>
<td>31 The explanations provided by the SQL-Tutor are tailored to my needs and the difficulties that I am experiencing.</td>
<td>3.10</td>
<td>0.84</td>
</tr>
<tr>
<td>32 The problems presented to me by the SQL-Tutor are based on my history (previous attempts).</td>
<td>2.93</td>
<td>0.88</td>
</tr>
<tr>
<td>Question</td>
<td>Average</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>---------</td>
<td>-------------------</td>
</tr>
<tr>
<td>34 The SQL-Tutor intervenes when I have difficulty while solving a problem.</td>
<td>2.39</td>
<td>0.83</td>
</tr>
<tr>
<td>35 The SQL-Tutor actively engages me in solving database problems.</td>
<td>3.97</td>
<td>0.57</td>
</tr>
<tr>
<td>36 I can decide at any point, to learn about any concept, clause, and function or choose the next problem to be solved or choose another database to work with.</td>
<td>4.03</td>
<td>0.76</td>
</tr>
<tr>
<td>37 The practice environment provided by the SQL-Tutor is sensitive to my needs and preferences</td>
<td>3.23</td>
<td>0.95</td>
</tr>
<tr>
<td>42 My solution (with/without errors) is understandable to the SQL-Tutor.</td>
<td>3.20</td>
<td>0.96</td>
</tr>
<tr>
<td>56 I know that the SQL-Tutor is intended as an interactive practice environment for students. Nevertheless I would like a scoring facility i.e. a mark for my attempt.</td>
<td>3.87</td>
<td>0.73</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 Do you agree that the SQL-Tutor can diagnose and communicate syntactic as well as semantic errors made by students?</td>
<td>37%</td>
<td>63%</td>
</tr>
<tr>
<td>15 When I call upon the SQL-Tutor to select a problem for me to solve, it either selects a new problem or presents me with a problem I failed to solve before.</td>
<td>3%</td>
<td>97%</td>
</tr>
<tr>
<td>16 After I solve a problem, the SQL-Tutor gives me an opportunity to solve the same problem again using an alternative solution</td>
<td>75%</td>
<td>25%</td>
</tr>
</tbody>
</table>

6.4.2.3 Analysis of student model

The analysis of the student model component centres around five evaluation questions. Many of the answers provided are a cross-reference to information provided in Chapter Three, since the student models are not accessible to the students.

What information about the student's knowledge and skills is stored in the student model?

The SQL-Tutor maintains a student model for each student containing general information about the student (name and knowledge level), a history of previously solved problems, and information about the usage of constraints obtained from student’s solution as described in section 3.4.3.

What information about the student's learning preferences is stored in the student model?

The student modeller contains no information on the student’s learning preferences.
What information about the student's past learning experience is stored in the student model?

It maintains a history of each constraint, as described in section 3.4.3, which contains information about how often the constraint was relevant for the ideal solution of a practice problem, how often it was relevant for the student's solution, and how often it was satisfied or violated. This information is stored in three indicators called 'relevant', 'used' and 'correct'. The pedagogical module uses this information to manage tutorial instruction.

Does the student model store information about the student's level of advancement?

Yes, it is updated to reflect successful/unsuccessful application of concepts for each problem-solving attempt of the student. The evidence to support this is the maintenance of a history for each constraint, as described in the answer to the preceding question.

Does the system monitor changes proposed by the student and comment on them if they seem to be unwise?

The answer to this question can be both 'yes' and 'no'. Yes, the system monitors changes proposed by the student when submitting a revised solution and comments on them if the constraints representing the domain knowledge are violated. No, the system does not monitor changes by the student while the query is in the process of being formulated.

The actual question in the questionnaire, pertaining to the student model, is tabulated below. The rating for the '%Yes' category indicates general assent on this issue.

<table>
<thead>
<tr>
<th>Question</th>
<th>% No</th>
<th>% Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>17  The history window of the SQL-Tutor maintains a log of all my previous attempts at solving problems during the current session.</td>
<td>3%</td>
<td>97%</td>
</tr>
</tbody>
</table>
6.4.2.4 Analysis of overall system control

This section is based on seven evaluation questions. Examples of actual feedback may be seen in section 3.4.4.1.

Does the system provide helpful feedback to student input?

Yes, the system does provide helpful feedback to students' solutions by diagnosing solutions as correct or incorrect and providing feedback on the errors committed. The average student rating on the given scale for this question was 3.50. This implies there is scope for improvement of system feedback.

Does the system treat all detected errors?

Yes, in the context of the SQL-Tutor, a detected error is a constraint violation and the system treats it with appropriate feedback by describing the error committed by students. These constraints relate to syntactic and semantic type errors, and are general in that their conditions can be tested against any problem as discussed in section 3.4.2.

How does the system respond if it cannot diagnose an error?

The tutoring system does not provide error messages for the errors it cannot diagnose. Those errors, if not picked up the students themselves, will go untreated.

When does the system intervene to remediate a misconception or a missing concept?

The system intervenes to remediate a misconception after the student has attempted a solution and the system detects errors in it.
Does the system attempt the remediation of a misconception or a missing concept when it recognises the student's need for it?

Yes, most definitely the system attempts the remediation of a misconception or missing concept when it recognises the student's need for it. Again it must be re-iterated that the SQL-Tutor does not follow the student step-by-step through the problem solving process, hence remediation is not attempted after individual problem solving steps. Instead, the need for remediation is recognised after the student has submitted a complete incorrect solution to a practice problem. Question 40, which addresses the timing of system remediation, received an average rating of 3.48. This figure indicates that students are not altogether convinced that the timing of system remediation is appropriate, but from the designer's point of view this approach is perfectly justifiable as discussed in section 3.4.4.1.

Does the system adapt to the student's advancement stage?

Yes, most certainly it does, since it uses the student model to ascertain student success or lack thereof with individual constraints, and then presents problems containing constraints that the student has not yet mastered or encountered. Furthermore, it advances students by presenting them with more complex problems after they have successfully attempted simpler problems. The average student rating for question 38 was 4.03, indicating that, students agree that the system adapts to their level of advancement.

Does the system adapt to the needs and preferences of the student?

The student modeller does not maintain learning preferences of students. It does, however, maintain general information about the student as described in section 3.4.3, which is used by the pedagogical module to guide tutorial instruction. This implies that the system uses its student models to adapt to the learning needs of each student.

The actual questions pertaining to overall system control are tabulated below together with respective averages and standard deviations of student ratings.
Evaluation of the SQL-Tutor: An Intelligent Tutoring System for the database query language SQL

<table>
<thead>
<tr>
<th>Question</th>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 The feedback (i.e. the system's responses to my answers) is informative and useful.</td>
<td>3.50</td>
<td>0.78</td>
</tr>
<tr>
<td>33 The SQL-Tutor gives me control in that it allows me to choose the problems that I want to solve.</td>
<td>4.00</td>
<td>0.95</td>
</tr>
<tr>
<td>38 The SQL-Tutor is tailored to my level of advancement i.e. it presents me with more complex problems after I have solved the simple ones.</td>
<td>4.03</td>
<td>0.57</td>
</tr>
<tr>
<td>40 The timing of the tutor's intervention was appropriate i.e. after I attempted to solve the problem.</td>
<td>3.48</td>
<td>0.91</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question</th>
<th>% No</th>
<th>% Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>39 The SQL-Tutor interrupts me unnecessarily.</td>
<td>97%</td>
<td>3%</td>
</tr>
</tbody>
</table>

6.4.2.5 Analysis of design principles

This section attempts to evaluate the SQL-Tutor's design against generic principles of computer-based tutor construction as advocated by O'Shea et al [1984] and summarised in tables 4.1 and 6.2.

The responses to the actual questions, pertaining to design principles, are tabulated below, in the form of respective averages and standard deviations.

<table>
<thead>
<tr>
<th>Question</th>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>44 The SQL-Tutor is largely robust in that it is able to detect and report on most errors including 'obvious mistakes'.</td>
<td>3.87</td>
<td>0.78</td>
</tr>
<tr>
<td>45 The SQL-Tutor is helpful in that it always provides some sort of help when I am stuck or unable to solve the problem.</td>
<td>3.77</td>
<td>0.90</td>
</tr>
<tr>
<td>46 The SQL-Tutor is fairly simple to use as it reduces the amount of typing necessary to achieve a given task.</td>
<td>4.37</td>
<td>0.56</td>
</tr>
<tr>
<td>47 SQL is perspicuous in that it does not provide me with a great many mystifying buttons to choose from.</td>
<td>3.97</td>
<td>0.68</td>
</tr>
<tr>
<td>48 The SQL-Tutor has state-of-the-art graphic capabilities.</td>
<td>2.80</td>
<td>0.81</td>
</tr>
<tr>
<td>49 The SQL-Tutor is navigable in that I know exactly where I am when using the system</td>
<td>3.80</td>
<td>0.61</td>
</tr>
<tr>
<td>50 The SQL-Tutor is absolutely consistent i.e. it always behaves in the same way in the same situations</td>
<td>3.77</td>
<td>0.63</td>
</tr>
<tr>
<td>51 The SQL-Tutor is fully transparent i.e. the effects of my actions are always displayed</td>
<td>3.57</td>
<td>0.73</td>
</tr>
<tr>
<td>52 The SQL-Tutor is flexible in that I can solve problems of my choice to gain the mastery I require.</td>
<td>3.90</td>
<td>0.88</td>
</tr>
<tr>
<td>53 The SQL-Tutor has only a single representation of the subject matter</td>
<td>3.33</td>
<td>0.96</td>
</tr>
</tbody>
</table>
Evaluation of the SQL-Tutor: An Intelligent Tutoring System for the database query language SQL

<table>
<thead>
<tr>
<th>Question</th>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>54 The SQL-Tutor is capable of leading me 'by the hand' by asking me leading questions in cases where it knows what I want to do but I am having difficulty doing it.</td>
<td>2.93</td>
<td>0.83</td>
</tr>
<tr>
<td>55 The Tutor is docile in that it is usually seems to be under my command.</td>
<td>3.33</td>
<td>0.80</td>
</tr>
</tbody>
</table>

Robustness:

The average rating for this question was 3.87, indicating that the SQL-Tutor is fairly robust in its ability to detect and report on most errors. The tutor has the ability to detect and provide explanations for syntactic and semantic errors. The possibility exists, however, that some errors will slip through the net, and go undetected. Students who disagreed with the statement that the SQL-Tutor was largely robust, offered the following reasons:
- 'I used a full stop instead of a comma and the tutor did not detect this error.'
- 'Detects errors but not always the cause of errors.'
- 'In some instances, I made an error and the tutor did not notice it.'

Helpfulness:

The average rating for this question was 3.77. The SQL-Tutor is perceived as helpful in that it provides feedback to students whenever they are unable to solve the problem. It is important to note that system feedback is provided only after a student has attempted the solution, and this fact may have been a contributory factor in student dissent or ambivalence. Furthermore the feedback mechanisms employed offer differing levels of information ranging from the minimal (diagnosing a solution to be correct/incorrect and citing the number of errors committed) to the more detailed (stating the clause(s) in error, providing explanation(s) of errors committed, and presenting the ideal solution upon request). System feedback is upgraded only after several unsuccessful attempts have been made by the student and this may have contributed to frustration. Students do have the option to directly select more detailed levels of system feedback, but some may not have exercised this.
Simplicity:

The average rating for simplicity on the given scale was a high 4.37. The students found it simple to use since it reduced the amount of typing necessary to implement a given task.

Perspicuity:

The average rating of 3.97 for question 47 indicates that the student found the SQL-Tutor to be perspicuous, in that it did not provide them with a great many mystifying buttons from which to choose.

Power:

The SQL-Tutor does not employ state-of-the-art graphics as evidenced by the relatively low student average rating of 2.80.

Navigability:

The question about to the SQL-Tutor’s navigability produced a 3.80 average rating, thereby confirming that the students knew exactly where they were when using the system.

Consistency:

The results of student feedback in respect of consistency indicated a general consensus that the SQL-Tutor behaved in the same way in the same situations.

Transparency:

The SQL-Tutor is fairly transparent in the sense that the effect of student actions (students’ solutions) are always displayed (system feedback). From the average rating of 3.57, one deduces a tendency towards student agreement. The SQL-Tutor, however, is not fully transparent in terms of its instructional goals and instructional strategies, which leaves the student a trifle unsure as to its true capabilities and its modus operandi.
Flexibility:

The average rating of 3.90 is a clear indication that students agree on the flexibility of the SQL-Tutor, in that it allows the more capable and experienced students to progress faster. These students can elect to use the student option, which allows them to solve problems of their choice. This flexibility, however, comes with a price in that a student may not cover all the constraints stored in the knowledge base.

Redundancy:

The SQL-Tutor represents only a single view of the subject matter. It thus lacks, or does not allow for, alternative representations of the same subject matter. This implies that if students are not altogether satisfied with the explanations generated by the system, they have no recourse to alternatives and this may lead to frustration. The average rating of 3.33 shows that the answer ‘maybe’ was the most popular choice for this question. It seems that the majority of students were unsure as to whether the SQL-Tutor supported single or multiple representations of the subject matter.

Omniscience:

The average rating was 2.93, with students tending to the answer ‘maybe’. The SQL-Tutor is omniscient in the sense that it guides students in problem solving by informing them whether their answers are correct or incorrect, as well as indicating the number of errors in their answers, thereby allowing students the opportunity to discover and correct their own errors. The system provides further support for the students’ repeated endeavours by the different levels of its feedback mechanism, offering more detailed explanations on where errors reside and the nature of errors committed. The tutor, however, does not lead the student by the hand by asking leading questions, hence, the relatively low average rating.

Docility:

The average rating of 3.33 is an indication that students, in the main, are unsure whether the tutor can be classified as docile or not. One can say that the tutor adopts a more docile approach where learners choose to take charge of their own learning by selecting the
problems to be solved, and by selecting the desired level of system feedback. Hence the
tutor's flexibility comes with a price in that the tutor adopts a more docile stance in the
learning process.

6.4.2.6 Analysis of usability

This section analyses usability properties of the SQL-Tutor as summarised in tables 4.1 and
6.2. The interface of the SQL-Tutor may be seen in section 3.4.1.

From the students' responses to the questionnaire, one can categorically state that the SQL-
Tutor is very easy to use and students, in fact, learnt to use it very quickly. The interaction
dialogue is related to the problem-solving task where students submit completed solutions to
the system and the system responds with feedback using a natural language format. The
interaction dialogue is fairly effective but limited in terms of the scope of the system. The
screen design is favoured, in that it does not display too much information. The main
window of the tutor offers a virtual tabletop for solving problems, which was greatly
appreciated by students. Memory load was thereby reduced, since the text of the problem to
be solved and the database schema were presented on the same window supporting the
solution template. The interaction style of the tutor supports pull-down menus, open menus,
windows; tool-tip features; and a point-and-click interface, which complies with known
standards for educational technology products.

<table>
<thead>
<tr>
<th>Question</th>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 The provision of SQL clauses (SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY) in the main window of the tutor reduces my memory load in that I do not have to remember the exact keywords used and the relative order of the clauses.</td>
<td>4.57</td>
<td>0.73</td>
</tr>
<tr>
<td>6 The provision of the SQL clauses (SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY) helps me to visualise the goal structure when solving problems</td>
<td>4.40</td>
<td>0.67</td>
</tr>
<tr>
<td>7 The information provided on a particular database schema (for example, MOVIES) and its associated tables and attributes in the main window assists me in formulating the SQL query command in that I do not have to remember table and attribute names.</td>
<td>4.50</td>
<td>0.63</td>
</tr>
<tr>
<td>8 The main window of the SQL tutor displays the text of the problem being solved and this serves as a reminder of the elements requested in the query.</td>
<td>4.40</td>
<td>0.50</td>
</tr>
<tr>
<td>9 The SQL-Tutor is complicated to use.</td>
<td>1.93</td>
<td>0.52</td>
</tr>
<tr>
<td>18 I learnt to use the SQL-Tutor (1. Could not use it all 2. Slowly 3. Gradually 4. Fairly Quickly 5. Very Quickly)</td>
<td>4.43</td>
<td>0.57</td>
</tr>
<tr>
<td>Question</td>
<td>Average</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>---------</td>
<td>-------------------</td>
</tr>
<tr>
<td>21 There is too much information on the screens, i.e. the screens are</td>
<td>1.90</td>
<td>0.40</td>
</tr>
<tr>
<td>too cramped</td>
<td></td>
<td></td>
</tr>
<tr>
<td>41 I can express my input (solution) into the system in a natural way</td>
<td>4.10</td>
<td>0.55</td>
</tr>
<tr>
<td>by using either the point and click interface of the SQL-Tutor or</td>
<td></td>
<td></td>
</tr>
<tr>
<td>typing in the solution.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43 The system’s feedback is expressed in a natural language that I can</td>
<td>4.00</td>
<td>0.69</td>
</tr>
<tr>
<td>easily understand.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question</th>
<th>% No</th>
<th>% Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>19 The SQL-Tutor provides me with a point and click interface that</td>
<td>3%</td>
<td>97%</td>
</tr>
<tr>
<td>minimises the typing for SQL query commands and thereby reduces typing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>errors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 It is easy to erase a solution with errors and restart by using the</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>CLEAR button provided by the SQL-Tutor.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.4.2.7 Analysis in respect of current directions in learning and
instructional theories

This section discusses the result of student evaluations with respect to the SQL-Tutor’s conformance to the overlapping aspects incorporated within the Hexa-C Metamodel, namely: cognition, constructivism, customisation, creativity, component display theory, and collaboration. The six C’s were discussed under evaluation criteria in Chapter Four, section 4.2.2.5 as current directions propounded by instructional experts [de Villiers, 1998; 1999a] and are summarised in tables 4.1 and 6.2. These were used to evaluate the SQL-Tutor with relation to its underlying instructional and learning theories.

Interviews were used as an evaluation instrument for obtaining student feedback, from a random sample of ten subjects drawn from Group 3. Each of these directions is discussed separately by way of a brief review of student responses to specific questions, followed by a general discussion of conformance.

1. Cognition

The following aspects were investigated with respect to cognition and cognitive learning, namely, reduced memory load, learning, retention and learning strategies, planning, interpreting, revising, critical thinking, and progress monitoring.
1.1 How did the provision of keywords and the relative order of clauses help you when solving problems?

All the students interviewed welcomed the provision of keywords and the relative order of clauses. They found these features beneficial in that:

- They served as memory aids, since they were not required to remember keywords and the order of clauses within the SELECT statement;
- The structure of the solution was provided;
- It saved them the task of typing in keywords of the various clauses;
- It prevented them from making errors with respect to keywords and clause order; and
- They could concentrate on understanding the requirements of the question and on solving the problem.

This reaffirms one of the theoretical principles upon which the tutor was built, that is, to eliminate memory overload and to provide the structure of the solution so as to facilitate problem solving.

1.2 In addition to clauses of the SELECT statement being displayed, the main window displays the text of the current problem as well as the schema of the chosen database. How does this additional information help you to solve problems?

The students were very pleased with the interface of the SQL-Tutor, as depicted in section 3.4.1, which provides all the information required to solve problems, namely: the text of the problem, the database schema, and the clauses of the SELECT statement, all in one screen. The major benefit of such an interface is that they were not required to access different screens in order to solve the problems.

As in 1.1 they were happy in that:

- They did not have to remember table and attribute names;
- They could access the definitions of tables and attributes using the tool tip feature of the tutor;
• They did not have to type in table and attribute names in problem solutions, since the point and click interface of the tutor allowed them to select and incorporate these names into the solution;
• They were able to access the description of the database schema by pointing and clicking;
• They made fewer typing errors; and
• They could continuously refer to the problem during the course of problem solving, since it was displayed on the same screen.

The design principle underlying the presentation of problem text and database schema on the main screen containing the structure of the solution is to reduce cognitive load and help students focus on higher-order critical thinking and problem-solving skills.

1.3 How does the SQL-Tutor assist you to learn/remember the concepts and elements of SQL?

Many students felt that the practice environment provided by the tutor and, in particular, the problem-solving activities offered them the opportunity to learn and remember the concepts and elements of SQL. They also mentioned the role of system feedback and error messages in helping them strengthen their grasp of SQL concepts. The menu-driven help facility and the point and click interface were also cited as useful sources of information. A few students suggested that the help facility should be extended to include SQL functions, and examples should be provided to illustrate how elements are used. One student suggested that background information on the SQL programming language should be incorporated for first-time users.

As discussed in section 3.5, the SQL-Tutor promotes three kinds of learning: conceptual learning, problem solving and meta-learning. It is apparent from the student responses received that they are aware of the conceptual and problem-solving learning modes, even though they did not express it in such terms. Conceptual learning involves learning about the concepts, elements, and query formulation rules by using interface controls, menu options, and system-generated feedback to student solutions. Problem-solving learning involves solving problems under the guidance of the tutor by a system of feedback where students can learn from their mistakes and remedy them. The students have overlooked the meta-learning mode, understandably so, since this mode is of a higher order and students are unaware that
system-generated error messages and the provision of system solutions are forms of self-explanation.

1.4 Does the Tutor support a 'learning by doing' approach? Elaborate.

There was an overwhelmingly positive response to the question of whether the tutor supports 'learning-by-doing'. The students elaborated that the tutor gave them the opportunity to solve problems, learn about errors in their solutions, and correct them.

From the responses received, one can conclude that the students were actively engaged in the learning process, encompassing problem solving, guided-discovery of errors, and remediation.

1.5 Do you find that the SQL-Tutor encourages you to critically evaluate your own solution to a problem and provides additional help only upon your request?

The students were in total agreement that the approach adopted by the SQL-Tutor encourages the learner to critically evaluate his/her solution to a problem and that further assistance was provided only on student request or by the system upon receipt of revised, yet still incorrect, solutions.

This is a further affirmation of one of the design goals of the tutor, namely to encourage critical thinking by affording the student, via different levels of system feedback, the opportunity of discovering and correcting errors.

1.6 Does the Tutor provide you with an adequate explanation of why your solution is incorrect?

This question was met with mixed reaction. There was consensus, that most of the time the explanation was adequate. Reservations were, however, expressed that in some instances:

- The tutoring system could not recognise alternative correct solutions; and
- As learners, they could not solve the more complex problems from the explanations given.
System explanations in the SQL-Tutor consist of error messages and the correct/ideal solution. Error messages are generated in response to syntactic and semantic errors committed by students. As discussed in section 3.4.2, domain knowledge in the SQL-Tutor is represented by constraints using constraint-based modelling (CBM) techniques. A constraint-based model represents domain knowledge as a set of constraints on correct solutions. Constraints are used to partition the universe of possible solutions into correct and incorrect ones. Two types of constraints are employed in the SQL-Tutor, namely, syntactic and semantic. The first type represents syntactic properties of queries and as such refers to only the student’s solution while the second type represents semantic properties of queries and compares the student’s solution to the ideal solution.

The students’ responses to this question call for a closer examination of the constraints encoded into the system in order to ascertain:

- Whether more constraints are needed to cater for unanticipated errors; and
- Whether there are ‘bugs’ present in the code implementing certain constraints, since error messages are directly linked to constraint violations.

1.7 Does the tutor monitor your progress and provide appropriate feedback in order for you to acquire the necessary SQL programming skills?

The majority of the students agreed that the tutor monitored their progress and provided appropriate feedback to facilitate acquisition of SQL programming skills. One student stated that, although he was given feedback, he was unaware that his progress was being monitored by the system. Another student articulated that the SQL-Tutor gives feedback, enforces concepts learnt by presenting students with problems covering the same concepts, and gives more time to practice in order to acquire mastery.

Summary

In evaluating the SQL-Tutor in respect of instructional design principles based on the cognitive learning theory [Hannafin, 1988, cited in De Villiers, 1998], one can state that the tutor:
• Supports orientation and recall of prior information by providing a help facility and all the information necessary to solve the problem on one screen;
• Supports the acquisition of SQL query formulation skills and promotes a learning-by-doing instructional strategy;
• Supports individualisation by maintaining student models;
• Supports flexible learning time by allowing students to work through the problems at their own pace without setting any time constraints; and
• Encourages positive learner attitudes by the positive tone of the feedback provided.

In addition, the evaluation reveals incorporation of cognitive strategies [Osman & Hannafin, 1992, cited in De Villiers, 1998] such as:
• Active planning, where query formulation in SQL requires learners to understand the requirements of the question and plan how to apply the elements of SQL to solve the problem;
• Self-monitoring, in that students can examine their progress via a session history feature which documents the types of errors made and the problems solved successfully; and
• Revision, where the student is given subsequent opportunities to correct errors.

2. Constructivism

This section examines how the tutor facilitates the learning of the SQL language, the presence of real-world situated learning, the active construction of knowledge, and the need for strengthening or improving on that knowledge.

2.1 How does the SQL-Tutor help you to learn the SQL programming language?

The interviewees cited various features of the SQL-Tutor that facilitated learning of the SQL programming language, namely:
• The inclusion of clauses of the SELECT statement and the relative order of the clauses;
• An environment for practising SQL skills;
• Feedback on errors made;
• The display of the database schema, problem and solution template all in one screen;
• Practising with different problems of varying complexity;
• A learning-by-doing approach; and
• Discovery of their own errors.

2.2 Do you enjoy actively participating with the tutor (as opposed to passive learning as in book or video learning)?

All respondents replied in the affirmative.

2.3 Were the problems that the tutor presented to you, real-world problems?

There was total consensus among the interviewees that the problems presented by the tutor were in fact real-world database systems such as movies, company, registration, etc.

2.4 Would it help to do additional problems? If so, how?

The learners interviewed all agreed that it would help to do more problems. They justified their answer by stating that ‘practice makes perfect’, so that their SQL query skills could be improved and consolidated.

Summary

Judging from the responses received from the learners, one can confidently say that the SQL-Tutor assists learners to construct their own knowledge by:
• Providing active and contextualised learning by actively engaging students in solving real-world problems using real-world database schemas and supplying system feedback in the context of the problem being solved;
• Providing ample opportunity for a continuous application of knowledge and concepts in a non-threatening environment, with no formalised testing or scoring; and
• Designing initial system feedback so that it is deliberately vague to facilitate student-initiative in the discovery and remediation of errors, thereby facilitating constructivist learning.
3. Customisation

This section examines whether the SQL-Tutor customises its tutoring; facilitates learning; supports personalised knowledge construction; allows student initiative in learning; encourages learners to persevere individually within their own time frames to reach goals, and develops problem-solving skills [Reigeluth, 1997, cited in De Villiers, 1998].

3.1 How did you feel about taking the initiative in learning by choosing a database, selecting problems to work on and discovering and correcting your errors under the guidance of the tutor?

The overall response to this question was very positive. The learners interviewed elaborated by stating that they:

- Felt in control of their own learning;
- Could move at their own pace;
- Could choose the problems they wanted to solve;
- Were given the guidance of the tutor when attempting to solve problems;
- Could skip certain problems and attempt them at a later time; and
- Received personalised feedback on their errors, which would not occur in a class-based situation, given the high number of students.

3.2 Does the SQL-Tutor support the individual by adapting to each learner’s profile (student model) and needs? Elaborate.

The majority of the interviewees agreed that the SQL-Tutor supports the individual by adapting to each learner’s profile (student model) and needs. They appreciated the fact that the tutor:

- Offers one-on-one tutoring;
- Keeps track of which problems they have solved and haven’t yet solved;
- Maintains a history of their performance at solving problems during a session; and
- Presents them with problems they have failed to solve or with new problems.
3.3 Do you find that maintaining a model of each learner (the tutor records all previous attempts at solving the problem) is a useful feature of the Tutor? Explain.

The answer to this question was overwhelmingly positive. The following elaboration were offered:

- Problems are provided which are similar to the ones they had difficulty with, this is made possible by the student model;
- Individualised tutoring is possible;
- The model keeps track of their errors and their learning needs, as well as problems successfully solved;
- The model shows them their strengths and helps them identify and overcome their weaknesses;
- Presents them with problems covering concepts they have not yet fully grasped and, in so doing, helps them improve;
- Knows and remembers the difficulties learners are experiencing, in contrast to a lecturer who does not have such an intimate knowledge of each learner; and
- The model offers tutoring which is not available in the lecture theatre/classroom.

3.4 Do you find that the tutor allows you to persevere individually within your own time frame to acquire the necessary SQL skills?

All the interviewees responded in the affirmative to this question. Additional comments were received such as ‘allows you to work at your own pace’, ‘Does not pressure you’, ‘No time limit’, ‘You can take as long as you want to solve the problem’, and ‘You are relaxed and this helps in learning’.

3.5 How did you find the feedback (including positive, negative, syntax and semantic error messages) to your typed solutions?

This question drew a mixture of responses. Many of the learners interviewed reported that the positive and negative feedback was both encouraging and helpful. The different levels of the system feedback ranged from ‘nice try, you only made 1 mistake’ to informing the learner where their errors are located and the nature of their errors. Concern was expressed that
some of the error messages lacked clarity in the sense that learners could not correct the solution using the information provided in the error messages. One student was not altogether convinced that the tutor accepts alternative correct solutions. Another student reported that the phraseology of the first level of feedback, namely 'nice try, you only made 1 mistake' tends to get a bit monotonous.

3.6 *Were you able to discover and remedy your errors quickly, given immediate and direct feedback?*

Most students reported in the affirmative. They found that for most of the problems they were easily able to discover and remedy their errors in the presence of immediate and direct feedback. However, for the more complex problems, they took longer to discover and remedy their errors. The students felt that discovery and remedy of errors also depended on other factors such as a sound grasp of concepts and the level of feedback chosen. One student reported that, in some instances, discovery and remedy of errors was not possible, since the system matched the student's solution against the ideal solution. Because the system recognises only certain specific solutions as being correct, other approaches are rejected.

3.7 *When did you use more advanced levels of feedback from the tutoring system?*

The students reported that they used the more advanced levels of system feedback after making repeated attempts and failing to solve the problem, when they couldn’t understand their own errors and needed additional help and information, and when they tackled the more complex problems.

3.8 *Did you, as a result of personalised instruction, get a better grasp of the sections on grouping, restricted grouping, join conditions and functions in SQL?*

There was a resounding ‘yes’ to this question. The students stated that, having initially understood the concepts in class, they then got had the opportunity of applying them by doing the tutor's problem-solving exercises. A few students felt that they needed more practice with respect to the sections on grouping and restricted grouping.
Summary

The overall response to each of these questions was positive. Due to the fact that the SQL-Tutor supports and maintains a model of each learner, customised learning is possible. The student model keeps a record of:

- Concepts that are fully understood, as indicated by successful application of those concepts in problem solving;
- Concepts that are not fully grasped by the student, as evidenced in errors made during problem solving; and
- Untested concepts, in the yet-to-be-tackled problems.

The system selects problems to be solved on the basis of the student model. In addition to system-selected problems, students have the option to seize the initiative and select the problems they want to solve, thereby taking responsibility for their own learning experience.

By providing a guided practice environment for solving problems, the SQL-Tutor sets as its primary goal, the task of developing problem-solving skills.

There are no time constraints imposed on students thereby allowing them to persevere individually within their own time frames.

4. Creativity

This section examines whether the instructional strategy of the tutor engages and motivates the learner by gaining attention, demonstrating relevance, instilling confidence and providing learner satisfaction as well as providing an innovative way of practising skill acquisition [Dick, 1995; Keller, 1987, cited in de Villiers, 1998].

4.1 What do you find different about the way the SQL-Tutor lets you practice?

The students responded with the following comments:

- Personalised attention;
- Solving problems under the guidance of the tutor;
Evaluation of the SQL-Tutor: An Intelligent Tutoring System for the database query language SQL

- Interactive one-on-one tutoring sessions;
- Working at their own pace;
- Choosing the problems they want to solve;
- Correcting errors and re-submitting revised solutions; and
- Discovering their errors.

4.2 Do you get bored quickly when you use it?

With the exception of one student, all replied in the negative. Two interviewees added that session duration impacts on boredom.

4.3 Does it give you, on the screen, all you need to solve problems? (i.e. a virtual table-top)?

There was complete consensus that the tutor provides a virtual tabletop for solving problems.

4.4 Comment on the ARCS Model (in particular, the attention and satisfaction aspects), with respect to use of the SQL-Tutor?

The learners emphasised that the SQL-Tutor held their attention and provided a satisfying learning experience.

4.5 Could you maintain concentration?

All replied that they could keep up concentration. Two students mentioned session duration as a factor impacting on concentration.

4.6 Are the problems presented by the tutor, relevant to the types of problems you are given in class assignments and tests?

The students stated that the problems were indeed relevant to those given in class assignments and tests.
4.7 Does the SQL-Tutor motivate you to make repeated attempts until you have successfully solved the problem?

All the students interviewed responded in the affirmative to the aspect of motivation and encouragement. Twenty percent mentioned that the same message repetitively, while encouraging in tone, annoyed them if they had not solved the problem after many attempts. Another student suggested that the system should be programmed so as to provide learners with the correct solution after they had made several attempts and failed.

4.8 Does it help you develop confidence in making SQL queries?

Very strong and positive response was received with relation to confidence being instilled in the learner.

4.9 How do you feel when you have successfully solved all the problems for a chosen database schema?

Some of the answers given to this question are ‘a sense of accomplishment’, ‘a sense of well-being’, and ‘a sense of achievement’.

Summary

From the responses received, there is no doubt that the SQL-Tutor engages and motivates learners and rates highly in terms of gaining attention, demonstrating relevance, instilling confidence and providing learner satisfaction [Keller, 1987, cited in de Villiers, 1998]. Furthermore, the learners appreciated the immediate and direct feedback that they received while practising with the SQL-Tutor, and the provision of a virtual-table top for problem solving.

5. Component Display Theory

This section examines whether the instructional strategy chosen for the SQL-Tutor is appropriate in terms of its instructional goals [Merrill, 1983, 1996a as cited in de Villiers, 1998].
5.1 The goal of the SQL-Tutor is to provide a guided-discovery learning environment. With reference to this goal, do you find the learning strategy employed (students apply their knowledge of SQL to solve problems, discover and remedy their errors under guidance) is appropriate for improving SQL programming skills and in keeping with the goal of the tutor?

All the learners agreed that the learning strategy employed is appropriate for improving SQL programming skills and is consistent with the instructional goal of the tutor, which is to provide a guided discovery-learning environment.

Summary

The instructional strategies employed by the tutor included: performance categories of 'remember' (knowledge and concepts covered in class and available via the tutor's help facility), 'use' (apply knowledge to solve problems), and 'find' (discover errors under the guidance of the tutor). These strategies were in keeping with the instructional goals of the SQL-Tutor.

6. Collaborative Learning

This section examines the need for, and the suitability of collaborative learning, with the SQL-Tutor.

6.1 Would it help to work together with another/other student(s) when using the SQL-Tutor?

Forty percent felt that it helps to work together with another student or other students when using the SQL-Tutor, especially for the more complex problems. The remaining 60% felt that it would not help to work with another/other students when using the tutor. One student in this group elaborated by saying that, with collaborative work, individual students would not be able to pinpoint what they don't know and understand.
6.2 Did you ever choose to work together with another/other student(s) while using the tutor? If so, describe the interaction.

Sixty percent of the respondents reported that they had consulted with another student while using the tutor. They interacted when they thought they were right and the system indicated they were wrong, when they needed clarification on errors made, where there was a common problem that both learners couldn’t solve individually, and when they couldn’t correct the error(s) diagnosed by the system.

Summary

From the results of the student evaluation, it is apparent that while a minority saw the need for interaction, the majority did interact for reasons outlined above.

It should be noted that the categories do overlap, in that certain concepts are applicable to more than one sub-heading. By and large, the SQL-Tutor shows strong adherence to the characteristics and principles advocated in the Hexa-C Metamodel.

6.4.2.8 General

This section examines students' responses to open-ended questions of a general nature with respect to the SQL-Tutor:

7.1 How do you feel about connecting the SQL-Tutor to a DBMS so that you can inspect tables and query results?

The students believed that this indeed should be done, so that the contents of general databases could be accessed and queried, with the facility to view the output produced by their queries.
7.2 Are there any facilities/features lacking in the tutor that you would like to see added to make it more effective?

The students suggested that the following should be added or improved in order to make the tutor more effective:

- A print facility;
- Use of multimedia;
- Clear error messages;
- Selection of problems based on student's prior performance;
- Test and scoring facility;
- Index to search the system;
- Recognition of alternative correct solutions;
- The incorporation of background information for first-time users.

Students were invited to make any further comments (favourable or otherwise) about the SQL-Tutor, including:

- The impression it made on them
- Frustrations they experienced
- In what way it helped
- Anything else they wanted us to know

Many comments were forthcoming and are summarised below.

On the positive side:

- The tutor was helpful and useful in that they were learning the same topic in class, so working with the tutor made a big difference in their understanding;
- The tutor was helpful and easy to use;
- The tutor provided a powerful, yet simple, computer-based training tool;
- The tutor's immediate and direct response to their solutions was regarded as a big advantage;
- They could access the system's solution immediately after attempting their own;
- The tutor provided practice problems, which enhanced their understanding of SQL;
• The error messages and ideal solutions provided by the tutor helped them to learn about and correct their own errors;
• They found it more fun than class-based learning;
• They could work at their own pace;
• The tutor diagnosed their answers as correct or incorrect and gave them additional information as to where their errors occurred, as well as a description of the errors made. They added that a lecturer could not achieve this in a classroom situation, given the high number of students;
• The tutor instilled a positive attitude about SQL;
• The SQL-Tutor provided a positive learning aid, which allowed them to work in an environment with little pressure;
• The SQL tutoring system is robust; and
• They found the SQL-Tutor to be a very good and impressive tutor.

On the negative side:
• The tutor was not user-friendly;
• Working with the tutoring system tended to get a bit tedious at times;
• The tutor did not satisfactorily adapt to different ways of solving problems; and
• It did not accept certain alternative correct answers.

Suggestions for improvement:
• The screen design should be made more aesthetically appealing;
• Error checking should be extended;
• Help should be more informative; and
• General information on SQL programming should be included to inform first-time users before being introduced to the practice environment.

These comments represent an overall judgement made by students on the functionality and usability of the SQL-Tutor. They have extolled its many virtues and at the same time provided positive criticism in the form of useful suggestions for improvement from a student perspective.
6.5 Conclusion

The application of the proposed methodology to the evaluation of the SQL-Tutor was successfully executed. Each phase of the evaluation framework was addressed in a major section of this chapter, covering pre-defined activities and producing specific outputs. Table 6.7 summarises this information:

Table 6.7 Application of evaluation framework

<table>
<thead>
<tr>
<th>Phase</th>
<th>Section</th>
<th>Discussion</th>
<th>Output / Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptive Modelling Phase</td>
<td>Section 6.2</td>
<td>Addresses SQL-Tutor's genealogy (ancestry) and problem habitat (uses).</td>
<td>Descriptive models: ITS/SQL Tutor genealogy (Fig 6.1): SQL-Tutor is a product of ITS technology; ITS is ancestrally related to Learning System technology specification.</td>
</tr>
<tr>
<td>Evaluation design phase</td>
<td>Section 6.3</td>
<td>Uses descriptive models to provide a set of evaluation criteria (for internal &amp; external evaluation); hypotheses and planned experiment (for external evaluation).</td>
<td>Evaluation criteria; Hypotheses; Experimental design</td>
</tr>
<tr>
<td>Evaluation implementation phase:</td>
<td>Section 6.4.1</td>
<td>Evaluates learning achievement by conducting experiment, analysing experimental evidence and reporting on findings. Evaluates learning affect via observation, questionnaires &amp; interviews.</td>
<td>Result of ‘no significant difference in academic performance’ for three groups, drawn from same class with same lecturer and instructional materials, but with different methods of solving SQL practice problems. Confirmation of the first null hypothesis (learning achievement) and refutation of the second null hypothesis (learning affect).</td>
</tr>
<tr>
<td>internal evaluation</td>
<td>Section 6.4.2</td>
<td>Evaluation of behavioural properties i.r.o.:</td>
<td>Results / Recommendations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. Domain model</td>
<td>Table 6.8 below.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Tutoring model</td>
<td>Table 6.9 below.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Student model</td>
<td>Table 6.10 below.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. Overall system control</td>
<td>Table 6.11 below.</td>
</tr>
</tbody>
</table>
General discussion and specific conclusions/recommendations are presented in tabular-form for each of the architectural components and features investigated in the internal evaluation of the SQL-Tutor. For purposes of clarity and readability, separate tables will be employed for each aspect of internal evaluation.

**Table 6.8 Domain Model**

<table>
<thead>
<tr>
<th>General discussion</th>
<th>Specific Conclusions / Recommendations</th>
</tr>
</thead>
</table>
| The *domain model*:
  - Maintains knowledge on the SELECT statement of the SQL database language;
  - Permits students to formulate single and multiple table queries for a chosen database and a set of practice problems;
  - Provides an explanation of a student solution via its feedback mechanisms;
  - Does not support alternative explanations of the same concept;
  - Does not answer arbitrary questions in SQL; and
  - Covers knowledge of common misconceptions in an implicit manner, by storing domain knowledge as a set of constraints on correct solutions. | The domain model of the SQL-Tutor could be extended with respect to:
  - Its scope, by including table creation and table maintenance;
  - Providing explanations for ideal system solutions;
  - Offering alternative explanations of the same concept; and
  - Answering arbitrary questions about the elements and concepts of the SQL database language. |
### Table 6.9 Tutoring model

<table>
<thead>
<tr>
<th>General discussion</th>
<th>Specific Conclusions / Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The tutoring model</strong> of the SQL-Tutor:</td>
<td>The tutoring model could be further developed to provide the following features/facilities:</td>
</tr>
<tr>
<td>• Has distinct teaching goals and instructional strategies;</td>
<td>• The teaching goals and instructional strategies should be made transparent to the student;</td>
</tr>
<tr>
<td>• Does not support alternative teaching strategies;</td>
<td>• Students should be aware that the philosophy underlying the practice environment is to encourage</td>
</tr>
<tr>
<td>• Tailors teaching strategies to the learning needs of the student;</td>
<td>self-discovery and remediation of errors;</td>
</tr>
<tr>
<td>• Intervenes after the student completes the problem-solving task;</td>
<td>• Successful usage of the system and correct perceptions of its feedback mechanisms depend on informed</td>
</tr>
<tr>
<td>• Allows students to initiate some new area of investigation within its limited scope;</td>
<td>student users;</td>
</tr>
<tr>
<td>• Actively engages the student in problem-solving activities;</td>
<td>• Alternative correct solutions should be stored in the system;</td>
</tr>
<tr>
<td>• Implicitly relates a diagnosed error to a misconception or a missing concept;</td>
<td>• Students should be given guidance in respect of their own methods for solving problems instead of being</td>
</tr>
<tr>
<td>• Does not incorporate remedial strategies in order to provide alternative remedial</td>
<td>forced to conform to the system’s methods;</td>
</tr>
<tr>
<td>teaching styles;</td>
<td>• The system should enquire of students whether they wish to discover their own errors, as many students</td>
</tr>
<tr>
<td>• Initial system feedback is deliberately vague to facilitate student-initiative in</td>
<td>are not inclined to do so;</td>
</tr>
<tr>
<td>the discovery and remediation of errors, thereby facilitating constructivist</td>
<td>• The system should let the students decide whether they prefer remedial instruction on a single or</td>
</tr>
<tr>
<td>learning;</td>
<td>multiple error(s) at one time;</td>
</tr>
<tr>
<td>• Tutoring guidance in the form of error flags and hints is provided only in the</td>
<td>• Students should be given remediation exercises to reinforce concepts with which they are struggling;</td>
</tr>
<tr>
<td>face of continued failure to solve problems;</td>
<td>and</td>
</tr>
<tr>
<td>• Employs a pedagogical strategy to inform students of the number of errors present</td>
<td>• The tutor could be extended beyond its role as a practice environment, to allow students to pose</td>
</tr>
<tr>
<td>in their solutions and targets only one error at a time for remedial instruction;</td>
<td>general questions about the elements, functions, expressions, set operations, joining, grouping and</td>
</tr>
<tr>
<td>• Serves as an implicit teacher in that its rich feedback mechanisms allows students</td>
<td>restricted grouping, and other complexities of SQL, beyond the facilities currently provided in the HELP</td>
</tr>
<tr>
<td>to reinforce concepts learnt;</td>
<td>menu.</td>
</tr>
</tbody>
</table>
Table 6.10 Student model

<table>
<thead>
<tr>
<th>General discussion</th>
<th>Specific Conclusions / Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>The student model of the SQL-Tutor:</td>
<td>The student model could be improved by:</td>
</tr>
<tr>
<td>• Maintains general information about the student, a history of previously solved problems, and the usage of constraints in solving problems;</td>
<td>• Incorporating information on student’s learning preferences;</td>
</tr>
<tr>
<td>• Maintains a history of each constraint, in terms of how often it was relevant to the ideal solutions, how often it was relevant to the students’ solutions, and how often it was violated or satisfied;</td>
<td>• Using information on past learning experiences to inform the tutoring component on what remedial instruction/exercises to provide, as well as when to provide such instruction; and</td>
</tr>
<tr>
<td>• Does not maintain information on the learning preferences of the student;</td>
<td>• Making the remedial instruction process transparent to the student.</td>
</tr>
<tr>
<td>• Stores information about the student’s level of advancement;</td>
<td></td>
</tr>
<tr>
<td>• Records constraint coverage by students;</td>
<td></td>
</tr>
<tr>
<td>• Provides feedback to students’ revised solutions;</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.11 Overall system control

<table>
<thead>
<tr>
<th>General Discussion</th>
<th>Specific Conclusions / Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>An analysis of student feedback, with respect to the overall system control of the SQL-Tutor, reveals that:</td>
<td>Overall system control could be improved by:</td>
</tr>
<tr>
<td>• System feedback is helpful;</td>
<td>• Incorporating some sort of system-generated error message if student errors cannot be diagnosed by the system;</td>
</tr>
<tr>
<td>• All detected errors are treated by the system;</td>
<td>• Students should not be able to access the complete solution to a problem without making a reasonable number of attempts to solve the problem;</td>
</tr>
<tr>
<td>• The system does not respond if it cannot diagnose an error;</td>
<td>• Giving due recognition to student learning preferences; and</td>
</tr>
<tr>
<td>• The system intervenes to remediate a misconception or missing concept after a student has submitted a solution; and</td>
<td>• Improving clarity of system-generated feedback.</td>
</tr>
<tr>
<td>• The system adapts to the learning needs of students but does not give recognition to the learning preferences of students.</td>
<td></td>
</tr>
</tbody>
</table>
Table 6.12 Design principles

<table>
<thead>
<tr>
<th>General Discussion</th>
<th>Specific Conclusions / Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student feedback, in respect of design principles, demonstrates that the SQL-Tutor is:</td>
<td>There is scope for improvement in terms of making the system:</td>
</tr>
<tr>
<td>• Fairly robust;</td>
<td>• Largely robust;</td>
</tr>
<tr>
<td>• Largely helpful;</td>
<td>• More powerful by incorporating state-of-the-art graphics;</td>
</tr>
<tr>
<td>• Very simple to use;</td>
<td>• More redundant by maintaining alternative representations of the same knowledge; and</td>
</tr>
<tr>
<td>• Perspicuous;</td>
<td>• Less docile; and</td>
</tr>
<tr>
<td>• Lacking in power, with a concomitant absence of state-of-the-art graphics;</td>
<td>• More transparent in terms of teaching goals and strategies.</td>
</tr>
<tr>
<td>• Navigable;</td>
<td></td>
</tr>
<tr>
<td>• Consistent;</td>
<td></td>
</tr>
<tr>
<td>• Flexible;</td>
<td></td>
</tr>
<tr>
<td>• Lacking in redundancy; and</td>
<td></td>
</tr>
<tr>
<td>• Perhaps docile, and omniscient.</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.13 Usability Properties

<table>
<thead>
<tr>
<th>General discussion</th>
<th>Specific conclusions / recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student feedback, in respect of usability principles &amp; paradigms, demonstrates that the SQL-Tutor:</td>
<td>The results of student feedback, with respect to the usability of the SQL-Tutor, indicate that usability</td>
</tr>
<tr>
<td>• Is very easy to use;</td>
<td>principles and properties were, by and large, applied to good effect.</td>
</tr>
<tr>
<td>• Promotes an interaction dialogue that is task-oriented, natural and fairly effective;</td>
<td>Suggestions for improvement:</td>
</tr>
<tr>
<td>• Uses screens that are not overloaded with information; and</td>
<td>• Screen design should be more appealing; and</td>
</tr>
<tr>
<td>• Uses interaction styles that supports windows, menus and tool-tip features, as well as a point-and click interface;</td>
<td>• Tutor’s role in interaction dialogue could be improved to serve the purpose of remedial instruction.</td>
</tr>
</tbody>
</table>
Table 6.14 Learning and instructional theories

<table>
<thead>
<tr>
<th>General discussion</th>
<th>Specific conclusions / recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student feedback, in respect of learning &amp; instructional theories, indicates that the SQL-Tutor:</td>
<td>From the results of student feedback, with respect to the SQL-Tutor’s conformance to the Hexa-C Metamodel, it appears that the tutor strongly supports the characteristics and principles advocated in the model.</td>
</tr>
<tr>
<td>Reduces memory &amp; cognitive overload;</td>
<td></td>
</tr>
<tr>
<td>Supports problem-solving and skill acquisition;</td>
<td></td>
</tr>
<tr>
<td>Requires student to interpret problems, plan, execute and revise solutions and monitor self progress;</td>
<td></td>
</tr>
<tr>
<td>Promotes real-world situated learning;</td>
<td></td>
</tr>
<tr>
<td>Encourages active construction of knowledge;</td>
<td></td>
</tr>
<tr>
<td>Facilitates learning by implicit teaching of domain concepts;</td>
<td></td>
</tr>
<tr>
<td>Customises its tutoring;</td>
<td></td>
</tr>
<tr>
<td>Demonstrates relevance;</td>
<td></td>
</tr>
<tr>
<td>Instils confidence;</td>
<td></td>
</tr>
<tr>
<td>Provides learner satisfaction;</td>
<td></td>
</tr>
<tr>
<td>Chooses instructional strategies in keeping with its goals;</td>
<td></td>
</tr>
<tr>
<td>Supports flexible learning time; and</td>
<td></td>
</tr>
<tr>
<td>Requires occasional collaboration.</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.15 General

<table>
<thead>
<tr>
<th>General discussion</th>
<th>Specific conclusion / recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the <strong>positive side</strong>, the SQL-Tutor was found to be:</td>
<td>Suggestions for improvement:</td>
</tr>
<tr>
<td>Helpful and useful;</td>
<td>• The screen design should be made more aesthetically appealing;</td>
</tr>
<tr>
<td>A powerful, yet simple and easy to use, computer-based training tool;</td>
<td>• Help facility should be extended beyond its current capabilities; and</td>
</tr>
<tr>
<td>More fun than class-based learning;</td>
<td>• Background information should be provided for first-time users.</td>
</tr>
<tr>
<td>A positive learning aid, which allowed students to work in an environment with little pressure;</td>
<td>• Inclusion of a print facility, test and scoring facility, as well as index to search the system;</td>
</tr>
<tr>
<td>On the <strong>negative side</strong>, the tutor:</td>
<td>• Use of multimedia; and</td>
</tr>
<tr>
<td>Tidious at times;</td>
<td>• Connection to a DBMS so that tables could be inspected and results queried.</td>
</tr>
</tbody>
</table>
The overall results of student feedback, with respect to internal evaluation, demonstrate that the SQL-Tutor:

- Has a well-defined architecture;
- Offers satisfactory tutoring, according to the criteria defined for a complete intelligent tutoring system;
- Adheres to generic or universal design principles of computer tutor construction;
- Applies usability principles and paradigms to good effect;
- Displays strong conformance to current learning and instructional theories.
CHAPTER SEVEN

Conclusion

'Building a tutor and not evaluating it is like building a boat and not taking it in the water' [Shute & Regian, 1993]. Evaluation can generate the same excitement as development, its results may be surprising if not humbling, yet with careful experimental design they will always be informative.

This MSc half-dissertation was an in-depth study of a highly sophisticated technology, namely ITS applications. It investigates important issues and milestones in their development, their implementation and architectures, behavioural properties, underlying learning and instructional theories, and in particular, issues surrounding their evaluation. These evaluation issues involve the identification of required characteristics and evaluation criteria for ICAI and ITSs, applied within an appropriate framework. The ITS technology overlaps with fields such as computer science, cognitive psychology, and educational research, which adds to its complexity.

The main problem that gave rise to this study is the low availability of literature and expertise on the practice of ITS evaluations, and the lack of effort expended on developing a generic methodology for evaluating ITSs within actual educational settings. The dual goals of the study were therefore to develop a generic methodology for evaluating ITSs, and to apply this methodology to the evaluation of an ITS for the database query language SQL. A generic methodology would integrate appropriate evaluation criteria with approaches, methods and techniques for evaluation, within a framework for evaluation. In addition, it would facilitate the transfer of this technology from its research fold into the educational sector. The potential of ITSs has to be realised and an evaluation methodology that is both comprehensive and rigorous can help to achieve this.
7.1 What has been accomplished?

This dissertation makes two important contributions to the evaluation of intelligent tutoring systems, namely:

1. It proposes a *generic methodology* encompassing internal and external approaches to evaluation; the use of sound evaluation criteria; the application of general methods and techniques within a framework for evaluating added value of a technology; and the adoption of principles for a good evaluative study.

2. It describes a complete *application of the methodology* to the evaluation of the SQL-Tutor, an ITS for the database query language SQL, in order to reveal its strengths and shortcomings.

Other subsidiary contributions include:

- A close examination of traditional, classical, new generation, and novel architectures for the development of ITSs;
- The relationship between the architecture of an ITS and its behaviour;
- The use of learning achievement and learning affect to assess the educational impact of ITSs;
- The adoption of a user-centred approach for evaluation coupled with a minimal degree of heuristic evaluation;
- The impact of current directions in learning and instructional theories on the development and evaluation of ITSs;
- The important role of the user interface and usability issues in computer based tutors; and
- The incorporation of universal principles for computer tutor construction;

7.2 Generic methodology for evaluating ITSs

The proposed evaluation methodology, framework and criteria are discussed in detail in Chapter Five. It has been demonstrated that internal and external approaches are more appropriate for the evaluation of ITSs than the traditional approaches of formative and summative evaluation. This is because researchers continue to experiment with various
forms of student modelling and knowledge representation techniques, with the intent to overcome the difficulties experienced in these areas. Hence ITSs are, to a large degree, still steeped in research and development with relatively few products being formally used in educational settings. The evaluation methods chosen, namely the use of criterion-based techniques for internal evaluation and experimental research for external evaluation, are general and simple to apply to the evaluation of any ITS. These methods are not prescribed to the exclusion of other evaluation methods, which provide valuable insights into the role of teaching and learning afforded by ITSs. The framework for evaluating ITSs is valuable in that it enables a comparison of the features and internal structure of the ITS technology with those of its historical precedents and peers, integrates different techniques for evaluation, and allows IT professionals and educators to make informed decisions about its added value.

7.3 Results and recommendations from evaluation of SQL-Tutor

A detailed analysis of the evaluation of SQL-Tutor according to the proposed methodology is provided in Chapter Six. The educational impact of the SQL-Tutor in terms of learning affect - motivation and satisfaction - was overwhelmingly positive, highlighting the affective value of the tutor in a real-world learning and practice situation.

With respect to learning achievement, a result of 'no significant difference in academic performance' was found when comparing the test results of 3 groups using different treatments. This result is not consistent with a previous evaluation of the SQL-Tutor performed by the developer, who reported 'a significant difference in academic achievement' [Mitrovic & Ohlsson, 1999]. The study conducted by Mitrovic compared the achievement of students who participated in the study with those who did not, with respect to their scores on a subsequent examination. It should be noted that random selection was not used, as the subjects in the experimental group were volunteers. Furthermore, the other subjects did not really constitute a control group, since they were not prevented from accessing or using the SQL-Tutor before the examination. The use of volunteers does open up the possibility that the experimental group subjects are the more enthusiastic, brighter, and keener students that would have, in any event, scored better of the two groups.
The ‘no significant difference phenomenon’ relates to the observation that media do not influence learning achievement, but that the kind of media used does impact strongly on aspects such as quality of learning, motivation, the knowledge construction process, and retention. There have been many reports of the ‘no significant difference phenomenon’ in respect of CAI/CBT training, ITS individualised instruction, on-line teaching and learning, distance education, cyber-learning, and virtual learning. The influence of media on academic achievement is hotly debated and there is a strong argument that the use of different media is not a decisive factor in learning achievement.

The results of student feedback to internal evaluation revealed the strengths and shortcomings of specific components or features of the SQL-Tutor. The following list offers suggestions for possible improvement:

- The incorporation of an expertise or problem-solver module to solve problems in SQL; the problem-solver must know a variety of algorithms used for problem-solving so that feedback can be generated which is relevant to the algorithm selected by the student;
- The use of some form of natural language processing in order to interpret problems and to answer queries posed by students, thereby achieving a more mixed-initiative dialogue between the learners and the system;
- Connection to a DBMS so that students can inspect values in database tables and view the output of queries (already accomplished in latest version of the SQL-Tutor);
- More tutorial support for novice learners;
- Extension of current functionality and usability; and
- Incorporation of other modes of learning, since discovery-learning may not be suitable for all types of students.

Finally, further research is required in terms of problem selection strategies and alternative/remedial instructional strategies.

7.4 Conclusion

There is no doubt that evaluation of educational technology feeds into the development cycle and informs the practice, without which there can be no significant progress and ultimate fulfilment of learner needs. Evaluation hence plays a crucial role in promoting faster diffusion of the ITS technology into the real-world of education and training.
The ultimate goal of this research would be achieved if this methodology, or aspects thereof were generally applied in evaluating ITS and ICAI applications in authentic settings of instruction and learning. The purpose of such evaluations should be to recommend and effect appropriate corrections, modifications and extensions to the components and functionality of the systems, so as to realise the full potential of artificial intelligence applications in facilitating education and training.
Bibliography


Bitter, G. and D. Wighton. 1987. The most important criteria used by the educational software evaluation consortium. The computing teacher, 14(6), 7-9.


Appendix A.1

Pre-test

The shorthand representation and description for the Movies database appears below. Use this representation to answer the questions that follow:

DIRECTOR (DIRNUMB, DIRNAME, DIRBORN, DIRDIED)

STAR (STARNUMB, STARNAME, BRTHPLCE, STARBORN, STARDIED)

MOVIE (MVNUMB, MVTITLE, YRMDE, MVTYPE, CRIT, MPAA, NOMS, AWRD, DIRNUMB)

MOVSTAR (MVNUMB, STARNUMB)

MEMBER (MMBNUMB, MMBNAME, MMBADDR, MMBCTY, MMBST, NUMRENT, BONUS, JOINDATE)

TAPE (TPNUMB, MVNUMB, PURDATE, TMSRNT, MMBNUMB)

The table DIRECTOR contains information on movie directors. Each director’s record contains the number assigned to him/her, his/her name, the year in which the director was born, and if appropriate, the year in which he/she died.

The STAR table keeps information on actors and actresses. Each row in this table contains the star’s number, name, birthplace, the year in which the star was born, and again, if appropriate, the year in which he or she died.

The movie table contains the following details: movie number, title, year produced, type of movie (comedy, drama, science fiction, and so on), critics’ rating, MPAA rating, number of academy award nominations received, number of academy awards won, and director number.

The table MOVSTAR is used to relate actors and actresses to the movies in which they appeared.

The MEMBER table keeps information on the video rental store members. Members’ numbers, names and addresses, the number of rentals the member has made, the number of bonus units the member is currently qualified for, and the date the member joined the club.

Information on the videotapes that the club owns is also maintained in the TAPE table. Details such as videotape number, the date the tape was purchased, the number of times the tape has been rented, and the number of the member who is currently renting the tape.
Write SQL commands to perform the following operations:

1. List the numbers and titles of all movies whose MPAA rating is PG and that were nominated for at least one academy award. (4)

2. List the numbers and titles of all movies whose type is Comedy, Religion, or Suspense. Use the SQL word IN when formulating the query. (4)

3. Find the numbers and names of all directors whose first name is Stanley. (4)

4. List the numbers and names of all directors who are still living. (3)

5. List the member numbers and names of all members who have rented between 10 and 20 tapes. (4)

6. List the numbers, names, addresses, and join dates of all members. Sort the output by join date. List the member who joined most recently first. (4)

7. Find out how many movies are of type Suspense, the total number of awards for which these movies were nominated, and the total number of awards they won. (6)

8. Find the maximum number of awards that any movie directed by director 5 was nominated for. Find all movies (any director) that won at least as many awards, as this maximum. (Do it with a single query that utilises a sub-query) (4)

9. List the numbers and names of all stars that have appeared in any movies directed by Alfred Hitchcock. (Hint: Director names are stored with the last name first. Thus you must search for Hitchcock, Alfred, for example.) (5)

10. List the numbers and names of any pairs of movies that are of the same type and have the same director. (7)

11. List the tape number and movie number for all tapes on which the movie is a comedy but are currently rented by someone other than Mark Peterson. (7)

12. List the tape number and movie number for all tapes on which the movie is a Comedy or that are currently rented by Mark Peterson (Peterson, Mark). (8)

13. For each director, list the director's number and the total number of awards won by comedies (movies whose type is Comedy) he or she has directed. List only the directors for whom the total number of awards is at least 1. (7)
Given the following relations and their respective descriptions, use SQL commands to satisfy the following queries:

**INVENTORY** (WAREHOUSE-NAME, PART-NO, QTY-IN-STOCK)

**PARTS** (PART-NO, DESCRIPTION, WEIGHT, MAX-DIM)

**REQUISITIONS** (REQ-NO, PROJ-NO, DATE-NEEDED, WHERE-NEEDED)

**REQ-LINES** (REQ-NO, PART-NO, QTY-NEEDED, QTY-FILLED)

**WAREHOUSES** (WAREHOUSE-NAME, LOCATION, MANAGER)

The database stores requisitions for parts. The requisitions are stored in relation REQUISITIONS and requisition contents are stored in REQ-LINES. The requisition data includes the number of the project that made the requisition, the date when the requested items are needed and the location where they are needed. Each occurrence in REQ-LINES represents a line in a requisition. Attribute REQ-NO identifies the requisition, and PART-NO defines the part ordered in that line. Each requisition line is for one part and there may be many requisition lines in one requisition. Part descriptions are stored in the relation PARTS. Relation INVENTORY stores the quantity of each part held in the organisation’s WAREHOUSES.

14. List all the requisition numbers for part descriptions equal to ‘drill’ and that are made by project ‘Project1’.

15. List all the requisition lines (giving REQ-NO and PART-NO) for which the entire number of parts requested can be fulfilled from one warehouse.

16. List all the requisition numbers that do not include part number ‘X11’.
Appendix A.2

Model Answer: Pre-test

Write SQL commands to perform the following operations:

1. SELECT MVNUMB, MVTITLE
   FROM MOVIE
   WHERE MPAA = 'PG'
   AND NOMS >= 1
   (4)

2. SELECT MVNUMB, MVTITLE
   FROM MOVIE
   WHERE MVTYPE IN ('Comedy', 'Religion', 'Suspense')
   (4)

3. SELECT DIRNUMB, DIRNAME
   FROM DIRECTOR
   WHERE DIRNAME LIKE 'Stanley%
   (4)

4. SELECT DIRNUMB, DIRNAME
   FROM DIRECTOR
   WHERE DIRDIED IS NULL
   (3)

5. SELECT MMBNUMB, MMBNAME
   FROM MEMBER
   WHERE NUMRENT BETWEEN 10 AND 20
   (4)

6. SELECT MMBNUMB, MMBNAME, MMBADDR, MMBCITY, MMBST,
   JOINDATE
   FROM MEMBER
   ORDER BY JOINDATE DESC
   (6)

7. SELECT COUNT (*), SUM (NOMS), SUM (AWRD)
   FROM MOVIE
   WHERE MVTYPE = 'Suspense'
   (4)

8. SELECT MVTITLE
   FROM MOVIE
   WHERE AWRD >=
   (SELECT MAX (NOMS)
    FROM MOVIE
    WHERE DIRNUMB = 5)
   (5)
9. SELECT STAR.STARNUMB, STARNAME 
    FROM STAR, MOVSTAR, MOVIE, DIRECTOR 
    WHERE STAR.STARNUMB = MOVSTAR.STARNUMB 
    AND MOVSTAR.MVNUMB = MOVIE.MVNUMB 
    AND MOVIE.DIRNUMB = DIRECTOR.DIRNUMB 
    AND DIRNAME = 'Hitchcock, Alfred'

10. SELECT F. MVNUMB, F.MVTITLE, S.MVNUMB, S.MVTITLE 
    FROM MOVIE F, MOVIE S 
    WHERE F.MVTYPE = S.MVTYPE 
    AND F.DIRNUMB = S.DIRNUMB 
    AND F.MVNUMB < S.MVNUMB

11. SELECT TPNUMB, TAPE.MVNUMB 
    FROM MOVIE, TAPE 
    WHERE MOVIE.MVNUMB = TAPE.MVNUMB 
    AND MVTYPE = 'Comedy' 
    AND MMBNUMB NOT IN 
    (SELECT MMBNUMB 
    FROM MEMBER 
    WHERE MMBNAME = 'Peterson, Mark')

12. SELECT TPNUMB, MOVIE.MVNUMB 
    FROM TAPE, MOVIE 
    WHERE TAPE.MVNUMB = MOVIE.MVNUMB 
    AND MVTYPE = 'Comedy' 
    UNION 
    SELECT TPNUMB, MVNUMB 
    FROM TAPE, MEMBER 
    WHERE TAPE.MMBNUMB = MEMBER.MMBNUMB 
    AND MMBNAME = 'Peterson, Mark'

13. SELECT DIRNUMB, SUM (AWRD) 
    FROM MOVIE 
    WHERE MVTYPE = 'COMEDY' 
    GROUP BY DIRNUMB 
    HAVING SUM (AWRD) >=1

14. SELECT REQ-NO 
    FROM REQUISITIONS 
    WHERE PROJ-NO = 'Project1' 
    AND REQ-NO IN 
    (SELECT REQ-NO 
    FROM REQ-LINES 
    WHERE PART-NO IN 
    (SELECT PART-NO 
    FROM PARTS 
    WHERE DESCRIPTION = 'drill'))
15. SELECT REQ-NO, PART-NO
    FROM REQ-LINES
    WHERE PART-NO IN
        (SELECT PART-NO
            FROM INVENTORY
            WHERE QTY-IN-STOCK > REQ-LINES.QTY-NEEDED)

16. SELECT REQ-NO
    FROM REQUISITIONS
    WHERE REQ-NO NOT IN
        (SELECT REQ-NO
            FROM REQ-LINES
            WHERE PART-NO = 'X11')
Appendix B.1

Post-test

The shorthand representation and description for the Movies database appears below. Use this representation to answer the questions that follow:

DIRECTOR (DIRNUMB, DIRNAME, DIRBORN, DIRDIED)

STAR (STARNUMB, STARLNAME, STARFNAME, BRTHPLCE, STARBORN, STARDIED)

MOVIE (MVNUMB, MVTITLE, YRMDE, MVTYPE, CRIT, MPAA, NOMS, AWRD, DIRNUMB)

MOVSTAR (MVNUMB, STARNUMB)

MEMBER (MMBNUMB, MMBNAME, MMBADDR, MMBCTY, MMBST, NUMRENT, BONUS, JOINDATE)

TAPE (TPNUMB, MVNUMB, PURDATE, TMSRNT, MMBNUMB)

The table DIRECTOR contains information on movie directors. Each director’s record contains the number assigned to him/her, his/her name, the year in which the director was born, and if appropriate, the year in which he/she died.

The STAR table keeps information on actors and actresses. Each row in this table contains the star’s number, name, birthplace, the year in which the star was born, and again, if appropriate, the year in which he or she died.

The movie table contains the following details: movie number, title, year produced, type of movie (comedy, drama, science fiction, and so on), critics’ rating, MPAA rating, number of academy award nominations received, number of academy awards won, and director number.

The table MOVSTAR is used to relate actors and actresses to the movies in which they appeared.

The MEMBER table keeps information on the video rental store members. Members’ numbers, names and addresses, the number of rentals the member has made, the number of bonus units the member is currently qualified for, and the date the member joined the club.

Information on the videotapes that the club owns is maintained in the TAPE table. This table contains details such as videotape number, the date the tape was purchased, the number of times the tape has been rented, and the number of the member who is currently renting the tape.
Write SQL commands to perform the following operations:

1. List the titles and number of movies that have won at least one academy award and have been made in or after 1988.

2. List the number and titles of all movies whose type is Comedy or Drama or Suspense.

3. For all members who live in Ilam, list their number and name.

4. List the names of all directors who are still living.

5. Retrieve the names of all directors born during the 1950's.

6. List the numbers, names, addresses, and join dates of all members. Sort the output by number descending and name ascending.

7. Find out how many comedies there are and how many awards they won.

8. List the numbers and names of all members who have rented more tapes than the average.

9. List the titles of all movies directed by Stanley Kubrick. Use the subquery technique.

10. List the name and addresses of all members currently renting Mel Brooks movies. Use the joining technique.

11. Find a list of any pairs of stars who have the same first name.

12. List the movie number and title for all movies that were nominated for more academy awards than any movie directed by Woody Allen.

13. For all directors who made more than 5 movies, list their number, names and total number of movies.

Use the following table definitions for the questions that follow:

```sql
CREATE TABLE ROOM
  (LOCATION     CHAR (4),
   ACCOM        CHAR (2),
   EXTENSION   SMALLINT,
   PATIENT-NO  INTEGER);

CREATE TABLE PATIENT
  (PATIENT-NO  INTEGER,
   PATIENT-NAME CHAR (15),
   DATE-DISCH   DATE,
   ... other data elements......)
```
CREATE TABLE PHY
    (PHY-ID CHAR (10),
     PHY-PHONE CHAR (8));

CREATE TABLE ITEM
    (ITEM-CODE SMALLINT,
     DESCRIPT CHAR (15),
     STD-CHG DECIMAL (7,2));

CREATE TABLE ATTENDS
    (PHY-ID CHAR (10),
     PATIENT-NO INTEGER,
     PROCEDURE CHAR (15));

CREATE TABLE BILLED
    (PATIENT-NO INTEGER,
     ITEM-CODE SMALLINT,
     CHARGE DECIMAL (7,2));

14. Which patients (display patient name) have been charged for the item Television? (6)

15. Which patients (display patient number) have not been treated by Dr Jefferson? (5)

16. Produce a list of charges for all patients (displaying patient number, name and charge) even those for whom there were no charges so far. (8)
APPENDIX B.2

Model Answer: Post-test

1. Select MVNUMB, MVTITLE
   FROM MOVIE
   WHERE AWRD >= 1 AND YRMDE >= 1988
   (4)

2. SELECT MVNUMB, MVTITLE
   FORM MOVIE
   WHERE MVTYPE IN ('Comedy', 'Drama', 'Suspense')
   (4)

3. SELECT MMBNUMB, MMBNAME
   FROM MEMBER
   WHERE MMBADDR LIKE '%Ilam%'
   (4)

4. SELECT DIRNAME
   FROM DIRECTOR
   WHERE DIRDIED IS NULL
   (3)

5. SELECT DIRNAME
   FROM DIRECTOR
   WHERE DIRBORN BETWEEN 1950 AND 1959
   (4)

6. SELECT MMBNUMB, MMBNAME, MMBADDR, MMBCTY, MMBST, JOINDATE
   FROM MEMBER
   ORDER BY MMBNUMB DESC, MMBNAME ASC
   (6)

7. SELECT COUNT (*), SUM (AWRD)
   FROM MOVIE
   WHERE MVTYPE = 'Comedy'
   (4)

8. SELECT MMBNUMB, MMBNAME
   FROM MEMBER
   WHERE NUMRENT >
   (SELECT AVG (NUMRENT)
    FROM MEMBER)
   (5)

9. SELECT MVTITLE
   FROM MOVIE
   WHERE DIRNUMB =
   (SELECT DIRNUMB
    FROM DIRECTOR
    WHERE DIRNAME = 'Stanley Kubrick')
   (5)
10. SELECT MMBNAME, MMBADDR
    FROM MEMBER, STAR, MOVSTAR, TAPE
    WHERE STARLNAME = 'Brooks' AND STARFNAME = 'Mel'
    AND STAR. STARNUMB = MOVSTAR.STARNUMB
    AND MOVSTAR.MVNUMB = TAPE.MVNUMB
    AND MEMBER.MMBNUMB = TAPE.MMBNUMB

11. SELECT S1.STARFNAME, S1.STARLNAME, S2.STARLNAME,
    S2.STARFNAME
    FROM STAR S1, STAR S2
    WHERE S1.FNAME = S2.FNAME AND S1.LNAME <> S2.LNAME

12. SELECT MVNUMB, MVTITLE
    FROM MOVIE
    WHERE NOMS > ALL
    (SELECT NOMS
     FROM MOVIE, DIRECTOR
     WHERE MOVIE.DIRNUMB = DIRECTOR.DIRNUMB)

13. SELECT DIRECTOR.DIRNUMB, DIRNAME, COUNT (*)
    FROM MOVIE, DIRECTOR
    WHERE MOVIE.DIRNUMB = DIRECTOR.DIRNUMB
    GROUP BY DIRECTOR.DIRNUMB, DIRNAME
    HAVING COUNT (*) > 5

14. SELECT PATIENT-NAME
    FROM PATIENT, BILLED, ITEM
    WHERE PATIENT.PATIENT-NO = BILLED.PATIENT-NO
    AND BILLED.ITEM-CODE = ITEM.ITEM-CODE
    AND DESCRIPT = 'Television'

15. SELECT PATIENT-NO
    FROM PATIENT P
    WHERE NOT EXISTS
    (SELECT *
     FROM ATTENDS A
     WHERE P.PATIENT-NO = A.PATIENT-NO
     AND PHY-ID = 'Dr Jefferson')

16. SELECT BILLED.PATIENT-NO, PATIENT-NAME, CHARGE
    FROM PATIENT, BILLED
    WHERE PATIENT.PATIENT-NO = BILLED.PATIENT-NO
    UNION
    SELECT PATIENT.PATIENT-NO, PATIENT-NAME, 0
    FROM PATIENT
    WHERE PATIENT-NO NOT IN
    (SELECT PATIENT-NO
     FROM PATIENT)
Appendix C

STUDENT EVALUATION OF THE SQL-TUTOR PRACTICE ENVIRONMENT

Name: ____________________________________________________________

Diploma: _________________________________________________________

Student Number: __________________________________________________

Address: _________________________________________________________

Phone Number (H): ________________________________________________

Other Courses/Modules (First Semester: 1999) __________________________

Please insert a cross in the appropriate block. If a question does not apply to you, omit it i.e. DO NOT ANSWER IT.

1. I like practising examples on the computer.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

2. It is easy to access HELP information such as description of databases, tables and/or attributes by directly selecting database, table/attribute names.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

3. The explanations pertaining to the various clauses (SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY) are a useful form of help.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

4. I can learn about the elements of SQL such as functions, expressions, predicates and operators by selecting appropriate options in the HELP menu.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>
5. The provision of SQL clauses (SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY) in the main window of the tutor reduces my memory load in that I do not have to remember the exact keywords used and the relative order of the clauses.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

6. The provision of the SQL clauses (SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY) helps me to visualise the goal structure when solving problems.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

7. The information provided on a particular database schema (for example MOVIES) and its associated tables and attributes in the main window assists me in formulating the SQL query command in that I do not have to remember table and attribute names.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

8. The main window of the SQL tutor displays the text of the problem being solved and this serves as a reminder of the elements requested in the query.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

9. The SQL-Tutor is complicated to use.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

10. The feedback (i.e. the system's responses to my answers) is informative and useful.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

11. The positive feedback gives me a feeling of achievement and motivates me to go on solving all the problems in the database.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

12. Which of the following feedback mechanisms did you most frequently use when the tutor indicated that your solution was incorrect?

<table>
<thead>
<tr>
<th>Error flag</th>
<th>Hint</th>
<th>Partial Solution</th>
<th>List all errors</th>
<th>Complete solution</th>
</tr>
</thead>
</table>

13. Do you think that the feedback mechanisms provided by the tutor gave you adequate help and advice on the error(s) made in order for you to correct your solution?

<table>
<thead>
<tr>
<th>Certainly not enough</th>
<th>Needed some more</th>
<th>About the right amount</th>
<th>Needed a bit less</th>
<th>Needed a lot less</th>
</tr>
</thead>
</table>

14. Do you agree that the SQL-Tutor can diagnose and communicate syntactic as well as semantic errors made by students?

| Yes | No |

If you disagree, state why.

15. Would you say that the SQL-Tutor either selects a new problem for you to solve or presents you with a problem you failed to solve before.

| Agree | Disagree |

16. After I solve a problem, the SQL-Tutor gives me an opportunity to solve the same problem again using an alternative solution.

| Agree | Disagree |

17. The history window of the SQL-Tutor maintains a log of all my previous attempts at solving problems during the current session.

| Agree | Disagree |

18. I learnt to use the SQL-Tutor

| Very quickly | Fairly quickly | Gradually | Slowly | Could not use it at all |

19. The SQL-Tutor provides me with a point and click interface that minimises the typing for SQL query commands and thereby reduces typing errors.

| Agree | Disagree |
20. It is easy to erase a solution with errors and restart afresh by using the CLEAR button provided by the SQL-Tutor.

<table>
<thead>
<tr>
<th>Agree</th>
<th>Disagree</th>
</tr>
</thead>
</table>

21. There is too much information on the screens, i.e. the screens are too crammed.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

22. I prefer doing exercises on paper.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

23. It is a waste of time doing interactive practise with the tutor when I can read worked examples in the textbook.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

24. The SQL-Tutor helped me to solve more advanced database problems.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

If you disagree, state why.

____________________________________________________________________

25. The SQL-Tutor contains the definition of more than one database.

<table>
<thead>
<tr>
<th>Agree</th>
<th>Disagree</th>
</tr>
</thead>
</table>

26. The SQL-Tutor contains a set of problems and ideal solutions for specified databases.

<table>
<thead>
<tr>
<th>Agree</th>
<th>Disagree</th>
</tr>
</thead>
</table>

27. Do you agree that the SQL-Tutor is able to answer arbitrary questions about SQL programming?

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
</table>

If so, give an example.

____________________________________________________________________
28. The SQL-Tutor gives a satisfactory explanation of a problem solution (including a problem solved by me).

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

Qualify your answer. ____________________________________________

29. The SQL-Tutor can give alternative explanations.

| Agree | Disagree |

If you agree, give an example. __________________________________

30. The SQL-Tutor stores each of my actions in the session history so that I can inspect it to see how well I am doing.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

31. The explanations provided by the SQL-Tutor are tailored to my needs and the difficulties that I am experiencing.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

32. The problems presented to me by the SQL-Tutor are based on my history (previous attempts).

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

33. The SQL-Tutor gives me control in that it allows me to choose the problems that I want to solve.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

34. The SQL-Tutor intervenes when I have difficulty while solving a problem.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

35. The SQL-Tutor actively engages me in solving database problems.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |
36. I can decide at any point, to learn about any concept, clause, and function or choose the next problem to be solved or choose another database to work with.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

37. The practice environment provided by the SQL-Tutor is sensitive to my needs and preferences.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

Give a reason for your answer. ________________________________________________

38. The SQL-Tutor is tailored to my level of advancement i.e. it presents me with more complex problems after I have solved the simple ones.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

39. The SQL-Tutor did not interrupt me unnecessarily.

<table>
<thead>
<tr>
<th>Agree</th>
<th>Disagree</th>
</tr>
</thead>
</table>

40. The timing of the tutor’s intervention was appropriate i.e. after I attempted to solve the problem.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

41. I can express my input (solution) into the system in a natural way by using either the point and click interface of the SQL-Tutor or typing in the solution.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

42. My solution (with/without errors) is understandable to the SQL-Tutor.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

Substantiate your answer. ________________________________________________
43. The system's feedback is expressed in a natural language that I can easily understand.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

44. The SQL-Tutor is largely robust in that it is able to detect and report on most errors including 'obvious mistakes'.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

If you disagree, give a reason for your answer. ___________________________________________

45. The SQL-Tutor is helpful in that it always provides some sort of help when I am stuck or unable to solve the problem.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

46. The SQL-Tutor is fairly simple to use as it reduces the amount of typing necessary to achieve a given task.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

47. SQL is perspicuous in that it does not provide me with a great many mystifying buttons to choose from.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

48. The SQL-Tutor has state-of-the-art graphic capabilities.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>

49. The SQL-Tutor is navigable in that I know exactly where I am when using the system.

<table>
<thead>
<tr>
<th>Strongly agree</th>
<th>Agree</th>
<th>Maybe</th>
<th>Disagree</th>
<th>Strongly disagree</th>
</tr>
</thead>
</table>
50. The SQL-Tutor is absolutely consistent i.e. it always behaves in the same way in the same situations.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

51. The SQL-Tutor is fully transparent i.e. the effects of my actions are always displayed.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

52. The SQL-Tutor is flexible in that I can solve problems of my choice to gain the mastery I require.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

53. The SQL-Tutor has only a single representation of the subject matter.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

54. The SQL-Tutor is capable of leading me 'by the hand' by asking me leading questions in cases where it knows what I want to do but I am having difficulty doing it.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

55. The Tutor is docile in that it is usually seems to be under my command.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

56. I know that the SQL-Tutor is intended as an interactive practice environment for students. Nevertheless I would like a scoring facility i.e. a mark for my attempt.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |

57. I would like similar practice environments for other courses in my diploma.

| Strongly agree | Agree | Maybe | Disagree | Strongly disagree |
58. I would like similar practice environments for other courses in my diploma.

Yes / No. If Yes, why? ______________________________________________________

59. Please make any further comments (favourable or otherwise) about the SQL-Tutor, including:

• The impression it made on you
• Frustrations you experienced
• In what way it helped
• Anything else you want us to know

THANK YOU FOR PARTICIPATING IN THIS STUDY
Appendix D

INTERVIEW

1. COGNITION

1.1 How did the provision of keywords and the relative order of clauses help you when solving problems?

1.2 In addition to clauses of the SELECT statement being displayed, the main window displays the text of the current problem, as well as the schema of the chosen database. How does this additional information help you to solve problems?

1.3 How does the SQL-Tutor assist you to learn/remember the concepts and elements of SQL?

1.4 Does the Tutor support a 'learning by doing' approach? Elaborate.

1.5 Do you find that the SQL-Tutor encourages you to critically evaluate your own solution to a problem and provides additional help only upon your request?

1.6 Does the Tutor provide you with an adequate explanation of why your solution is incorrect?

1.7 Does the tutor monitor your progress and provide appropriate feedback in order for you to acquire the necessary SQL programming skills?

2. CONSTRUCTIVISM

2.1 How does the SQL-Tutor help you to learn the SQL programming language?

2.2 Do you enjoy actively participating with the tutor (as opposed to passive learning as in book or video learning)?

2.3 Were the problems that the tutor presented to you, 'real world' problems?

2.4 Would it help to do additional problems? If so, how?

3. CUSTOMIZATION

3.1 How did you feel about taking the initiative in learning, by choosing a database, selecting problems to work on, and discovering and correcting your errors under the guidance of a tutor?
3.2 Does the SQL-Tutor support the individual by adapting to each learner’s profile (student model) and needs? Elaborate.

3.3 Do you find that maintaining a model of each learner (the tutor records all previous attempts at solving the problem) is a useful feature of the Tutor? Explain.

3.4 Do you find that the tutor allows you to persevere individually within your own time frame to acquire the necessary SQL skills?

3.5 How did you find the feedback (including positive, negative, syntax and semantic error messages) to your typed solutions?

3.6 Were you able to discover and remedy your errors quickly given immediate and direct feedback?

3.7 When did you use more advanced levels of feedback from the tutoring system?

3.8 Did you as a result of personalised instruction get a better grasp of the sections on grouping, restricted grouping, join conditions and functions in SQL?

4. CREATIVITY

4.1 What do you find different about the way the SQL-Tutor lets you practice?

4.2 Do you get bored quickly when you use it?

4.3 Does it give you, on the screen, all you need to solve problems? (i.e. virtual table-top)?

4.4 Comment on the ARCS (in particular, the attention and satisfaction aspects), with respect to the use of the SQL-Tutor?

4.5 Could you maintain concentration?

4.6 Are the problems presented by the tutor, relevant to the types of problems you are given in class assignments and tests?

4.7 Does the SQL-Tutor positively encourage you to make repeated attempts until you have successfully solved the problem?

4.8 Does it help you to develop confidence in making SQL queries?

4.9 How do you feel when you have successfully solved all the problems for a chosen database schema?
5. COMPONENT DISPLAY THEORY

5.1 The goal of the SQL-Tutor is to provide a guided discovery learning environment. With reference to this goal, do you find the learning strategy employed (students apply their knowledge of SQL to solve problems, discover and remedy their errors under guidance) is appropriate for improving SQL programming skills and in keeping with the goal of the tutor?

6. COLLABORATIVE LEARNING

6.1 Would it help to work together with another/other student(s) when using the SQL-Tutor?

6.2 Did you ever choose to work together with another/other student(s) while using the tutor? If so, describe the interaction.

7. GENERAL

7.1 How do feel about connecting the SQL-Tutor to a DBMS so that you can inspect tables and query results?

7.2 Do you think there are any facilities/features missing in the tutor that could be added to make it more effective.
Appendix E

OBSERVATION

Body Language (posture, facial expression, etc):

Fluency of physical reactions and responses:

Speed (note fluency/speed/achievement relationship):

Concentration and perseverance:

Unprompted verbalisation /Reading out aloud:

Requests for elaboration/help:

Are there extraneous factors/vibes? (i.e. desire to master skills, rushed/don't-care attitude)
APPENDIX F

Examples of successful interactions with the SQL-Tutor

Figure F.1: Use of computed columns
Examples of successful interactions with the SQL-Tutor

For all customers who live in lam, list their number and name.

```
SELECT LNAME, FNAME, NUMBER
FROM CUSTOMER
WHERE ADDRESS LIKE '%lam%'
```

Figure F.2: Use of SQL word LIKE
Examples of successful interactions with the SQL-Tutor

Figure F.3: Use of NOT operator

```sql
SELECT MOVIE.TITLE
FROM MOVIE
WHERE NOT (MOVIE.CRITICS = 'NR')
```
Examples of successful interactions with the SQL-Tutor

![SQL-Tutor interface](image)

**Figure F.4: Use of nulls**
List the titles of all movies, arranged in descending order of the number of Academy Awards won.

```
SELECT TITLE, AAWON
FROM MOVIE
WHERE
GROUP-BY
HAVING
ORDER-BY AAWON DESC
```

Figure F.5: Sorting with single key
Examples of successful interactions with the SQL-Tutor

Figure F.6: Use of sub-query
Examples of successful interactions with the SQL-Tutor

Figure F.7: Use of aliases
Examples of successful interactions with the SQL-Tutor

List the movie number and title for all movies that were nominated for more Academy Awards than any movie directed by Woody Allen.

```
SELECT NUMBER, TITLE 
FROM MOVIE 
WHERE AANOM > ALL( SELECT AANOM FROM MOVIE, DIRECTOR WHERE DIRECTOR = 'Woody Allen') 
GROUP BY 
HAVING 
ORDER BY
```

Figure F. 8: Use of SQL word **ALL**