

This is to certify that the

dissertation entitled


An Architecture for the Generation of Intelligent
Tutoring Systems from Reusable Components
and Knowledge-Based Systems

presented by

Eman El-Sheikh

has been accepted towards fulfillment
of the requirements for

Doctoral degree in Computer Science
& Engineering


Major professor

Date 9 May 2002

LIBRARY
Michigan State
University

PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.
MAY BE RECALLED with earlier due date if requested.

DATE DUE	DATE DUE	DATE DUE

**AN ARCHITECTURE FOR THE GENERATION OF INTELLIGENT TUTORING
SYSTEMS FROM REUSABLE COMPONENTS AND KNOWLEDGE-BASED
SYSTEMS**

By

Eman Mustafa El-Sheikh

A DISSERTATION

**Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of**

DOCTOR OF PHILOSOPHY

Department of Computer Science and Engineering

2002

ABSTRACT

AN ARCHITECTURE FOR THE GENERATION OF INTELLIGENT TUTORING SYSTEMS FROM REUSABLE COMPONENTS AND KNOWLEDGE-BASED SYSTEMS

By

Eman Mustafa El-Sheikh

There is a growing demand for principled and useful instructional software applications in both academic and industrial settings. The need for effective tutoring and training is increasingly important in technical fields, which demand the learning of complex tasks and the use of large knowledge stores. In the last two decades, intelligent tutoring systems (ITSs) have been proven to be highly effective as learning aides, and numerous ITS research and development efforts were initiated. However, few tutoring systems have made the transition to the commercial market. The main reasons for this failure to deliver are that the development of ITSs is difficult, time-consuming, and costly. There is a need for easier, more cost-effective means of developing tutoring systems.

In this dissertation, a novel methodology and architecture for generating intelligent tutoring systems for a wide range of domains is described. The architecture incorporates an ITS shell that interacts with any generic task-based expert system to produce a tutoring system for the domain knowledge

represented in that system. The focus is on the issue of reusability. The knowledge-rich structure of generic tasks can be reused for instructional purposes, allowing the tutoring of domain and problem solving knowledge embedded within an expert system. The integration of this reusable knowledge with other reusable ITS components creates a powerful environment for the generation of tutoring systems for different domains.

The architecture developed has many features needed in an effective ITS authoring environment. Among other features, it employs a runnable deep model of domain expertise, facilitates fine-grained student diagnosis, and offers an easy method for building expert systems and generating ITSs from them. It was used to generate two tutoring systems: an Excel tools tutor and a composite materials fabrication technology tutor. Evaluation studies of the architecture and tutoring systems generated showed that the architecture allows ITS authors to generate instructionally effective tutors in a simple and straightforward process. The central contributions of this research are: (1) an architecture that generates intelligent tutoring systems from generic task expert systems, (2) a proof of concept implementation of the architecture, and (3) a concrete demonstration of the reusability of the knowledge stored within a generic task knowledge-based system.

**© Copyright by
Eman Mustafa El-Sheikh
2002**

For Kareem, my sunshine

ACKNOWLEDGEMENTS

I would like to thank all those who have inspired me to always reach higher, in everything that I do. Certainly, the most influential figure on my research work has been Prof. Jon Sticklen. As my thesis advisor, Jon always managed to give me just enough advice to help keep me progressing, while still giving me enough responsibility and leeway to figure things out on my own. I have learned a tremendous amount from Jon, not just about work, but about life.

I would like to specifically thank the other members of my guidance committee: Prof. Pat Dickson, Prof. Carrie Heeter, and Prof. Bill Punch. In spite of their very demanding schedules, they always found the time for stimulating discussions when I came calling. Their useful input and valuable comments helped shape and refine this research work.

I would also like to thank my former and current colleagues at the Intelligent Systems Laboratory for their help and support during my graduate studies at Michigan State University. Whether bouncing ideas off of each other, or drinking aged coffee together, you have helped make my time in the ISL a lot more interesting and useful. A special thanks to Ahmed Kamel, Kris Schroeder, Rob Hawkins, and Taner Eskil.

This work has been supported in part by the National Science Foundation under the Technology Reinvestment Program (Grant number: eee-9410783-613224) and the Rapid Prototyping Initiative (Grant number: MIP-942-0351), and by DARPA under the RADEO project (Grant number: N00014-96-1-0678).

Finally, I can't possibly say enough to thank my parents, Mustafa and Gihan, and my wonderful husband, Mohannad, for their never-ending support. It was their support, love, and encouragement that always kept me going, even when the end of graduate school seemed to be nowhere in sight. Thanks also to my sister, Rhonda, and brother, Ashraf, for their help and support. And last but definitely not least, I would like to thank my wonderful son, Kareem, for giving me the most important title of all: mommy. No matter how many long days I spent at the lab, Kareem always greeted me with a welcoming smile and gave me something to be happy and thankful for. You are, and will always be, my sunshine.

TABLE OF CONTENTS

LIST OF TABLES	xiii
LIST OF FIGURES	xiv
KEY TO ABBREVIATIONS	xvii
Chapter 1. Introduction.....	1
1.1 Motivation	1
1.2 Learning Effectiveness of Intelligent Tutoring Systems	2
1.3 ITS Development Problem	3
1.4 Generic Task Leverage for Reusability and ITS Authoring	5
1.5 Expected Impact of This Research.....	6
1.6 Organization of Dissertation	7
Chapter 2. Background	8
2.1 Background on Intelligent Tutoring Systems	8
2.1.1 The History of ITS Development.....	9
2.1.2 Effectiveness of ITSs for Learning.....	12
2.1.3 Basic Architecture of an ITS	16
2.1.4 Current and Future Issues for ITSs	18
2.2 The Task-Specific Architecture and Generic Task Expert Systems Development Methodologies	20
2.3 Task-Specific Architectures	21
2.3.1 KADS and CommonKADS	23
2.3.2 PROTÉGÉ-II.....	24

2.3.3	EXPECT	25
2.3.4	Generic Tasks	26
2.3.4.1	Structured Matching	27
2.3.4.2	Hierarchical Classification	28
2.3.4.3	Routine Design	29
2.3.4.4	Functional Modeling	30
2.3.4.5	The Knowledge-Level Architecture	30
2.3.5	Analysis of the Task-Specific Approaches.....	32
2.4	Learning and Cognition Issues for ITS Development and Use	33
2.4.1	Key Findings on Learning	33
2.4.2	The Role of Knowledge Structuring in the Learning Process	34
2.4.3	The Transfer of Knowledge	35
2.4.4	Issues in Using Technology to Support Learning	36
2.5	Analysis	37
Chapter 3.	Related Research on ITS Authoring Environments	38
3.1	Commercial Tools.....	39
3.2	AI-based ITS Authoring Systems.....	41
3.2.1	Early Attempts	45
3.2.2	General-purpose Authoring Environments	46
3.2.3	COTS and Component-based Tools	48
3.2.4	Instructional Strategy Authoring	49
3.2.5	Interface Authoring	50
3.2.6	Simulation Systems Authoring.....	51

3.2.7	Ontology-based Authoring.....	53
3.2.8	Domain-specific Authoring.....	54
3.2.9	Task-specific Authoring	55
3.2.10	Summary of ITS Authoring Systems.....	56
3.3	Related Research on Explanation for Generic Tasks	57
3.4	Lessons Learned	60
Chapter 4.	Problem Definition	63
4.1	Motivation	63
4.2	General Issues Related to ITS Authoring Environments	65
4.3	Objectives within the KBS Field.....	69
4.4	Objectives within the ITS Field	70
4.5	Educational Objectives	73
4.6	Problem Statement.....	75
Chapter 5.	An Architecture for Generating Intelligent Tutoring Systems	77
5.1	Extending the Generic Task Framework for Tutoring Support.....	77
5.2	Completing the Architecture	82
5.2.1	The Reusable Tutoring Components of the ITS Shell	82
5.2.1.1	Student Model.....	82
5.2.1.2	Instructional Manager	84
5.2.1.3	User Interface	86
5.2.2	The Extended Domain Knowledge Component.....	87
5.3	Specializing the ITS Architecture for Hierarchical Classification.....	88
5.3.1	Problem Solving Knowledge.....	90

5.3.1.1	Knowledge Embedded Within Each Agent	90
5.3.1.2	Reasoning Strategy	91
5.3.1.3	Problem Solving State	92
5.3.2	Domain Knowledge	93
5.3.2.1	Classification Structure	93
5.3.2.2	Input Information.....	94
5.3.2.3	Output Information	95
5.3.3	Knowledge Storage and Transfer.....	95
5.4	Comparison of the Generic Task Approach for ITS Generation to the KADS Approach.....	96
5.4.1	Description of the KADS Approach	97
5.4.2	Comparing the GT and KADS Approaches for Tutoring Support.....	98
5.4.3	Analysis of the Two Approaches	102
Chapter 6.	Tahuti: A Software Systems for Generating Intelligent Tutoring Systems.....	103
6.1	Implementation of the Tahuti System	103
6.2	Using Tahuti to Generate Intelligent Tutoring Systems	105
6.2.1	Developing the Expert System	106
6.2.2	Constructing the Extended Knowledge Module.....	112
6.2.3	Generating the Intelligent Tutoring System	113
6.3	Examples of ITSs Generated by Tahuti.....	115
6.3.1	The Excel Tutor.....	117

6.3.2	The Composite Materials Tutor	124
6.4	Constraints of the Tahuti System.....	131
Chapter 7.	Evaluation of the Tahuti ITS Architecture	133
7.1	Evaluating the Process of Generating ITSs.....	133
7.2	Evaluating the ITSs Generated.....	135
7.2.1	Pilot Study	136
7.2.1.1	Setup of the Study	136
7.2.1.2	Results of the Study.....	137
7.2.2	Lessons Learned	138
Chapter 8.	Conclusions.....	140
8.1	Contributions to the KBS Field.....	140
8.2	Contributions to the ITS Field	142
8.3	Educational Contributions.....	144
8.4	Summary of Contributions	146
8.5	Future Directions	147
8.6	Concluding Remarks	148
APPENDIX A.	Representation of the Excel Tools Expert System	151
APPENDIX B.	Details of the Pilot Study	157
BIBLIOGRAPHY	183

LIST OF TABLES

Table 1.	Summary of ITS authoring environments, listed by category	57
Table 2.	Summary of research contributions	146
Table 3.	Test scores for group A.....	179
Table 4.	Test scores for group B.....	179

LIST OF FIGURES

Figure 1.	The rise of computer-based learning.....	11
Figure 2.	Distributions for different learning conditions.....	13
Figure 3.	Components of an ITS	16
Figure 4.	Design space for ITS authoring systems.....	43
Figure 5.	System-user interaction model.....	78
Figure 6.	Architecture for generating ITSs from GT-based expert systems..	79
Figure 7.	IPT model for the student model	84
Figure 8.	Basic layout of the user interface	86
Figure 9.	ITS architecture specialized for HC-based systems.....	89
Figure 10.	ITS development spectrum.....	100
Figure 11.	Tahuti system architecture	104
Figure 12.	Creating the problem solver	107
Figure 13.	Problem solver window.....	107
Figure 14.	Database window	108
Figure 15.	Variable definition window	109
Figure 16.	Classification hierarchy.....	109
Figure 17.	Table matcher window	110
Figure 18.	Input window	111
Figure 19.	Problem solver output window.....	112
Figure 20.	Template for the extended knowledge file.....	113
Figure 21.	The load menu of the Tahuti ITS architecture	114

Figure 22.	Linking the expert system to the architecture	115
Figure 23.	Linking the extended knowledge file to the architecture	115
Figure 24.	Tahuti architecture instantiated to generate the Excel Tutor	116
Figure 25.	Tahuti architecture instantiated to generate the Composite Materials Tutor.....	116
Figure 26.	Excel Tutor – main user interface.....	118
Figure 27.	An example presented by the Excel Tutor.....	118
Figure 28.	The classification hierarchy for the Excel tools.....	119
Figure 29.	A question posed to the learner by the Excel Tutor.....	120
Figure 30.	The Excel Tutor's explanation of correct answer.....	121
Figure 31.	The Excel Tutor shows the learner a hint and asks again	121
Figure 32.	An example of feedback given by the Excel Tutor.....	122
Figure 33.	The Excel Tutor presents more examples.....	123
Figure 34.	Summary report displayed by the Excel Tutor.....	124
Figure 35.	Composite Materials Tutor – main user interface	125
Figure 36.	An example presented by the CM Tutor.....	125
Figure 37.	A practice question asked by the CM Tutor.....	126
Figure 38.	Examples of feedback provided by the CM Tutor.....	127
Figure 39.	The CM Tutor gives a hint and asks again	128
Figure 40.	Explanation of correct answer provided by the CM Tutor.....	129
Figure 41.	Hierarchy of fabrication technologies shown to the learner by the CM Tutor.....	130
Figure 42.	Summary report presented by CM Tutor	131

Figure 43.	Flowchart for evaluation plan done in the pilot study.....	137
Figure 44.	Fig. 1. An example from the tutoring system	163
Figure 45.	Fig. 2. Classification hierarchy for Excel tools	164
Figure 46.	Fig. 3. Classification hierarchy showing a path that matched	165

KEY TO ABBREVIATIONS

Abbreviation	Long Form
AI	Artificial Intelligence
CAI	Computer-assisted Instruction
CBT	Computer-based Training
CM	Composite Materials
COFATE2	Composite Materials Fabrication Technology Selector
COTS	Commercial Off the Shelf Systems
CSE	Computer Science and Engineering
GT	Generic Task
GUI	Graphical User Interface
HC	Hierarchical Classification
ICAI	Intelligent Computer-assisted Instruction
IEEE	Institute for Electrical and Electronic Engineers
ILE	Interactive Learning Environment
IPT	Information Processing Task
ISL	Intelligent Systems Laboratory
ITS	Intelligent Tutoring System
KADS	Knowledge Acquisition and Documentation System <i>or</i> Knowledge Analysis and Design Support
KBS	Knowledge-based System
KLA	Knowledge Level Architecture
LTSC	Learning Technology Standards Committee
NSF	National Science Foundation
PI	Programmed Instruction
PS	Problem Solver
RD	Routine Design
SM	Structured Matching
TRP	Technology Reinvestment Program
TSA	Task-specific Architecture
VW	VisualWorks

Chapter 1. Introduction

The goal of this research work is the efficient development of intelligent learning environments. More specifically, the focus is to develop an effective architecture in which any generic task-based expert system can be used as a knowledge source from which an intelligent tutoring system can be generated that covers the domain represented in that expert system.

This chapter presents an overview of this research. The motivation, problem statement, solution framework, and expected impact will be sketched here, and detailed in later chapters.

1.1 Motivation

The need for effective tutoring and training is mounting, given the increasing knowledge demand in academia and industry. Rapid progress in science and technology has created a need for people who can learn knowledge-intensive domains and solve complex problems. Many computer-assisted instruction (CAI) techniques exist that can present instruction, and interact with students in a tutor-like fashion, individually, or in small groups. The incorporation of artificial intelligence techniques and expert systems technology to CAI systems gave rise to intelligent tutoring systems (ITSs), i.e., systems that model the learner's understanding of a topic and adapt the instruction accordingly. Although ITS research has been carried out for nearly two decades, few tutoring systems have made the transition to the commercial market. The main reasons for this failure to

deliver are that the development of ITSs is difficult, time-consuming, and costly. Thus, there is a need for easier, more cost-effective means of developing tutoring systems.

In this research, a novel approach was taken to develop an intelligent tutoring system methodology. The goal is to develop an ITS authoring methodology and architecture that interacts with any generic task expert system, to produce a tutoring system for the domain knowledge represented in that system (El-Sheikh, Kamel et al., 1996; El-Sheikh and Sticklen, 1999). There are two potential paths in following this approach. First, in the case of an existing generic task system, a tutoring system can be easily generated, in a largely automated manner. Second, in the case that no generic task system is available, the tools and methodology of building such knowledge-based systems are readily available. In either case, more cost-effective development of a tutoring system is likely.

1.2 Learning Effectiveness of Intelligent Tutoring Systems

For many years, researchers have argued that individualized learning offers the most effective and efficient learning for most students (Bloom, 1956; Burton and Brown, 1982; Cohen, Kulik et al., 1982; Bloom, 1984; Woolf, 1987; Juel, 1996). Intelligent tutoring systems epitomize this principle of individualized instruction. Recent studies have found that ITSs can be highly effective learning aides (Shute and Regian, 1990). Shute evaluated several ITSs to judge how they live up to the two main promises of ITSs: (1) to provide more effective and efficient learning in relation to traditional instructional techniques, and (2) to reduce the

range of learning outcome measures where a majority of individuals are elevated to high performance levels. Results of such studies show that tutoring systems do accelerate learning with no degradation in final outcome. For example, in one such study, students working with the Airforce electronics troubleshooting tutor for only 20 hours, gained proficiency equivalent to that of trainees with 48 months of on-the-job experience (Lesgold, Lajoie et al., 1990). In another study, students using the LISP tutor completed programming exercises in 30% less time than those receiving traditional classroom instruction and scored 43% higher on the final exam (Anderson, 1990).

1.3 ITS Development Problem

Although ITSs are becoming more common and proving to be increasingly effective, a serious problem exists in the current methodology of developing intelligent tutoring systems. Each application is usually developed independently, from scratch, and is very time-consuming and difficult to build. In one study of educators using basic tools to build an ITS, results indicated that each hour of instruction required approximately 100 person-hours of development time (Murray, 1993). Although the time required is significant, it still compares favorably with the 100 – 300 person-hours per instruction hour necessary for building traditional CAI.

Another problem is that the tutoring knowledge, both domain and pedagogical knowledge, is usually hard-coded into individual applications. There is very little reuse of tutoring components between applications because the field lacks a standard language for representing the tutoring knowledge, a standard

interface to allow applications to access the knowledge and a set of tools to allow designers to manipulate the knowledge. In describing the state of building ITSs at the First International Conference on Intelligent Tutoring Systems, Clancey and Joerger (Clancey and Joerger, 1988) complained that "...the reality today is that the endeavor is one that only experienced programmers (or experts trained to be programmers) can accomplish. Indeed, research of the past decade has only further increased our standards of knowledge representation desirable for teaching, while the tools for constructing such programs lag far behind or are not generally available." Unfortunately, the same problem still exists over a decade later.

Authoring tools for ITSs are not yet commercially available, but authoring systems are available for traditional CAI and multimedia-based training. However, these systems lack the sophistication required to build "intelligent" tutors. Commercial authoring systems give instructional designers and domain experts tools to produce visually appealing and interactive screens, but do not provide a means of developing a rich and deep representation of the domain knowledge and pedagogy. Indeed, most commercial systems allow only a shallow representation of content.

In addition, in current research-based authoring systems, there exists a gap between the tutoring content and the underlying knowledge organization. There is a need for ITS authoring tools that can bridge this gap by making the organization of the knowledge used for tutoring more explicit.

1.4 Generic Task Leverage for Reusability and ITS Authoring

The motivation for our work comes from the need for reusable intelligent tutoring systems and from the leverage that the generic task (GT) development methodology offers in solving this problem (Chandrasekaran, 1986). The assumption of the GT approach is that there are basic “tasks” - problem solving strategies and corresponding knowledge representation templates - from which complex problem solving may be decomposed. Our goal is to develop an ITS architecture that can interact with a GT-based system, and produce an effective tutorial covering the domain knowledge represented in the problem solver (El-Sheikh and Sticklen, 1998; El-Sheikh, 1999). GT systems are strongly committed to both a semantically meaningful domain knowledge representation, and to an explicit inferencing strategy. The backing intuition of this work is that the explicit knowledge representation and inference strategies in GT systems can be accessed effectively by an agent external to the specific GT and used by that agent to generate an effective ITS system. It is the purpose of the research that is reported here to test that intuition.

This approach facilitates the reuse of tutoring components. The ITS shell can be used in conjunction with any GT-based expert system, effectively allowing the same tutoring components to be plugged in with different domain knowledge bases. In other words, the tutoring overlay can be used to generate an ITS for a domain by linking to a GT expert system for that domain. The same tutoring overlay can be used as-is to generate an ITS for other domains by linking to different GT expert systems. The assumption we are making is that the ITS shell

can be domain knowledge free, allowing it to be reused for different domains. Our approach makes the underlying knowledge organization of the expert systems explicit, and reuses both the problem solving and domain knowledge for tutoring.

1.5 Expected Impact of This Research

The major anticipated accomplishment of this work is the development of an architecture for generating ITSs for various domains. Specifically, the solution proposed has at its core GT-based expert systems, and reuses the other tutoring components to produce tutoring systems. More generally, a technique will be formulated for leveraging the knowledge representation and structure of the generic task framework for tutoring. Thus, the top-level accomplishment envisioned is the development of a framework for an ITS extension to the GT approach.

The main objective addressed is achieving reusability. However, the ITS authoring environment should also be easy to use, generate pedagogically sound ITSs, and facilitate rapid prototyping and evaluation. Moreover, it should combine the problem solving and domain knowledge extracted from the expert system with the appropriate pedagogical knowledge to produce effective instruction. Such an authoring system will help bridge the gap between the artificial intelligence and education fields by integrating effective instructional techniques and learning situations with the adaptiveness and robustness of intelligent systems to produce learning environments.

1.6 Organization of Dissertation

The remainder of this dissertation describes the research work in greater detail. The next chapter details the background on intelligent tutoring systems, generic tasks, and learning needed to address the problem. Chapter 3 describes previous and related research efforts on ITS development tools and authoring environments. Chapter 4 presents the formal problem definition and chapter 5 presents the architecture for developing reusable ITSs. Chapter 6 presents the architecture developed for generating ITSs and also describes how the shell is used to generate ITSs. Chapter 7 gives an account of the evaluation of the ITS architecture. Finally, chapter 8 discusses the contributions of this work to the Intelligent Tutoring Systems (ITS), Knowledge-based Systems (KBS), and Education fields, and also discusses future research directions for this work.

Chapter 2. Background

There has been a growing interest in the research and development of intelligent tutoring systems, with many thrusts towards advancing the field. This chapter outlines background information and terminology for ITSs and generic tasks. The first section presents a review of the field of ITSs, including a history of its development, a widely accepted definition of what an ITS is, the current status of ITS development, and future research and development issues. After that, the idea of generic tasks as a task-specific approach is explained, along with descriptions of other task-specific approaches. Finally, the chapter concludes with a discussion of learning and cognitive issues that are relevant to the development and use of intelligent tutoring systems.

2.1 Background on Intelligent Tutoring Systems

Intelligent tutoring systems are computer-based instructional systems that attempt to determine information about a student's learning status, and use that information to dynamically adapt the instruction to fit the student's needs (Wenger, 1987; Yazdani, 1987; Frasson and Gauthier, 1988; Psotka, Massey et al., 1988; Anderson, Boyle et al., 1990; Goodyear, 1990; Larkin and Chabay, 1991; Venezky and Osin, 1991; Farr and Psotka, 1992; Winkels, 1992; Benyon and Murray, 1993; Petrushin, 1995; Urban-Lurain, 1996). ITSs are also commonly referred to as *knowledge-based tutors*, because they are modular, and have separate knowledge bases for domain knowledge (which specify what

to teach) and instructional strategies (which specify how to teach) (Murray, 1996).

ITS design is founded on two fundamental assumptions about learning:

- Individualized instruction by a competent tutor is far superior to classroom-style learning because both the content and style of instruction can be continuously adapted to best meet the needs of the learner (Bloom, 1984).
- Students learn better in situations that closely approximate the actual situations in which they will use their knowledge. In other words, students learn best “by doing,” learn via their mistakes, and learn by constructing knowledge in a very individualized way (Bruner, 1966; Ginsburg and Oppen, 1979; Kafai and Resnick, 1996).

In the last decade, ITSs have moved out of the research labs and into use in the classrooms and workplace. However, they have come a long way, so the following section reviews their rise.

2.1.1 The History of ITS Development

The field of ITSs has been around for over 20 years, but its evolution stemmed from earlier accomplishments in computers and education. Around the mid-1900's, general-purpose digital computers were first developed, which were characterized by a built-in ability to make logical decisions and a built-in device for easy storage and manipulation of data? These computers paved the way for the Artificial Intelligence (AI) movement around the 1950's and 1960's, in search of intelligent machines. Then, researchers such as Alan Turing, Marvin Minsky, John McCarthy, and Alan Newell, thought that computers that could “think” as

humans do could be produced soon, constrained only by the development of bigger, faster computers. Concurrent with the gradual emergence of computers, there was an important advance in the education field. Educational psychologists began reporting that carefully designed individualized tutoring produced the most effective learning for most people. Thus, it was quite a natural development to apply computers to the task of individualized teaching. From the 1970's until now, ITSs have been heralded as the most promising approach to delivering such individualized instruction.

Figure 1 shows a timeline highlighting important milestones in the rise of intelligent tutoring systems. The first notion of computer-based instruction came about as a result of *programmed instruction* (PI). Established in the early 1960's, PI was used to refer to any instructional methodology that utilized a systematic approach to problem decomposition and teaching (Shute and Psotka, 1996). In a typical PI approach, as learners were led through the problems in the curriculum, overt responses were obtained at every step; incorrect responses were immediately corrected, and learners were always informed before moving on to a different content area. Programmed instruction that was administered using computer programs was called *computer-assisted instruction* (CAI), which also became known as *computer-based training* (CBT). CAI was similar to PI in that both have well-defined curricula. However, CAI provided the added benefit of conditional branching, whereas PI allowed only fixed branching.

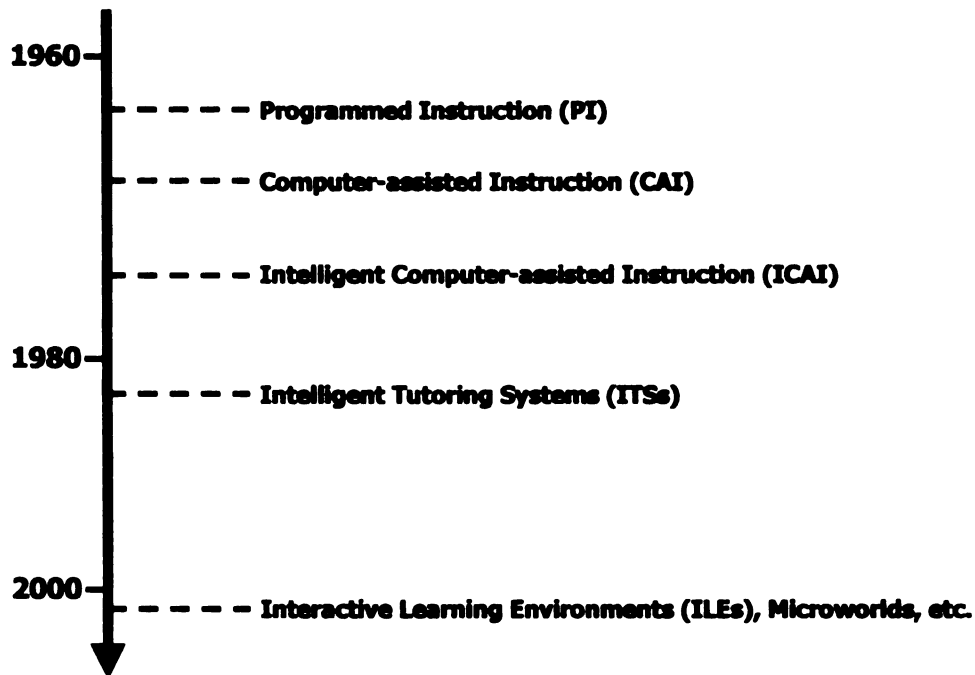


Figure 1. The rise of computer-based learning

As more sophisticated AI techniques were employed to build CAI systems, a distinction was made between simple CAI and more adaptive *“intelligent” computer-assisted instruction* (ICAI). Although ICAI could provide more powerful instruction, the approach still suffered from a major flaw. Most ICAI systems could not differentiate between remedial and original instruction. So, for example, if a student cannot determine the right answer the first time, he or she will face the same problem again if the same instruction is used over. Researchers recognized the need to diagnose the errors or misconceptions that a student could encounter, and tailor the instruction accordingly, giving rise to intelligent tutoring systems. This feature is one of the major distinctions between ICAI and ITS. AI programming techniques provided “intelligence” in ITSs by going beyond what was explicitly programmed, understanding students’ inputs, and generating rational responses based on reasoning from the inputs and the system’s own

database. Not only could intelligent tutors identify and remediate errors in a learner's knowledge structure, but they could also make use of complex branching techniques, in which the branching is algorithmic, not enumerated or pre-defined. The complexity associated with an ITS provided an increase in flexibility of instruction and interaction that justified the label "intelligent."

An early specification of an ITS was provided by Sleeman and Brown (Sleeman and Brown, 1982), who stated that an ITS must possess:

- Knowledge of the domain (expert model)
- Knowledge of the learner (student model)
- Knowledge of the instructional strategies (tutor)

It is interesting to note that this specification remains basically the same nearly two decades later. These requirements are manifested in the form of knowledge-based components of an ITS, which will be described in section 2.1.3.

2.1.2 Effectiveness of ITSs for Learning

Problems have been identified with conventional teaching methods, e.g., a teacher presenting material to a large group of students. In one study, Bloom (Bloom, 1984) affirmed that this form of teaching provides one of the least effective methods of learning. As teaching becomes more individualized and focused, learning is enhanced. Bloom identified the "two sigma problem" - to achieve two standard deviation improvements with tutoring over traditional instruction. He compared students' scores on achievement tests using three forms of instruction: conventional teaching, mastery teaching, and individualized tutoring. Mastery teaching is an instructional technique whereby a teacher

supplements a lecture with diagnostic tests to determine where students are having problems, and adjusts the instruction accordingly. The result of this comparison is shown in figure 2. Students receiving conventional teaching scored in the 50th percentile, students receiving mastery teaching scored in the 84th percentile, while students receiving individualized tutoring scored in the 98th percentile (a 2 standard deviation increase). Bloom replicated these results four times with three different age groups for two different domains, and thus, provided concrete evidence that tutoring is one of the most effective educational delivery methods available.

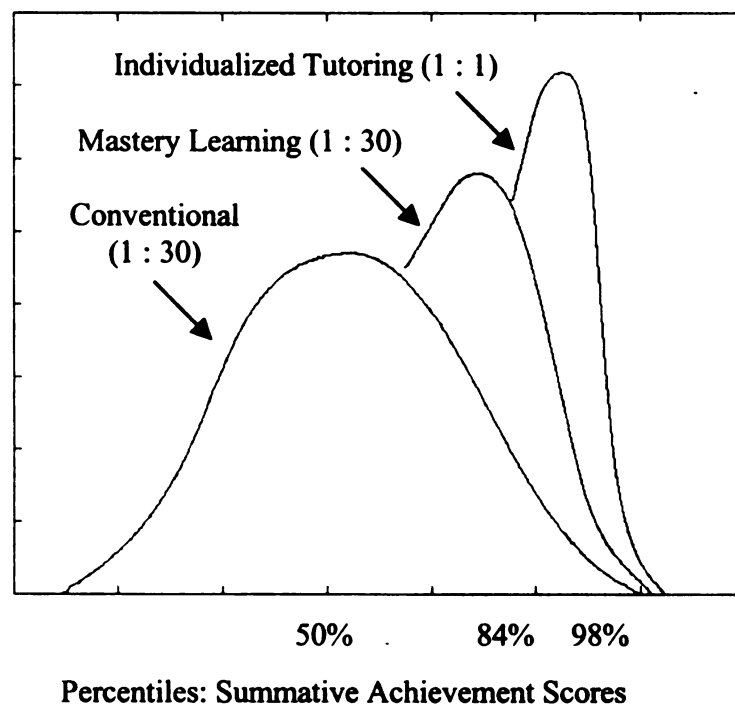


Figure 2. Distributions for different learning conditions
(Adapted from Bloom, 1984)

Intelligent tutoring systems attempt to provide more effective learning through individualized instruction. The two main promises of ITSs are:

- To engender more effective and efficient learning in relation to conventional teaching formats.
- To reduce the range of learning outcome measures where a majority of students is elevated to high performance levels.

For ITSs to keep their promises, they must meet the goals of the “two sigma problem.” Therefore, controlled evaluation studies are needed to determine their learning effectiveness. A few examples of systematic controlled evaluations of ITSs reported in the literature include: the LISP tutor (Anderson, 1990), instructing LISP programming skills; Smithtown (Shute and Glaser, 1990), a discovery world that teaches scientific inquiry skills in the context of microeconomics; Sherlock (Lesgold, Lajoie et al., 1990), a tutor for avionics troubleshooting; Pascal ITS (Shute, 1991), which teaches Pascal programming skills; Stat Lady (Shute, Gawlick-Grendell et al., 1993), instructing statistical procedures; and the Geometry Tutor (Anderson, Boyle et al., 1985), teaching Geometry theorems.

Shute (Shute and Psotka, 1996) examined the results of these evaluations, which showed that these tutors do accelerate learning, with no degradation in outcome performance. The tutors should be evaluated with respect to the promises of ITSs. With regards to the first goal, the tutors mentioned were shown to accelerate students' learning rates:

1. Students working with the LISP tutor learned the knowledge and skills in 33-66% less time than it took the control group.

2. Students working with Smithtown learned the same material in half the time it took a classroom-instructed group.
3. Students working with Sherlock learned, in 20 hours, skills comparable to technicians with 4 years of experience.
4. Students learning from Pascal ITS acquired the same knowledge and skills through traditional instruction in one third the time.
5. Students using Stat Lady learned significantly more knowledge and performed at least as well (and much better, in some cases), compared to students in a traditional lecture environment.
6. Evaluation of students working with the Geometry tutor produced a different kind of result, namely that students working with the tutor could more easily share experiences and make use of each others experiences and problem solving strategies.

In all cases, individuals using the ITSs learned faster, and performed at least as well as those learning from traditional teaching environments. The second goal is to produce a reduction in the range of outcome scores. Although this was harder to assess, the results showed that these tutors could not only reduce the range of outcome scores, but also make learning less dependent on aptitudes, thereby providing every student with a fair chance at learning.

Overall, the evaluation results are encouraging, especially given the enormous differences among the six tutors in design structure as well as evaluation methods. To further reconfirm the effectiveness of ITSs, more

principled approaches to ITS design and evaluation are needed (Regian and Shute, 1994; Bloom, Linton et al., 1997; Siemer and Angelides, 1998).

2.1.3 Basic Architecture of an ITS

Although there is no standard for an ITS architecture, most ITSs reported in the literature have four main components (Sleeman and Brown, 1982; Wenger, 1987; Yazdani, 1987; Polson and Richardson, 1988; Costa, 1992; Wasson, 1997). These are the student model, the pedagogical module, the expert model, and the communication module or interface. These four software components and their interactions are illustrated in figure 3.

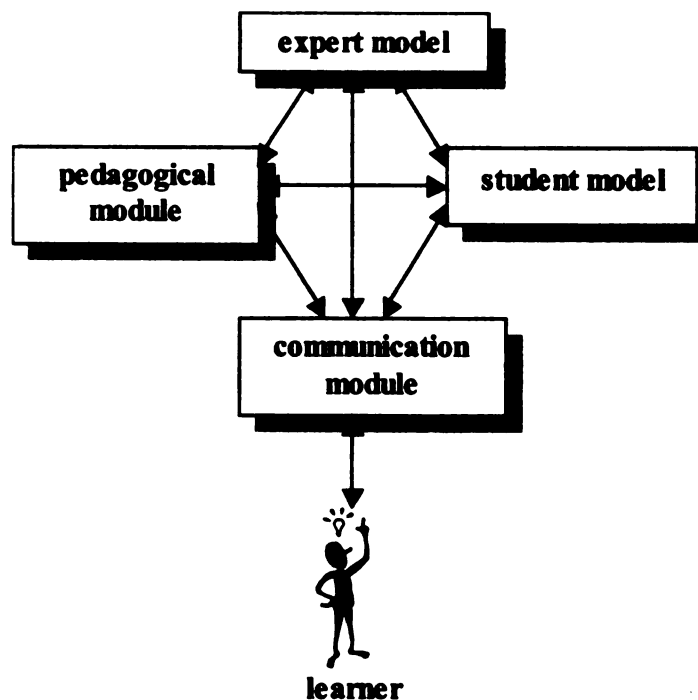


Figure 3. Components of an ITS

The *student model* stores information that is specific to each individual learner. At a minimum, such a model tracks how well a student is performing on the material being taught. A possible addition to this is to also record

misconceptions. Since the purpose of the student model is to provide data for the pedagogical module of the system, all of the information gathered should be able to be used by the tutor. Recently, some researchers have argued that student modeling capabilities are not necessary for building effective ITSs (Lajoie and Derry, 1993; Gugerty, 1997).

The *pedagogical module* provides a model of the teaching process. For example, information about when to review, when to present a new topic, and which topic to present is controlled by this module. As mentioned earlier, the student model is used as input to this component, so the pedagogical decisions reflect the differing needs of each student.

The *expert model* contains the domain knowledge, the information being taught to the learner. However, it is more than just a representation of the data; it is a model of how someone skilled in a particular domain represents the knowledge. Most commonly, this takes the form of a runnable expert model, i.e. one that is capable of solving problems in the domain (Clancey and Soloway, 1990; Shute and Psotka, 1996; Murray, 1998). By using an expert model, the tutor can compare the learner's solution to the expert's solution, pinpointing the places where the learner has difficulties. This component contains information the tutor is teaching, and is the most important since without it, there would be nothing to teach the student. This aspect of the expert model is of great importance to this work, and will be further analyzed in chapter 4. Generally, it requires significant knowledge engineering to represent a domain so that other parts of the tutor can access it. One related research issue is how to represent

knowledge so that it easily scales up to larger domains. Another open question is how to represent domain knowledge other than facts and procedures, such as concepts and mental models.

Interactions with the learner, including the dialogue and the screen layouts, are controlled by the *communication module*. For example, it determines how the material should be presented to the student in the most effective way. This component has not been researched as much as the others, but there has been some promising work in this area (Burns, Parlett et al., 1991; Winkels, 1992; Frasson, Gauthier et al., 1996).

These four components, the student model, the pedagogical module, the expert model, and the communication module interact to provide the individualized educational experience promised by ITS technology. But, what does the “I” in an ITS signify? To be labeled “intelligent,” an ITS must be able to (a) accurately diagnose students’ knowledge structures and preferences using principles, rather than pre-programmed responses, to decide what to do next, and then (b) adapt instruction accordingly. In other words, an ITS focuses more on the fluctuating cognitive needs of a single learner over time, rather than on stable individual differences.

2.1.4 Current and Future Issues for ITSs

Important advances in computer science, education, and psychology will pave the way for computers with instructional capabilities superior to conventional human-only teaching techniques (Burns, Parlett et al., 1991). Although the future of ITSs looks promising, significant issues and problems still remain. The 20+

year history of the field of ITSs has witnessed great achievements and great debates. Each decade brought its own set of results, and sprung its own set of new issues. In the 1970's, when only simple tutoring systems were built, the research focus was on problem generation, knowledge representation, and simple student modeling. As more effective tutors were developed, the emphasis in the 1980's moved to developing more powerful student modeling techniques, such as model-tracing and buggy libraries; use of more sophisticated reasoning techniques like case-based reasoning; discovery worlds; simulations; natural language processing; and building authoring systems. Lack of agreement on how to meet these new challenges spurred further controversial issues in the 1990's. The current debates include:

- How much learner control should be allowed in tutoring systems?
- Should learners interact with ITSs individually or collaboratively?
- What is the nature of learning? Is it situated, unique, and ongoing? Or is it symbolic and following an information-processing model?
- Does virtual reality (VR) uniquely contribute to learning beyond CAI, ITS, or even multimedia?

For ITSs to have a great impact on education, these and other issues must be resolved. To take advantage of newer, more effective instructional techniques, ITSs of the future will have to allow for increased learner initiative and between-learner collaboration (Shute and Psotka, 1996). ITSs must also assess learning as it transfers to authentic tasks, not standardized tests, and establish connections across fields, so topics are not learned in isolation. A more

fruitful approach for ITS development may be to develop specific cognitive tools, for a given domain or applicable across domains. Such a transition would allow future ITSs to be everywhere, as embedded assistants, to explain, critique, provide online support, coach, and perform other ITS activities.

2.2 The Task-Specific Architecture and Generic Task Expert Systems Development Methodologies

Early Artificial Intelligence research followed what is currently referred to as the “first generation” approach to knowledge-based systems. This approach is characterized by the use of general-purpose tools to model intelligence. A typical example is the widely used technique of *rule-based* or *production systems* (Newell and Simon, 1972; Shortliffe, 1976; McDermott, 1982), which represent knowledge in the form of a set of “if-then” rules, and utilizes a general inferencing mechanism, such as forward or backward chaining.

With the widespread use of rule-based systems, several problems with this approach began to emerge. One problem was that the inferencing mechanism adopted was too rigid. Reasoning was usually limited to forward or backward chaining, allowing little flexibility for solving problems that required a different mode of reasoning. In addition, a general-purpose inferencing mechanism did not help in analyzing the problem. Another problem encountered was that not all types of knowledge could be naturally represented in the form of rules. Moreover, as production systems got larger and more complex, it became harder to describe what the system does at a “knowledge level,” independently of its implementation.

Recognizing these problems, several schools of thought emerged forming what is currently known as second-generation knowledge-based systems (Chandrasekaran, 1983; Chandrasekaran, 1986; Wielinga and Breuker, 1986; McDermott, 1988; Sticklen, 1989; Steels, 1990). While these schools differ in detail, they share a common philosophy, each having deviated from the general tools approach by developing *task-specific* taxonomies of problem solving types or categories. The common assumption of these approaches is that human problem solving can be classified into categories of problems. Each category shares a common methodology, while the knowledge needed for solving individual problems would differ from one problem to another.

2.3 Task-Specific Architectures

By grouping problems into task categories, task-specific approaches avoid the generality of first-generation techniques, while avoiding the pitfall of having to design a new technique for every problem. In other words, by carefully analyzing a class of problems, a framework can be formulated that can be applied for analyzing similar problems. Furthermore, task-specific architectures facilitate the knowledge acquisition process by focusing on the high level descriptions of problem solving and not on the low level implementation specifics. By implementing the results of the analysis in the form of an expert system development tool, the analysis and the development of another expert system can be greatly simplified. The tools thus serve to guide the analysis of the problem.

While the different task-specific approaches share a common philosophy, they differ in detail. For example, the generic task (GT) approach is based on the hypothesis that knowledge should be represented differently according to its intended use (Chandrasekaran, 1986). While this mode of knowledge representation can lead to duplication in knowledge stored, it also leads to a more efficient mode of problem solving, since the knowledge in each instance is represented in a form suitable for its immediate use. By representing knowledge according to its intended use, it also becomes more readily usable for effectively transferring the system's expertise to less experienced humans. The generic task approach identifies a number of domain-independent tasks or methods and defines an implementation tool for each of these tasks, which embeds control knowledge specific to the task. As such, these tools can be used to facilitate the acquisition of the requisite domain knowledge directly in the form in which it will be used.

On the other hand, other task-specific approaches such as the KADS approach (Wielinga, Schreiber et al., 1992) view knowledge as being independent from its intended use. Instead, knowledge is represented in an abstract neutral format that is expanded at run-time to generate the constructs needed. Some of the recognized task-specific approaches are described next. For a more detailed account of other task-specific approaches, see (McDermott, 1988; Steels, 1990; Wielinga, Schreiber et al., 1992).

2.3.1 KADS and CommonKADS

The CommonKADS or KADS-2 approach (Knowledge Acquisition and Documentation System) (Schreiber, Wielinga et al., 1994), the successor of KADS (Knowledge Analysis and Design Support) (Wielinga and Schreiber, 1994), views the development of KBS as a modeling activity. KADS provides a general framework for the analysis of expert problem solving and the development of knowledge-based systems.

According to the CommonKADS methodology, the development of KBS requires the construction of a number of models that represent different views on problem-solving behavior. For example, the construction of the organizational model requires specifying the business resources, structural units, business processes, and the relationship between these components. Other models recommended for the development of KBS are: task model, model of capabilities, model of communication, model of expertise, and model of design. The central activity in the process of KBS construction is building the model of expertise, which is described using the following components:

- The *domain ontologies* specify the concepts, properties of concepts, and relations, etc. of the problem domain. The domain ontology provides a vocabulary to describe the *domain model*.
- The *inference knowledge* represents the knowledge-based inferences, which are performed during problem solving. Inference knowledge is defined by *inference functions* (e.g., classification) and *knowledge roles* (e.g., the domain knowledge which forms input and output of the inference function). The

inferences are also associated with assumptions on the availability and various properties of domain knowledge.

- The *task knowledge* is defined from the functional perspective by specifying the goals in terms of input/output relations and its *control structure*. The control structure represents the decomposition of the task and procedural ordering on the inferences.

2.3.2 PROTÉGÉ-II

PROTÉGÉ-II (Tu, Eriksson et al., 1995) is a methodology and a tool for generating knowledge acquisition tools. PROTÉGÉ-II, similar to the KADS approach, views knowledge as being independent from its intended use. Knowledge is represented using domain ontologies, while problem-solving methods are represented separately in method ontologies. The mapping between the domain data and problem-solving methods is considered a part of the system development process. However, PROTÉGÉ-II provides a means for the explicit encoding of domain-dependent control knowledge in the control model, which distinguishes it from the KADS approach. This ability allows leveraging of the application domain specifics during the problem-solving process.

The approach relies on a library of reusable basic problem-solving methods called mechanisms. Mechanisms can be configured into more complex methods using a method specification language. Methods are used for solving tasks, which, in turn, are represented through their input and output. The method is either recursively decomposable into a set of subtasks or is a basic method.

Each of the subtasks may have more than one method for its solution. The task decomposition in PROTÉGÉ-II is similar to the KADS task-method decomposition. The difference between the KADS approach is that in PROTÉGÉ-II, mechanisms contain no computational assumptions about the procedure or the required knowledge.

The PROTÉGÉ-II approach uses three types of ontologies:

- Domain ontologies, which model concepts and relations in the problem domain
- Method ontologies, which model concepts related to the domain-independent problem solving methods: input and output, control structure, etc.
- Application ontologies, which combine domain and method ontologies for particular applications

2.3.3 EXPECT

EXPECT (Swartout, Gil et al., 1999; Valente, Gil et al., 1999) is a framework for developing knowledge-based systems using self-organized method libraries. EXPECT's knowledge base includes:

- Domain knowledge: Domain ontologies and domain facts are represented using semantic networks.
- Problem solving knowledge: The methods are specified through the capability description (description of problems that can be solved) and method body (procedural description of steps for achieving method capability).

EXPECT uses case grammar-style formalisms to describe method capabilities and problem-solving goals. This formal specification allows the:

- development of libraries of problem-solving methods that are self organizing
- creation of a problem solver for a specific task automatically by matching the goals to a description of the methods in the library

The generation of KBS in EXPECT starts with an initial high-level goal of the problem solver. This goal is used to find the method in the library with matching capabilities. The method is then instantiated. If the method's body contains sub-goals, the process described above is repeated.

The EXPECT approach is similar to the other task specific approaches described previously: the libraries of problem-solving methods are developed based on a functional description of the methods. Unlike other approaches, the EXPECT framework offers a formal scheme based on case grammar to describe the capabilities of the problem-solving methods. This framework allows the construction of self-organizing libraries and supports reuse. Despite the ambitious goals of the EXPECT project, it is difficult to judge its scalability, since no big problem solvers have yet been developed with it. It is possible that an unambiguous mapping from an informal task description to a formal specification can be very difficult or even impossible for large, complex problem solvers.

2.3.4 Generic Tasks

The idea of generic tasks can be understood at one level as a semantically motivated approach to developing reusable software - in particular reusable shells for knowledge-based system analysis and implementation. Each GT is defined by a unique combination of: (1) a well-defined description of GT input and output form, (2) a description of the knowledge structure which must be

followed for the GT, and (3) a description of the inference strategy utilized by the GT (Chandrasekaran, Johnston et al., 1992; Chandrasekaran and J. R. Josephson, 1997). To develop a system following this approach, a knowledge engineer first performs a task decomposition of the problem, which proceeds until a sub-task matches an individual generic task, or another method (e.g., a numerical simulator) is identified to perform the sub-task. The knowledge engineer then implements the identified instances of atomic GT building blocks using off-the-shelf GT shells by obtaining the appropriate domain knowledge to fill in the identified GT knowledge structure. Having a pre-enumerated set of generic tasks and corresponding knowledge engineering shells from which to choose guides the knowledge engineer during the analysis phase of system development.

Several of these atomic generic tasks are currently available, and are implemented in the Intelligent Systems Lab toolkit. These include structured or hypothesis matching (Chandrasekaran, 1986), hierarchical classification (Gomez and Chandrasekaran, 1981) and routine design (Brown, 1991), which are described below in more detail.

2.3.4.1 Structured Matching

Structured matching (SM) (Chandrasekaran, 1986) is a simple inferencing mechanism for performing inferences of the form: "If conditions A, B, C... hold, then result X is valid." Rules of this form are combined and organized into patterns corresponding to a particular portion of the problem domain, constituting a domain-driven structure. The structured matching task involves hierarchical

symbolic abstraction. An abstraction of the data is computed in the form of a degree of fit; it is symbolic because the abstraction is presented as a discrete qualitative measure of fit (e.g., strongly match, weakly match, against, etc.) and hierarchical because the final abstraction is evaluated from intermediate abstractions, which in turn can be derived from nested sub-abstractions. This task of matching hypotheses against data is a general type of reasoning useful in a number of contexts, and is used in conjunction with other generic tasks.

2.3.4.2 Hierarchical Classification

Hierarchical classification (HC) (Gomez and Chandrasekaran, 1981; Bylander and Mittal, 1986) is intuitively a knowledge representation and inferencing technique for selecting among a number of hierarchically organized options. Knowledge is organized in the form of a hierarchy of pre-specified categories. The higher level categories represent the more general hypotheses (e.g., a mechanical problem in diagnosing an automobile), while the lower level categories represent more specific hypotheses (e.g., a clogged valve or a bad spark plug). Inferencing uses an algorithm known as “establish-refine” in which each category attempts to establish itself by matching patterns of observed data against pre-defined matching patterns. Once a category is established, it is refined by having its sub-categories attempt to establish themselves. Pruning the hierarchy at high levels of generality eliminates some of the computational complexity inherent in the classification problem. While automobile engine diagnosis and medical diagnosis are very different in terms of the knowledge

involved, they both can utilize the same type of classification problem solving. Analysis of one of the two problems thereby contributes to analysis of the other.

2.3.4.3 Routine Design

The *routine design* (RD) approach (Brown, 1991) was introduced to handle a certain class of design problems, namely “routine” design problems, i.e., problems that are of a well-understood nature. In such problems, the knowledge necessary to solve a problem is well known. However, the specific course of action necessary for solving any particular instance of the problem is not necessarily known in advance. As such, the result of a routine design problem can be a novel design, but the method used to achieve the design cannot be novel.

This method is based on a hierarchy of cooperating specialists, each responsible for an identified part of a complete design. A design specialist represents a particular piece of design knowledge about some part of the overall design. The higher level specialists in the hierarchy typically represent more conceptual aspects of the design process, whereas the lower level specialists represent more parametric aspects.

In routine design, each specialist follows one of a set of pre-specified plans. Plans prescribe the problem solving actions to be followed and are defined at the time of building a design system. While the plans of the design process are known in advance, their combination and ordering is determined only during the problem solving. Although each specialist has a pre-defined set of plans to choose from, the actual plan to be followed is dynamically chosen during run-

time based on the design input requirements as well as the results of the design established so far. The choice of plans for every specialist can therefore lead to the generation of a design, which is novel, since specialists can use different plans that have not previously been used together. However, RD is not suitable for completely novel design problems.

2.3.4.4 Functional Modeling

Functional modeling (FM) captures role-oriented causal knowledge about how a “device in the world works.” Unlike other identified generic tasks, functional modeling does not have a prescribed inference strategy. Most explored use of FM has been to support a type of qualitative simulation. (Sticklen and Chandrasekaran, 1985; Chandrasekaran, 1993; Hawkins, McDowell et al., 1993).

2.3.4.5 The Knowledge-Level Architecture

Although task-specific architectures and generic tasks are very useful for solving real world problems, many problems cannot be solved by a single problem solving mechanism. Some problems are multi-task in nature, requiring several distinct tasks to solve them. Thus, there is a need to facilitate the interaction of multiple problem solving types.

Prior to providing a mechanism for multiple task interaction, there must be a common ground for comparison. The *Knowledge Level* as proposed by Newell provides the first step in determining this baseline (Newell, 1982). Newell stated the Knowledge Level Hypothesis as:

There exists a distinct computer system level lying immediately above the symbol level, which is characterized by knowledge as the medium and the principle of rationality as the behavior.

The Knowledge Level provides a way of understanding a problem solving agent apart from the implementation details. Although this allows a deeper understanding of problem solving than that possible through a symbolic level analysis, it does not always permit the prediction of the behavior of an agent. This deficiency is due to the lack of any considerations of the problem solving control.

The *Knowledge Level Architecture* (KLA) (Sticklen, 1989) provides an organizational overlay to the basic generic task approach to facilitate integration. The underlying hypothesis builds on Newell's proposal by allowing the discussion of knowledge organization and control at the Knowledge Level. This organization and control are ensconced in the KLA hypothesis:

If a problem solving agent may be decomposed into the cooperative efforts of a number of subagents, the larger agent can be understood at the Knowledge Level by giving a Knowledge Level description of the subagents and specifying the architecture the composition follows.

The KLA leads to the specification of a system by explicitly representing the interactions between its agents. There are two defining aspects of the KLA: first, there is a distinct message protocol that exists between problem solvers. The message protocol between two cooperating agents is defined in terms of the functionality of the agents. In other words, the protocol provides a means for the agents to request work (from other agents) and respond in a vocabulary that the other agents can understand. Second, to allow communication between cooperating problem solvers, communication channels are established. By decomposing the agent into subagents and fixing the communication paths, the KLA provides a way of organizing the knowledge of the agents.

Overall, the GT approach, extended with the KLA, conceives the overall job of knowledge-based system development on three levels:

1. The KLA level: At the highest conceptual level the goal is to represent the interaction of problem solving modules. The vocabulary is of the communication channels between the individual modules and the pair wise request types between the module types.
2. The atomic GT level: At this middle level the goal is to represent the representation vocabulary and inference strategies of the atomic generic tasks.

The implemented problem solving module level: At this level, knowledge engineers utilize the frameworks offered at the TSA level and the atomic GT level to guide problem analysis and knowledge acquisition and build working systems.

2.3.5 Analysis of the Task-Specific Approaches

This section outlined prevalent task-specific approaches. These approaches can be divided into two groups based on their position toward knowledge representation. KADS, PROTÉGÉ, and EXPECT advocate a use-independent domain knowledge representation, whereas the GT approach has task-specific ontological commitments. Central to GT is the hypothesis that “the knowledge should be represented differently according to its intended use” (Chandrasekaran, 1986). While this mode of knowledge representation can lead to duplication in the knowledge stored, it also results in more efficient problem solving, since the knowledge in each instance is represented in a form most suitable for its immediate use. Moreover, such a use-oriented representation

supports more effective transfer of human problem-solving knowledge to a system's representation.

The differences among the task-specific approaches have important implications for tutoring support. Although all the frameworks can support the generation of ITS, a GT framework is expected to be of greater leverage in generating ITSs because of the larger problem solving *granularity* of the GT approach compared to the other approaches. The inferencing mechanisms in the other approaches are more fine-grained than their GT counterparts, so the tutoring support must be specified at a lower level. In addition, in the GT approach, a system developer is strongly tied to a backing theory of problem solving that provides a good semantic foundation for knowledge-based systems development. These stronger commitments made in a GT-based system enable a *general* ITS overlay to be developed to generate ITSs on an automatic and largely knowledge-free basis.

2.4 Learning and Cognition Issues for ITS Development and Use

This section examines several findings about cognition and the learning process, and their implications for the development and use of intelligent tutoring systems.

2.4.1 Key Findings on Learning

The science of learning aims to provide knowledge to significantly improve people's abilities to become active learners who seek to understand complex subject matter and are better prepared to transfer what they have learned to new problems and settings. Although this is a major challenge, many recent findings are paving the way towards improving our understanding of learners and learning

(Bransford, Brown et al., 2000). Three major findings that have strong implications for learning are:

- Learners have preconceptions about how the world works. If their initial understanding is not referenced or activated during the learning process, they may fail to understand any new concepts or information.
- To learn about and be competent in a subject area, learners must:
 - (a) have a deep foundation of factual knowledge
 - (b) understand facts and ideas in the context of a conceptual framework
 - (c) organize knowledge in ways that facilitate retrieval and application
- A “metacognitive” approach to instruction can help students learn to take control of their own learning by defining learning goals and monitoring their progress in achieving them.

2.4.2 The Role of Knowledge Structuring in the Learning Process

In the section above, one of the key findings was that for people to become competent in a domain, they need to not only have a deep knowledge base of information related to that domain, but they must also be able to understand that knowledge within the context of a conceptual framework, and be able to organize that knowledge in a manner that facilitates its use. This finding emerges from research that compares the performance of experts to novices and from research on learning and transfer. Experts develop a richly structured knowledge base that they use to transform factual information into usable knowledge. Experts are also able to easily access relevant knowledge because their conceptual framework allows them to quickly identify what is relevant.

A key finding in the learning and transfer literature is that organizing information into a conceptual framework allows for greater transfer of knowledge. By developing a conceptual framework, students are able to apply their knowledge in new situations and to learn related information more quickly. For example, a student who has learned problem solving information for one topic in the context of a conceptual framework approaches the task of learning problem solving for another topic in a more relevant manner that helps guide the acquisition of new information.

2.4.3 The Transfer of Knowledge

There exists a relationship between the learning and transfer of knowledge to new situations. Transfer is usually a function of the relationships between what is learned and what is tested. For students to transfer knowledge successfully across domains, they must conceive of their knowledge base as growing continuously, instead of in discrete steps.

Recent research by Singley and Anderson indicates that the transfer of knowledge between tasks is a function of the degree to which the tasks share cognitive elements (Singley and Anderson, 1989). In their study, Singley and Anderson taught students several text editors, one after the other. They found that students learned subsequent text editors more rapidly and that the number of procedural elements shared by the two text editors predicted the amount of transfer. Their results showed that there was large transfer across editors that were very different in surface structures but that had common abstract structures.

Singley and Anderson were able to generate similar results for the transfer of mathematical competence across multiple domains.

2.4.4 Issues in Using Technology to Support Learning

Emerging computer-based technologies hold great promise as a means of supporting and promoting learning. There are several ways that such technology can be used to help meet the challenges of developing effective learning environments:

- Bringing real-world problems to the learning environment.
- Providing “scaffolding” support to learners during the learning process.
Scaffolding allows learners to participate in complex cognitive experiences, such as model-based learning, that is more difficult without technical support.
- Increasing opportunities for learners to receive feedback and guidance from software tutors and learning environments.
- Building local and global communities of teachers, administrators, students, and other interested learners.
- Expanding opportunities for teachers’ learning.

Learning environments need to be developed and implemented with a full understanding of the principles of learning and developmental psychology. In addition, these new learning environments need to be assessed carefully, including how their use can facilitate learning, and the cognitive, social, and learning consequences of using these new tools.

2.5 Analysis

This research work is at the crossroads of three research fields: intelligent tutoring systems, knowledge-based systems, and education. In this chapter, we have described background information from each of the three fields that is important for understanding this research work. In this dissertation, we describe a novel approach for generating intelligent tutoring from existing expert systems and reusable tutoring components. Our approach builds upon important aspects from the three disciplines.

First, the approach utilizes the standard architecture of an ITS, including the expert model, student model, instructional manager, and user interface components. In fact, the architecture developed *reuses* the ITS components for different domains. Second, the architecture can generate ITSs for different domains by interfacing with different knowledge-based systems. The expert model component of the ITS can interact with any generic task expert system and extract domain and problem solving knowledge for tutoring. Third, in section 2.4.3, we indicated that the transfer of knowledge between tasks is a function of the degree to which the tasks share cognitive elements. Since the architecture can generate ITSs for a specific class of tasks that share structure and reasoning strategy, it can identify and utilize learning strategies that are appropriate for task-specific tutoring.

Chapter 3. Related Research on ITS Authoring Environments

As the need for computer-based learning systems is becoming more widespread, a variety of tools and authoring systems for building tutors are being developed, in both commercial and academic settings. The tools available commercially are mostly general-purpose; they can be used to build tutorials, courseware, reference manuals, and other educational software. Various research labs at academic and other institutions have produced development tools that are intended primarily for building learning software. Many of these systems hope to make tutoring systems more available and more effective by allowing teachers and trainers to build tutoring systems with their authoring environments.

One issue that needs to be clearly defined is the notion of who the user is. For ITS authoring environments, there are two levels of users. At the first level, the users are the people who wish to use ITS authoring environments to develop tutoring systems for specific purposes. These users will be called *authors*, and can include teachers, trainers, instructional designers, domain experts, etc. At the second level, the users are those who will use the tutoring systems developed for learning about a specific topic, task, or domain. These users will be called *learners*.

This chapter reviews other efforts aimed at building development tools, authoring environments, and shells for tutoring systems, and analyzes some of their benefits and limitations. The first section describes some commercially available general-purpose tools that can be used to develop computer-based

learning material. Then, Artificial Intelligence (AI) based systems that have been developed mostly in research labs are reviewed.

3.1 Commercial Tools

The commercial market currently includes a wide variety of general-purpose authoring tools, which can be used to build educational software systems. Many of these tools provide graphical user-interfaces (GUIs) and support multimedia development. Commercial tools are most commonly used to build computer-based courseware.

Tools currently available are of two types: scripting language tools and graphical development tools. Scripting language tools include HyperCard (developed by Apple Computers, Inc.) and Toolbook and its CBT Systems Edition for building courseware (developed by Asymetrix Learning Systems, Inc.), among others. To develop a system using these tools, authors perform two tasks. First, they create interface elements, such as buttons, boxes, menus, and graphics. Then, users define a “script” for each interface element, which represents the action that the interface element will take when active. For example, a script for a button could define the action the system takes when the button is clicked. Systems developed using these tools usually take the form of a series of pages or “cards.” A card contains a chunk of information, including text, graphics or other media types. Cards are presented to the user according to the user’s actions.

Graphical development tools, such as Authorware (developed by Macromedia, Inc.) and IconAuthor (developed by Asymetrix Learning Systems,

Inc.), provide similar features, but without the need for scripting. Users build systems by choosing from among different icon types and then linking and manipulating them graphically. In other words, the user develops a system by building a system flowchart using graphical objects. These tools are generally easier to use than scripting tools. In addition, they provide a scripting facility to allow more powerful systems to be built.

Graphical and scripting tools described above have two main drawbacks. First, both types of tools focus on building a system through the user interface. Rather than facilitate the development of teaching systems based on principles of instructional design, these tools constrain the development in terms of the system's interface. In other words, design is done in terms of physical components, rather than at a conceptual level. Thus, people who use such tools to develop computer-based courseware must find a way to map their domain content and teaching techniques directly to interface components. The second drawback of most commercial tools is that they lack the sophistication to build "intelligent" tutoring systems. Most tools do not incorporate or make use of any AI techniques, and rely on flowcharting and other simple techniques to represent the knowledge of a tutor.

To build more powerful, flexible, and adaptive tutors, authoring tools must make a paradigm shift from traditional "story board" representations of instructional material to more powerful and flexible knowledge-based representations (Murray, 1996). Commercially available authoring tools utilize a representation of instructional content and flow that is explicit, non-modular, and

fairly linear. Murray refers to these systems as “story board” systems, because they are based on enumerating all of the screens and having explicit links from each screen to the next. In contrast, in a knowledge-based representation, the instructional content is separated from the specifications of how and when that content is presented to the student, so that the content can be reused in multiple ways. Moreover, with AI-based tools, instructional actions can be designed using pedagogically relevant building blocks, rather than at the level of the media, using text, pictures, button clicks, and similar objects.

3.2 AI-based ITS Authoring Systems

Driven by the need for achieving cost-effective and reusable ITS shells, many research efforts are underway to develop authoring tools or environments for tutoring systems. ITS authoring tools can speed up ITS development and reduce costs by providing essential infrastructure along with a conceptual framework within which to design and build educational systems. However, a commitment to an authoring environment’s underlying philosophy entails other commitments that may or may not be suitable for application in a certain domain. Several recent reports have discussed design, implementation, and delivery issues for ITS authoring (Bloom, 1995; Murray, 1996; Rowley, 1996).

The ITS authoring design space can be envisioned as in figure 4 (Murray, 1999). The dimensions are system power and flexibility (x), system usability (y), and system fidelity and cost (z). The power/flexibility dimension has two aspects: (1) breadth, the generality of the framework for building diverse tutors, and (2) depth of system’s knowledge base and model. Breadth and depth are often at

odds, because the authoring system's generality must often be limited to allow deep representations of the tutoring knowledge (Murray, 1999). The usability dimension also has two aspects: (1) learnability, how easy the system is to learn, and (2) productivity, how easy it is to use. Learnability and productivity are also often opposing, since a system that is designed to be easily used by novices may not provide the powerful features required to build effective tutors. Fidelity implies the degree to which the tutor built matches its target domain. And cost refers to the amount of resources needed to develop the authoring system.

The design space for ITS authoring tools is large. In addition, the dimensions of this design space are often conflicting with each other. This is because as systems become more general purpose, they become more complex, and generally, more difficult to use. Systems that are too specific will generally not allow enough flexibility, and thus provide little reuse. The tradeoffs revolve around the usability and flexibility of more general tools, versus the power that is derived from knowledge-rich tools based on explicit and detailed models. A good solution will fall towards the middle of the three-dimensional space, providing enough power and flexibility, while still maintaining a good degree of usability and fidelity, and also maximizing cost-effectiveness. Note that the important metric of "instructional effectiveness" is not included in the ITS authoring design space, because this depends heavily on how the authoring system is used to produce a tutor. It is assumed that the production of ITSs that are instructionally effective is an inherent design goal of authoring systems.

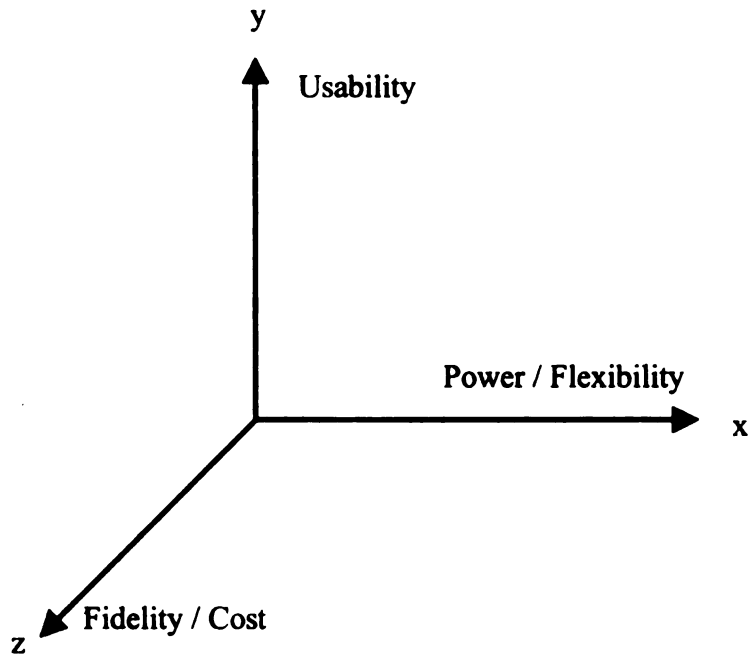


Figure 4. Design space for ITS authoring systems

The process of creating ITS authoring tools is complex and challenging. The difficulties in creating powerful and usable tools reflect the complexities of the target application. ITSs require domain knowledge, pedagogical reasoning skills, understanding of the user and the user's tasks and goals, interactivity, and an engaging user interface (Bell and Redfield, 1998). These factors make it unlikely that a single tool can be developed that effectively addresses these disparate areas across a broad range of applications.

There is much diversity within the ITS research and development community as to what constitutes an ITS authoring system. This is generally due to a lack of agreement within the field on ITS standards and ontologies. This issue is currently being addressed by the IEEE Learning Technology Standards Committee (P1484 - LTSC). LTSC has been chartered by the IEEE Computer Society standards activity board. The mission of the IEEE LTSC committee is "to

develop technical standards, recommended practices, and guides for software components, tools, technologies and design methods that facilitate the development, deployment, maintenance and interoperation of computer implementations of education and training components and systems.” This international consortium consisting of people from academic institutions, research groups, and major corporations, is collaborating to develop a standard model of a learning environment. The committee has a long way to go towards developing a standard learning environment model before it can work on a standard model for *authoring* learning environments. For more information on this effort, see <http://grouper.ieee.org/p1484/>.

The diversity within the field is expressed in the many names associated with ITS authoring environments. Some researchers are building *development tools*, which can be used to build individual ITS components, and, in some cases, even integrate those components. Other research efforts are aimed at building *authoring systems*, which are intended to be used by instructional designers, teachers, or domain experts, to build complete tutoring systems. Yet, other researchers are engaged in developing *shells*, which are domain-independent frameworks for building ITSs and representing the necessary knowledge. ITS authoring systems differ from shells in that they incorporate user interfaces that allow nonprogrammers to formalize and visualize their knowledge in building ITSs. In this dissertation, the term *authoring environments* will be used to refer to all of the different types defined above. The rest of this section describes various research efforts for developing ITS authoring environments that have been

reported. The systems discussed are categorized according to their central methodology, which is defined in each section.

3.2.1 Early Attempts

The 1980s were characterized by enormous growth and momentum in the ITS field. Driven by the need for more tutoring systems, and the difficulty of developing them, some researchers invested in the idea of developing authoring tools for ITSs – tools that could make tutors more available and easier to build. The overall goal was to provide software tools that enabled relative computer novices to take advantage of computers for developing instruction.

One of the earliest efforts dates back to Clancey's efforts to produce a tutoring system shell from parts of GUIDON (Clancey, 1987). The same tutoring rules developed for GUIDON were later used to build a system for teaching a student how to select the appropriate structural analysis software packages to run in order to analyze a particular structure. Based on this earlier work, Training Express was an effort to build a practical tool for creating simple versions of these types of intelligent tutoring systems (Clancey and Joerger, 1988). Although this system had its limitations, it paved the way for other authoring research and development thrusts. Other early attempts included:

- PIXIE: a domain-independent ITS shell for diagnosing and remediating student errors in a particular domain (Sleeman, 1987).
- IDE (Instructional Design Environment): a design and development system for building curriculum-based tutoring systems (Russell, Moran et al., 1988).

- ID Expert: an expert system for instructional design and development (Merrill, 1989).
- TDK (Tutor Development Kit): a programming shell in the form of a set of tools that help programmers to construct ITSs (Anderson and Pelletier, 1991).
- KAFITS (Knowledge Acquisition Framework for ITS): a set of tools for the acquisition of ITS domain and teaching knowledge (Murray, 1991).
- ABC-1.5 (an Author Builds a Course): an authoring programming language for the implementation of ITSs (Petrushin, 1992).

Although these early attempts were not entirely successful, one outcome they did achieve was to spur a flurry of further research on authoring environments for ITSs.

3.2.2 General-purpose Authoring Environments

General-purpose authoring environments attempt to provide the tools necessary to build all the ingredients of an intelligent tutoring system. This type of environment will include tools for acquiring and representing domain knowledge, creating instructional strategies, and building student models. General-purpose systems provide the most flexibility because they can be used by instructional designers, teachers, or domain experts to produce a wide variety of tutoring systems and educational software. Unfortunately, their generality imposes two significant drawbacks, compared to other types of ITS authoring environments.

First, they are more difficult to develop. One of the fundamental goals of building authoring systems for ITSs is to make tutoring systems easier to develop. Trying to build extremely general and flexible authoring environments is,

in itself, a complex, time-consuming, and tedious process, and thus, will not bring us any closer to the original goal. Indeed, many attempts to build these types of systems suffer a futile end. Second, these systems are generally hard to use. The more features they provide and functions they accomplish, the more cognitive load they put on authors using such a system. Even authors with good computer programming experience may need training to learn how to use a general-purpose system.

Due to the difficulty involved in developing and using them, not many such authoring environments exist. One example of a general-purpose authoring environment is Eon (Murray, 1998). Eon is an integrated set of powerful tools for developing complete knowledge-based tutors. Murray admits that the system may be too laborious to use for developing tutoring systems directly:

"After using the tools for several tutors and with several authors, I find in that the authoring system, although fairly large (the user documentation is several hundred pages long), is composed of tools that are quite learnable and usable. However, the process of starting from scratch to build an ITS is still a formidable one." (Murray, 1998) (p. 54)

The solution Murray suggests is to use Eon as a meta-authoring tool to build special-purpose authoring environments that, in turn, can be used to create tutors for specific classes of domains or tasks, or other special purpose. Special-purpose authoring tools can be packaged with pre-built libraries of components tailored to specific needs. In other words, they can include default teaching strategies, student modeling rules, interactive screens, and a topic structure that is tailored to a specific type of domain or task.

3.2.3 COTS and Component-based Tools

Recognizing that authoring an ITS development environment is a multi-faceted task and is unlikely to be accomplished within a single system, several research groups initiated efforts to build separate authoring tools directed at individual ITS components. In this way, components for which good authoring tools already exist may be identified, so the development task will be simplified through reuse of the existing components. An example of this type of work is the architecture for plug-in tutor agents and component-based learning environments under development at Carnegie Mellon University (Ritter and Koedinger, 1996; Ritter and Blessing, 1998). In contrast to the “all-inclusive” architecture of general-purpose authoring environments, plug-in tutor agents do not contain all aspects of the learning environment. Instead, tutor agents can be embedded within existing software tools that lack educational support. Components can then be connected together to form a complete learning environment. The CMU architecture provides a tool called the Visual Translator, which allows users to link their tools to pre-existing tutor agents.

The ESSCOTS approach (McArthur, Lewis et al., 1995) aims to build educational software environments (ESS) around commercial off-the-shelf (COTS) software products. ESSCOTS systems provide scaffolding for commercial software, so that students can take better advantage of them. The systems can adapt to the learner’s current skill level by continuously monitoring the learner’s performance. Another system that falls under this category is the Computer Integrated Learning Environment (CILE) project, which aims to build

and integrate Intelligent Tutoring Tools (ILLs) to form comprehensive tutoring systems (Patel and Kinshuk, 1996).

Although this approach can produce fruitful results, several issues must first be resolved. First, standards for communication between the components of learning environments must be developed. Cheikes recognized this problem as part of the MITRE Corporation's effort to build an agent-based architecture for ITSs (Cheikes, 1995; Cheikes, 1995). Second, specialized authoring tools that embody these communication standards must then be developed. Only then will it be possible to build real learning environments that can take advantage of existing ITS components and commercial tools.

3.2.4 Instructional Strategy Authoring

These types of authoring environments focus on providing a set of instructional strategies to be used within an ITS. In some systems, authors can select from a pre-defined set of general teaching methods, while in other systems, authors can build their own instructional techniques. The Teaching Executive (Jona and Korcuska, 1996), REDEEM (Major, Ainsworth et al., 1997; Ainsworth, Underwood et al., 2000), CREAM-Tools (Nkambou, Gauthier et al., 1996), COCA (Major, 1995) and GTE (Van Marcke, 1992; Van Marcke and Vedelaar, 1995) are examples of authoring environments that focus on instructional strategies development and use.

In the Teaching Executive, authors can choose from pre-authored libraries of teaching strategies to build tutors with. REDEEM (Reusable Educational Design Environment and Engineering Methodology) is a suite of software tools

that teachers can use to describe course material, construct teaching strategies, and classify students in order to develop a complete learning environment. With REDEEM, authors can assemble instructional strategy components to form a complete strategy. In contrast, GTE (Generic Tutoring Environment) only allows authors to select whole strategies from a large set of reusable techniques. Other systems of this type include Genitor (Kameas and Pintelas, 1997) and EDDI (Marcenac, 1992), which also include support for authoring a learner model.

The main drawback associated with this type of system is that they assume that every topic can be taught in essentially the same manner. Such systems generally adopt a course-like approach to instruction; learners are presented with material as a set of “course notes,” and then tested periodically on the material. Various instructional techniques are employed to help learners progress through the “course.” The underlying conjecture that the same type of instruction can work well for all domains is not pedagogically well founded.

3.2.5 Interface Authoring

Some authoring systems exist to help in building the user interface and communication components of an ITS. The aim of tools within this category is to automate (entirely or partially) the interface construction process, since creating a user interface can be a programming-intensive task. Some take the approach that the interaction between user and tool is typically guided by a model. These systems are targeted for use in generating interfaces for graphical browsing of either domain models or knowledge bases. Such systems include GENIUS (Janssen, Weisbecker et al., 1993), HUMANOID (Puerta, Eriksson et al., 1994),

and Mecano (Szekely, Luo et al., 1993). However, most systems usually focus on developing rich multimedia content, rather than on intelligent tutoring systems. More recently, some researchers are trying to alleviate this problem by building multimedia authoring tools that make use of intelligent models and separate the instructional content from the learning strategies (Wong and Chan, 1997).

3.2.6 Simulation Systems Authoring

This section describes various authoring environments developed specifically for tutoring in device or simulation-based environments. They are similar to domain-specific and task-specific authoring systems, which will be described in the following sections, but target only simulation domains or tasks. Simulation-based environments generally differ from more traditional learning environments in two ways. First, the nature of the domain model is different, usually represented as a quantitative, executable model of the system to be learned. Such a model is designed for the purpose of efficiently and accurately calculating the simulation variables, and normally does not have instructional features. Second, the learner interaction with a simulation-based system is different. In traditional ITSs, interaction usually takes the form of system-initiated dialogue, with the learner reacting to the information presented. In simulations, both the system and learner may initiate a sequence of events. However, once started, the simulation will proceed independently of the learner's actions, and events resulting from the ongoing simulation may result in learning opportunities. Antao et al. (Antao, Broderson et al., 1992) present a general framework for building ITSs for simulation systems.

Authoring environments that exist for device simulation and equipment training systems include: RIDES (Munro, Johnson et al., 1997), XAIDA (Redfield, 1997; Hsieh, Halff et al., 1999), SMISLE (Joolingen and Jong, 1996), and SimQuest (Joolingen, King et al., 1997). Joolingen later extended his work to develop an architecture for collaborative discovery learning (Joolingen, 2000). RIDES is an integrated software environment for developing and delivering instruction and practice that is based on graphical simulations. Using RIDES, authors can build graphical simulation models of devices and then build interactive lessons in the context of those models. RIDES is a successor of earlier development efforts by the same research group, namely IMTS, RAPIDS, and RAPIDS II.

XAIDA, the Experimental Advanced Design Advisor is a system for the development of computer-based maintenance training in four areas: the physical characteristics of a device, its theory of operation, operating and maintenance procedures, and troubleshooting. XAIDA acquires knowledge of a device from a subject matter expert and applies common maintenance-training procedures to generate interactive training from the description. XAIDA relies on an instructional device known as a transaction shell, an instructional procedure applicable to particular instructional objectives of a specific type. XAIDA employs a different transaction shell for each of the four above-mentioned areas, and each shell employs a knowledge structure appropriate to the shell.

The SMISLE authoring system provides domain experts with a set of tools for designing simulation-based learning environments that support various

instructional techniques, including model progression, assignments, hypothesis scratchpads, and explanations. SimQuest, a follow-on project by the same research group, builds on the SMISLE approach, allowing authors to access libraries to select appropriate elements for different learning environments, with the addition of providing more flexible authoring, flexible exchange of material between different authors, and conceptual support for the author.

3.2.7 Ontology-based Authoring

Ontology-based authoring tools make use of an ontology as a theoretical foundation for the ITS development process. Ontologies can help in the ITS authoring process by providing continuity from the authors' conceptual understanding of an educational task to the ITS design and development process (Hayashi, Ikeda et al., 2000). The vocabulary and concepts of an ITS authoring ontology specifies the operational semantics of the learning content and enables the conceptual-level simulation of the learning content. The success of ontology-based authoring approaches depends on the existence of a level of agreement in the research community on the definition of ITS authoring standards. Moreover, the development of such ontologies does not provide any guidance to the authors on the actual design and development process.

Ikeda et al are working on the development of an ontology-based authoring tool named SmartTrainer/AT (Ikeda, Hayashi et al., 1999; Jin, Chen et al., 1999). SmartTrainer/AT incorporates a training task ontology as a fundamental knowledge source for yielding intelligent functions to support the authoring process. Authors can use the ontology to write "training scenarios"

using the SmartTrainer system. The tool also includes a conceptual level simulation feature that supports authoring by showing the behavior of the learning content as a structured behavior along the design intention.

3.2.8 Domain-specific Authoring

The objective of domain-specific authoring systems is to provide an environment for developing ITSs for a specific domain or content area. The environment, including pre-defined teaching strategies, system-learner interactions, and interface components, is geared towards a particular domain. Developers of domain-specific authoring environments try to determine the best instructional settings for a specific domain or topic and build environments that exploit those factors. Examples of domain-specific systems include the LEAP shell (Bloom, 1995; Dooley, Meiskey et al., 1995; Sparks, Dooley et al., 1999), Casper (Kass, 1994), and ITS-Engineering (Srisethanil and Baker, 1995). Also, Lajoie et al. are working on an authoring tool for developing computer-based learning environments (CBLEs) for specifically for medical diagnosis (Lajoie, Faremo et al., 2001).

The advantage that comes with domain-specific environments is that, because they target a particular domain, they can produce effective learning outcomes if they are built right and provide enough flexibility. The disadvantage is their lack of reusability for other domains. What counts as effective instruction for one domain will generally not work well for another domain. It is interesting to note that most domain-specific systems do provide some amount of reusability for the particular domain they address. For example, LEAP was developed as a

shell for building training systems for customer service employees at US West, but some of its components are reusable for other customer service training applications. In this sense, LEAP can be considered to be a task-specific shell to be used for developing ITSs for tutoring interaction skills. Also, ITS-Engineering was tested for construction engineering, but may be used for other engineering domains. However, the problem remains that the reusability provided by such systems does not transfer well to other domains.

3.2.9 Task-specific Authoring

Task-specific authoring environments aim to provide an environment for developing ITSs for a class of tasks. They are similar to domain-specific environments in that they incorporate pre-defined notions of teaching strategies, system-learner interactions, and interface components. However, these features are intended to support a specific class of tasks rather than a domain.

Several task-specific authoring environments have been developed within the ITS community. MOPed-II (Guralnick, 1996) is an authoring tool for building procedural task training systems. Another such environment is IDLE-Tool, the Investigate and Decide Learning Environments Tool (Bell, 1998), formerly known as the Goal-based Scenario (GBS) Builder. IDLE-Tool supports the design and implementation of educational software for *investigate and decide* tasks. Bell defines an investigate and decide task as a type of goal-based scenario in which the learner's task involves performing some investigations and then making a decision based on the supporting knowledge acquired from those investigations. Having recognized that IDLE-Tool lacks any real knowledge of the investigation

process, Bell's recent work focuses on adding an investigation map (IMAP) component to create a more knowledge-rich authoring tool (Bell, 1999). Another example of task-specific authoring environments is TRAINER (Reinhardt and Schewe, 1995), a shell for developing case-oriented training systems. TRAINER works by linking to an expert system to acquire and retrieve domain knowledge, and has been used to develop training systems for medical diagnosis.

Task-specific authoring systems offer the most flexibility, while still being powerful enough to build intelligent tutors. They are generally easier to use than general-purpose or component-based systems, because they target a particular class of tasks, and thus support a development environment that is best suited for those tasks. In addition, they afford more reusability than domain-specific environments because they can be used to develop tutoring systems for a wide range of domains, within a class of tasks. Moreover, task-specific authoring environments are likely to be more pedagogically sound than other type of authoring environments because they are capable of utilizing the most effective instructional and communication strategies for the class of tasks they address. Overall, the task-specific authoring approach falls very near the median of the ITS authoring systems design spectrum depicted in figure 3.

3.2.10 Summary of ITS Authoring Systems

A summary of the ITS authoring categories described above is shown in table 1. The table gives a brief description and some example systems of each category.

Table 1. Summary of ITS authoring environments, listed by category

Category	Description	Examples
Early attempts	Set the path for numerous ITS authoring research efforts	Guidon, PIXIE, IDE, TDK, KAFITS, ABC-1.5
General-purpose authoring environments	Provide tools to build all the ingredients of an ITS. Usually include tools for acquiring and representing domain knowledge, creating instructional strategies, and building student models	Eon
COTS and component-based tools	Attempt to build authoring tools for individual ITS components or commercial products	Plug-in tutor, ESSCOTS
Instructional strategy authoring	Provide a set of instructional strategies to be used within an ITS	The Teaching Executive, GTE, REDEEM, CREAM-Tools
Interface authoring	Help in building the user interface and communication components of an ITS	GENIUS, HUMANOID,
Simulation systems authoring	Authoring environments specifically for tutoring in device or simulation-based environments	RIDES, XAIDA, SMISLE, SimQuest
Ontology-based authoring	Make use of an ontology as a theoretical foundation for the ITS development process	SmartTrainer/AT
Domain-specific authoring	Provide an environment for developing ITSs for a specific domain or content area	LEAP, Casper, ITS-Engineering
Task-specific authoring	Provide an environment for developing ITSs for a class of tasks	MOPed-II, IDLE-Tool, TRAINER

3.3 Related Research on Explanation for Generic Tasks

Earlier research on generating explanations for generic task-based expert systems is also of relevance to this work. Explanation is one aspect of user interfaces for expert systems. Good explanations can make an expert system's advice more understandable and more believable. Chandrasekaran et al.

distinguished three components of the explanation problem (Chandrasekaran, Tanner et al., 1988):

1. How an expert system represents its own problem solving activity and retrieves relevant portions in response to user queries.
2. How the user's goals, state of knowledge, etc., are used to adapt the output to the user's needs.
3. How an appropriate human-computer interface displays and presents the information to a user in an effective way.

It is important to note that the solution for the first item above will affect the other two. In other words, if a bad representation is adopted, then no matter how good the user modeling and user interface are, the system will produce bad explanations. Thus, how well an expert system understands its own representation will determine the quality of the explanations it generates. Work by Chandrasekaran et al. focused on this aspect. They argued that the explanation of problem solving activity for the generic task class of expert systems is composed of three parts:

1. Explaining why certain decisions were or were not made. This has to do with how the data relates to the knowledge for making specific decisions.
2. Explaining the elements of the knowledge base itself, such as explaining the rationale behind using a particular piece of knowledge.
3. Explaining the problem solving activity and control behavior of the expert system. This would typically be at a higher level of abstraction than the explanations of the first type.

Much of this work concentrated on the roles of the task structure and domain functional models for explanation (Tanner and Keuneke, 1991). Other research efforts were directed at generating explanations for a particular class of tasks, for example, the work on explanation for routine design problem solving (Kassatly and Brown, 1987).

The generic task approach is well suited for producing explanations because it provides a theory that captures the relationship between the task and the problem solving strategy. The problem solving process is represented at the right level of abstraction for generating explanations of the decision-making and problem solving activity. Explaining knowledge elements and justifying problem solving knowledge can be done by developing functionally represented deep models of the problem solving process.

Although this previous work is not directly related to developing authoring systems for ITSs, it is relevant because of the approach adopted. This work emphasizes the importance of choosing an appropriate knowledge representation for the domain knowledge embedded in an expert system for generating explanations. From the perspective of developing an ITS authoring tool, this issue is important because it determines how effective the representation of the expert model component of an ITS will be for teaching, and ultimately, the quality of the tutoring systems produced by an authoring system. Thus, finding a good knowledge representation is the first step involved in developing a reusable system for generating tutoring systems.

3.4 Lessons Learned

This chapter has presented a variety of authoring environments, and discussed different methodologies that exist for ITS authoring systems. Each approach has some benefits and limitations. From the authoring tools described above and reported in the literature, conclusions can be drawn about the features that an “ideal” system should possess. A good ITS authoring environment should offer the following characteristics:

- Facilitate the development of pedagogically sound instructional and learning environments.
- Provide ease of use, so that authors can use it to develop tutoring systems with a minimal amount of training and computer programming skills.
- Support rapid prototyping of learning environments to allow authors to design, build, and test their systems quickly and efficiently.
- Allow the reuse of tutoring components to promote the development of flexible, cost-effective ITSs.

Thus, a good authoring system should fall towards the middle of the three-dimensional design space shown earlier in figure 4, providing enough usability and flexibility, yet powerful enough to produce “knowledgeable” or “intelligent” tutors. One solution that seems to fulfill this objective, and can achieve the goals specified above is the task-specific authoring approach. As discussed above, because they target a specific category of tasks, they are generally easier to use, offer a high level of reusability, and can produce effective instruction.

An important design issue for ITS authoring systems is the appropriate degree of generality. As systems become more general purpose, they are often more complex, and generally, more difficult to use. Systems that are too specific will generally not allow enough flexibility, and thus provide little reuse. In an analysis of the state of the art of ITS authoring tools, (Murray, 1999) identifies that an appropriate level of abstractness for ITS authoring tools may be at the knowledge level of generic tasks as proposed by (Chandrasekaran, 1986), rather than at a more surface level of task types.

The approach proposed in this dissertation falls under the task-specific authoring category. The overall goal is to develop an ITS authoring environment that can interact with any classification-based generic task expert system, and produce an intelligent tutoring system for the domain topic addressed by that system. The focus is on the class of classification-based generic tasks. An authoring environment for such expert systems can generate ITSs for a wide range of domains, provided the domain applications can be represented as generic tasks. This approach facilitates the reuse of tutoring components for various domains. The ITS authoring environment can be used in conjunction with any classification GT-based expert system, effectively allowing the same tutoring components to be plugged in with different domain knowledge bases. Another important aspect of this approach is that it allows the teaching of both factual knowledge and process knowledge, both of which are extracted from the expert system. The ITSs generated use these two knowledge types to teach domain knowledge as well as problem solving skills: the tutoring system presents

knowledge about the domain in the context of problem solving using the process followed by the expert system. Chapter 5 presents a more detailed description of the approach.

Chapter 4. Problem Definition

In this chapter, a concise definition is given of the problem addressed in this research work. The first section summarizes the motivation behind this work. Section 4.2 discusses some broad research issues related to the development of ITS shells. Next, the artificial intelligence objectives, including both the ITS and KBS perspectives, and education objectives of the work are outlined in sections 4.3, 4.4, and 4.5. Finally, the chapter concludes with a statement of the problem.

4.1 Motivation

The motivation for this work stems from the need for easier and more cost-effective means of producing intelligent tutoring systems. This need is driven by the current demand for a wide array of educationally well-founded and useful instructional applications, in both academic and industrial settings. Research results have indicated that individualized instruction is more effective than traditional learning, such as classroom-type instruction, because both the content and style of the instruction can be continuously adapted to best meet the needs of the learner. Early computer-based training (CBT) and computer-assisted instruction (CAI) attempts to develop systems that incorporated individualized instruction fell short because they lacked the flexibility and learner-centered orientation that add a dynamic and adaptive dimension to self-paced instruction. The notion of intelligent tutoring systems evolved to deal with these shortcomings. Recently, numerous ITSs were proven to be highly effective as

learning aides (Shute and Psotka, 1996), and thus ITSs started to receive widespread attention.

Although the demand for ITSs continues to grow, the number of ITSs developed has not significantly increased. This is largely due to the numerous and challenging complexities involved in authoring ITSs. Each ITS must still be built from scratch, usually at a significant cost in time and money. Very few authoring aides for ITSs exist. Some authoring systems are commercially available for building traditional CAI and CBT systems, but these tools lack the sophistication required to build “intelligent” tutors. Thus, there is a need for research and development efforts aimed at alleviating the ITS supply and demand problem, through the construction of ITS authoring tools.

This research work is motivated by the need for more cost-effective strategies for developing ITSs and from the leverage that the generic task development methodology offers in solving this problem. The focus is on the issue of reusability: knowledge reusability and tutoring components reusability, to be more precise. Generic task-based expert systems all share a strong structure that includes a semantically meaningful knowledge representation method as well as a structured inferencing strategy. Their knowledge-rich structure can be reused for instructional purposes, allowing the tutoring of domain knowledge embedded within the expert system and problem solving skills utilized by the expert system in solving the problem. By integrating this reusable knowledge with other reusable ITS components, a powerful authoring environment is created for the generation of tutoring systems for various domains. In effect, such an

authoring environment can be linked to any generic task-based expert system, allowing the same tutoring components to be plugged in with different knowledge bases.

4.2 General Issues Related to ITS Authoring Environments

Before describing the specific objectives of this research, it is useful to consider some general issues related to the context of ITS authoring research and development. These questions help to identify the broad scope of ITS authoring environments. The main issues are:

- How broadly applicable should an ITS authoring tool be?

This question deals with the suitability of general-purpose tools, which have wider applicability, compared to task-specific systems, which are tailored to specific curricular or domain needs. The tradeoffs revolve around the usability and flexibility of more general tools, versus the power that is derived from knowledge-rich tools based on explicit and detailed models. General-purpose ITS authoring environments aim to support the development of a broad range of applications, relying on weak, but general-purpose, models of instruction and of the learner's task. In contrast, specialized tools target a narrow range of ITS applications but can offer more powerful support for authors.

- How usable should an ITS authoring environment be?

Usability involves two aspects: learnability and productivity (Murray, 1996). Learnability is how easy an authoring system is to learn how to use. Productivity is how quickly and effectively an author can use the environment to develop an ITS. Learnability and productivity are often at odds because a

system that is designed to be learned easily by novices may not provide the powerful features that experienced users need to efficiently produce tutoring systems. The question is, 'Where along the usability spectrum should an authoring system fall?'

- **Who is the user?**

What is the nature of the intended user, i.e., the author? Earlier, an author was defined as a person interested in developing an ITS for a particular topic, such as a teacher, trainer, instructional designer, or domain expert. Should a system be designed for use by a specific set of users, for example, teachers, or should it target a wider, more general audience? Clearly, any author will need to have expertise in the target domain. However, depending on the specific class of users the system was designed for, authors may also need skills in programming, instructional design, and knowledge engineering. It may be necessary that the author is actually a team of users, each having some expertise or skill. Another important question is whether users, other than those mentioned above, should be considered potential authors of tutoring systems. For example, should the user scope be broadened to include editors, publishers, or even educational theorists? The answer to this question will require determining if the needs of such users can be adequately fulfilled by current ITS authoring methodologies.

- **What constitutes authoring?**

Should authoring be viewed as a large-scale composition task, with the objective of developing a complete ITS authoring environment? Alternatively,

should the focus be on specific software components such as tutoring components or domain content? In addition, the functional expectations for each authored component must be determined. For example, for the domain content, will the environment allow the author to develop the definition and representation of the domain knowledge? Will an authored pedagogical module facilitate the manipulation of instructional strategies and control? Can the authoring environment provide an author with the flexibility of choosing the student modeling primitives or techniques that he or she prefers? Moreover, for the communication module, how much flexibility in choosing the appropriate interface components and presentation aspects will the authoring tool allow? These and other questions must be answered to determine the output requirements of the authoring process.

- How can the development of an ITS authoring environment help promote its widespread applicability?

The impediments to widespread use of authoring systems must be identified, along with ways to overcome them. The greatest obstacle is the lack of standards. Standards for ITS authoring environments are needed to: (a) develop a common, shared ontology for ITS authoring, (b) facilitate the development of ITSs from reusable components, and (c) identify appropriate instructional strategies or target domains. Another question is whether knowledge sharing can be achieved using a knowledge communication language such as KIF, the Knowledge Interchange Format language. Overall, standards can increase the degree of interoperability of authoring tools, which

can lead to more widespread usage. Collaborative efforts among the ITS community are needed to develop and agree on a set of common standards for ITS authoring.

- How should an ITS authoring environment be evaluated?

A central question concerns how to judge the success of an authoring environment. What criteria should be used to evaluate ITS authoring tools? For example, some ITS authoring environments are judged by the learning effectiveness of the resulting ITSs. However, this approach may not work well since it depends on how an author will use the environment to produce a tutoring system. A skilled author might use it to produce an ITS with a high measure of learning effectiveness, while a novice author might produce an ineffective tutor. Thus, the learning effectiveness measure depends on the author, not really on the authoring environment. In that case, maybe the evaluation criterion should be the usability of the authoring system; environments could be judged by how easy to learn and use they are. Some people have even argued that the success of an ITS authoring system could be judged by its sales record and widespread usage. Much research work is still needed to identify appropriate methods of evaluating ITS authoring systems.

Before describing our approach to ITS authoring in more detail, the specific objectives of this research are first discussed. These objectives are presented as the goals that will be addressed, from the knowledge-based systems, intelligent tutoring systems, and education perspectives.

4.3 Objectives within the KBS Field

The main objective of this work within the knowledge-based systems research and development field is to:

- Extend the task-specific framework to support tutoring through the reuse of the knowledge embedded within generic task-based expert systems

The primary focus of this research work is to develop an architecture that integrates an ITS shell with a task-specific architecture for generating tutoring systems. The ITS shell uses problem solving knowledge about a specific task structure and domain knowledge embedded within existing expert systems. This eliminates the need for authoring the domain knowledge component of a tutor. This also implies that the ITS's tutoring capabilities are directly dependent on the underlying problem solving capability of the expert system.

The Generic Task approach has several appealing implications for ITS generation. First, the large-grain approach of the Generic Task methodology allows the reuse of high-level problem solving concepts and strategies for tutoring support; it allows the tutoring focus to be on problem solving *tasks*, rather than on lower level knowledge constructs used by small-grain approaches. This enables a task-specific tutoring approach, as described in section 3.2.9. Second, the underlying knowledge representation and reasoning method of a GT system provides a semantically rich foundation for the extraction of tutoring knowledge. A generic task expert system utilizes a task-based knowledge model of the domain, in which the problem is successively decomposed into smaller problems. The system's inferencing strategy allows reasoning about the problem in terms of its

sub-problems, each within its own context. The ITS shell can make use of the GT system's knowledge representation and inferencing method to support the tutoring process. Third, the ITS shell can be specialized for task-specific tutoring so that the other ITS components are reusable, without requiring any modification, for different task-specific problem solving domains.

4.4 Objectives within the ITS Field

This dissertation also has several objectives in the intelligent tutoring systems community. The ITS objectives addressed by this work include:

- **Knowledge representation and reuse**

The issue of reusing knowledge embedded within an expert system is an important issue that needs to be addressed. The knowledge representation selected determines the usefulness of the target application. For the approach of ITS authoring presented here, the tutoring knowledge will only be as good as the knowledge representation used by the expert system. Thus, the goal here is twofold: in general, to find an appropriate way of representing and utilizing the domain knowledge of an expert system, and, in particular, to determine how that knowledge can be effectively reused for tutoring.

- **Teaching factual domain and problem solving knowledge**

Another objective of this research work is to provide a learning environment that allows the teaching of both factual knowledge and process knowledge. Both types of knowledge can be extracted from the expert system, and accordingly reused for tutoring. The goal for the ITSs generated would be to use these two knowledge types to teach factual domain knowledge as well as

problem solving skills. Since our approach involves generating the tutoring content from the expert system; this is significantly different from other approaches that focus on authoring the tutoring content. Generating the content from the underlying expert system makes process knowledge available for tutoring. The expert system's domain knowledge can be used to generate factual knowledge for tutoring, and the expert system's control knowledge can be used to generate process knowledge for tutoring problem solving skills. The tutoring systems would attempt to present knowledge about the domain in the context of problem solving, by utilizing and presenting the process followed by the expert system.

- **Reuse of tutoring components**

The key component of the ITS shell that allows reuse is the expert model, which interfaces to a GT-based expert system to extract tutoring knowledge. However, to be a complete ITS authoring environment, this model must work in conjunction with a tutoring module and student model to interact with the learner. To achieve maximum flexibility, the other ITS components, namely the student model, pedagogical module, and communication module, must be reusable. The adopted task-specific approach allows these components to be reused as-is for different classification-type problem solving domains. In addition, these components should be flexible enough to suit the specific needs of different learning situations.

- **Support for rapid prototyping of tutoring systems**

The ITS authoring environment should support rapid prototyping, allowing authors to quickly design, implement, and test a tutoring system. Rapid prototyping can provide many benefits by allowing the authoring environment to be used as a design and evaluation tool for developing learning systems. This approach will allow authors to quickly see the fruits of their labor, and identify errors, inconsistencies, or omissions in the design that can be corrected in the next iteration, rather than having to wait until the entire system is built to discover such bugs.

- **Explicit knowledge modeling**

There is a need to make the tutoring knowledge used by ITS authoring environments explicit. The solution proposed is guided by a strong ontological commitment - to the GT framework overall, and to the vocabulary of problem solving in each individual GT. One question that must be answered is whether or not any additional domain knowledge will be needed to produce an effective ITS other than that which is already represented in an instance of a GT-based expert system. The modeling of the domain knowledge required for tutoring, including pedagogical and performance knowledge, is important in the generation of effective tutors.

- **Separation of instructional content from instructional strategies**

The knowledge to be tutored should be separated from the instructional strategies that will be used to teach it. The instructional content should be modular and thus reusable for different purposes. This would allow the same content to be used in different parts of the tutorial. For example, a modular

unit of tutoring knowledge can be used as part of an overview of the tutoring system's topic, and also as an example later on. Moreover, by separating instructional content from instructional strategies, the ITS authoring system can provide multiple and abstracted views of the tutoring content. This would allow examples to be presented at varying levels of detail or for different purposes.

4.5 Educational Objectives

This research also has several educational objectives. The specific issues addressed are:

- **Generation of pedagogically sound learning environments**

An ITS authoring system should facilitate the development of pedagogically sound instructional and learning environments. Not all instructional techniques work for all types of learning. Indeed, many studies have been conducted to determine the most appropriate types of instruction for various learning activities (Bloom, 1956; Gagne, 1985; Joyce and Weil, 1986; Kyllonen and Shute, 1989). The main educational objective of this work is to generate tutoring systems that are pedagogically sound, by utilizing instructional strategies that are appropriate for task-specific tutoring of problem solving domains. Moreover, the authoring system should provide guidance for the authoring process itself. For example, the system can help the author specify learning goals, or reduce the author's effort in developing an ITS by performing some of the authoring steps itself.

- **Development of an easy to use ITS authoring environment**

A good authoring environment must be easy to use, so that authors can use it to develop tutoring systems with a minimal amount of training and computer programming skills. Most users of ITS authoring environments will be: (a) educational specialists, such as teachers, trainers, and instructional designers; or (b) domain experts, i.e., people who have expertise in a certain domain or topic. These authors may not have the computer or programming skills to drive a complex ITS authoring environment. Thus, the authoring environment should utilize a user interface and/or development environment that authors would feel comfortable using.

- **Development of a learner-centered approach**

The authoring environment's instructional strategies should employ a learner-centered approach to education. This implies that the tutoring systems generated should allow students to navigate to the topics they want to learn and to ask for hints, examples, etc. The modularization of instructional content and instructional strategies facilitates such a learner-centered approach.

- **Use as an evaluation tool**

Within the context of educational software, there is a need for coping with how computer-based learning, given the rapid advancement in technology, changes the way in which learning and education is conducted. This requires more effective means of evaluating learning environments. Typically, learning is evaluated by field testing with real students, an approach which can be time consuming and inaccurate. A more efficient method would be to develop a

model of the learning environment, and use that model to simulate different learning situations. For example, results of simulating the learning environment for different instructional strategies could lead to an understanding of which instructional technique is more suited for a certain learning task, or a certain type of student. As another example, varying the learner knowledge state, while keeping the rest of the environment constant, can provide insight on novice-expert shifts in learning tasks. The ITS authoring shell could serve as a component of a larger environment for the modeling and evaluation of different learning conditions. Such an environment could be envisioned as a virtual laboratory for experimentation with new learning approaches technologies.

4.6 Problem Statement

The sections above discuss the general research objectives that this dissertation work will try to address. The main problem addressed is the need for reusable ITS shells and components. A solution to this problem lies in the generation of tutoring systems for existing expert systems and knowledge bases. The specific goals of this dissertation are to:

- Develop a framework for a tutoring extension to the GT approach. More specifically, to formulate a technique for leveraging the knowledge representation and structure of the Generic Task framework for tutoring.
- Develop an ITS authoring shell for classification-based GT expert systems. The purpose of this shell is to generate intelligent tutoring systems for various

domains by interfacing with existing expert systems, and reusing the other tutoring components.

- Generate intelligent tutoring systems for two demonstration domains using the ITS shell developed.
- Analyze the ITS authoring shell and tutoring systems generated, with respect to the research objectives.

Chapter 5. An Architecture for Generating Intelligent Tutoring Systems

In this chapter, we describe the development of a methodology and architecture for generating intelligent tutoring systems for different domains by interfacing with existing expert systems, and reusing the other tutoring components. The technique focuses on leveraging the knowledge representation and structure of the generic task framework for tutoring, and thus serves as a framework for an ITS extension to the GT approach. Next, we describe the architecture developed, and how it addresses the issue of reusability.

5.1 Extending the Generic Task Framework for Tutoring Support

The GT framework is extended by developing an ITS architecture that can interact with any GT-type problem solver to produce a tutoring system for the domain addressed by the problem solver. The learner interacts with both the tutoring system shell (to receive instruction, feedback, and guidance), and the expert system (to solve problems and look at examples), in an integrated environment as shown in figure 5.

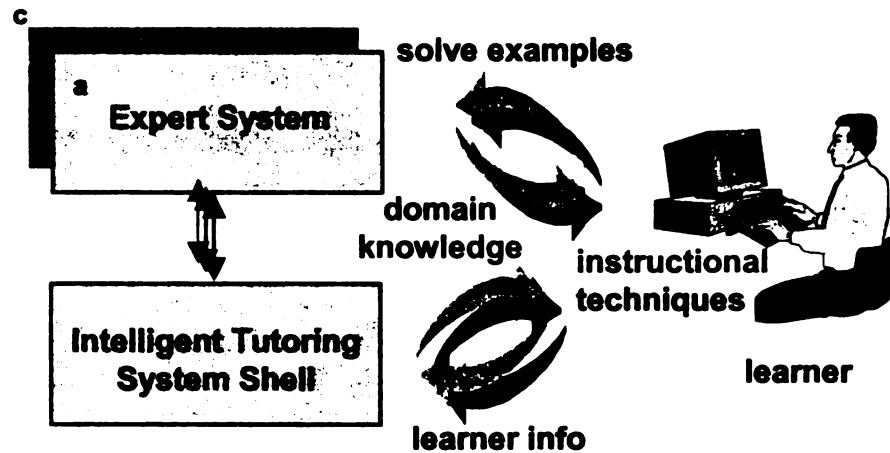


Figure 5. System-user interaction model

The architecture for a tutoring extension to the generic task framework is shown in figure 6. The architecture consists of three main components: (1) a GT expert system, (2) a component for extended domain knowledge for tutoring, and (3) an ITS shell. The GT expert system is used by the tutoring shell and the learner to derive problem solving knowledge and solve examples. The extended domain knowledge component stores knowledge about the domain that is necessary for tutoring, but not available from the expert system, such as pedagogical knowledge. The ITS shell has four main components: the expert model, student model, instructional manager, and user interface.

This discussion focuses on how the architecture achieves reusability and how the ITS shell interacts with the expert system. The ITS shell and extended domain knowledge component will be explained in more detail in the next section.

The expert model component of the ITS shell understands the structure of a GT expert system and extracts knowledge from it. Rather than re-implement

the expert model for each domain, the ITS shell interfaces with a GT system, through the knowledge link depicted in figure 6, to extract the necessary knowledge for each domain. This facilitates the reuse of the instructional manager, student model, and user interface components for different domains.

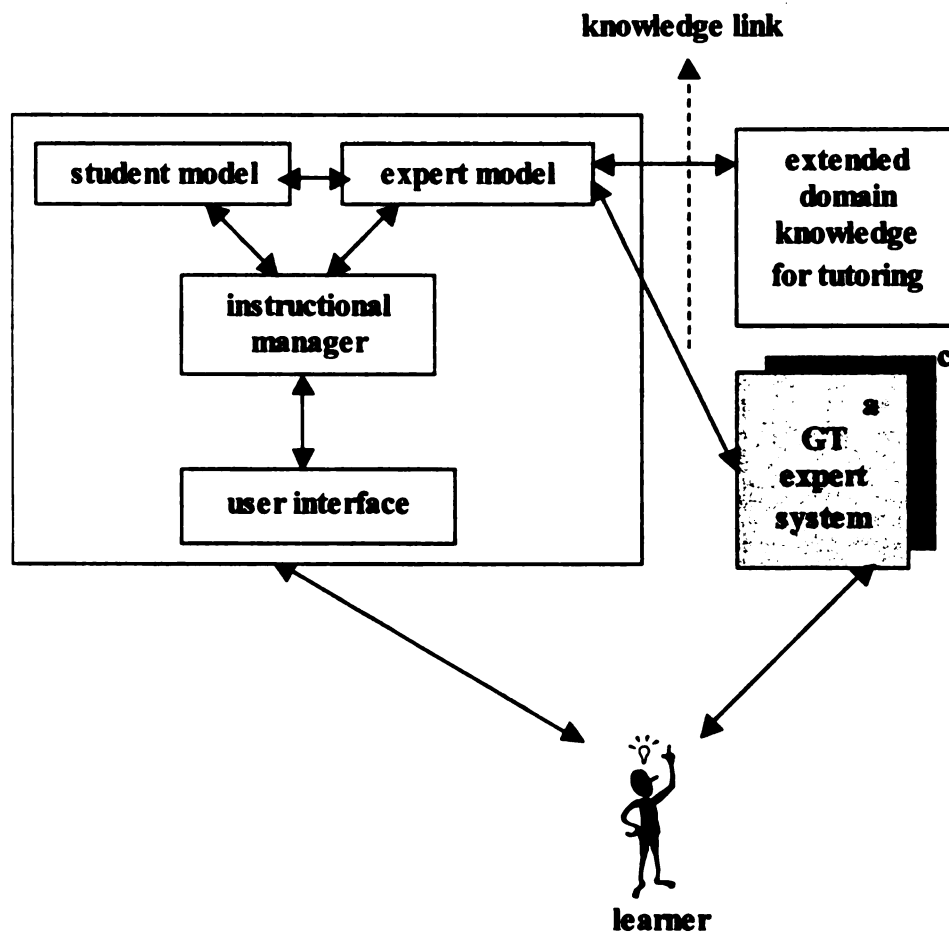


Figure 6. Architecture for generating ITSs from GT-based expert systems

Linking the ITS's expert model to the problem solver deserves special consideration. Rather than encode domain knowledge explicitly, the expert model extracts and utilizes of the domain knowledge available in the expert system. Thus, the quality of the tutoring knowledge is affected by the knowledge representation used by the expert system. The GT methodology's strong

commitment to both a semantically meaningful knowledge representation method, and a structured inferencing strategy, allows the extraction of well-defined tutoring knowledge. The expert model extracts three types of knowledge:

- **Decision-making knowledge**

This describes how the data relates to the knowledge. This knowledge characterizes how the domain knowledge is represented within the expert system. The expert model can extract the problem solving structure and relationships among the agents of the expert system, which it uses to produce tutoring content about the expert system's decision-making process.

- **Knowledge of the elements in the domain data base**

This includes knowledge of the data base types. An expert system employs user-defined variables in the problem solving process. These data variables can be different types, such as numerical, text, logic, etc. The expert model has knowledge of the variables used by the expert system in problem solving, including each variable's name, type, description, legal values, and current value.

- **Knowledge of the problem solving strategy and control behavior**

This knowledge encodes the task structure of a generic task expert system. This knowledge is what allows the expert model to "understand" the problem solving process of the expert system. This knowledge is the same for all expert systems that are instances of the same generic task type. For example, for an HC-based expert system, this knowledge would consist of the

“establish-refine” inferencing strategy. The expert model uses this knowledge to trace through the problem solving process, as required during tutoring.

Generic task expert systems have well defined knowledge structures and reasoning processes that can be reused for tutoring support. A typical GT system is composed of agents, each of which has a specific goal, purpose, and plan of action. The expert system solves problems using a case-by-case approach. Individual cases can be extracted from the expert system, either to present as examples or problems for the learner to solve. The expert model of the ITS shell can extract the following types of knowledge about a case for tutoring:

- **Case name and description**

This knowledge is used for presenting examples to the learner.

- **Input variables and values**

This knowledge is used for presenting examples and also for posing questions to the learner.

- **Explanation of the output generated**

This knowledge includes a description of the agents along the output path for the given input. It is used for presenting examples and also for posing questions to the learner.

The expert model of the tutoring shell uses this knowledge, along with an encoding of the expert system’s structure, to formulate tutoring knowledge as required by the ITS.

5.2 Completing the Architecture

What has been described so far is the key component of the ITS shell that allows reusability, namely the expert model, and how it interfaces to a GT-based expert system. To complete the ITS shell, this expert model must work in conjunction with the student model, instructional manager, and user interface to interact with the learner. In addition, the architecture includes an extended domain knowledge component that is used for tutoring support. These components are described in the next two sections.

5.2.1 The Reusable Tutoring Components of the ITS Shell

The ITS shell has four main components: the expert model, student model, instructional manager, and user interface. The expert model component and how it interacts with a GT expert system was explained in section 5.1.

5.2.1.1 Student Model

Student modeling is often viewed as the key to individualized knowledge-based instruction (Greer and McCalla, 1994). Within the context of the ITS, the student model plays an important role, in that it allows the tutoring system to adapt to the needs of individual learners. The architecture adopts an approach for student modeling that simplifies the student modeling process.

The student model is developed as a task model of the expert system. The student model uses the expert system to make decisions about how an expert would solve problems in the domain. The student model compares the performance of the student during problem solving to the expert system's solution to make inferences about how the learner is learning. An overlay model

is used to assign performance scores to the problems that the learner solves. Each question that the learner answers is assigned a score based on how many hints and/or attempts the learner needed, and on whether or not the learner was able to determine the correct answer. Each topic has an overall score that is computed as the average score of all the questions covered on that topic. The instructional manager uses this information provided by the student model to direct the instruction. For example, if the learner's overall topic score is low, the tutor presents more examples. If the score is high, the tutor can ask the learner to solve more questions, or move on to the next topic.

Figure 7 shows the Information Processing Task (IPT) Model for the student modeling component. The student model uses information provided from the expert system, instructional manager, and extended knowledge component to keep an accurate model of the learner's knowledge level and capabilities, and also to guide the instructional strategy.

This approach has important benefits. It can provide explanations of the learner's behavior, knowledge, and errors, as well as explanations of the reasoning process. In addition, using a runnable and deep model of expertise allows fine-grained student diagnosis and modeling. As a result, the tutor can give learners very specific feedback and hints that explain the problem solving process when their behavior diverges from that of the expert system's. Moreover, if the learner is having difficulties, he or she can ask the tutor to perform the next step or even to solve the problem completely.

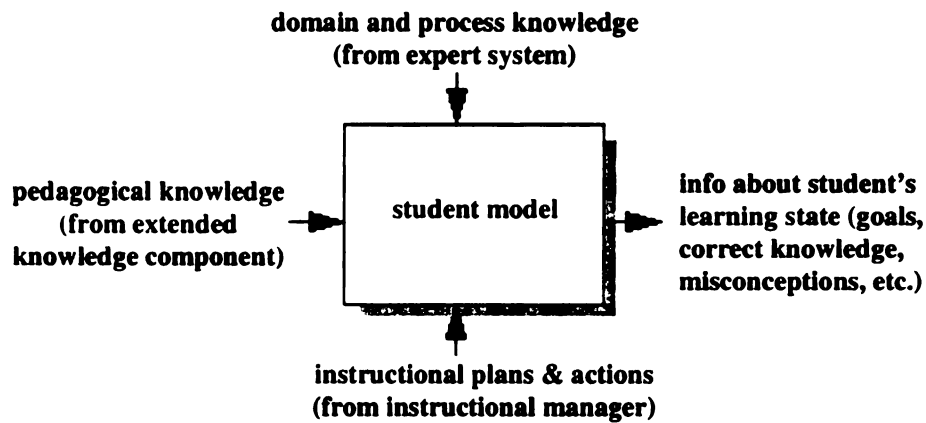


Figure 7. IPT model for the student model

5.2.1.2 Instructional Manager

The ITS architecture incorporates a pedagogical approach that is appropriate for tutoring domains using problem solving. The instructional manager uses two main instructional strategies, learning by doing and example-based teaching. These teaching strategies are well suited for teaching complex, knowledge-intensive domains that focus on learning knowledge and problem solving skills, such as engineering domains. Moreover, such strategies are a good match for this framework, since the learner can interact with both the expert system to solve problems and the tutoring system. Learning by doing is implemented within the architecture by having the learner solve real problems using the expert system, with the tutor watching over as a guide. In the other learning mode, the tutor makes use of the knowledge base of the expert system, in which the input-output sets are stored as individual cases. The instructional manager can present prototypical cases as examples to the user, which serve as a basis for learning from new situations. Additionally, it can present new examples, posed as questions, and ask the learner to solve them. The goal is to help the user

develop knowledge of how to solve problems in the domain, by looking at and solving examples.

The instructional manager uses an instructional plan that incorporates a cognitive apprenticeship approach; learners move from looking at examples to solving problems as their competence level of the domain increases. The curriculum includes:

- Presentation of an overview on the domain topic. This gives the learner an introduction to problem solving in the domain, including a description of the input variables and output alternatives of the problem solving process.
- Presentation of problem solving examples on each topic of the domain.
- Asking the learner to solve problems covering the domain topics.

The curriculum gives the author flexibility in determining the content of the examples and questions presented to the user. The author selects these from the set of cases defined in the expert system. The author can also determine the number of questions to ask the user for each topic. The pedagogical strategy also includes techniques for giving the learner feedback and hints. When the learner solves a problem during a tutoring session, the tutor gives the learner appropriate feedback according to whether the learner solved it correctly or not. If the learner's answer is incorrect, the tutor gives the learner a hint specific to the mistake committed, and then re-asks the question. The pedagogical strategy supports giving the learner up to 3 levels of hints and attempts to solve a problem, after which the tutor presents the correct answer if the learner still did not answer the question correctly.

5.2.1.3 User Interface

The architecture employs a simple user interface that has been designed for tutoring using a problem solving approach. Since the tutoring content is mainly generated from the expert system, the user interface is tailored to allow the learner to interact with both the ITS shell and expert system in a transparent manner. The domain knowledge presented through the user interface is generated from the underlying expert system. The basic layout of the main window of the user interface is shown in figure 8 and the actual interface is shown in chapter 6. The main bar on the top is for the menus and title of the tutor. The sub-window on the left will show the curriculum and current topic. The sub-window on the right will display the domain content to be tutored. This content comes in part from the expert system and in part from the extended domain knowledge component, which is explained next.

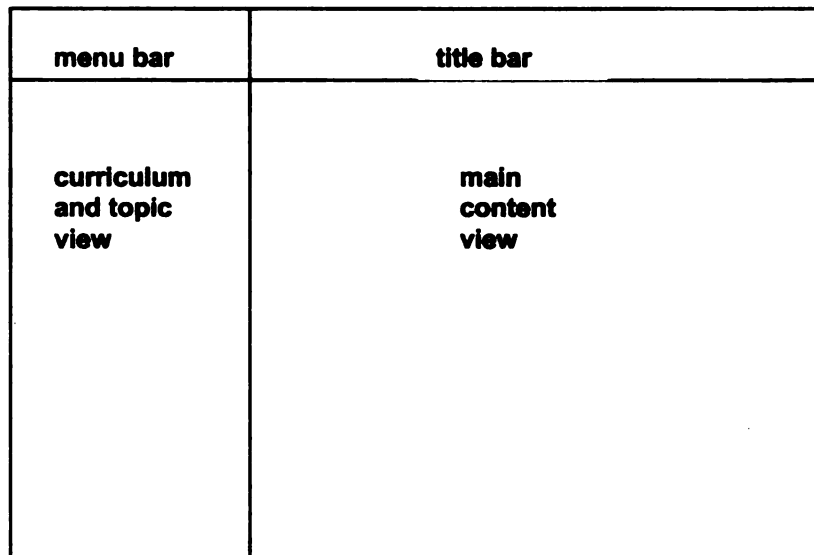


Figure 8. Basic layout of the user interface

5.2.2 The Extended Domain Knowledge Component

For the ITS architecture to produce effective tutoring, it needs the appropriate domain knowledge and pedagogical knowledge about the domain. The expert system contains most of the domain knowledge needed for tutoring. However, since the main purpose of the expert system is to solve problems, it lacks some knowledge required for tutoring. To complete the ITS architecture, this additional knowledge is stored in the extended domain knowledge component. This knowledge includes:

- A high-level description of the problem solving domain.

The expert system is designed to solve problems in the domain, and the ITS shell extracts knowledge about the domain and the problem solving process for tutoring. The extended knowledge component stores a high-level overview about the problem or domain, which cannot be extracted from the expert system. This knowledge is used by the ITS shell in presenting an introduction on the domain to the user.

- The list of examples used in the curriculum.
- The list of questions used in the curriculum.

These two types of knowledge are pedagogical knowledge related to the domain. They are not available from the expert system, which only contains problem solving knowledge about the domain. The ITS author specifies what topics to be included in the tutoring curriculum by defining the list of examples and questions in the extended knowledge component.

5.3 . Specializing the ITS Architecture for Hierarchical Classification

This section describes how the tutoring architecture explained in section 5.1 can be specialized for hierarchical classification-based (HC) GT expert systems, which were described in section 2.3.4.2. Figure 9 depicts how the ITS architecture would be specialized for HC-based expert systems. To facilitate the specialization, certain components of the architecture require modification. First, the expert system would be a specific instance of an HC-based expert system. Also, the ITS shell needs specific knowledge about the knowledge representation and problem solving behavior of the hierarchical classification system. This implies that the expert model must have the knowledge necessary to understand and interact with an HC-based system. The instructional manager, student model, and user interface must be able to use this knowledge as required for tutoring. Note that some components of the architecture are task-specific, HC-specific in this case, while some are task-free. Note also that some components are domain-specific while others are domain-free. Specifically:

- The expert system component is task-specific and domain-specific. It is used for problem solving using a task-specific approach about a specific domain.
- The ITS shell component is task-specific but domain-free. This feature is what makes the shell reusable for different domains.
- The extended knowledge component is task-free but domain specific. This component stores information such as a high-level textual description of the domain, and the list of examples and questions to use in the curriculum. This

knowledge is specific to the domain but does not include any task-specific knowledge.

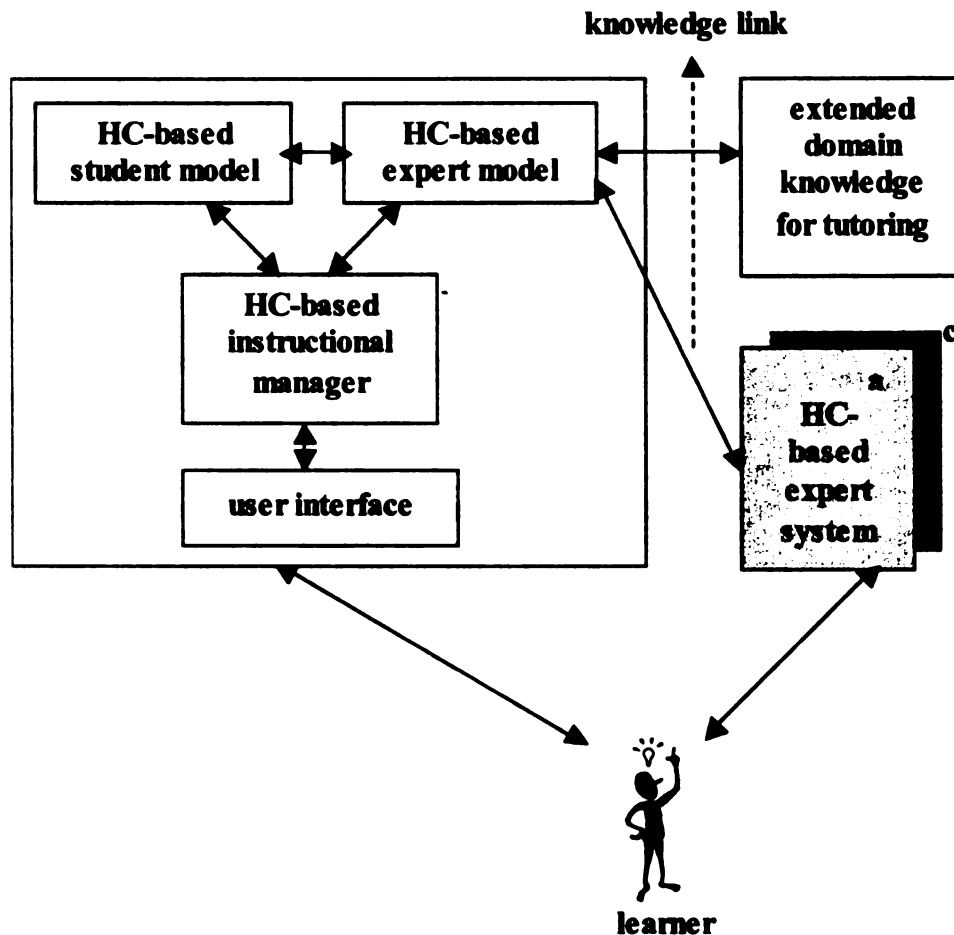


Figure 9. ITS architecture specialized for HC-based systems

Next, the knowledge that the expert model needs about the expert system for tutoring purposes is described. This description uses COFATE2 (Moy, McDowell et al., 1994), a composite material selector system, as an example of an HC-based expert system. The domain expert component of the ITS shell will need the following knowledge about an HC problem solver:

1. Problem solving knowledge
 - Knowledge embedded within each HC agent type

- Reasoning strategy
 - Problem solving state
2. Domain knowledge
- Classification structure
 - Input information
 - Output information

5.3.1 Problem Solving Knowledge

The ITS shell needs knowledge about the problem solving task structure used by the expert system. This knowledge exists within the ITS shell and enables it to interact with any expert system that shares the same HC task structure. The shell contains three components of problem solving knowledge: knowledge embedded within each HC agent, reasoning strategy, and problem solving state.

5.3.1.1 Knowledge Embedded Within Each Agent

An HC system has one main agent type, namely specialists. The ITS shell stores knowledge templates for each agent type that represent the knowledge “known” by each agent. These templates utilize specific knowledge embedded within a problem solver to be completely useable. This specific problem solving knowledge is shown in brackets (<...>) in the description below.

Specialist Knowledge:

In the context of <purpose of specialist>, this specialist will attempt to establish itself by the knowledge available in the <name of table matcher> table.

If established, the specialist will invoke its sub-specialists: <name(s) of sub-specialists>.

Example:

In the context of *establishing the hypothesis that the tooling cost is high*, this specialist will attempt to establish itself by the knowledge available in the *high tooling cost* table.

If established, the specialist will invoke its sub-specialists: *compression molding, pultrusion, and transfer molding*.

Specialist purpose: to establish the hypothesis that <specialist's attribute>

Example: to establish the hypothesis that *the tooling cost is high*

5.3.1.2 Reasoning Strategy

The reasoning strategy of an HC expert system can be represented as a sequence of inferencing actions. Each action maps to a specific problem solving goal. Goals are decomposed into sub-goals as the inferencing strategy progresses. In the description below, inferencing actions are shown in CAPS and specific problem solver knowledge in brackets (<...>). The ITS shell stores the reasoning strategy of an HC problem solver as follows:

1. INVOKE <top-level specialist>
2. <specialist> attempts to ESTABLISH <self> based on table matcher knowledge
3. If <specialist> is established, then <specialist> sends ESTABLISH-REFINE message to each <sub-specialist> (if any)

- For each <sub-specialist>, perform steps 2-3.

4. TERMINATE problem solving when no further leaf-level <specialists> remain to be invoked¹.

5.3.1.3 Problem Solving State

The expert model of the ITS shell can extract a snapshot of the problem solving state to keep track of what state the problem solver is in any given time. The problem solving state is comprised of both knowledge about the task structure and knowledge specific to the expert system. The problem solving state knowledge consists of:

- **Partial classification result**

This information includes the classification specifications satisfied so far. This is stored as the output parameters (described below) with any values determined at that point. The output parameters consist of the leaf level specialists, along with their establish-refine results.

- **Classification pathway**

The classification pathway represents the reasoning path that has been traversed by the expert system at any given time. The collection of all specialists along the reasoning path represents the emerging classification pathway. The reasoning path is generated by executing the reasoning strategy on the classification structure of the problem solver.

- **Current goal(s)**

Any goals or sub-goals that are considered active at that point are stored in the problem solving state. The active goals are determined as the reasoning path is generated.

5.3.2 Domain Knowledge

The ITS shell also needs domain knowledge about the problem the expert system is trying to solve. This knowledge exists within the expert system and needs to be transferred to the ITS shell to enable tutoring. The problem solver contains three components of domain knowledge: classification structure, input information, and output information.

5.3.2.1 Classification Structure

The ITS shell needs to know the structure of the classification problem. This consists of the hierarchical decomposition of the problem into specialists with goals, and sub-specialists with sub-goals. The structure of the COFATE2 problem solver is:

- <1> cofate 1
 - <2> High Throughput 2
 - <3> High Tooling Cost 3
 - <4> Compression Molding 4
 - <4> Pultrusion 5
 - <4> Transfer Molding 6
 - <3> Low Tooling Cost 7
 - <4> Blow Molding 8
 - <3> Medium Tooling Cost 9

¹ Not all implementations of HC operate this way. Some terminate when the first leaf-level

—————	<4> Extrusion	10
—————	<4> Filament Winding	11
—————	<4> Injection Molding	12
———	<2> Low Throughput	13
—————	<3> High Geometric Complexity	14
—————	<4> Hand Layup	15
—————	<4> Hand Sprayup	16
—————	<4> Prepreg Autoclave	17
—————	<3> Low Geometric Complexity	18
—————	<4> Autotape Autoclave	19
—————	<3> Medium Geometric Complexity	20
—————	<4> Pressure Molding	21
—————	<4> Vacuum Molding	22
———	<2> Medium Throughput	23
—————	<3> High Performance Properties	24
—————	<4> Resin Transfer Molding	25
—————	<3> Low Performance Properties	26
—————	<4> Reaction Injection Molding	27
—————	<3> Medium Performance Properties	28
—————	<4> Structural Reaction Injection Molding	29

In other words, the ITS shell needs an ordering of the problem decomposition into agents. The information about each agent should include its type, super-agents (if any), and sub-agents (if any).

5.3.2.2 Input Information

The ITS shell needs information about the input requirements of the classification problem. This includes all the input variables and their values. The problem

solver stores one set of input values for each case that it can solve. This domain information is transferred to the ITS shell as needed. For each case, the ITS shell needs to know the input variables that have been defined, their types (for example, string or yes/no variable), their values, and any additional information needed (for example, upper and lower limits for numerical variables, or possible values for one-of variables).

5.3.2.3 *Output Information*

The output information required by the ITS shell includes the final classification result and pathway for each case defined. These knowledge elements represent the final classification result after the problem solving has been completed.

5.3.3 Knowledge Storage and Transfer

The ITS shell needs general knowledge about the HC task structure, as well as specific knowledge about a particular problem solver. General problem solving knowledge is stored in the ITS shell itself, within the expert model. As described above, problem solving knowledge is of three types: agent knowledge, reasoning strategy, and problem solving state. The templates used to store the knowledge embedded within each agent type are fixed, and is stored as is by the shell. Specific knowledge is transferred to the templates as needed by the tutoring system. The reasoning strategy is entirely contained within the shell. As for the problem solving state, this information will change often based on the current goals, and so is transferred to the ITS shell as needed. The ITS shell uses a query-based transfer mechanism for the template knowledge and the problem

solving state, whereby the ITS shell queries the expert system for such knowledge as needed during a tutoring session.

The domain-related problem solving knowledge is mostly contained in the expert system, and, thus, needs to be transferred to the ITS shell. Domain knowledge is comprised of three components: classification structure, input information, and output information. The classification hierarchy of a problem solver is fixed, and can be transferred to the shell all at once. The input and output information are different for each case represented within the expert system. This information is transferred to the ITS shell on a case-by-case basis. The ITS shell loads a complete case all at once for use in tutoring, and loads different cases as needed, depending on the instructional objective.

5.4 Comparison of the Generic Task Approach for ITS Generation to the KADS Approach

The architecture developed and described in this chapter follows the Generic Task approach for knowledge-based systems development. It integrates an ITS shell with any generic task-based expert system, to produce a tutoring system for the domain knowledge represented in that system. Another task-specific approach is the Knowledge Acquisition and Documentation System (KADS), which was described in section 2.2.1. The focus of this section is to compare ITS generation using the GT approach with the KADS approach, conjecturing about how the KADS approach could be used to support tutoring. We analyze ITS generation using the GT approach with the conjectured KADS approach for the purpose of analyzing their similarities, differences, strengths, and weaknesses.

5.4.1 Description of the KADS Approach

KADS is a methodology for knowledge-based systems development that originated in Europe (Wielinga and Breuker, 1986). Originally, the KADS project was initiated to provide support for analyzing and describing problem solving behavior in terms of generic inferences. Over time, the KADS view has evolved into a more comprehensive framework, KADS-2 (Wielinga, Schreiber et al., 1992), which has many layers spanning analysis through strategies to implementation, and the acronym also came to stand for Knowledge Analysis and Design Support. The KADS methodology includes a life cycle model for task decomposition of the problem, a set of modeling languages and frameworks for describing the objects in the methodology, and a set of tools and techniques for data analysis, model construction, and knowledge representation.

In KADS, the development of a knowledge-based system is viewed as a modeling activity. The five basic principles underlying the KADS approach are: (1) the introduction of partial models as a means to cope with the complexity of the knowledge engineering process, (2) the KADS four-layer framework for modeling the required expertise, (3) the reusability of generic model components as templates supporting top-down knowledge acquisition, (4) the process of differentiating simple models into more complex ones, and (5) the importance of structure preserving transformation of models of expertise into design and implementation.

The most well known ingredient of KADS is the four-layer model: a conceptual framework for describing the problem-solving expertise in a particular

domain using several layers of abstraction. A major assumption is that generic parts of such a model of expertise can potentially be reused as templates in similar domains – these are the interpretation models. Another important aspect of KADS is that it distinguishes between a conceptual model of expertise independent of a particular implementation, and a design model specifying how an expertise model is operationalized with particular computational and representational techniques. This gives the knowledge engineer flexibility in specifying the required problem solving expertise, but also creates the additional task of building a system that correctly and efficiently executes the specification.

5.4.2 Comparing the GT and KADS Approaches for Tutoring Support

At a high level, generating tutoring systems using the KADS approach is similar to the GT approach: both are done at the task level, whether by extending task structures or models. This similarity stems from their common philosophy, namely the classification of human problem solving into task-specific categories, and the reuse of such generic tasks for solving problems in different domains. The two approaches are also similar in that they both involve augmentation of the knowledge structures, task decompositions, and inference methods to support tutoring.

Although similar in high-level methods, the two approaches differ in the specifics of how tutoring generation is achieved. In the KADS approach, the developer would specify the ITS extensions separately for the analysis space (conceptual model) and the design space (design model). Moreover, KADS stores domain knowledge in a task-neutral format that is independent of its

intended use, unlike the GT approach that assumes that knowledge should be represented in a format suitable for the way in which it will be used.

An important distinction is that the inference primitives in KADS are more fine-grained than their GT counterparts, so the interface to the ITS shell must be specified at a lower level, equivalent to the level of internal operators in the GT methodology. The ITS architecture can be specified in a more formal and concrete manner than with the GT approach, since the interactions between the ITS shell and expert system will be defined at a finer grain size. This facilitates the process of building the tutoring system architecture, since models of primitive inferences are available for the developer to use in defining the ITS-KADS interactions.

However, the interaction between the ITS shell and a KADS-based system is complicated, since it must be done at the level of the inference layer. KADS supports the specification of tasks through the reuse of library-stored inference models. This implies that the ITS shell must support a model of interaction between the shell and the task for each inference model. In comparison, the GT approach specifies inferencing semantics for each class of tasks. This allows the GT-based ITS shell to represent a single task model for each task category.

Consider the scope of ITS development, as shown by the spectrum in figure 10. At one end of the spectrum, ITSs must be built from scratch for every domain, not allowing any reuse. At the other end, ITS authoring tools are flexible enough to allow the reuse of domain knowledge and tutoring components for any domain. The GT approach falls more towards the general-purpose end of the

spectrum than the KADS approach. For an ITS shell to interact with a GT-based system, it must understand and include support for each basic generic task. Such a shell can generate ITSs for any domain that can be represented as a generic task. The KADS approach falls closer to the middle of the spectrum. Although this approach does not require starting from scratch for each ITS to be built, it does require the implementation of a different ITS framework for each inference model defined in the KADS library. Each time a new inference strategy is introduced, the ITS shell would need to be updated to include support for that strategy in the interaction framework.

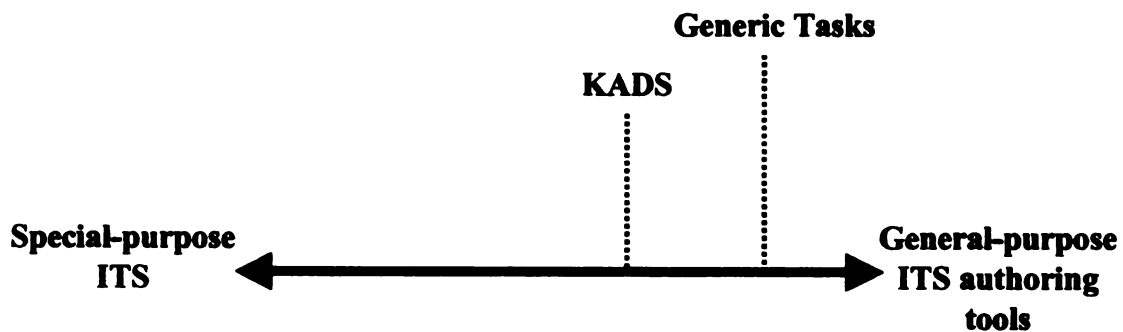


Figure 10. ITS development spectrum

It is important to understand the impact of this distinction in methodologies on the capability of each approach to generate ITSs. Both approaches require multiple frameworks within the ITS architecture to model a knowledge-based system. The GT approach requires a framework for each base GT task and the KADS approach requires a framework for each inference model. In the case of the GT approach, a single framework for each generic task is sufficient because a fixed set of semantics is enforced for each task, so all the associated inferencing strategies can be identified and encoded into that framework.

However, in the KADS approach, generic inference models are reused to build task-based systems. These reusable models are at the inference level – one level below the task level. These models are at the level of the internal operators in the GT approach. Thus, in the KADS approach, the ITS architecture must include many more finer-grained frameworks, one for each inference model, to support interaction with a task-based system.

One of the advantages of the KADS approach is its task-neutral knowledge representation format. Although such task-neutrality has not been demonstrated in working KADS systems, the approach assumes that such task-neutral representation is possible. Domain knowledge is stored independently of its intended use and, thus, the ITS shell can share knowledge across different tasks. The advantage is the elimination of duplicate knowledge representations if the same domain is to be tutored from different task perspectives. However, this advantage could be a practical disadvantage since a task-neutral knowledge format does not provide a semantically rich representation for tutoring task-based domains. The quality of the tutoring knowledge is affected by the knowledge representation used within the KADS approach. A second advantage of the KADS approach is that it provides a more formal design, development, and evaluation methodology for knowledge-based systems, which would allow the development of a more formal specification of the ITS architecture.

ITS generation using the KADS approach also has several disadvantages. For one, the migration from conceptual specification and design to implementation of a tutoring system is more challenging. The GT approach offers

direct support for implementation in the form building blocks, which can be extended to include tutoring support. The implementation task is not as straightforward in the KADS framework since there is a gap between the conceptual modeling and design phases, and the implementation of a system that meets the required specifications. Another disadvantage alluded to above stems from the general task-neutral representation of domain knowledge adopted. The issue is how usable that knowledge is for tutoring from a task-specific viewpoint, and whether it needs to be represented in a more suitable format. A more detailed account of ITS generation using the KADS approach is given in (El-Sheikh and Sticklen, 1999).

5.4.3 Analysis of the Two Approaches

The previous section compared two recognized task-specific methodologies, GTs and KADS, for ITS generation from a theoretical viewpoint. Although there are other candidates that might be compared, the GT/KADS comparison proves illustrative. The analysis reveals that both frameworks can support the generation of ITSs for different domains. The salient result is that a GT framework is of greater leverage in generating ITSs because of the larger problem solving *granularity* of the GT approach compared to the KADS approach. In a sense, because of stronger semantic commitments taken following the GT approach, a system developer is strongly tied to a backing theory of problem solving. Yet, once implemented, the stronger commitments made in a GT-based system enable a *general* ITS overlay to be developed to generate ITSs on an automatic and largely knowledge-free basis.

Chapter 6. Tahuti: A Software Systems for Generating Intelligent Tutoring Systems

This chapter describes Tahuti² – a software system for generating ITSs that implements the architecture developed in the previous chapter. By integrating reusable tutoring components with GT-based expert systems, Tahuti can dynamically generate tutoring systems for a wide range of domains. This chapter discusses the implementation of Tahuti, how it is used to generate intelligent tutoring systems, and describes some ITSs generated using Tahuti.

6.1 Implementation of the Tahuti System

Tahuti consists of three main components: the HC-based expert system, the ITS shell, and the extended knowledge module, as shown in figure 11. The conceptual details of these components were described in the previous chapter. All the components of the architecture were developed in Smalltalk using the VisualWorks development environment. The system runs as a stand-alone environment from a CD and is platform-independent.

The expert system is developed as an instance of a hierarchical classification-based GT system. It is built using the Integrated Generic Task Toolset developed and used at the Intelligent Systems Laboratory, Michigan State University (Sticklen, Penney et al., 2000), a comprehensive toolset for

² The architecture is named after Tahuti, the ancient Egyptian god of knowledge, wisdom, learning, and magic. ITS development and use, I believe, involves a bit of all four aspects.

knowledge-based systems analysis and implementation. It includes tools for building Routine Design-based (RD) and Hierarchical Classification-based (HC) systems, among others. A detailed account of how the expert system component of Tahuti is developed using the HC tool is given in section 6.2.1. Tutoring systems for different domains can be generated by plugging different expert systems into Tahuti.

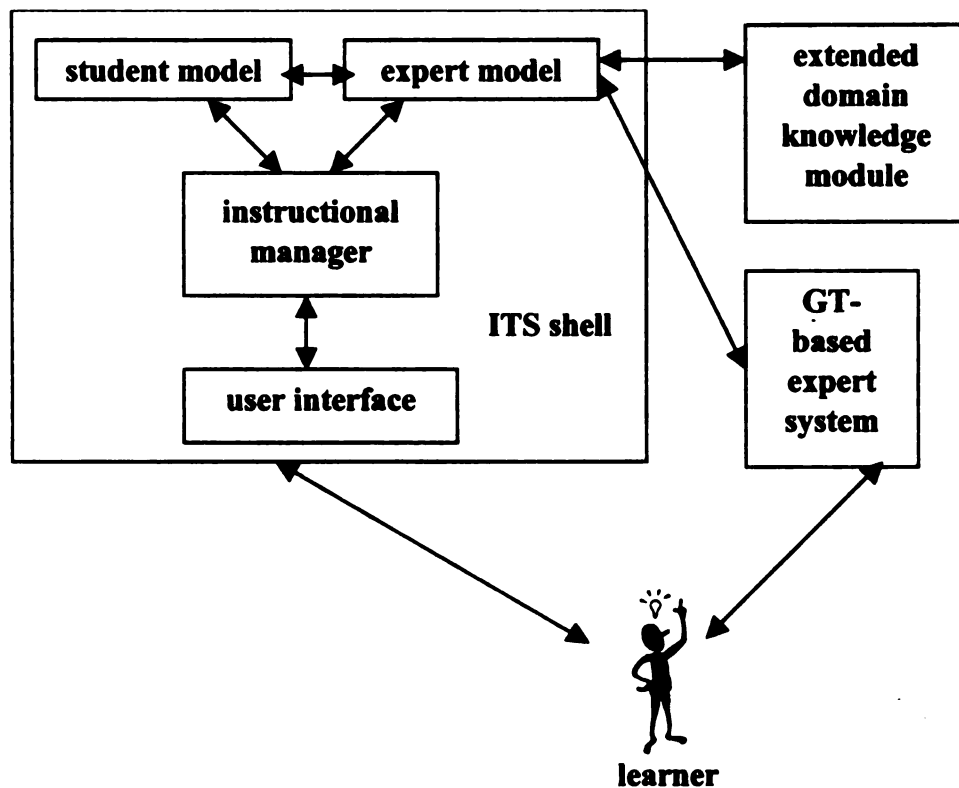


Figure 11. Tahuti system architecture

The ITS shell component is implemented as Smalltalk modules, and consists of four sub-components: the expert model, student model, instructional manager, and user interface. The expert model has the mechanisms needed for interacting with the GT system to extract domain and problem solving knowledge as needed for tutoring. The student model is implemented as an overlay model to

keep track of the learner's performance on the topics covered by the tutor. It makes use of the expert system as a model of expertise, and makes decisions about the student's learning by comparing the student's behavior to that of the expert system. The instructional manager coordinates the delivery of instruction to the learner through the user interface and manages the actions of the other components of the ITS shell as needed in carrying out the curriculum. The user interface adopts a simple layout that is tailored to tutoring using a problem solving approach. The ITS shell implemented incorporates the features of the architecture that were described in the previous chapter.

The extended knowledge module is implemented as a structured text file. This module contains domain knowledge that is needed for tutoring but not available from the expert system. This includes a high level description of the problem to be solved and the topics to be covered in the curriculum as examples and questions. The expert model component of the ITS shell uses this file to derive the additional knowledge needed for tutoring. A detailed description of how the extended knowledge component of Tahuti is constructed is given in section 6.2.2.

6.2 Using Tahuti to Generate Intelligent Tutoring Systems

To generate an ITS for a certain domain using the Tahuti system, an ITS author needs to perform three main steps:

1. Identify an existing GT-based expert system for that domain. If one does not exist, the author can develop the expert system using the ISL's Integrated Generic Task Toolset.

2. Construct the extended knowledge module.
3. Generate the intelligent tutoring system.

These steps are described in greater detail in the next three sections.

6.2.1 Developing the Expert System

Tahuti is designed to be used with existing GT expert systems, as well as with any domain problem that can be represented as one. To develop an expert system that will be used as part of the architecture, an author uses the HC tool of the ISL's Integrated Generic Task Toolset. This section gives a brief account of how to develop an expert system using the HC tool. For a more detailed description, refer to (Moy, McDowell et al., 1994). The description uses the development of an expert system for problem solving with Microsoft Excel tools as an example. This expert system was integrated with the Tahuti architecture to generate an ITS for Excel tools. A complete representation of the Excel tools expert system is given in Appendix A.

The basic steps involved in developing an expert system are:

1. The first step is to open the GT Toolset and create a new HC-based problem solver, as shown in figure 12, of the type *SimpleHCPS*.

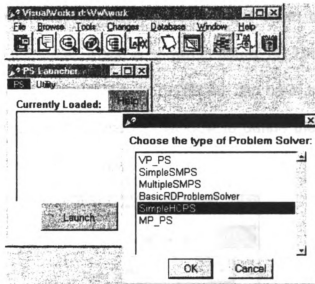


Figure 12. Creating the problem solver

2. The above step opens a problem solver window, as shown in figure 13. This window shows the cases defined for the problem solver. Each case consists of a set of data values, which the problem solver can run to generate an output solution. The Case menu is used to define cases for the problem solver, which will later be presented by the tutor as examples and questions.

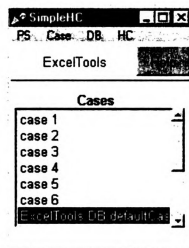


Figure 13. Problem solver window

3. From the problem solver window, select the *DB* menu option to open the problem solver's database, which is shown in figure 14. The database defines the variables used in problem solving. Variables can be of six types, such as a numerical or string variable. Use this window to define the variables and set their values for each defined case.

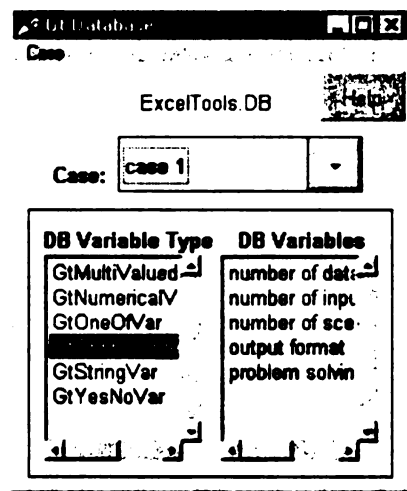


Figure 14. Database window

4. Each variable defined in the step above is then initialized using the variable definition window, shown in figure 15. This allows the author to set the legal values for a variable, a description of it, and the question to ask the user when this variable is needed during problem solving.

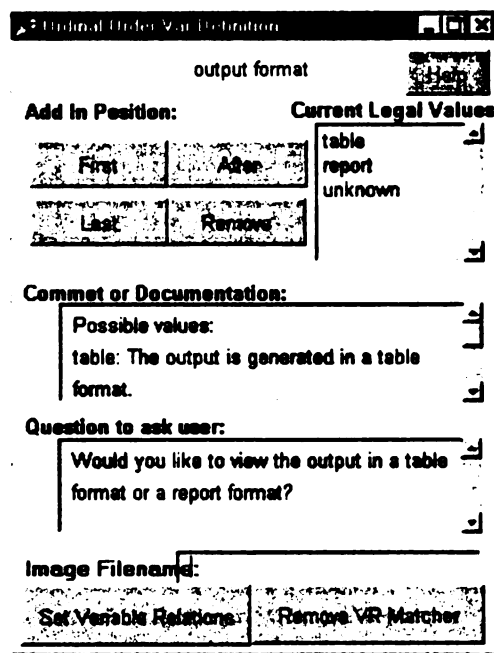


Figure 15. Variable definition window

5. The next step is to decompose the problem into a hierarchical classification of specialists, as shown in figure 16. The top-level specialist represents problem to be solved and the leaf-level specialists represent the possible output alternatives. The problem solver traverses this hierarchical classification, using the data values from a defined case, to determine the appropriate output solution for that case.

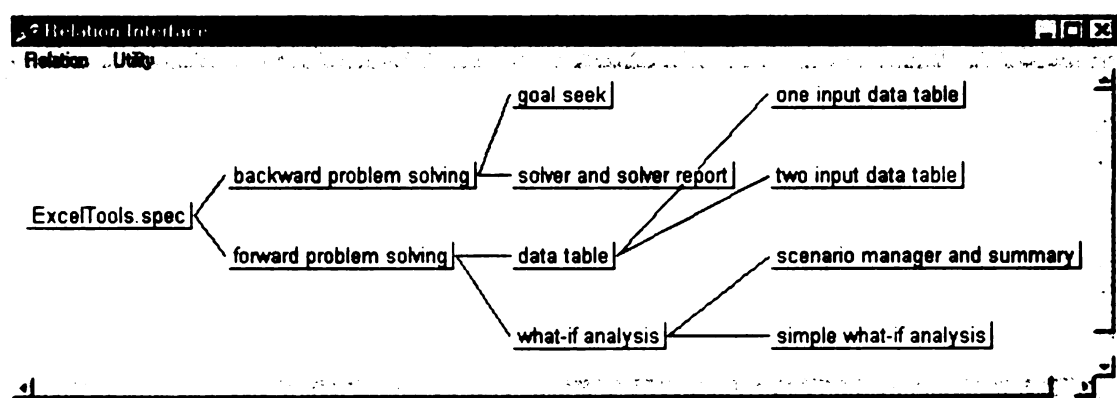


Figure 16. Classification hierarchy

6. Each specialist defined uses a table matcher, shown in figure 17, to make decisions about the data. This table is used to determine if the specialist matches the given data values. The columns of a table matcher represent the variables that affect the specialist, and each row represents a combination of their values. The last column shows the result of processing each row. In the table shown in figure 17, the second row can be interpreted as: 'If the number of input values = multiple, and the problem solving direction = backward, then the result of this specialist = match.' Each specialist defined in the step above must have a corresponding table matcher.

Table Matcher Interface

Rows: 5 Columns: 3

goal seek.sm.ExcelTools.tt

	number of input values	problem solving direction	result
1	= single [F]	= backward	strongly match
2	= multiple [T]	= backward [T]	match **
3	= unknown	= backward	weakly match
4	?	= backward	against
5	?	~= backward	strongly against

Run Clear Browse Comment

Figure 17. Table matcher window

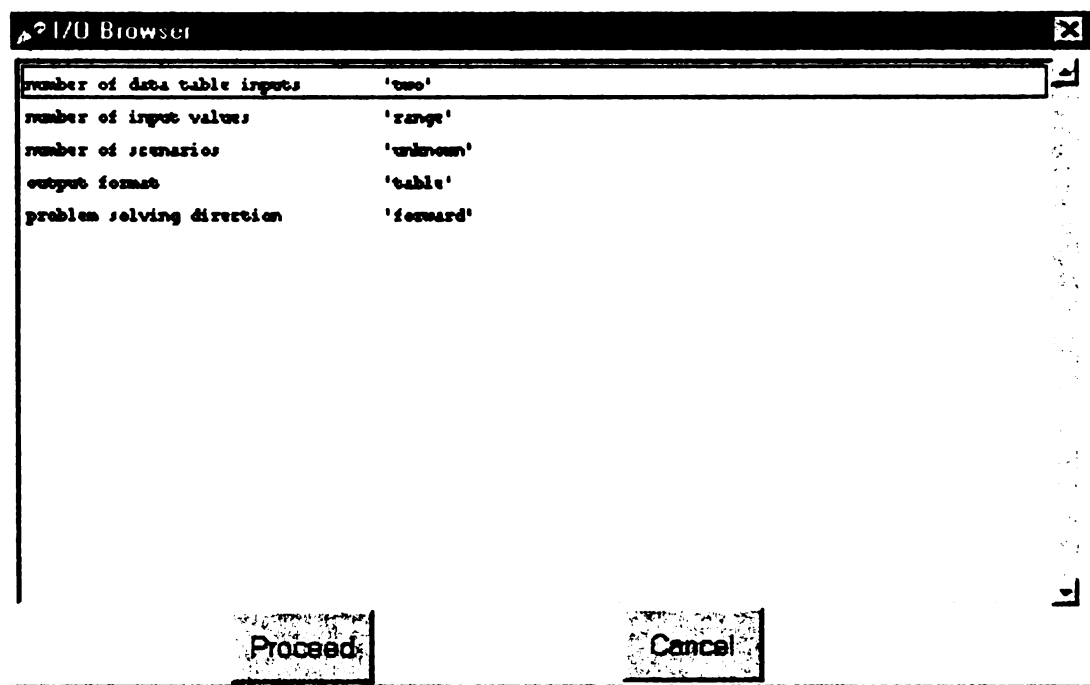


Figure 18. Input window

After the above six steps have been completed, the expert system is ready to be used and integrated into the Tahuti architecture. To run the expert system, select *Run PS* from the *PS* menu of figure 12. This brings up the input window, shown in figure 18. This window shows the variables' values for the current case, which can be modified or left as-is.

Clicking on the Proceed button in figure 18 runs the expert system and generates the output window, shown in figure 19. This window shows the classification hierarchy, augmented with information about the result of each specialist. Traversing the path that *matched* to a leaf-level specialist identifies the output solution for the given case.

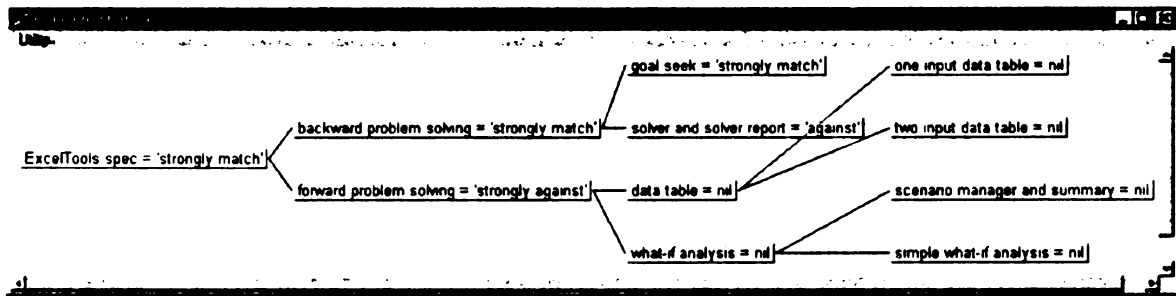


Figure 19. Problem solver output window

6.2.2 Constructing the Extended Knowledge Module

The extended knowledge module contains domain knowledge that is needed for tutoring but not available from the expert system. The expert model component of the ITS shell uses this file to derive the additional knowledge needed for tutoring. This module is represented as a structured text file. This file includes the following information:

- A high level description of the problem solving purpose.
- A description of the output alternatives that problem solving can produce.
- The example topics to be covered by the curriculum.
- The practice topics and questions to be covered by the curriculum.

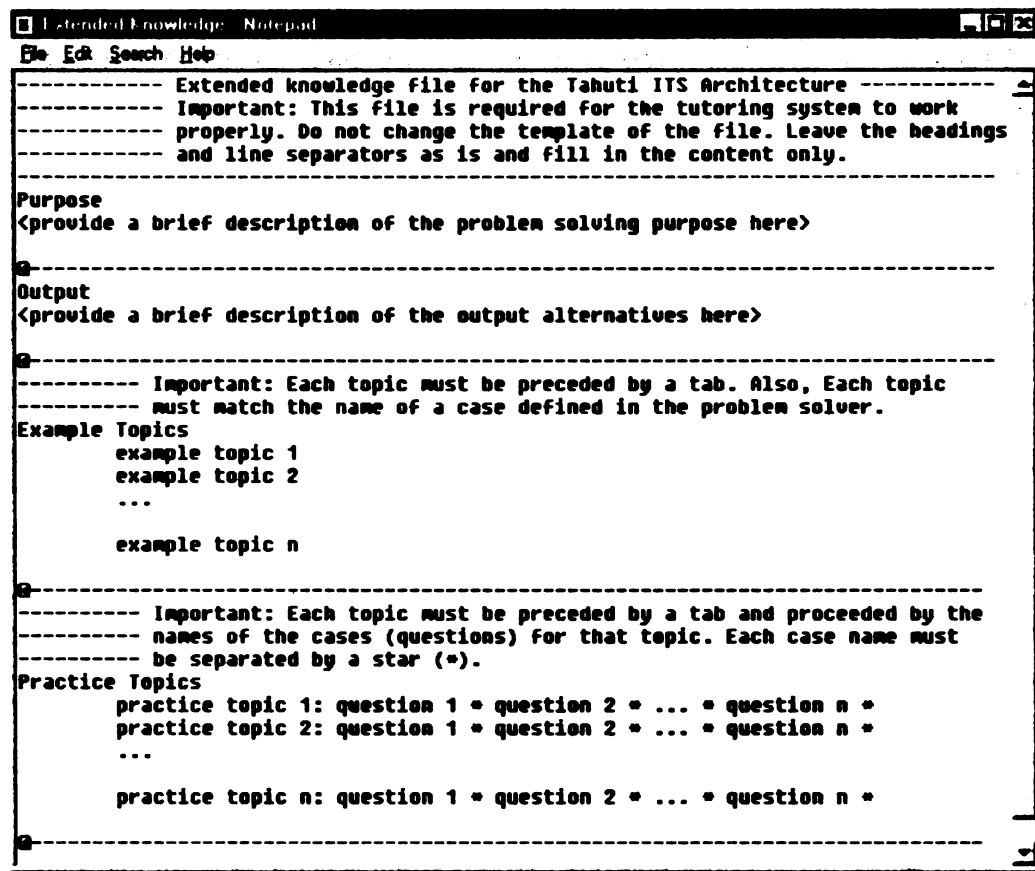


Figure 20. Template for the extended knowledge file

The template for the extended knowledge file is shown in figure 20. The ITS author fills in the template with the required information. The author is given flexibility in determining the number and ordering of the example and practice topics covered by the curriculum. The author also determines how many questions the tutor asks learners on each topic. Each example topic and question defined in the curriculum must match the name of a case defined in the expert system.

6.2.3 Generating the Intelligent Tutoring System

If the expert system and extended knowledge module have been developed, generating the ITS is a simple procedure. The procedure involves linking the

expert system and extended knowledge module to the ITS shell in two straightforward steps.

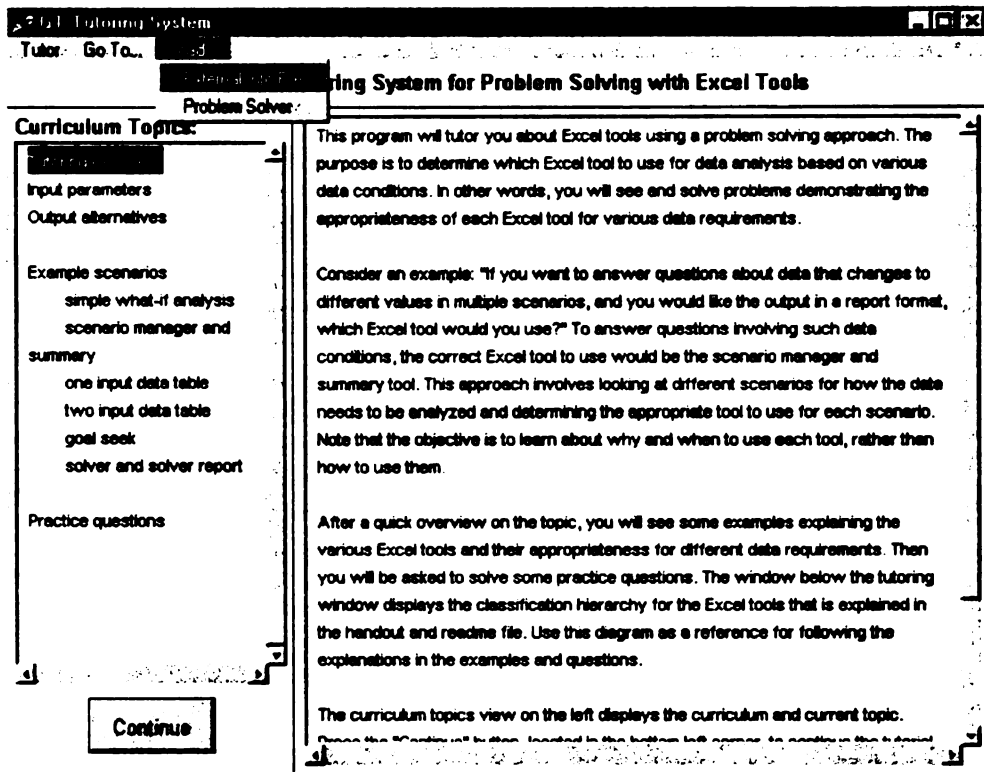


Figure 21. The load menu of the Tahuti ITS architecture

The main user interface of the ITS architecture features a load menu, shown in figure 21, that is used for linking the components of the ITS architecture. The ITS author uses this menu to specify the name of the expert system, as shown in figure 22, and the name of the extended knowledge file, as shown in figure 23. Once the components of the architecture have been integrated, the tutor is generated dynamically and is ready to use. The features of the tutors generated are described in the next section.

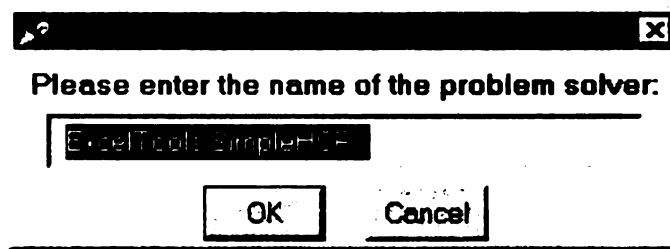


Figure 22. Linking the expert system to the architecture

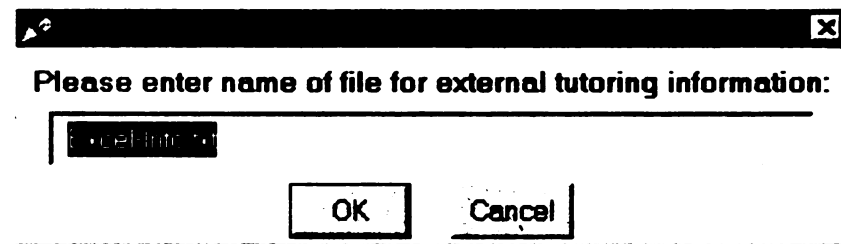


Figure 23. Linking the extended knowledge file to the architecture

6.3 Examples of ITSs Generated by Tahuti

This section describes two working tutors that were generated using the Tahuti architecture. The tutors cover two very different domains. The first domain is the set of Excel problem solving and what-if analysis tools. The second domain is the fabrication of composite materials systems. Figures 24 and 25 illustrate how the Tahuti architecture is instantiated to generate each tutor. The architecture generates tutors for different domains by plugging in different expert systems. The ITS shell itself is reusable for different domains.

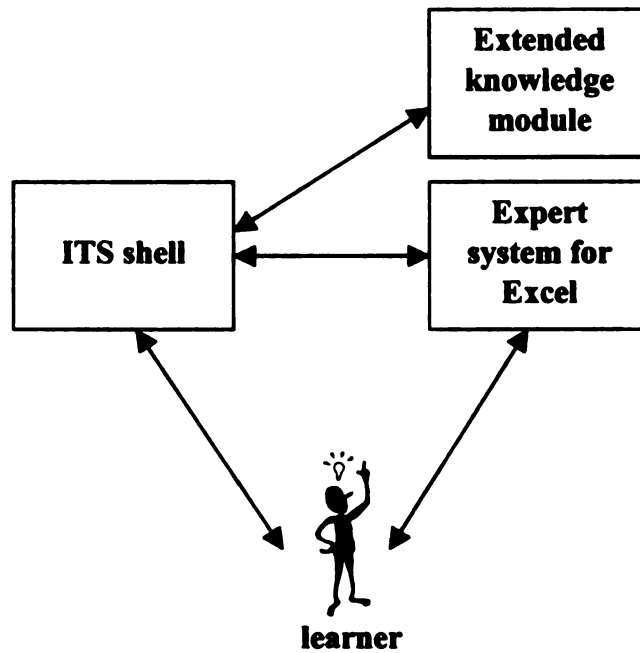


Figure 24. Tahuti architecture instantiated to generate the Excel Tutor

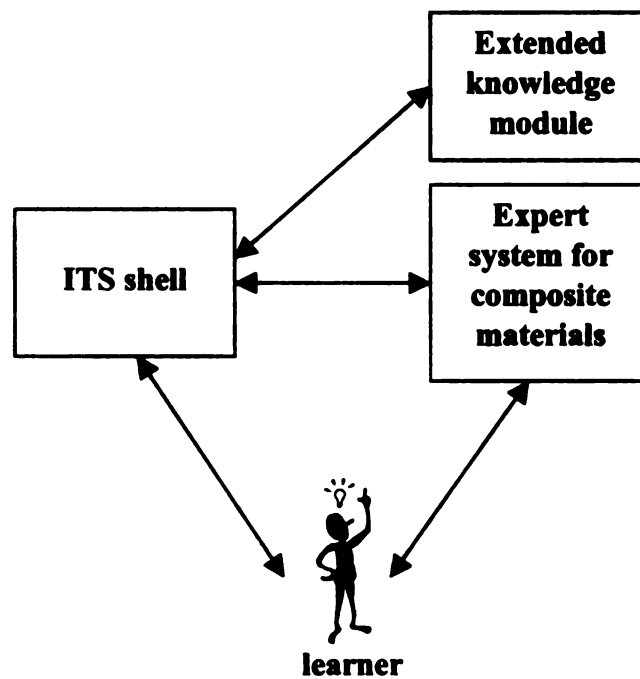


Figure 25. Tahuti architecture instantiated to generate the Composite Materials Tutor

6.3.1 The Excel Tutor

This section describes the Excel Tutor: an ITS generated by the Tahuti architecture for tutoring about the Microsoft Excel what-if analysis and problem solving tools. Specifically, the tutor helps learners determine which Excel tool to use for data analysis based on various data conditions. The ITS uses a problem solving approach to tutoring. In other words, the tutor presents examples and questions demonstrating the appropriateness of each Excel tool for various data requirements.

The main user interface is shown in figure 26. The view on the left displays the curriculum and current topic, and the view on the right shows the main tutoring content. The learner presses the "Continue" button, located in the bottom left corner, to continue the tutorial session after completing each screen. The "Go To..." menu can be used to jump to a specific section of the tutorial.

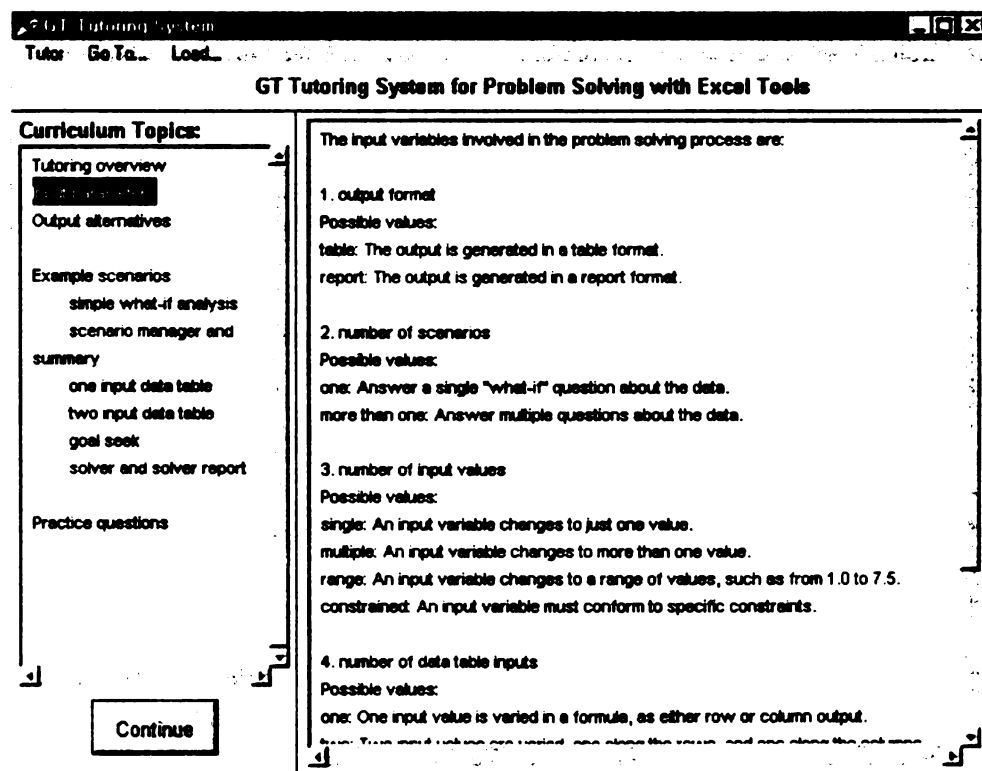


Figure 26. Excel Tutor – main user interface

A tutoring session starts with an overview on the topic. This includes a general description of solving problems with the Excel tools, and a description of the input parameters and output alternatives involved in the problem solving process. Figure 26 shows the input description provided by the tutor.

The ITS then presents examples that explain the various Excel tools and their appropriateness for different data requirements. An example presented by the Excel Tutor on the solver tool is illustrated in figure 27. The first part of the example shows the values for the input variables. The next part of the example shows the correct tool to use for the input conditions and the explanation for selecting this tool.

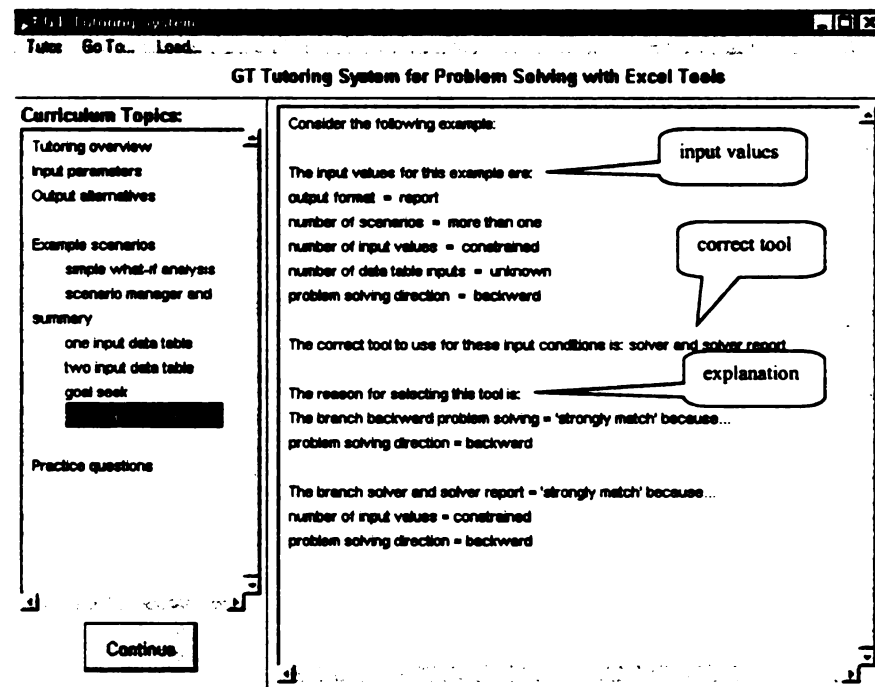


Figure 27. An example presented by the Excel Tutor

The explanations generated by the tutor are based on a classification approach to problem solving. A window below the main tutoring window displays

the classification hierarchy for the Excel tools, which is depicted in figure 28. Learners use this diagram as a reference for following the explanations in the examples and questions. This hierarchy provides a cognitive structure for learners to map their knowledge onto as they learn.

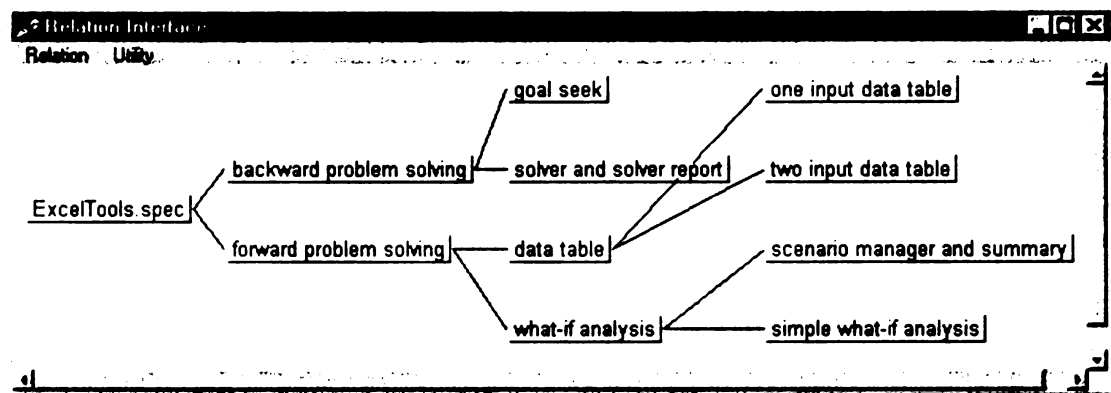


Figure 28. The classification hierarchy for the Excel tools

In figure 28, the root node is the set of Excel tools and each of the leaf nodes represents an individual tool. Determining the appropriate Excel tool to use for a set of data conditions involves starting at the root node and trying to find the correct path down the hierarchy that matches the data conditions. The tutor explains the path of nodes that matched the data conditions for that example. For each branch that matched, the tutoring system explains the input conditions that caused it to match. The final branch shows the output result: the appropriate tool selected for the given data conditions.

After presenting the examples, the tutor asks the learner to solve some practice questions. A question posed by the tutor is shown in figure 29. The tutor presents a set of data conditions, and asks the learner to select the appropriate data analysis tool for those conditions. If the learner selects the correct tool, the tutor gives the learner appropriate feedback and provides an explanation of the

solution, shown in figure 30. Examples of feedback provided by the tutor. If the learner's answer is incorrect, the tutor provides an appropriate hint, and asks the learner again, as shown in figure 31.

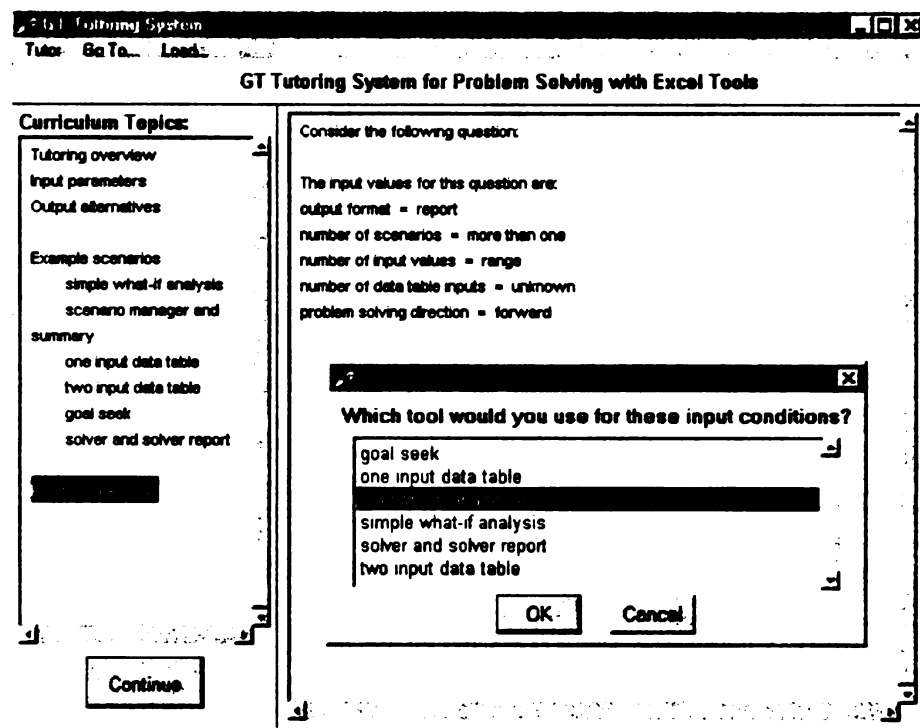


Figure 29. A question posed to the learner by the Excel Tutor

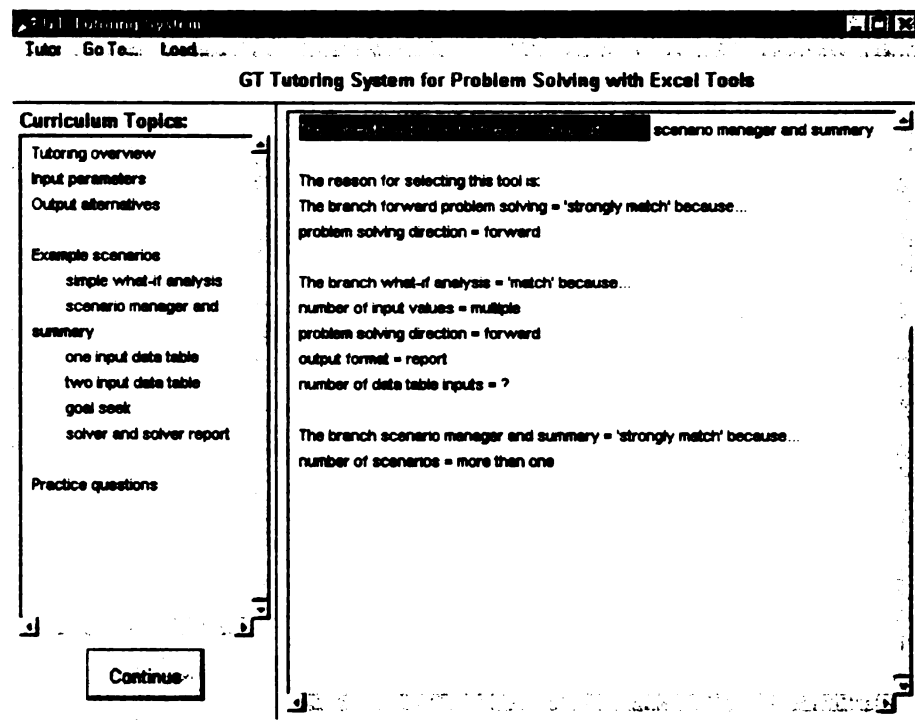


Figure 30. The Excel Tutor's explanation of correct answer

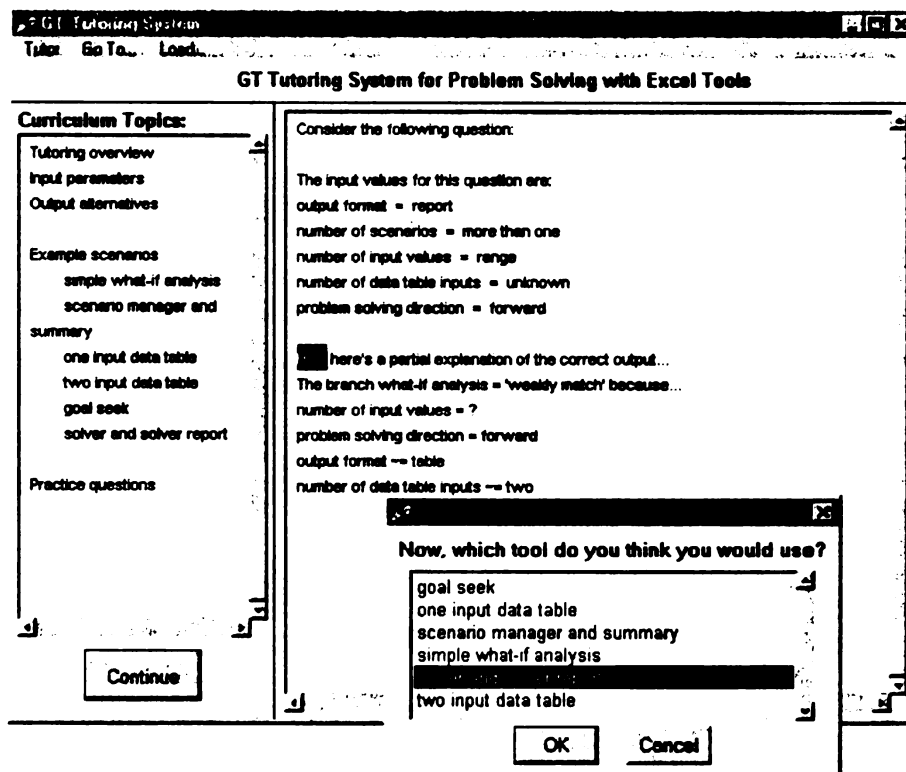


Figure 31. The Excel Tutor shows the learner a hint and asks again

The hint gives the learner a suggestion about how to proceed in determining the correct tool, based on where the learner is in the problem solving process. If after giving the learner several hints and attempts to solve the question, the learner still cannot determine the correct answer, the tutor gives the learner appropriate feedback, as shown in figure 32. It then displays the correct answer, along with an explanation of that solution. This explanation is similar to the one shown earlier in figure 30.

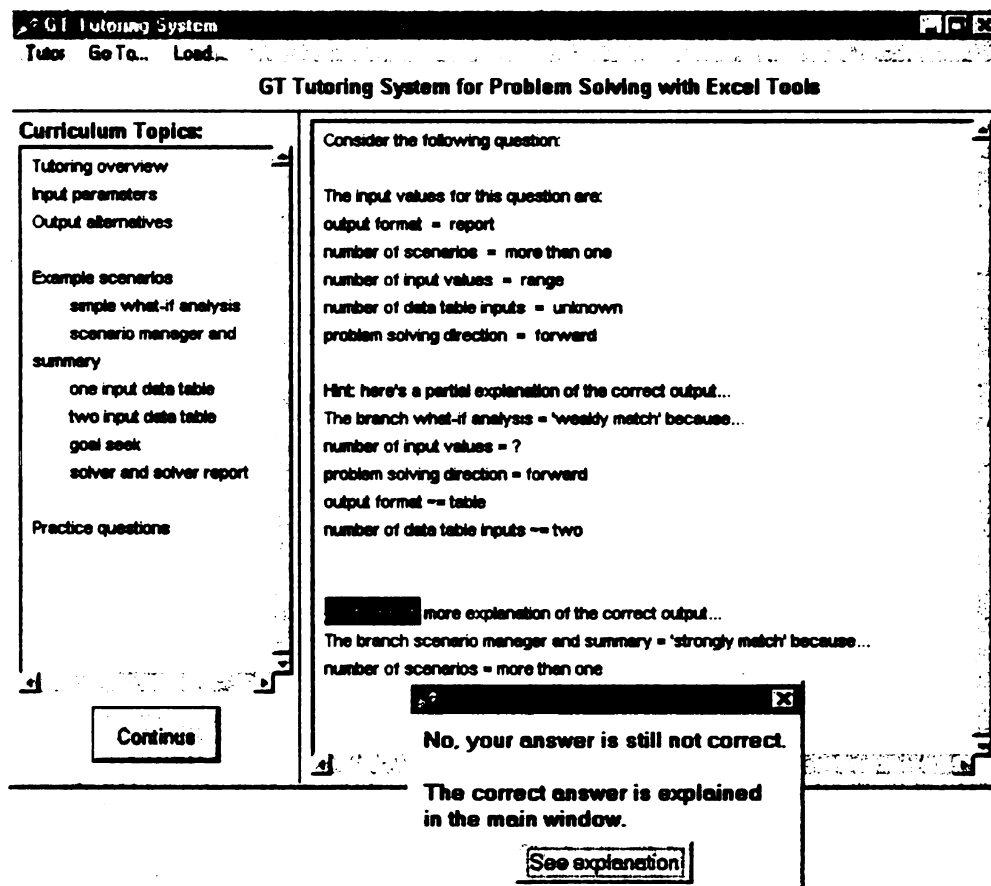


Figure 32. An example of feedback given by the Excel Tutor

The Excel Tutor scores the learner's performance on each question, based on how many hints and/or attempts the learner needed to solve the problem. The tutor computes the learner's overall score for each topic and uses

those scores to direct the instruction. If a learner's score on a topic is very low, the tutor presents more examples on the topic before asking more questions, as illustrated in figure 33. The tutor moves to a new topic when the learner has become competent in a topic. Competency is defined by the author of the ITS. For the Excel Tutor, it means that the learner has answered at least two questions correctly on the topic and has an overall score of 70% or higher on that topic. When the learner has finished covering all the topics, the tutor displays a report summarizing the learner's performance on the topics. This report is illustrated in figure 34.

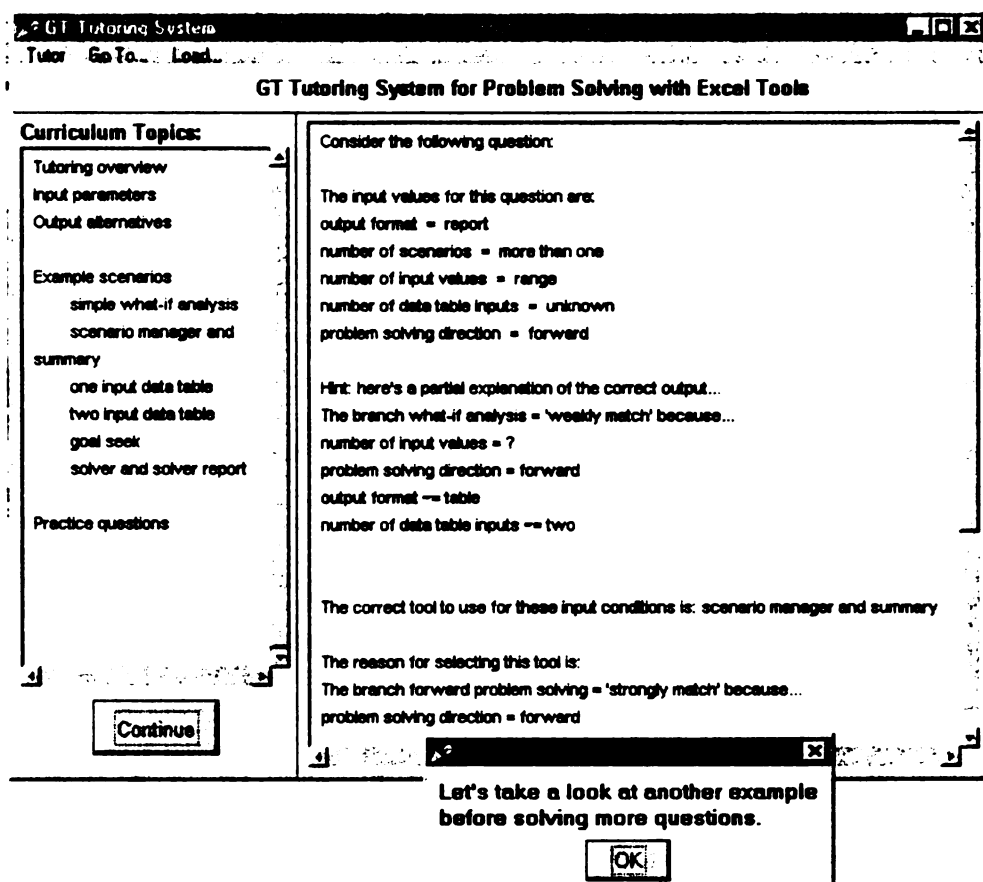


Figure 33. The Excel Tutor presents more examples

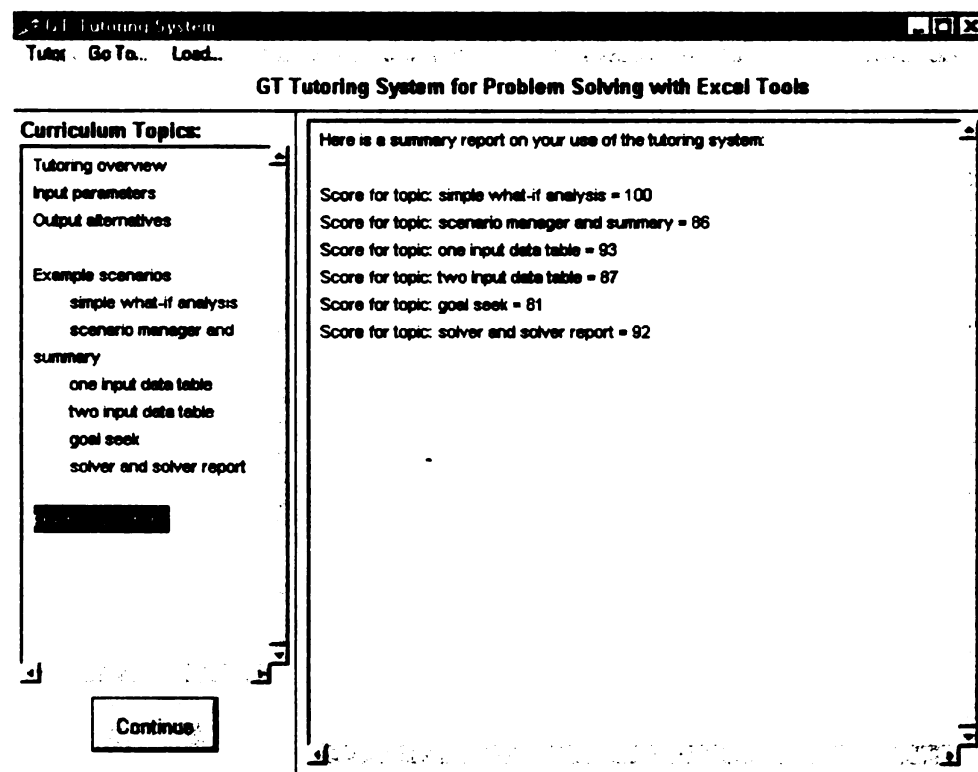


Figure 34. Summary report displayed by the Excel Tutor

6.3.2 The Composite Materials Tutor

This section describes another tutor generated by the Tahuti architecture for the domain of composite materials. Specifically, the Composite Materials (CM) tutor covers the domain of selecting appropriate fabrication technologies for systems designed out of composite materials. The tutor is generated using an existing expert system named COFATE2 (Moy, McDowell et al., 1994). COFATE2 was developed using the HC tool of the Intelligent Systems Laboratory's Integrated Generic Task Toolset (Sticklen, Penney et al., 2000).

The main interface of the CM Tutor is shown in figure 35. The tutoring session begins with an overview about the topic, which includes a description of the problem solving purpose, input parameters, and output alternatives.

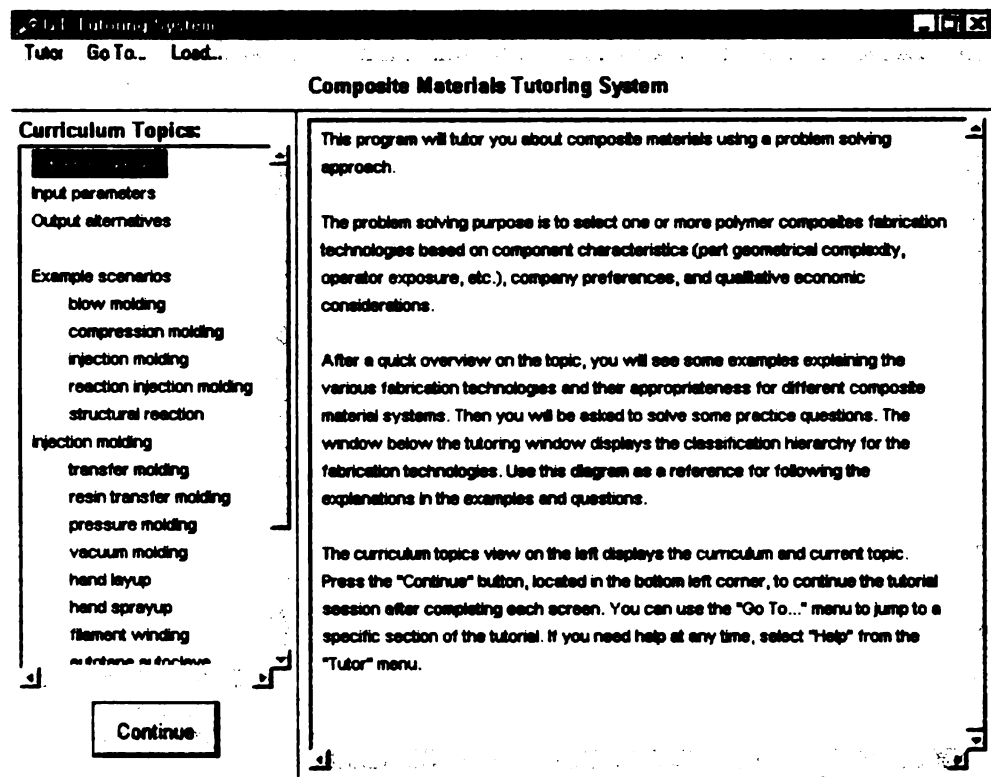


Figure 35. Composite Materials Tutor – main user interface

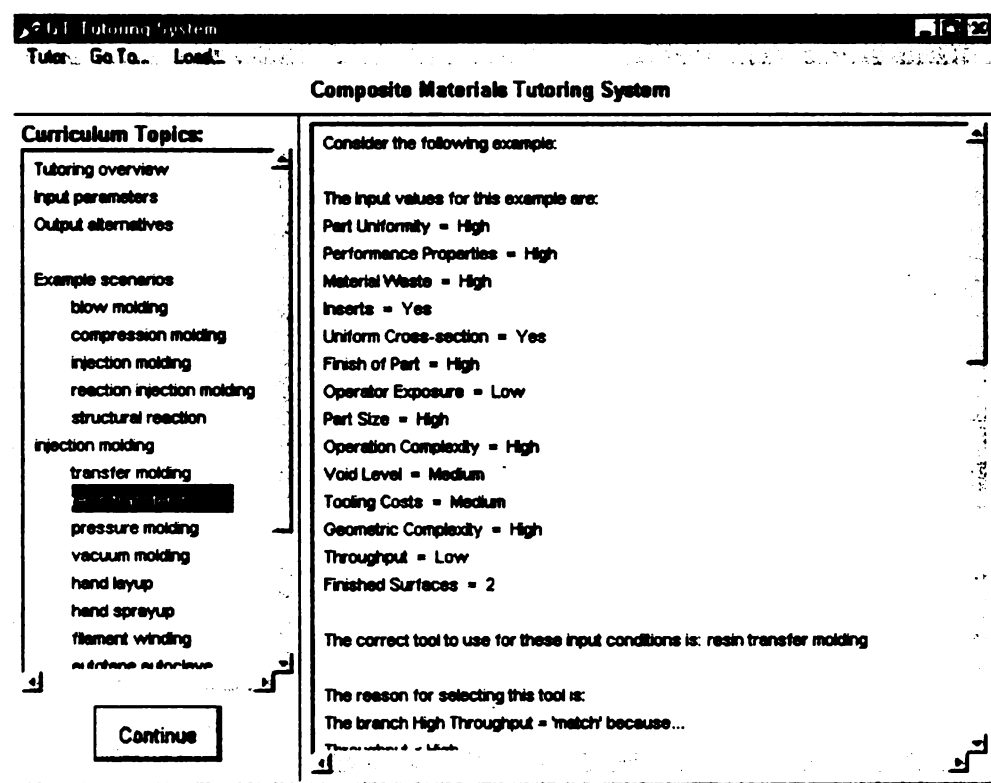


Figure 36. An example presented by the CM Tutor

The tutor then presents examples, like the one shown in figure 36. The tutor presents an example for each topic defined by the ITS author in the curriculum. After that, the tutor asks the learner to solve practice questions, like the one shown in figure 37. For each question, the tutor presents a problem solving scenario, and asks the learner to select the appropriate fabrication technology.

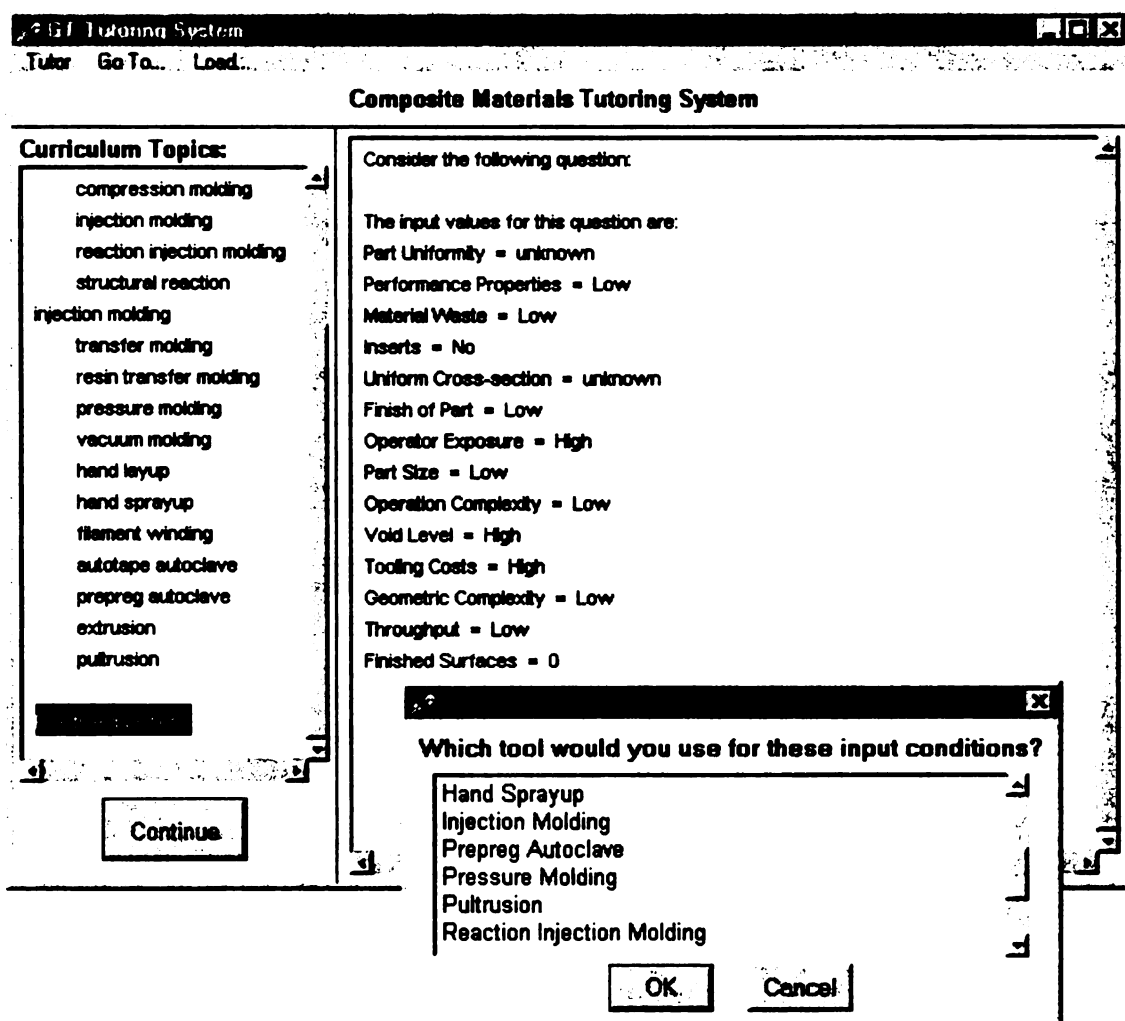


Figure 37. A practice question asked by the CM Tutor

The tutor presents appropriate feedback according to the current instructional objective and learner's performance. Examples of feedback provided by the tutor are shown in figure 38.

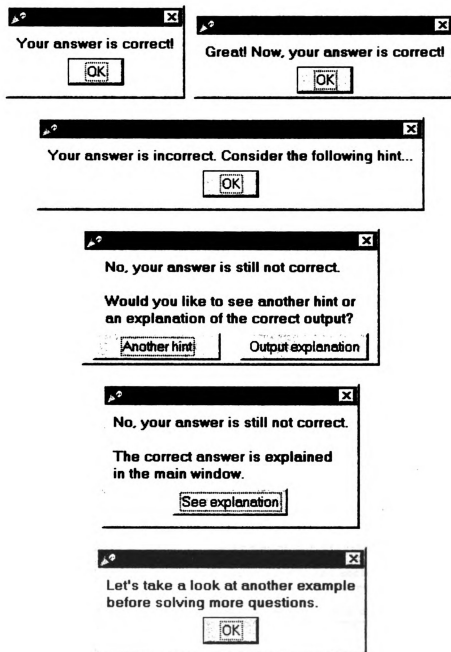


Figure 38. Examples of feedback provided by the CM Tutor

If the learner selects an incorrect answer, the tutor gives the learner one or more hints, and asks again, as shown in figure 39. If the learner selects the correct answer or fails to do so after several attempts, the tutors presents an explanation of the correct answer, as shown in figure 40.

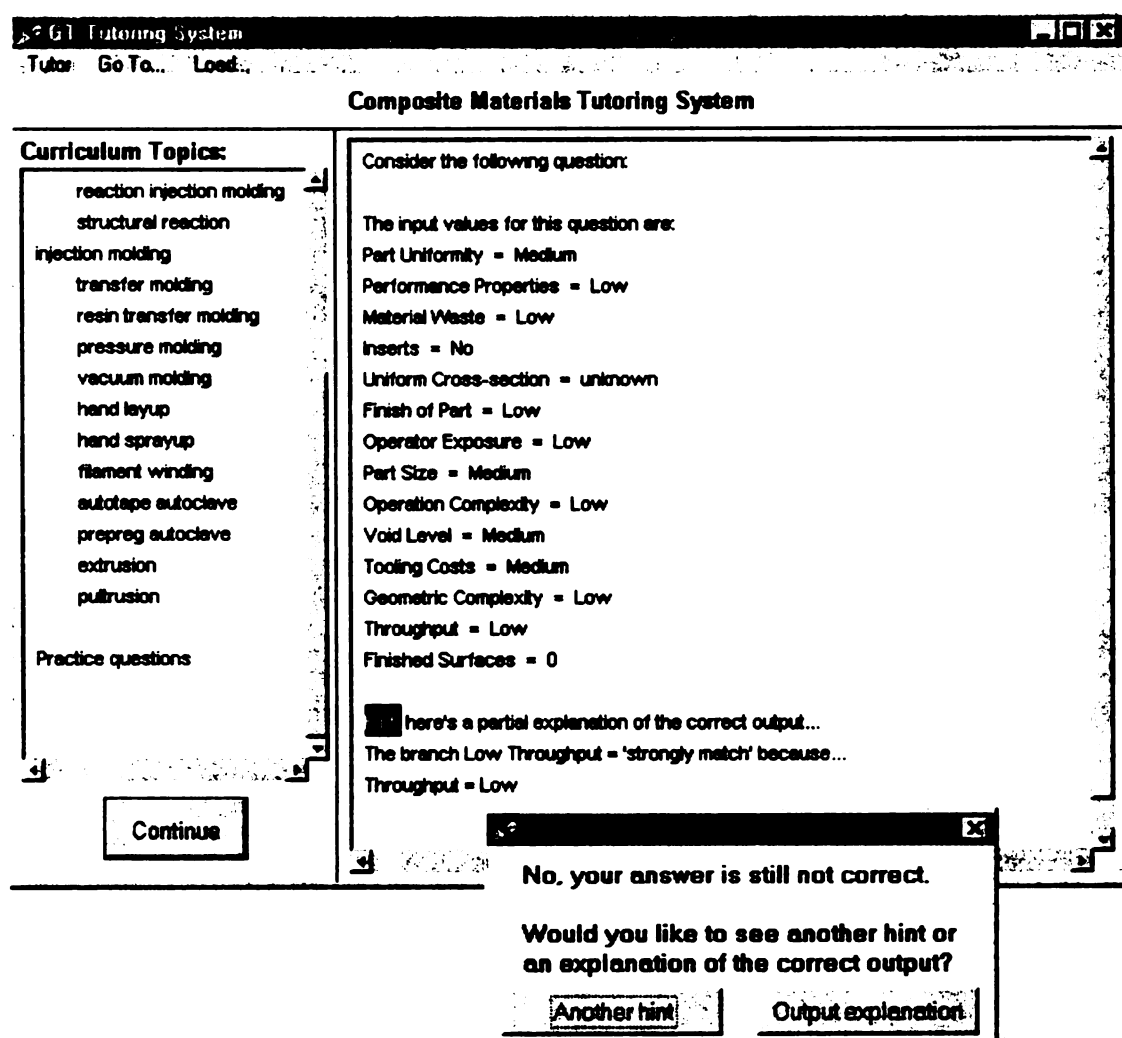


Figure 39. The CM Tutor gives a hint and asks again

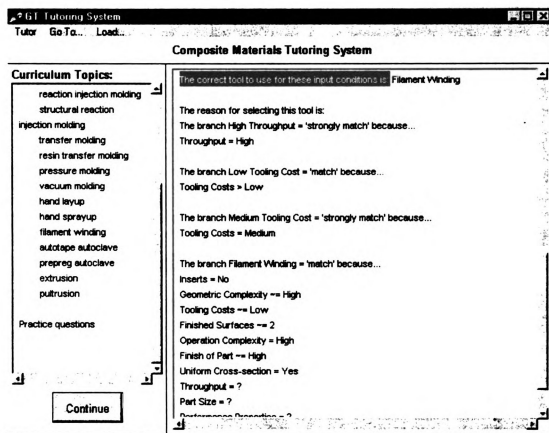


Figure 40. Explanation of correct answer provided by the CM Tutor

The hints and explanations provided by the tutor are based on the hierarchical classification of fabrication technologies, which is illustrated in figure 41. This classification structure is an integral part of the architecture. It is used by the expert system during problem solving. It is used by the tutor in generating the hints and explanations. And it is used by the learners as a cognitive structure to map their knowledge onto as they learn.

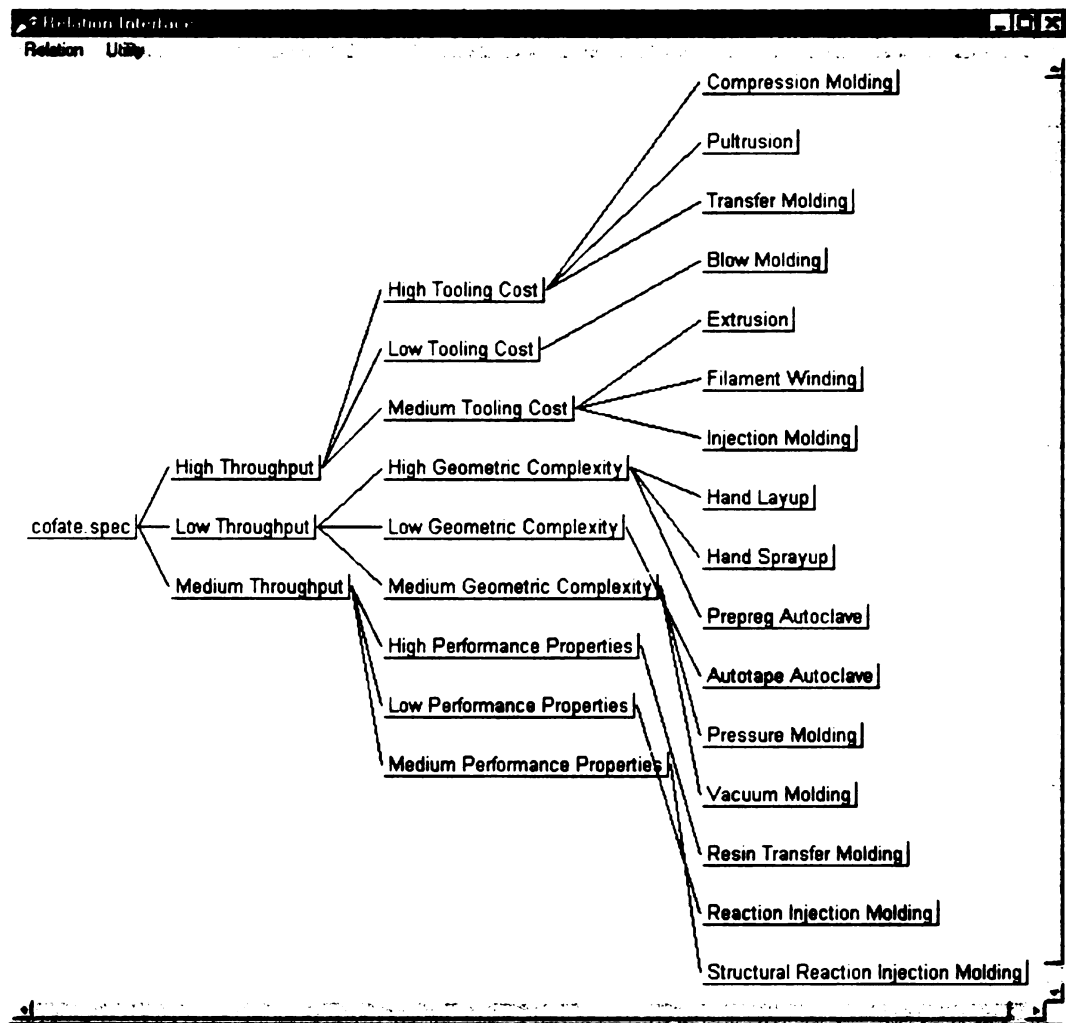


Figure 41. Hierarchy of fabrication technologies shown to the learner by the CM Tutor

When the learner has achieved competency on the curriculum topics, the tutoring session ends, and the tutor displays a summary report, shown in figure 42.

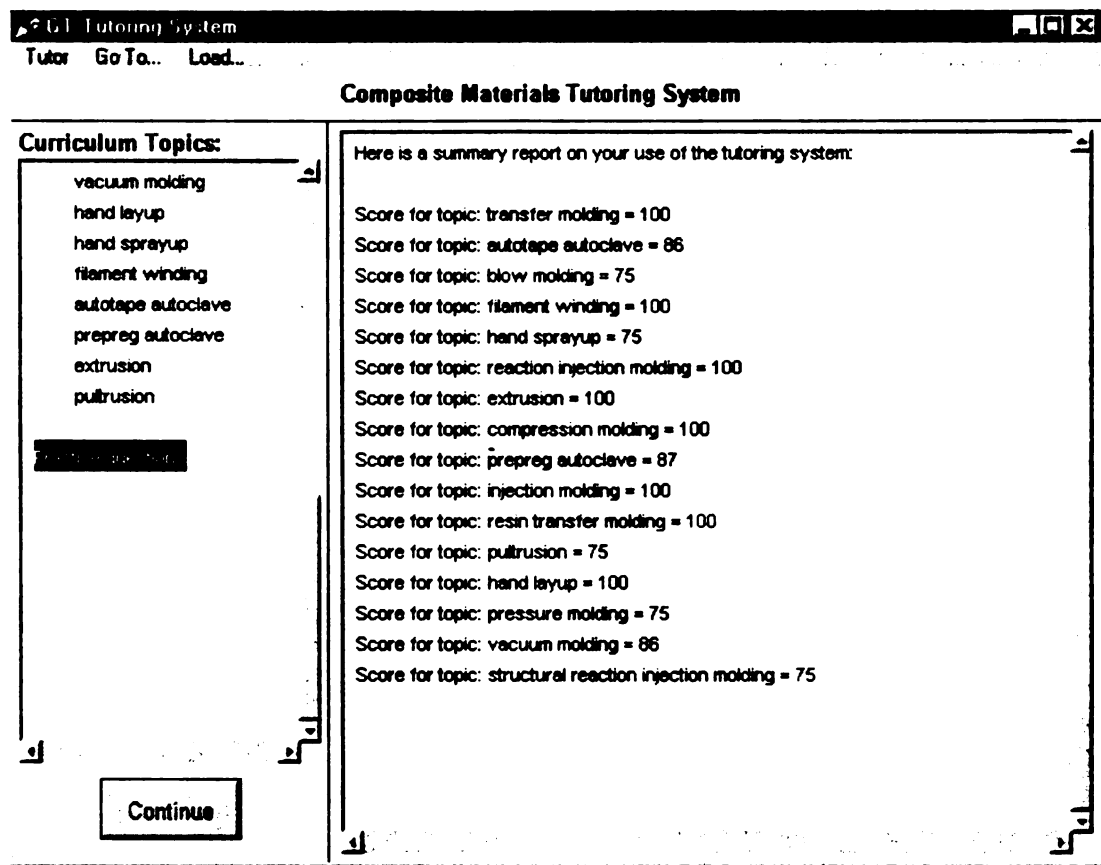


Figure 42. Summary report presented by CM Tutor

6.4 Constraints of the Tahuti System

Although Tahuti can generate tutors for a wide range of domains, it has constraints on the type of tutors generated. Specifically, the system can generate an ITS for any domain that can be represented as a hierarchical classification-based generic task. The Tahuti architecture can specifically interact with any expert system built using the hierarchical classification (HC) tool of the Integrated Generic Task Toolset. The tutors generated using Tahuti that were described above cover domains that come from very different fields. However, they both share the same problem solving methodology, namely, a classification-based approach. Tahuti cannot generate tutors for domains represented as other types

of generic tasks, such as routine design (RD) tasks. In addition, it cannot generate tutors for domains that are represented as integrated generic tasks. However, the architecture developed in chapter 5 provides the groundwork for the development of software architectures for generating ITSs from other types of generic tasks.

Chapter 7. Evaluation of the Tahuti ITS Architecture

This chapter describes the evaluation of the Tahuti architecture for generating intelligent tutoring systems. The evaluation of the architecture involves two components: first, evaluating the process of generating ITSs using Tahuti, and second, evaluating the effectiveness of the ITSs generated using Tahuti. The next two sections describe these evaluations.

7.1 Evaluating the Process of Generating ITSs

A fundamental part of evaluating the architecture is the consideration of the process of using the architecture to generate ITSs. The process of generating an ITS using the architecture can be much easier and quicker than building an equivalent system from scratch. In addition, the development can be done by domain experts rather than by programmers.

The fact that the Tahuti architecture employs a tutoring shell overlay on top of expert systems allows domain experts to easily structure their knowledge. Moreover, the representation used by the architecture facilitates the effective reuse of problem solving and domain knowledge for tutoring. Tahuti removes much of the programming needed to build a typical intelligent tutoring system by using an expert systems foundation and reusable tutoring components. Indeed, the Tahuti architecture fundamentally reduces the ITS development process to a knowledge acquisition and organization process within a well-structured environment.

The Tahuti architecture *generates* ITSs, as opposed to other ITS authoring environments, which provide frameworks for *authoring* ITSs. The process of generating an ITS is inherently simpler than authoring an ITS. The ITS author will have to perform the knowledge acquisition necessary to develop the expert system, and the architecture performs much of the authoring process: it generates the tutoring content, instructional strategies, and user interface. To generate an ITS using Tahuti, a domain expert basically needs to perform two steps. The first step is to represent the domain knowledge in an expert system using the GT toolset. The second step is to define the topics covered by the curriculum.

Two tutoring systems were generated using the Tahuti architecture as proof of principle: the Excel Tutor and the Composite Materials Tutor, which were described in section 6.3. The following discussion describes the process needed to generate these two systems.

The Excel Tutor was generated for the domain of the Excel what-if tools, and covers the use of six tools. The expert system for the Excel tools did not exist previously, and was developed using the GT Toolset. The expert system is a hierarchical classification-based system; structurally, it is composed of 11 specialists. Representing the domain as a classification structure, and developing the corresponding expert system took approximately two hours. Constructing the extended knowledge file was a relatively straightforward task, and was completed within an hour. After approximately three hours of development time, the Excel Tutor had been generated and was ready to use. Since we were

already familiar with the GT Toolset and approach, the development effort was very easy for us. Someone not familiar with it will likely need to spend some time at the beginning to learn about the GT Toolset.

The Composite Materials Tutor was generated for the domain of composite materials fabrication technologies and covers sixteen different fabrication technologies. The expert system for classifying the technologies, COFATE2 had already been developed by a domain expert (Moy, McDowell et al., 1994). The expert system is a hierarchical classification-based system and is composed of 29 specialists. The only tasks remaining were: (1) to extend the expert system by defining cases to be used as examples and practiced questions, and (2) to construct the extended knowledge file. These tasks took two hours to complete, after which the Composite Materials Tutor was generated and ready to use.

7.2 Evaluating the ITSs Generated

To determine the usefulness of the Tahuti architecture, it is necessary to assess the usefulness of its end products, namely the ITSs generated by the architecture. Evaluating the intelligent tutoring systems involves judging their instructional effectiveness.

One of the ITSs generated using Tahuti, the Excel Tutor, was chosen for analysis through pilot studies in an undergraduate engineering course at Michigan State University. This tutor was selected because the domain it covers, the Excel 'what-if' tools, is part of the curriculum for CSE 131: Introduction to Technical Computing. This allows the tutor to be integrated into the course

curriculum, which provides a realistic test-bed for evaluating the instructional effectiveness of the ITS.

7.2.1 Pilot Study

7.2.1.1 Setup of the Study

A pilot study of the Excel Tutor was performed during the 2001 summer semester at MSU. Students enrolled in CSE 131: Introduction to Technical Computing were asked to participate in the study. CSE 131 students are mostly freshman or sophomore engineering students. The summer offering of the course has two sections. Participants in section 1 were considered as group A, and participants in section 2 as group B. The study was designed so that the two groups would be of approximately the same size. However, due to enrollment changes, the size of group A turned out to be 20 students, while the size of group B was 7 students. The low and unequal sample size put the experiment into the class of pilot studies.

The data collection took place in the context of a computer-equipped instructional lab, the same lab that students use for doing course-related lab exercises. The evaluation procedure followed a crossover design, which is depicted in figure 43. Both groups received classroom and lab instruction on the topic of problem solving with the Excel tools. Group A then used the Excel Tutor for approximately 30 minutes, while group B worked on lab exercises on the same topic for the same amount of time. Both groups then took the same 15-minute test on their knowledge of the topic.

The groups then alternated instruction type: group B used the Excel Tutor, while group A worked on lab exercises. Both groups then took a second 15-minute test on their knowledge of the topic. All participants were then asked to complete a short questionnaire on their experience with the tutoring system.

Test 1 and test 2 were generated to be of the same skill level and format. Both tests include 10 multiple choice questions in random order that cover the Excel tools. The lab exercises consisted of students working individually on problems, with some assistance from the teaching assistant. The details of the pilot study, including procedures, instruments, and samples of data collected are given in Appendix B.

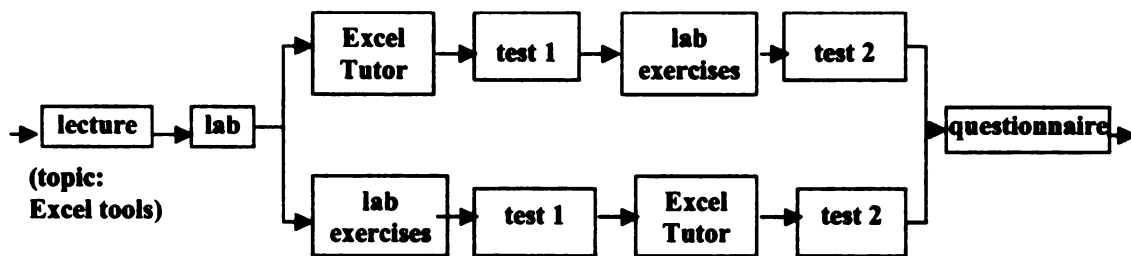


Figure 43. Flowchart for evaluation plan done in the pilot study

7.2.1.2 Results of the Study

Data was collected from four sources: (1) test 1 scores, (2) test 2 scores, (3) questionnaires, and (4) log files generated from the tutor.

The test scores from the pilot study could not be used to derive any statistical results. The results of the test scores for groups A and B are given in Appendix B.

The responses on the questionnaires filled out by the students were mostly positive. Most students indicated that they required a minimum or

reasonable amount of time to learn about the tutoring system. More than half of the students said that they learned new knowledge about the Excel tools from using the Excel Tutor and that the tutor was a helpful addition to lecture and lab instruction.

An examination of the students' log files generated by the Excel Tutor indicated that most students covered all the curriculum topics and benefited from the feedback and adaptiveness provided by the tutor. The log files showed that the students' scores on the practice questions asked by the tutor improved as they received guidance and feedback from the tutor. A sample log file is shown in Appendix B.

Although we could not derive any statistical analysis results from the pilot study, the study shed light on how to evaluate our approach and its potential. An interesting issue that it raised is the ability of tutoring systems to help weak students improve their performance. We expect that the results of an experimental study would show that the tutor could help weak students considerably.

7.2.2 Lessons Learned

The pilot study falls under exploratory evaluation methods, and, thus, cannot be used to make formal claims about the instructional effectiveness of the ITS. This stems from several characteristics of the nature of the study. First, the population size was very small, with a total of 27 students in both groups. The two groups also turned out to be very different in size, with 20 students in group A and only 7 students in group B. Moreover, certain measures should have been taken to

control external factors that could affect the pilot study's results. For example, the lab exercises component of the study was designed to let the students work on solving problems independently. In the implementation of the study, some students sought help from the teaching assistant in solving the lab exercises. This could have caused the students' baseline scores to be higher than expected.

Stronger claims about the Excel Tutor's instructional effectiveness can be made using an experimental evaluation study. Such experimental research requires random assignment of participants to conditions and statistically significant groups. Although the pilot study did not facilitate such circumstances, it did indicate the tutoring system's promising potential for improving learning outcomes. It also paves the way for the design of an experimental research study to further test the ITS's instructional effectiveness.

Chapter 8. Conclusions

This chapter discusses the major contributions of this research work. The central contribution is the development of an architecture that enables the generation of tutoring systems for a wide range of domains. Explicitly, provided the target domain for learning can be represented as a task-specific expert system of the hierarchical classification type, a tutoring system may be generated for the domain represented by that system.

This dissertation addresses three different research and development communities: the knowledge-based systems field, the intelligent tutoring systems field, and the education field. The following three sections discuss the contributions of this research to each of those fields, with respect to the objectives defined in chapter 4. The research contributions are summarized in section 8.4. Section 8.5 discusses research paths opened by this work, and section 8.6 presents some concluding remarks.

8.1 Contributions to the KBS Field

The main deliverable of this research is an architecture for the generation of intelligent tutoring systems. This architecture is an integration of a task-specific ITS shell and generic task-based expert systems.

The research emphasis from the KBS viewpoint has been on the interaction of the task-specific ITS shell with a GT expert system for generating tutoring systems, and on the resulting knowledge linkage issues. The ITS shell

uses problem solving knowledge about a classification task structure and domain knowledge embedded within the expert system. This allows the reuse of the ITS shell for different domains. This also implies that the generated ITS's tutoring capabilities are dependent on the underlying problem solving capability of the expert system. The architecture exploits the underlying expert system's domain knowledge and problem solving strategy for generating effective tutoring.

The main research contribution to the KBS field is the extension of the task-specific framework to support tutoring through the reuse of the knowledge embedded within generic task-based expert systems, which was detailed in chapter 5. The concept of reusing knowledge embedded within knowledge-based systems is an important research issue, and can facilitate the reuse of knowledge for other purposes, such as tutoring. Yet, effective knowledge reuse is often a difficult task, and has been rarely achieved.

Our approach to extending the task-specific framework has several appealing features for ITS generation. First, the large-grain approach of the GT methodology allows the reuse of high-level problem solving concepts and strategies for tutoring support; it allows the tutoring focus to be on problem solving *tasks*, rather than on lower level knowledge constructs used by small-grain approaches. Second, the underlying knowledge representation and reasoning method of a GT system provides a semantically rich foundation for the extraction of tutoring knowledge. Moreover, the architecture employs a task-specific ITS shell that is reusable as-is for different task-specific problem solving systems.

8.2 Contributions to the ITS Field

This research also addresses several fundamental issues in the intelligent tutoring systems area. The major contribution of this research is the development of a task-specific methodology and architecture for generating ITSs. The development of this architecture has important findings for the ITS authoring community.

An important contribution is the use of an ITS shell as an overlay on existing expert systems. The knowledge representation used by the expert system directly affects the effectiveness of the tutor generated. A generic task expert system uses a semantically rich knowledge representation. The architecture extracts this knowledge and effectively reuses it for tutoring purposes.

Another important contribution is the architecture's success as a reusable ITS shell. The key component of the ITS shell that allows reuse is the expert model, which interfaces to a GT-based expert system to extract tutoring knowledge. The architecture allows the other ITS components, namely the student model, instructional manager, and user interface, to be reused as-is for different classification-based problem solving domains.

The architecture developed also supports rapid prototyping, allowing authors to quickly design, implement, and test a tutoring system. The author needs to represent the domain knowledge in an expert system and define the curriculum topics, after which the ITS is ready to be tested. In this manner, the

architecture can be used as a design and evaluation tool for developing learning environments.

The methodology developed has a strong ontological foundation, in the GT framework overall, and in the vocabulary of problem solving in an individual expert system. The architecture explicitly models the expert system's knowledge as domain and problem solving knowledge, and reuses it as tutoring knowledge. One question that was posed earlier in this dissertation is whether or not any additional domain knowledge is needed to produce an effective ITS other than that which is already represented in a GT-based expert system. The answer to this question is yes; some additional knowledge is needed for generating effective tutoring. This knowledge comes from the extended knowledge module, and consists of: (1) high-level descriptive knowledge of the problem solving, and (2) pedagogical knowledge, such as curriculum topics. These two knowledge types are not needed by the expert system for problem solving, but are required for the expert system's knowledge to be used for tutoring.

The adopted approach separates the instructional content from the instructional strategies that will be used to teach it. The instructional strategies are housed in the ITS shell, while all the tutoring content is derived from the expert system and extended knowledge module. This allows the same content to be used in different parts of the tutorial. For example, a modular unit of tutoring knowledge can be used as part of the tutoring system's overview, and also as an example later on. Moreover, by separating instructional content from instructional strategies, the architecture can provide multiple and abstracted views of the

tutoring content. For example, a unit of tutoring knowledge can be presented as a hint or an explanation, depending on the instructional objective.

Another important contribution of this research is the student modeling approach employed by the architecture. The integration of the ITS shell with an expert system allows the expert system to be used as a *model of domain expertise* against which the tutor can compare a student's performance and guide the tutoring accordingly. In addition, this approach not only allows the tutor to explain a student's performance, by tracing the expert system's problem solving path, but also allows the tutor to give the learner fine-grained feedback and explanations.

8.3 Educational Contributions

The Tahuti architecture allows authors to easily generate intelligent tutoring systems that can be used as instructional aides to provide effective instruction and improve learning outcomes. The architecture addresses several fundamental educational issues.

The architecture facilitates the generation of pedagogically sound learning environments as defined in the educational community. The educational findings in the literature summarized in section 2.4 reported that, for people to become competent in a domain, they need to understand the domain knowledge within the context of a conceptual framework, and organize that knowledge in a manner that facilitates its use. The ITSs generated provide a classification structure for learners to use as a conceptual structure in the learning process. This structure allows learners to easily access relevant knowledge and transform factual

information into usable problem solving knowledge. Moreover, by developing a conceptual framework, learners are able to apply their knowledge in new situations and to learn related information more quickly.

In addition, instructional strategies that are appropriate for tutoring task-specific problem solving domains have been identified and used in the ITS shell. These strategies include example-based learning and learning by doing. Moreover, the architecture guides ITS authors in generating effective learning environments. For example, the author specifies the contents and order of the curriculum topics and the system determines the appropriate instructional technique for those topics.

Another important feature of the architecture is that it is relatively easy to use. If an expert system for the target domain is available, an author can generate a tutoring system in three straightforward steps. If an expert system for the target domain is not available, an ITS author can develop a system using the Integrated Generic Task Toolset with a minimal amount of training and computer programming skills. The architecture employs a simple user interface and development process that teachers, instructional designers, or domain experts can use to generate ITSs in a straightforward way.

The architecture's instructional strategies employ a learner-centered approach to education. The tutoring systems generated follow a curriculum that covers all the topics, but also allow students to navigate to the topics they want to learn about. In solving practice questions, students can ask for hints, second chances, or more examples, etc. The use of a runnable deep model of

instructional content, and its separation from the instructional strategies facilitates such a learner-centered approach.

By modeling different learning conditions, the Tahuti architecture can be used by ITS researchers and developers as a tool for evaluating different learning situations. For example, varying the learner's knowledge state, while keeping the rest of the environment constant, can provide insights on novice-expert shifts in learning tasks. As another example, results of modeling a learning environment for different instructional strategies could lead to an understanding of which instructional technique is more suited for a certain learning task, or a certain type of student.

8.4 Summary of Contributions

Table 4 summarizes the research contributions of this dissertation, listed by field.

Table 2. Summary of research contributions

KBS Contributions
Extension of the task-specific framework to support the generation of ITSs
Reuse of the domain and problem solving knowledge embedded within GT systems
ITS Contributions
Generation of ITSs as an overlay on expert systems
Reuse of ITS components (student model, instructional manager, and user interface)
Support for rapid prototyping
Use of multiple and reusable views of the instructional content, which are separate from the instructional strategies
Utilization of a simple yet powerful student modeling technique

Education Contributions
Generation of pedagogically sound learning environments
Development of an easy to use ITS authoring framework
Use of a learner-centered approach to education
Use as an evaluation tool for different learning situations

8.5 Future Directions

This research work opens up several new research paths as future extensions:

- Use the Tahuti architecture to generate tutoring systems for other domains, and test those systems in classroom and other instructional settings. We have generated tutoring systems for two test-bed domains. Generating more ITSs and evaluating the use of those systems as learning aides will allow draw further conclusions about the usefulness of our approach.
- Extend the architecture to generate ITSs for domains represented as other types of generic tasks or as integrated generic tasks. The architecture supports the generation of tutoring systems for task-based expert systems of the hierarchical classification type. Extending the architecture to support other types of generic task expert systems, such as routine design or functional modeling, or expert systems that are represented as an integrated set of problem solvers will allow ITSs to be generated for a wider range of domains and uses.
- Extend the architecture to include a user interface shell that would allow ITS authors to augment the domain knowledge provided by the expert system with graphics or visual knowledge that can enhance the tutoring. The domain

content displayed through the architecture's user interface is extracted from the expert system, and is mostly text-based. The ability to augment that content with other resources, such as pictures, animations, video clips, etc., will enhance the architecture's usability.

- Extend the architecture to provide support for user-defined instructional strategies. The architecture currently supports instructional strategies that are appropriate for tutoring task-specific problem solving domains. Allowing ITS authors to define other instructional strategies would facilitate the generation of more flexible tutoring systems that can tutor using a wider variety of instructional techniques.
- Enable the use of the architecture as an evaluation tool for different learning environments. In this manner, the architecture could serve as a powerful tool for the design and evaluation of different learning environments, and as an important component of a virtual laboratory for experimentation with new learning approaches and technologies.

8.6 Concluding Remarks

In this dissertation, a novel approach for generating intelligent tutoring systems for a wide range of domains has been presented. The architecture developed has many desirable features of an ITS authoring environment. Among other features, it employs a runnable deep model of domain expertise, facilitates fine-grained student diagnosis, and offers an easy method for building expert systems and generating ITSs from them.

In the last two decades, the ITS research and development community has made considerable progress towards understanding and solving the key issues involved in developing ITS authoring tools. The development efforts have used diverse approaches, some of which have demonstrated significant success in limited cases. However, more work is needed to take ITS authoring tools from the level of research vehicles to a level of wide-spread use and commercialization.

To achieve such wide-spread use, the ITS community should be committed to creating collaborative research and development efforts that will allow ITS researchers to work together on developing better ITS authoring methodologies and tools. The IEEE Learning Technology Standards Committee (<http://grouper.ieee.org/p1484/>) is one step in that direction. This international consortium is collaborating to develop technical standards and recommended practices for the development and deployment of learning environments. The efforts of this or similar committees aimed at ITS authoring tools will give the community greater momentum towards achieving the goal of making such tools more efficient and widespread.

Finally, it is important to note that the demand for ITS authoring tools stems from the increasing demand for intelligent tutoring systems that are educationally-effective and cost-effective. From this perspective, the two main objectives of ITS authoring tools are: (1) to reduce the cost involved in building ITSs, and (2) to facilitate the development of enough systems that will confirm the educational effectiveness of ITSs.

This research has addressed both these objectives. The methodology and architecture developed facilitates the easy and fast generation of instructionally effective ITSs. This research has also addressed the fundamental issues involved in the development and success of ITS authoring tools. More fundamentally, it has helped bridge the gap between the artificial intelligence and educational fields by integrating effective instructional techniques with the adaptiveness and robustness of knowledge-based systems to generate useful learning environments.

APPENDIX A. Representation of the Excel Tools Expert System

This appendix gives a complete representation of the Excel tools expert system's knowledge and control structure, which was discussed in section 6.2.1. The information presented below was automatically generated by the expert system, and has been re-formatted for the purpose of this presentation.

```
***** DATA BASE *****
-----GtOrdinalOrderVar-----
```

```
*****number of data table inputs
```

LEGAL VALUES:

one
two
unknown

COMMENT:

Possible values:

one: One input value is varied in a formula, as either row or column output.
two: Two input values are varied, one along the rows, and one along the columns.

QUESTION:

How many inputs are to be varied in a data table?

```
*****number of input values
```

LEGAL VALUES:

single
multiple
range
constrained
unknown

COMMENT:

Possible values:

single: An input variable changes to just one value.
multiple: An input variable changes to more than one value.
range: An input variable changes to a range of values, such as from 1.0 to 7.5.
constrained: An input variable must conform to specific constraints.

QUESTION:

How many values can an input variable have?

single: An input variable can change to just one value.

multiple: An input variable can have multiple values.
range: An input variable can have a range of values.
constrained: An input variable must conform to specific constraints.

*****number of scenarios

LEGAL VALUES:

one
more than one
unknown

COMMENT:

Possible values:
one: Answer a single "what-if" question about the data.
more than one: Answer multiple questions about the data.

QUESTION:

How many data analysis scenarios would you like to perform? In other words, how many questions would you like to answer about the data?

*****output format

LEGAL VALUES:

table
report
unknown

COMMENT:

Possible values:
table: The output is generated in a table format.
report: The output is generated in a report format.

QUESTION:

Would you like to view the output in a table format or a report format?

*****problem solving direction

LEGAL VALUES:

backward
forward
unknown

COMMENT:

Possible values:
backward: You specify a solution and find the appropriate input values.
forward: You specify the input values and find the solution.

QUESTION:

What direction does problem solving proceed in?
backward: You specify a solution and find the appropriate input values.
forward: You specify the input values and find the solution.

***** KNOWLEDGE HIERARCHY *****

```

---- <1> ExcelTools 1
----- <2> backward problem solving 2
----- <3> goal seek 3
----- <3> solver and solver report 4
----- <2> forward problem solving 5
----- <3> data table 6
----- <4> one input data table 7
----- <4> two input data table 8
----- <3> what-if analysis 9
----- <4> scenario manager and summary 10
----- <4> simple what-if analysis 11

```

***** KNOWLEDGE REPRESENTATION *****

----- 1-----

ExcelTools

HIERARCHY OF TABLES:

```

---- <1> ExcelTools.spec.sm.ExcelTools.tt

```

TABLES:

ExcelTools.spec.sm.ExcelTools.tt

problem solving direction	result
?	strongly match **

----- 2-----

backward problem solving

HIERARCHY OF TABLES:

```

---- <1> backward problem solving.sm.ExcelTools.tt

```

TABLES:

backward problem solving.sm.ExcelTools.tt

problem solving direction	result
= backward [F]	strongly match
= forward [T]	strongly against **
= unknown	against

----- 3-----

goal seek

HIERARCHY OF TABLES:

---- <1> goal seek.sm.ExcelTools.tt

TABLES:

goal seek.sm.ExcelTools.tt

no. of input values	problem solving dir.	result
= single [F]	= backward	strongly match
= multiple [T]	= backward [T]	match **
= unknown	= backward	weakly match
?	= backward	against
?	~= backward	strongly against

----- 4-----

solver and solver report

HIERARCHY OF TABLES:

---- <1> solver and solver report.sm.tt

TABLES:

solver and solver report.sm.tt

no. of input values	problem solving dir.	result
= multiple [T]	= backward [T]	strongly match **
= constrained	= backward	strongly match
= range	= backward	match
= constrained	?	weakly match
?	~= backward	strongly against
?	= backward	against

----- 5-----

forward problem solving

HIERARCHY OF TABLES:

---- <1> forward problem solving.sm.ExcelTools.tt

TABLES:

forward problem solving.sm.ExcelTools.tt

problem solving direction	result
= forward [T]	strongly match **
= backward	strongly against
= unknown	match

----- 6-----

data table

HIERARCHY OF TABLES:

---- <1> data table.sm.ExcelTools.tt

TABLES:

data table.sm.ExcelTools.tt

input values	ps direction	output format	table inputs	result
= multiple [F]	= forward	= table	?	strongly match
= range [T]	= forward[T]	= table [T]	?	strongly match
= multiple	= forward	= table	= one	match
= multiple	= forward	~= report	= two	match
?	= forward	?	?	against
?	~= forward	?	?	strongly against

----- 7-----

one input data table

HIERARCHY OF TABLES:

---- <1> one input data table.sm.ExcelTools.tt

TABLES:

one input data table.sm.ExcelTools.tt

number of data table inputs	result
= one [F]	strongly match
= unknown [F]	match
= two [T]	strongly against **

----- 8-----

two input data table

HIERARCHY OF TABLES:

---- <1> two input data table.sm.ExcelTools.tt

TABLES:

two input data table.sm.ExcelTools.tt

number of data table inputs	result
= two [T]	strongly match **
?	against

----- 9-----

what-if analysis

HIERARCHY OF TABLES:

---- <1> what-if analysis.sm.ExcelTools.tt

TABLES:

what-if analysis.sm.ExcelTools.tt

input values	ps direction	output format	table inputs	result
= single [F]	= forward	= report	?	strongly match
= multiple[F]	= forward	= report	?	match
= unknown [F]	= forward	= report	?	weakly match
?	= forward [T]	~= table [F]	~= two	weakly match
?	= unknown [F]	~= table	?	match
?	= backward[F]	?	?	strongly against
?	?	= table [T]	?	strongly against

----- 10-----

scenario manager and summary

HIERARCHY OF TABLES:

---- <1> scenario manager and summary.sm.ExcelTools.tt

TABLES:

scenario manager and summary.sm.ExcelTools.tt

number of scenarios	result
= more than one [T]	strongly match **
?	against

----- 11-----

simple what-if analysis

HIERARCHY OF TABLES:

---- <1> simple what-if analysis.sm.ExcelTools.tt

TABLES:

simple what-if analysis.sm.ExcelTools.tt

number of scenarios	result
= one [F]	strongly match
= unknown [F]	match
?	against **

APPENDIX B. Details of the Pilot Study

This appendix includes the procedures, instruments, and samples of data collected in the pilot study, which was described in section 7.2.1.

Data Collection Procedure

1. Data collection will be during one week of the summer semester and take place in the computer lab designated for CSE 131, during their allotted lab sessions. The course has two sections. Participants in section 1 will be considered as group A, and participants in section 2 as group B.
2. The secondary investigator will meet the participants in the computer lab at the start of the first lab session and will explain the purpose and details of the study.
3. At the start of the lab session, participants will be administered the written consent form. Any questions will be answered to make sure that they understand the procedure and they are interested in participating. Steps 2 and 3 will be done in the first 30 minutes of the lab session.
4. Group A will then use the tutoring system for approximately 60 minutes. That includes time to receive instruction about how to use the system. Group B will work on lab exercises related to the topic.
5. In the final 30 minutes of the lab session, both groups will take the same computer-based test (test 1) on the topic of problem solving with Excel.

6. In the next lab session (two days later), the secondary investigator will again meet the participants in the computer lab.
7. Group B will then use the tutoring system for approximately 60 minutes, and group A will work on the lab exercises for the same amount of time.
8. Both groups will then take the same computer-based test (test 2) on the topic of problem solving with Excel.
9. After participants have completed the test, they will be asked to complete an anonymous written questionnaire about their experience and opinion of the tutoring system.
10. Finally, participants will be informed that if they have any questions or concerns about the study, or they would like to have a copy of the results of the study, they can contact the researcher at the e-mail address and phone number given to them in the consent form.

Procedures for the Protection of Human Subjects

Following procedures laid out by the University Committee on Research Involving Human Subjects (UCRIHS) at MSU, the following measures will be taken to protect the rights of the participants as human subjects:

- Participants will use an identification number during data collection. This information will be held strictly confidential. This information is necessary because the tests and tutoring system will be incorporated into the class curriculum. So the course instructor will need to identify students for recording

their test grades as part of the course requirements. The identification numbers will be used only for this purpose.

- By signing the consent form, the participants acknowledge that they are willing to participate in the study and thus give their permission for the researchers to use the data collected. Data stored and analyzed for the research study will discard identifiers and consist of only anonymous data.
- All data reporting will be done only in aggregate form.
- Participation is voluntarily and refusal to participate or discontinuing their participation at any time during the study will have no effect on their course evaluation.
- They are informed that they can contact and consult the researchers or UCRIHS Chair if they have any questions.

Instruments and Documents Used in the Pilot Study

- Consent form
- Tutoring system's user's guide
- Test 1
- Test 2
- Problem set for lab exercises
- Questionnaire
- Sample log file

Informed Consent Form For A Project on the Impact of Using An Intelligent Tutoring System in CSE 131

We are interested in understanding the impact of using an intelligent tutoring system as an instructional aide in college-level courses. As part of your class requirements this semester, you will receive lectures, lab instruction, and use a tutoring system on problem solving with Excel. You will also take two short tests on the topic of problem solving with Excel. The study will use data from three sources: (1) data from your use of the tutoring system, (2) data from your test results, and (3) data from a short questionnaire about your experience with the tutoring system. The questionnaire will take approximately 5 minutes of your time. The data collected will be analyzed to generate research results about the effectiveness of the tutoring system as an instructional aide and also about the integration of such tutoring systems into course curriculums.

By consenting to participate in this study, you agree to allow us to use your data for generating research results. Note that using the tutoring system and taking two tests on the topic of problem solving with Excel are part of your course requirements, and participation in those activities are obligatory. Consenting to participate in the study indicates that you permit us to collect the data generated from your use of the tutoring system and test results, with an understanding that all data stored for the research study will be anonymous.

Participation is voluntary. You may choose not to participate and you may withdraw your participation at any time, including withdrawal of any information you have provided. All data collection will be strictly confidential and all data storage and analysis will be anonymous. Your name will **not** be used in any report of the research findings. Your privacy will be protected to the maximum extent allowable by law.

If you have any questions regarding your participation in this study, please feel free to contact Jon Sticklen (tel: 353-3711 or email: sticklen@cse.msu.edu). If you have any questions regarding participants' rights as human subjects of research, please contact David E. Wright (tel: 355-2180), Chair of the University Committee on Research Involving Human Subjects.

Thank you for helping us improve course instruction through the use of intelligent educational software. We ask that you sign and date this form indicating that you have been informed of the procedures of this research study and consent to participating in it.

Name: _____
Signature: _____
Date: _____

Tutoring System for Excel Tools

User's Guide

Starting the tutoring system

1. Insert the floppy disk into drive A. This disk will be used during the tutorial session.
2. Insert the tutoring system CD into the CD drive and open the 'CSE 131' folder.
3. Double click on the file called 'start.im'. A window labeled 'Open With' will appear that asks for the program to open this file with. Select 'Other...' from the options, and in the window that comes up, locate and select the file 'visual.exe' in the 'vw' folder on the CD.
4. Click on 'Ok' to start the tutoring system.

Using the tutoring system

This program is designed to help you identify which of the Excel what-if tools to use for a given problem you want to solve. The task of deciding which what-if tool to use is a problem that, in the real world, is often the most difficult part of using the what-if tool.

The tutoring system will present an overview of what you are going to learn, including a description of the input variables involved in the task of deciding which what-if tool to use, and of the Excel what-if tools themselves. The tutor will then give you examples that show the appropriate data conditions for

using each of the Excel tools. The format of the examples is explained below. Then you will be asked to solve some practice questions. Try to finish all the practice questions until the tutoring system tells you that you are done.

Press the "Continue" button, located in the bottom left corner, to continue the tutorial session after completing each screen. Please follow the tutoring system's curriculum, but you can use the "Go To..." menu in the menu bar on the top to jump back to a specific section of the tutorial. If you need help at any time, select "Help" from the "Tutor" menu.

A tutoring example

Consider the following example: 'If you want to answer a single 'what-if' type question, which Excel tool would you use?' Questions involving such decision conditions can be answered using what-if analysis, and thus the correct Excel tool to use is the simple what-if analysis tool. This is a problem solving approach to learning about the Excel tools. It involves looking at different scenarios for how the data needs to be analyzed and determining the appropriate tool to use for each scenario. Note that the objective is to learn about *why and when* to use each tool, rather than how to use them.

Figure 1 shows an example from the tutoring system. The curriculum topics view on the left displays the curriculum and current topic. The main window on the right shows an example. The first part of the example shows the values for the input variables. There are five input variables involved in determining which tool to use. These input variables are defined in the tutoring

system. The next part of the example shows the correct tool to use for the input conditions and the explanation for selecting this tool.

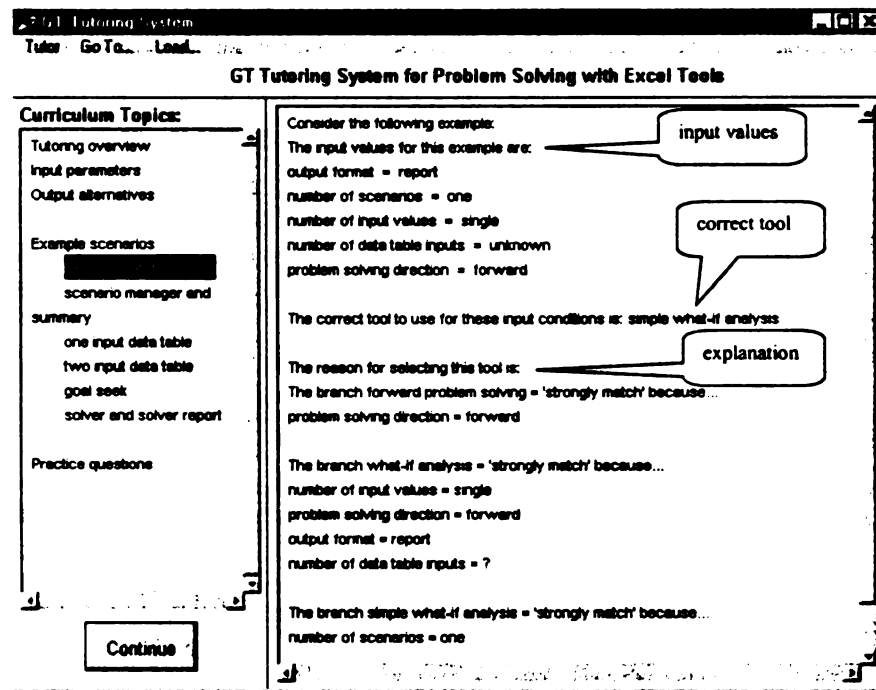


Figure 44. Fig. 1. An example from the tutoring system

The explanation is based on a classification approach to problem solving. The classification hierarchy for the Excel tools is shown in figure 2. The root node is the set of Excel tools and each of the leaf nodes represents an individual Excel tool. At each branch, the data conditions related to that branch are analyzed. If the input values match the data conditions for that branch, we say that the branch 'matched.' A branch's result can be one of the following values: 'strongly match', 'match', 'weakly match', 'strongly against', 'against', or 'weakly against' depending on how well the input values correspond to the branch's data conditions. The matching branch is added to the classification path and that branch's child nodes are then evaluated to see if they match as well. Determining

the appropriate Excel tool to use for a set of data conditions involves starting at the root node and trying to find the correct path down the hierarchy that matches the data conditions. The final leaf node that matched will be the correct Excel tool to use for the given data conditions. The explanation in figure 1 explains the path of nodes that matched the data conditions for that example. For each branch that matched, the tutoring system explains the input conditions that caused it to match. The final branch shows the output result, namely that the simple what-if analysis tool was selected as the appropriate tool to use for the given data conditions. Figure 3 shows the classification hierarchy for Excel tools, with the matching path for the example in figure 1.

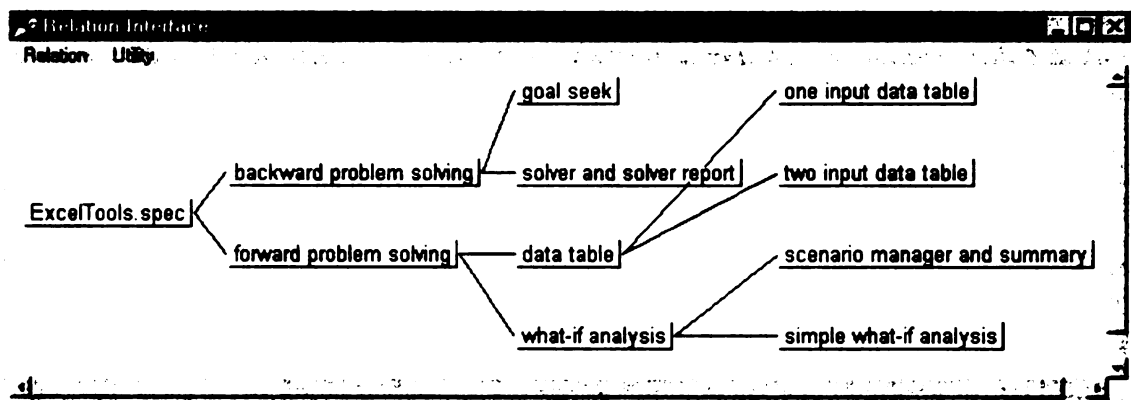


Figure 45. Fig. 2. Classification hierarchy for Excel tools

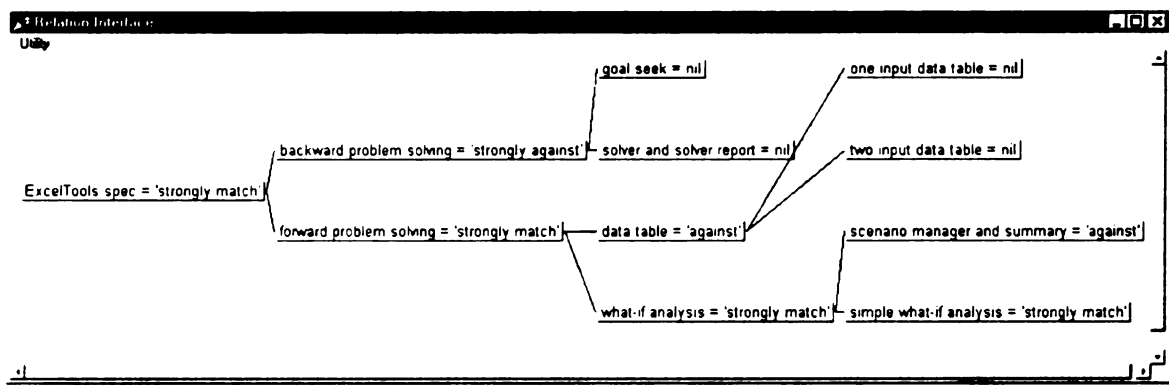


Figure 46. Fig. 3. Classification hierarchy showing a path that matched

Exiting the tutoring system

1. Close the main tutoring system window. This step is very important!
2. Find the window labeled 'VisualWorks'.
3. From the File menu, select the 'Exit VisualWorks...' option. In the dialog box that appears, select Exit.
4. Remove the CD and floppy disk and return them to the instructor.

CSE 131 - Summer 2001
Quiz # 1 on Excel Tools

Please select only one answer for each question below. For each question, indicate the **name** of the appropriate Excel tool that you would use for the given data conditions. The possible answers are:

- simple what-if analysis
- scenario manager and summary
- one input data table
- two input data table
- goal seek
- solver and solver report

1. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = single
number of data table inputs = unknown
output format = report
number of scenarios = one
problem solving direction = forward

Appropriate Excel tool to use: _____

2. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = range
number of data table inputs = one
output format = table
number of scenarios = unknown
problem solving direction = forward

Appropriate Excel tool to use: _____

3. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = constrained
number of data table inputs = unknown
output format = report
number of scenarios = more than one
problem solving direction = backward

Appropriate Excel tool to use: _____

4. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = multiple
number of data table inputs = unknown
output format = report
number of scenarios = more than one
problem solving direction = forward

Appropriate Excel tool to use: _____

5. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = single
number of data table inputs = unknown
output format = unknown
number of scenarios = one
problem solving direction = backward

Appropriate Excel tool to use: _____

6. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = range
number of data table inputs = two
output format = table
number of scenarios = unknown
problem solving direction = forward

Appropriate Excel tool to use: _____

7. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = range
number of data table inputs = unknown
output format = report
number of scenarios = more than one
problem solving direction = forward

Appropriate Excel tool to use: _____

8. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = range
number of data table inputs = one
output format = table
number of scenarios = more than one
problem solving direction = forward

Appropriate Excel tool to use: _____

9. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = single
number of data table inputs = unknown
output format = unknown
number of scenarios = one
problem solving direction = forward

Appropriate Excel tool to use: _____

10. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = single
number of data table inputs = unknown
output format = table
number of scenarios = one
problem solving direction = backward

Appropriate Excel tool to use: _____

CSE 131 - Summer 2001
Quiz # 2 on Excel Tools

Please select only one answer for each question below. For each question, indicate the **name** of the appropriate Excel tool that you would use for the given data conditions. The possible answers are:

- simple what-if analysis
- scenario manager and summary
- one input data table
- two input data table
- goal seek
- solver and solver report

1. Given the following data conditions, which Excel tool would you use to perform the data analysis?
- number of input values = multiple
 - number of data table inputs = two
 - output format = table
 - number of scenarios = more than one
 - problem solving direction = forward

Appropriate Excel tool to use: _____

2. Given the following data conditions, which Excel tool would you use to perform the data analysis?
- number of input values = range
 - number of data table inputs = unknown
 - output format = report
 - number of scenarios = more than one
 - problem solving direction = forward

Appropriate Excel tool to use: _____

3. Given the following data conditions, which Excel tool would you use to perform the data analysis?
- number of input values = multiple
 - number of data table inputs = unknown
 - output format = report
 - number of scenarios = more than one
 - problem solving direction = backward

Appropriate Excel tool to use: _____

4. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = single
number of data table inputs = unknown
output format = unknown
number of scenarios = one
problem solving direction = forward

Appropriate Excel tool to use: _____

5. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = single
number of data table inputs = unknown
output format = table
number of scenarios = one
problem solving direction = backward

Appropriate Excel tool to use: _____

6. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = range
number of data table inputs = one
output format = table
number of scenarios = more than one
problem solving direction = forward

Appropriate Excel tool to use: _____

7. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = range
number of data table inputs = two
output format = table
number of scenarios = unknown
problem solving direction = forward

Appropriate Excel tool to use: _____

8. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = multiple
number of data table inputs = unknown
output format = report
number of scenarios = more than one
problem solving direction = forward

Appropriate Excel tool to use: _____

9. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = constrained
number of data table inputs = unknown
output format = report
number of scenarios = more than one
problem solving direction = backward

Appropriate Excel tool to use: _____

10. Given the following data conditions, which Excel tool would you use to perform the data analysis?

number of input values = range
number of data table inputs = one
output format = table
number of scenarios = unknown
problem solving direction = forward

Appropriate Excel tool to use: _____

Exercises on Excel Tools

To solve the following exercises, you will use a problem solving approach. These problems are designed to help you identify which Excel tool to use for data analysis for a given problem you want to solve.

Consider an example: "If you want to answer questions about data that changes to multiple values in more than one scenario, and you would like the output in a report format, which Excel tool would you use?" To answer questions involving such data conditions, the correct Excel tool to use would be the scenario manager and summary tool. This approach involves looking at different scenarios for how the data needs to be analyzed and determining the appropriate tool to use for each scenario. Note that the objective is to learn about why and when to use each tool, rather than how to use them.

We can decide which Excel tool to use for data analysis based on five parameters:

- Number of input values
 - This specifies the number of values that a data variable can have. A data variable can have a single value, multiple values, a range of values, or constrained values.
- Number of scenarios
 - This specifies the number of data analysis scenarios that the user would like to perform.
- Number of data table inputs
 - This specifies the number of inputs for a data table, either one or two.
- Output format
 - This specifies the type of output generated from the data the data analysis, either in report or table format.
- Problem solving direction
 - This specifies if the problem solving is forward (find a solution given the inputs) or backward (find the appropriate input values for a given solution).

The available Excel what-if tools are:

- Simple what-if analysis
- Scenario manager and summary
- One input data table
- Two input data table
- Goal seek
- Solver and solver report

Consider the following data model for a car loan:

Car Loan Payment Model

Inputs

Loan Amount	\$	25,000
Annual Interest Rate		10.50%
Term in months		48

Outputs

Monthly Payment:	\$	640.08
Total Payments:	\$	30,724.06
Total Interest:	\$	5,724.06

- If you would like to know how the monthly car payments would change as interest rates increase by increments of 0.25%, which Excel tool would you use?
- If you would like to know what happens to the monthly payments if the loan term is 5 years instead of 4 years, which Excel tool would you use?
- If wonder what your total payment will be if the interest is 10%, which Excel tool would you use?
- If you would like to generate a report summarizing the results of the above two questions, which Excel tool would you use?
- If your boss wants to know what happens if varying interests rates are applied to varying loan terms, which Excel tool would you use?
- If you wonder what would happen to the monthly payments if a less-expensive car with a lower loan amount is used, which Excel tool would you use?

- Your boss wants to know how much money the company could save by keeping the total payment amount equal to \$28,000. Which Excel tool would you use to answer his question?

Consider the following data model for a multiple car loan:

Multiple Car Loan Model

Inputs			
Annual Interest Rate	10%		
Term in months	48		
	<u>Car Type 1</u>	<u>Car Type 2</u>	<u>Car Type 3</u>
Price	\$24,000.00	\$21,000.00	\$16,000.00
Passengers	9	6	4
Quantity to Purchase	4	3	4
Outputs			
	<u>Car Type 1</u>	<u>Car Type 2</u>	<u>Car Type 3</u>
Loan Amount	\$96,000.00	\$63,000.00	\$64,000.00
Monthly Payment	\$2,434.81	\$1,597.84	\$1,623.21
Total Passengers	36	18	16
Total Loan Amounts	\$223,000.00		
Total Monthly Payments	\$5,655.86		
Total Interest Amount	\$48,481.09		

- If you are interested in Total Monthly Payments corresponding to Annual Interest Rates ranging from 5% to 20%, which Excel tool would you use?
- If you are not willing to pay a Total Interest Amount of more than \$15,000 and you are looking for a financial institution that offers a lower Annual Interest Rate, which Excel tool would you use?
- If you wonder what would be the Total Passengers if we bought 5 cars of Car Type 2 instead of 3, which Excel tool would you use?

- Your boss would like to see the differences in Price and Total Passengers for the following cases. Which Excel tool would you use to create the report he wants?
- 3 of Car Type 1, 2 of Car Type 2, 1 of Car Type 3
- 4 of Car Type 1, 1 of Car Type 2, 1 of Car Type 3

Questionnaire
Using A Tutoring System for Learning Excel Tools

Thank you very much for participating in this research project. Your feedback is crucial for furthering the use of tutoring systems in classes and the development of improved tutoring systems. We would appreciate if you could take the time to fill this questionnaire. The questionnaire is anonymous. Please circle only one answer for each question.

Answer key for questions 3 - 10:

1=strongly agree, 2=agree, 3=neutral, 4=disagree, 5= strongly disagree

1. What is your previous problem solving experience with Excel?

- (a) only lectures
- (b) lectures plus some practical work
- (c) extensive use

2. How much time did you need to learn about the tutoring system itself and its functions?

- (a) a minimum amount of time
- (b) a reasonable amount of time
- (c) most of the tutoring session

3. I learned new knowledge about problem solving with Excel tools from using the system.

1 2 3 4 5

4. I found using the tutoring system to be a helpful addition to attending the lecture.

1 2 3 4 5

5. I found using the tutoring system to be a helpful addition to lab instruction.

1 2 3 4 5

6. I would like to use the tutoring system more.

1 2 3 4 5

7. I would recommend the tutoring system to other students.

1 2 3 4 5

8. The tutoring system's user interface was easy to use.

1 2 3 4 5

Please specify the reasons:

9. The tutoring system's explanations were useful.

1 2 3 4 5

Please specify the reasons:

10. I would prefer more details in the explanations.

1 2 3 4 5

11. What do you like in particular about the tutoring system?

12. How can the user interface be improved for you?

13. How can the tutoring system be improved for you?

14. Did you encounter any software problems or crashes?

a) yes b) no

If yes, please specify:

Thank you for completing this questionnaire!

Test Scores from the Pilot Study

Table 3. Test scores for group A

Student #	Test 1	Test 2
A1	8	7
A2	10	9
A3	10	10
A4	9	10
A5	9	9
A6	6	10
A7	10	9
A8	10	10
A9	10	10
A10	9	9
A11	10	10
A12	10	10
A13	7	8
A14	10	10
A15	6	9
A16	7	9
A17	5	6
A18	10	10
A19	10	10
A20	6	8

Table 4. Test scores for group B

Student #	Test 1	Test 2
B1	9	9
B2	10	10
B3	9	10
B4	8	8
B5	10	10
B6	7	10
B7	10	10

Sample Log File Generated From the Excel Tutor

Topic: Overview: input

Topic: Overview: output

Topic: Present examples

Example: simple what-if analysis

Topic: Present examples

Example: scenario manager and summary

Topic: Present examples

Example: one input data table

Topic: Present examples

Example: two input data table

Topic: Present examples

Example: goal seek

Topic: Present examples

Example: solver and solver report

Topic: Present examples

Topic: Practice questions

Question: simple what-if analysis

Example: simple what-if analysis

score = 75

Topic: Practice questions

Question: simple what-if analysis

Example: case 1

score = 50

Topic: Practice questions

Question: simple what-if analysis

Example: simple what-if analysis

score = 62

Topic: Practice questions

Question: simple what-if analysis

Example: case 1

score = 81

Topic: Practice questions
Question: scenario manager and summary
Example: scenario manager and summary
score = 50

Topic: Practice questions
Question: scenario manager and summary
Example: case 2
score = 75

Topic: Practice questions
Question: one input data table
Example: one input data table
score = 100

Topic: Practice questions
Question: one input data table
Example: case 3
score = 100

Topic: Practice questions
Question: two input data table
Example: two input data table
score = 100

Topic: Practice questions
Question: two input data table
Example: case 4
score = 100

Topic: Practice questions
Question: goal seek
Example: goal seek
score = 75

Topic: Practice questions
Question: goal seek
Example: case 5
score = 87

Topic: Practice questions
Question: solver and solver report
Example: solver and solver report
score = 100

Topic: Practice questions
Question: solver and solver report
Example: case 6
score = 100

Here is a summary report on your use of the tutoring system:

Score for topic: simple what-if analysis = 81
Score for topic: scenario manager and summary = 75
Score for topic: one input data table = 100
Score for topic: two input data table = 100
Score for topic: goal seek = 87
Score for topic: solver and solver report = 100

Topic: Practice questions

BIBLIOGRAPHY

Ainsworth, S., J. Underwood, et al. (2000). Using an ITS Authoring Tool to Explore Educators' Use of Instructional

Strategies. ITS 2000: Fifth International Conference on Intelligent Tutoring Systems, Montreal, Canada, Springer-Verlag.

Anderson, J. R. (1990). Analysis of Student Performance with the LISP Tutor. Diagnostic Monitoring of Skill and Knowledge Acquisition. N. Frederiksen, R. Glaser, A. M. Lesgold and M. Shafro. Hillsdale, NJ, Lawrence Erlbaum Associates.

Anderson, J. R., C. Boyle, et al. (1985). The Geometry Tutor. Proceedings of IJCAI-85: the International Joint Conference on Artificial Intelligence, Los Angeles, CA.

Anderson, J. R., C. F. Boyle, et al. (1990). "Cognitive Modeling and Intelligent Tutoring." Artificial Intelligence 42(1): 7-49.

Anderson, J. R. and R. Pelletier (1991). A Development System for Model-Tracing Tutors. International Conference of the Learning Sciences, Charlottesville, VA, Association for the Advancement of Computing in Education.

Antao, B. A. A., A. J. Broderson, et al. (1992). "Building Intelligent Tutorial Systems for Teaching Simulation in Engineering Education." IEEE Transactions on Education 35(1): 50-56.

Bell, B. (1998). "Investigate and Decide Learning Environments: Specializing Task Models for Authoring Tool Design." The Journal of the Learning Sciences 7(1): 65-105.

Bell, B. (1999). "Supporting educational software design with knowledge-rich tools." International Journal of Artificial Intelligence in Education 10: 46-74.

Bell, B. and C. Redfield (1998). "Guest Editors' Introduction, Special Issue on Authoring Tools for Intelligent Tutoring Systems." The Journal of the Learning Sciences 7(1): 1-4.

Benyon, D. and D. Murray (1993). "Adaptive Systems: From Intelligent Tutoring to Autonomous Agents." Knowledge-Based Systems 6(4): 197-219.

Bloom, B. S. (1956). Taxonomy of Educational Objectives. New York, NY, McKay.

Bloom, B. S. (1984). "The 2-Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-one Tutoring." Educational Researcher 13: 4-16.

Bloom, C. (1995). Roadblocks to Successful ITS Authoring in Industry. AIED-95 Workshop on Authoring Shells for Intelligent Tutoring Systems, Washington, DC.

Bloom, C., F. Linton, et al. (1997). "Using Evaluation in the Design of an Intelligent Tutoring System." Journal of Interactive Learning Research 8(2): 235-276.

Bransford, J. D., A. L. Brown, et al. (2000). How People Learn. Washington, D.C., National Academy Press.

Brown, D. (1991). Design. Encyclopedia of AI, 2nd Edition. S. Shapiro. New York, NY, Wiley-Interscience Publishers.

Bruner, J. (1966). Toward a Theory of Instruction. Cambridge, MA, Harvard University Press.

Burns, H., J. W. Parlett, et al. (1991). Intelligent Tutoring Systems: Evolutions in Design. Hillsdale, NJ, Lawrence Erlbaum Associates.

Burton, R. R. and J. S. Brown (1982). An Investigation of Computer Coaching for Informal Learning Activities. Intelligent Tutoring Systems. D. H. Sleeman and J. S. Brown. London, UK, Academic Press.

Bylander, T. and S. Mittal (1986). "CSRL: A Language for Classificatory Problem Solving." AI Magazine 7(2): 66-77.

Chandrasekaran, B. (1983). "Towards a Taxonomy of Problem Solving Types." AI Magazine 4(winter/spring): 9-17.

Chandrasekaran, B. (1986). "Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design." IEEE Expert 1(3): 23-30.

Chandrasekaran, B. (1993). The Functional Representation Language: A Framework for Reasoning, Ohio State University.

Chandrasekaran, B. and J. R. J. R. Josephson (1997). The Ontology of Tasks and Methods. Symposium on Ontological Engineering, Stanford, CA, AAAI.

Chandrasekaran, B., T. R. Johnston, et al. (1992). "Task Structure Analysis for Knowledge Modeling." Communications of the ACM 35(9): 124-136.

Chandrasekaran, B., M. C. Tanner, et al. (1988). Explanation: The Role of Control Strategies and Deep Models. Expert Systems: The User Interface. J. A. Hendler. Norwood, NJ, Ablex Publishing: 219-247.

Cheikes, B. (1995). GIA: An Agent-Based Architecture for Intelligent Tutoring Systems. Workshop on Intelligent Information Agents, CIKM '95: Conference on Information and Knowledge Management, Baltimore, Maryland.

Cheikes, B. A. (1995). Should ITS Designers Be Looking For A Few Good Agents? AIED-95 Workshop on Authoring Shells for Intelligent Tutoring Systems, Washington, DC.

Clancey, W. (1987). Knowledge-based Tutoring: The GUIDON Program. Cambridge, MA, MIT Press.

Clancey, W. and K. Joerger (1988). A Practical Authoring Shell for Apprenticeship Learning. Proceedings of ITS'88: First International Conference on Intelligent Tutoring Systems, Montreal, Canada.

Clancey, W. J. and E. Soloway (1990). Artificial Intelligence and Learning Environments. Cambridge, MA, MIT Press.

Cohen, P. A., J. Kulik, et al. (1982). "Educational Outcomes of Tutoring: A Meta-Analysis of Findings." American Educational Research Journal 19(2): 237-248.

Costa, E. (1992). New Directions for Intelligent Tutoring Systems. Berlin, Springer-Verlag.

Dooley, S. A., L. Meiskey, et al. (1995). Developing Usable Intelligent Tutoring System Shells. AIED-95 Workshop on Authoring Shells for Intelligent Tutoring Systems, Washington, DC.

El-Sheikh, E. (1999). Development of a Methodology and Software Shell for the Automatic Generation of Intelligent Tutoring Systems from Existing Generic Task-based Expert Systems. AAAI-99: Sixteenth National Conference on Artificial Intelligence, 1999 SIGART/AAAI Doctoral Consortium, Orlando, Florida, AAAI Press.

El-Sheikh, E., A. Kamel, et al. (1996). An ITS Shell Leveraging the Generic Task Approach to Problem Solving. Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs, ITS'96: Third International Conference on Intelligent Tutoring Systems, Montreal, Canada.

El-Sheikh, E. and J. Sticklen (1998). A Framework for Developing Intelligent Tutoring Systems Incorporating Reusability. IEA-98-AIE: 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Benicassim, Castellon, Spain, Springer-Verlag (Lecture Notes in Artificial Intelligence, vol. 1415).

El-Sheikh, E. and J. Sticklen (1999). Leveraging a Task-specific Approach for Intelligent Tutoring System Generation: Comparing the Generic Tasks and KADS

Frameworks. 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Cairo, Egypt.

Farr, M. J. and J. Psotka (1992). Intelligent Instruction by Computer: Theory and Practice. Washington, DC, Taylor and Francis.

Fraas, J. (1983). Basic Concepts in Educational Research. Lanham, MD, University Press of America.

Frasson, C. and G. Gauthier (1988). Intelligent Tutoring Systems: At the Crossroad of Artificial Intelligence and Education. Norwood, NJ, Ablex Publishing.

Frasson, C., G. Gauthier, et al. (1996). Intelligent Tutoring Systems ITS '96, Montreal, Canada, Springer-Verlag.

Gagne, R. (1985). The Conditions of Learning and Theory of Instruction. New York, NY, Holt, Rinehard & Winston.

Ginsburg, H. and S. Oppen (1979). Piaget's Theory of Intellectual Development. Englewood Cliffs, NJ, Prentice Hall.

Gomez, F. and B. Chandrasekaran (1981). "Knowledge Organization and Distribution for Medical Diagnosis." IEEE Transactions on Systems, Man, and Cybernetics **SMC-11**(1): 34-42.

Goodyear, P. (1990). Teaching Knowledge and Intelligent Tutoring. Norwood, NJ, Ablex.

Greer, J. E. and G. I. McCalla (1994). Student Modeling: The Key to Individualized Knowledge-Based Instruction. Berlin, Springer-Verlag.

Gugerty, L. (1997). "Non-Diagnostic Intelligent Tutoring Systems: Teaching Without Student Models." Instructional Science **25**: 409-432.

Guralnick, D. A. (1996). An Authoring Tool for Procedural Task Training, Northwestern University.

Hawkins, R., J. McDowell, et al. (1993). Function-Based Modeling and Troubleshooting. AAAI Workshop on Reasoning about Function.

Hayashi, Y., M. Ikeda, et al. (2000). Is What You Write What You Get?: An Operational Model of Training Scenario. ITS 2000: Fifth International Conference on Intelligent Tutoring Systems, Montreal, Canada, Springer-Verlag.

Hsieh, P. Y., H. M. Half, et al. (1999). "Four easy pieces: development systems for knowledge-based generative instruction." International Journal of Artificial Intelligence in Education **10**: 1-45.

Ikeda, M., Y. Hayashi, et al. (1999). Ontology More Than a Shared Vocabulary. Proc. of the workshop on Ontologies for Intelligent Educational Systems, AIED-99, Le Mans, France.

Janssen, C., A. Weisbecker, et al. (1993). Generating User Interfaces from Data Models and Dialogue Net Specifications. INTERCHI '93. Amsterdam, ACM Press.

Jin, L., W. Chen, et al. (1999). An Ontology-aware Authoring Tool, Functional Structure and Guidance Generation. AIED-99, Le Mans, France.

Jona, M. and M. Korcusk (1996). Same Architecture New Domain (SAND): An Alternative Methodology for Building High-Quality Reusable Tutoring Systems. ITS'96 Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs, Montreal, Canada.

Joolingen, W. R. v. (2000). Designing for Collaborative Discovery Learning. ITS 2000: Fifth International Conference on Intelligent Tutoring Systems, Montreal, Canada, Springer-Verlag.

Joolingen, W. R. v. and T. d. Jong (1996). "Design and implementation of simulation-based discovery environments: the SMISLE solution." International Journal of Artificial Intelligence and Education 7: 253-277.

Joolingen, W. R. v., S. King, et al. (1997). The SimQuest Authoring System for Simulation-based Discovery Environments. Knowledge and Media in Learning Systems. B. d. Boulay and R. Mizoguchi. Amsterdam, IOS Press: 79-87.

Joyce, B. and M. Weil (1986). Models of Teaching. Englewood Cliffs, NJ, Prentice Hall.

Juel, C. (1996). Learning to Learn From Effective Tutors. Innovations in Learning: New Environments for Education. L. Schauble and R. Glaser. Mahwah, NJ, Lawrence Erlbaum Associates.

Kafai, Y. and M. Resnick (1996). Constructionism in Practice: Designing, Thinking, and Learning in a Digital World. Mahwah, NJ, Lawrence Erlbaum Associates.

Kameas, A. and P. Pintelas (1997). "The Functional Architecture and Interaction Model of a Generator of Intelligent Tutoring Applications." Journal of Systems Software 36: 233-245.

Kass, A. (1994). The Casper Project: Integrating Simulation, Case Presentation, and Socratic Tutoring to Teach Diagnostic Problem Solving in Complex Domains, Institute for the Learning Sciences, Northwestern University, Evanston, IL.

Kassatly, A. and D. C. Brown (1987). Explanation for Routine Design Problem Solving. Presented at the 2nd International Conference on the Applications of Artificial Intelligence in Engineering, Boston, MA.

Kerlinger, F. N. (1973). Foundations of Behavioral Research. New York, NY, Holt, Rinehart, and Winston, Inc.

Kyllonen, P. C. and V. J. Shute (1989). A Taxonomy of Learning Skills. Learning and Individual Differences. P. L. Ackerman, R. J. Sternberg and R. Glaser. New York, NY, W. H. Freeman: 117-163.

Lajoie, S., S. Faremo, et al. (2001). A Knowledge-Based Approach to Designing Authoring Tools: From Tutor to Author. AI-ED 2001: 10th International Conference on Artificial Intelligence in Education, San Antonio, Texas.

Lajoie, S. P. and S. J. Derry (1993). Computers as Cognitive Tools. Hillsdale, NJ, Lawrence Erlbaum Associates.

Larkin, J. and R. Chabay (1991). Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared Issues and Complementary Approaches. Hillsdale, NJ, Lawrence Erlbaum Associates.

Lesgold, A., S. Lajoie, et al. (1990). A Coached Practice Environment for an Electronics Troubleshooting Job. Computer Assisted Instruction and Intelligent Tutoring Systems Establishing Communication and Collaboration. C. a. S. Larkin. Hillsdale, NJ, Lawrence Erlbaum.

Major, N. (1995). How Generic Can Authoring Shells Become? AIED-95 Workshop on Authoring Shells for Intelligent Tutoring Systems, Washington, DC.

Major, N., S. Ainsworth, et al. (1997). "REDEEM: Exploiting Symbiosis Between Psychology and Authoring Environments." International Journal of Artificial Intelligence in Education 8.

Marcenac, P. (1992). "An Authoring System for ITS Which is Based on a Generic Level of Tutoring Strategies." Lecture Notes in Computer Science 602: 428-440.

McArthur, D., M. W. Lewis, et al. (1995). "ESSCOTS for Learning: Transforming Commercial Software into Powerful Educational Tools." Journal of Artificial Intelligence in Education 6(1): 3-33.

McDermott, J. (1982). "R1: A Rule-Based Configurer of Computer Systems." Artificial Intelligence 19(2): 39-88.

McDermott, J. (1988). Preliminary Steps Towards a Taxonomy of Problem-Solving Methods. Automating Knowledge Acquisition for Expert Systems. S. Marcus. Boston, MA, Kluwer Academic Publishers.

Merrill, M. D. (1989). "An Instructional Design Expert System." Computer-Based Instruction 16(3): 95-101.

Moy, B., J. McDowell, et al. (1994). Expansion of an Intelligent Decision Support System for Process Selection and Process Design in Polymer Composites. 26th International SAMPE Technical Conference, Atlanta, Georgia.

Munro, A., M. C. Johnson, et al. (1997). "Authoring Simulation-Centered Tutors with RIDES." International Journal of Artificial Intelligence in Education 8.

Murray, T. (1991). Facilitating Teacher Participation in Intelligent Computer Tutor Design: Tools and Design Methods, University of Massachusetts, Amherst.

Murray, T. (1993). "Formative Qualitative Evaluation for Exploratory ITS Research." Journal of Artificial Intelligence in Education 4(2): 179-208.

Murray, T. (1996). From Story Boards to Knowledge Bases: The First Paradigm Shift in Making CAI "Intelligent". ED-MEDIA 96 Educational Multimedia and Hypermedia Conference, Charlottesville, VA.

Murray, T. (1996). Having It All, Maybe: Design Tradeoffs in ITS Authoring Tools. ITS '96 Third Intl. Conference on Intelligent Tutoring Systems, Montreal, Canada, Springer-Verlag.

Murray, T. (1998). "Authoring Knowledge-Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design." The Journal of the Learning Sciences 7(1): 5-64.

Murray, T. (1999). "Authoring intelligent tutoring systems: an analysis of the state of the art." International Journal of Artificial Intelligence in Education 10: 98-129.

Newell (1982). "The Knowledge Level." Artificial Intelligence 18: 87-127.

Newell, A. and H. A. Simon (1972). Human Problem Solving. Englewood Cliffs, NJ, Prentice-Hall.

Nkambou, R., G. Gauthier, et al. (1996). CREAM-Tools: An Authoring Environment for Curriculum and Course Building in an Intelligent Tutoring System. CALISCE'96, 3rd International Conference on Computer Aided Learning and Instruction in Science Engineering, San Sebastian, Spain.

Patel, A. and Kinshuk (1996). Intelligent Tutoring Tools - A Problem Solving Framework for Learning and Assessment. 1996 Frontiers in Education Conference, Salt Lake City, UT.

Petrushin, V. A. (1992). An Authoring Language for Intelligent Tutoring Systems Implementation. Proc. of the East/West Conference on Emerging Computer Technologies in Education.

Petrushin, V. A. (1995). "Intelligent Tutoring Systems: Architecture and Methods of Implementation. A Survey." Journal of Computer and Systems Sciences International 33(1): 117-139.

Polson, M. C. and J. J. Richardson (1988). Foundations of Intelligent Tutoring Systems. Hillsdale, NJ, Lawrence Erlbaum Associates.

Psotka, K., L. D. Massey, et al. (1988). Intelligent Tutoring Systems: Lessons Learned. Hillsdale, NJ, Lawrence Erlbaum Associates.

Puerta, A. R., H. Eriksson, et al. (1994). Model-based Automated Generation of User Interfaces. Proceedings of AAAI'94: the Twelfth National Conference on Artificial Intelligence, Seattle, WA.

Redfield, C. (1997). An ITS Authoring Tool: Experimental Advanced Instructional Design Advisor. Papers from the 1997 AAAI Fall Symposium on Intelligent Tutoring System Authoring Tools, Boston, MA, AAAI.

Regian, J. W. and V. J. Shute (1994). Evaluating Intelligent Tutoring Systems. Technology Assessment in Education and Training. E. L. Baker and J. H. F. O'Neil. Hillsdale, NJ, Lawrence Erlbaum Associates: 79-96.

Reinhardt, B. and S. Schewe (1995). A Shell for Intelligent Tutoring Systems. AI-ED 95 7th World Conference on Artificial Intelligence in Education, Washington, DC.

Ritter, S. and S. B. Blessing (1998). "Authoring Tools for Component-Based Learning Environments." The Journal of the Learning Sciences 7(1): 107-132.

Ritter, S. and K. R. Koedinger (1996). "An Architecture for Plug-in Tutor Agents." Journal of Artificial Intelligence in Education 7(3/4): 315-347.

Rowley, K. (1996). Increasing the Effectiveness of ITS Research and Development. ITS'96 Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs, Montreal, Canada.

Russell, D., T. Moran, et al. (1988). The Instructional Design Environment. Intelligent Tutoring Systems: Lessons Learned. J. Psotka, L. D. Massey and S. A. Mutter. Hillsdale, NJ, Lawrence Erlbaum Associates: 203-228.

Schreiber, A. T., J. B. Wielinga, et al. (1994). "CommonKADS: A Comprehensive Methodology for KBS Development." IEEE Expert 9(6): 28-37.

Shortliffe, E. H. (1976). Computer-based Medical Consultations: MYCIN. New York, NY, Elsevier.

Shute, V. and J. Psotka (1996). Intelligent Tutoring Systems: Past, Present, and Future. Handbook of Research for Educational Communications and Technology. D. Jonassen. New York, NY, Macmillan.

Shute, V. J. (1991). "Who is Likely to Acquire Programming Skills?" Journal of Educational Computing Research 1: 1-24.

Shute, V. J., L. A. Gawlick-Grendell, et al. (1993). An Experiential System for Learning Probability: Stat Lady. Annual Meeting of the American Educational Research Association, Atlanta, GA.

Shute, V. J. and J. W. Regian (1990). Rose garden promises of intelligent tutoring systems: Blossom or thorn? Proceedings from the Space Operations, Applications and Research Symposium, Albuquerque, NM.

Shute, V. R. and R. Glaser (1990). "A Large-scale Evaluation of an Intelligent Discovery World: Smithtown." Interactive Learning Environments 1: 51-76.

Siemer, J. and M. C. Angelides (1998). "A Comprehensive Method for the Evaluation of Complete Intelligent Tutoring Systems." Decision Support Systems 22: 85-102.

Singley, K. and J. R. Anderson (1989). The Transfer of Cognitive Skill. Cambridge, MA, Harvard University Press.

Sleeman, D. (1987). PIXIE: A Shell for Developing Intelligent Tutoring Systems. Artificial Intelligence and Education. R. W. Lawler and M. Yazdani. Norwood, NJ, Ablex Publishers: 239-265.

Sleeman, D. H. and J. S. Brown (1982). Intelligent Tutoring Systems. London, UK, Academic Press.

Sparks, R., S. Dooley, et al. (1999). "The LEAP authoring tool: supporting complex courseware authoring through reuse, rapid prototyping, and interactive visualizations." International Journal of Artificial Intelligence in Education 10: 75-97.

Srisethanil, C. and N. Baker (1995). ITS-Engineering: Providing Adaptive Teaching in the Engineering Tutor. 1995 Frontiers in Education Conference, Atlanta, GA.

Steels, L. (1990). "Components of Expertise." AI Magazine 11(2): 28-49.

Sticklen, J. (1989). "Problem Solving Architecture at the Knowledge Level." Journal of Theoretical and Applied Artificial Intelligence 1: 1-52.

Sticklen, J. and B. Chandrasekaran (1985). Use of Deep Level Reasoning in Medical Diagnosis. Government Symposium in Expert Systems.

Sticklen, J., C. Penney, et al. (2000). Integrated Generic Task Toolset. East Lansing, Intelligent Systems Laboratory, Michigan State University.

Swartout, B., Y. Gil, et al. (1999). Representing Capabilities of Problem Solving Methods. IJCAI-99, Workshop on Ontologies and Problem-solving Methods, Stockholm, Sweden.

Szekely, P., P. Luo, et al. (1993). Beyond Interface Builders: Model-based Interface Tools. INTERCHI '93. Amsterdam, ACM Press.

Tanner, M. C. and A. M. Keuneke (1991). "Explanations in Knowledge Systems: The Roles of Task Structures and Domain Functional Models." IEEE Expert 6(1): 50-57.

Tu, S. W., H. Eriksson, et al. (1995). "Ontology-based Configuration of Problem-solving Methods and Generation of Knowledge -acquisition Tools: Application of PROTEGE-II to Protocol-based Decision Support." Artificial Intelligence in Medicine 7: 257-289.

Urban-Lurain, M. (1996). Intelligent Tutoring Systems: An Historic Review in the Context of the Development of Artificial Intelligence and Educational Psychology, Michigan State University.

Valente, A., Y. Gil, et al. (1999). LIBRA: A Library of Knowledge Components for EXPECT. Knowledge Acquisition Workshop'99, Banff, Canada.

Van Marcke, K. (1992). Instructional Expertise. ITS'92: Second International Conference on Intelligent Tutoring Systems, Montreal, Canada, Springer-Verlag.

Van Marcke, K. and H. Vedelaar (1995). Learner Adaptivity in Generic Instructional Strategies. Proceedings of AI-ED 95: World Conference on Artificial Intelligence in Education, Washington, DC, AACE.

Venezky, R. and L. Osin (1991). The Intelligent Design of Computer-Assisted Instruction. New York, NY, Longman.

Wasson, B. (1997). "Advanced Educational Technologies: The Learning Environment." Computers in Human Behavior 13(4): 571-594.

Wenger, E. (1987). Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge. Los Altos, CA, Morgan Kaufmann Publishers, Inc.

Wielinga, B., A. Schreiber, et al. (1992). "KADS: A Modeling Approach to Knowledge Engineering." Knowledge Acquisition 4(1): 5-54.

Wielinga, B. J. and J. A. Breuker (1986). Models of Expertise. Advances in Artificial Intelligence. B. D. Boulay, D. Hoggs and L. Steels. Amsterdam, North-Holland.

Wielinga, B. J. and A. T. Schreiber (1994). Conceptual Modeling of Large Reusable Knowledge Bases. Berlin, Germany, Springer Verlag.

Winkels, R. (1992). Explorations in Intelligent Tutoring and Help. Amsterdam, IOS Press.

Wong, W. K. and T. W. Chan (1997). "A Multimedia Authoring System for Crafting Topic Hierarchy, Learning Strategies, and Intelligent Models." International Journal of Artificial Intelligence in Education 8: 71-96.

Woolf, B. P. (1987). A Survey of Intelligent Tutoring Systems. Proceedings of the Northeast Artificial Intelligence Consortium, Blue Mountain Lake, NY.

Yazdani, M. (1987). Intelligent Tutoring Systems: An Overview. Artificial Intelligence and Education: Learning Environments and Tutoring Systems. R. W. Lawler and M. Yazdani. Norwood, NJ, Ablex Publishing. 1: 183-201.

MICHIGAN STATE UNIVERSITY LIBRARIES



3 1293 02334 2425