

1-1-1990

Knowledge-based tutors : an artificial intelligence approach to education.

Beverly Park Woolf

University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_1

Recommended Citation

Woolf, Beverly Park, "Knowledge-based tutors : an artificial intelligence approach to education." (1990). *Doctoral Dissertations 1896 - February 2014*. 4710.

https://scholarworks.umass.edu/dissertations_1/4710

This Open Access Dissertation is brought to you for free and open access by ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations 1896 - February 2014 by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.

UMASS/AMHERST



312066008658525

KNOWLEDGE-BASED TUTORS:
AN ARTIFICIAL INTELLIGENCE APPROACH TO EDUCATION

A Dissertation Presented

by

BEVERLY PARK WOOLF

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements for the degree of

DOCTOR OF EDUCATION

February 1990

School of Education

©Copyright by Beverly Park Woolf 1990
All Rights Reserved

This work was supported in part by:

The National Science Foundation, Materials Development Research, No. 8751362

The Air Force Systems Command, Rome Air Development Center, Griffiss AFB, New York, 13441 and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under contract No. F30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC).

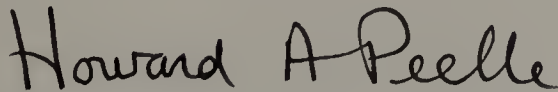
KNOWLEDGE-BASED TUTORS:
AN ARTIFICIAL INTELLIGENCE APPROACH TO EDUCATION

A Dissertation Presented

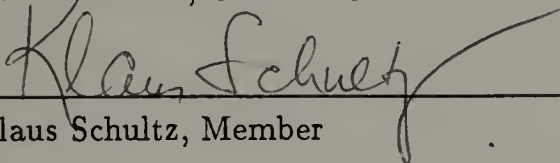
by

BEVERLY PARK WOOLF

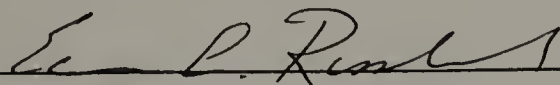
Approved as to style and content by:



Howard Peelle, Chair of Committee



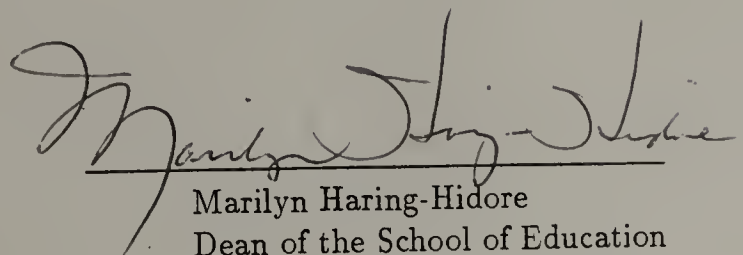
Klaus Schultz, Member



Edwina Rissland, Outside Member



Michael Arbib, Outside Member


Marilyn Haring-Hidore
Dean of the School of Education

Dedicated to my Children

Luna and Ellie Woolf

ACKNOWLEDGEMENTS

I am particularly indebted to my committee for tolerating the unorthodox processes behind my doctoral program. Many of the members have been my colleagues for several years and took the time to explore this research area with me at my request. Each member has contributed generously to the evolution of the work. Howard Peelle has been especially attentive, engaging in numerous and laborious sessions to work out my intent. Michael Arbib contributed greatly to the early organization of the document and has been a steady supporter of my activities. Klaus Schultz has been my tireless partner in research and shares with me the mission of communicating this work to the larger educational community. Edwina Rissland has been generous as a good friend and associate for several years.

I have been blessed with several excellent secretaries who have made this work and me look good. Susan Parker has been a tireless and meticulous worker on every phase of the document. She either monitored or performed all the tedious work involved in generating the manuscript. Her precision and attention to detail are remarkable. Gwyn Mitchell has been my conscientious secretary for many years and has typed many of the original pieces.

I thank Stephen Woolf for being considerate and patient during the rougher parts of this extended process. I particularly thank children everywhere, including my own two, for providing outstanding exemplars of good students and for stimulating me to put forth my best efforts on their behalf.

ABSTRACT

KNOWLEDGE-BASED TUTORS: AN ARTIFICIAL INTELLIGENCE APPROACH TO EDUCATION

FEBRUARY, 1990

BEVERLY PARK WOOLF, B.A., SMITH COLLEGE

M.S., UNIVERSITY OF MASSACHUSETTS

Ph.D., Ed.D., UNIVERSITY OF MASSACHUSETTS

Directed by: Professor Howard Peelle

A vehicle is suggested for bringing information technology into education. Knowledge-based systems are proposed as a way to explore, reason about, and synthesize large knowledge bases. These systems utilize resources such as artificial intelligence, multimedia, and electronic communication to reason about *what, with whom, and how* they should teach in order to tailor knowledge and communication to individual students. Teaching material does not consist of a repertoire of prespecified responses; rather, reasoning about the student and the complexity of the subject matter informs the system's response so that inferences made by the machine become key features of the system's response. Currently, such systems can reason about a student's presumed knowledge, can solve the problems given to the student, and can begin to recognize plausible student misconceptions.

This document provides a practical hands-on guide for people who are considering building knowledge-based systems. It identifies the requisite resources, personnel, hardware and software and describes artificial intelligence methodologies and tools that might become available. The document is directed both at increased production of knowledge-based systems and also at improving the dialogue among computer scientists, educators, researchers, and classroom practitioners around the issue of information technology in the schools.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	v
ABSTRACT	vi
LIST OF FIGURES	xi
CHAPTER	
1. CHALLENGES TO EDUCATION IN AN INFORMATION SOCIETY	1
1.1 The Information Society	1
1.2 Challenges to Education	3
1.3 Societal Factors that Obstruct Education	5
1.4 The Role of Education in an Information Society	9
1.5 The Role of Knowledge-based Tutors in an Information Society	13
1.5.1 A Classroom Problem	13
1.5.2 One Promise of Knowledge-based Tutors	15
1.5.3 New Roles for Teachers	16
1.6 Changes to Education in the Information Age	17
2. KNOWLEDGE-BASED TUTORS	19
2.1 Three Example Systems	20
2.1.1 Recovery Boiler Tutor for Teaching Complex Industrial Processes	20
2.1.2 Caleb for Teaching a Second Language	27
2.1.3 Physics Tutor	32
2.2 Additional Examples	36
2.2.1 More Effective and Efficient Teaching	37

2.2.2	Engaging Students in Real-World Problems	38
2.2.3	Complex Communication Networks	39
2.2.4	Complex Information Management	41
2.2.5	Individualized Teaching	44
2.3	Knowledge-Based Tutors versus Computer-Aided Instructional Systems	45
2.3.1	Comparing Two Instructional Systems	48
2.3.2	The Technology Behind Knowledge-based Tutors and CAI systems	49
2.4	Components of a Knowledge-based Tutor	51
2.5	Summary and Discussion	58
3.	EPISTEMOLOGICAL ISSUES	59
3.1	Introduction	59
3.2	Knowledge Engineering	60
3.3	Tutoring as Qualitative Modeling	63
3.3.1	Modeling Cognitive Processes	66
3.3.2	Modeling Tutoring Expertise	69
3.3.3	Modeling Reasoning in Science	72
3.3.4	Modeling Communication Knowledge	74
3.4	Indexing Information to be Taught	74
3.5	Summary	79
4.	DEVELOPMENT ISSUES	80
4.1	Introduction	80
4.2	Organizing the Development Process	81
4.2.1	Knowledge Engineering	85
4.3	Stages of Development	87
4.3.1	Stage 1: Problem Identification and Definition	87

4.3.2	Stage 2: Paper Design and Prototype Development	88
4.3.3	Stage 3: Implementation	89
4.3.4	Stage 4: Test, Evaluate, and Revise Cycles	91
4.3.5	Stage 5: Integration	91
4.3.6	Stage 6: Evaluation and Maintenance	91
4.4	Gathering a Team	94
4.4.1	Environmental Input	95
4.4.2	Teaching Input	97
4.4.3	Cognitive Input	99
4.4.4	Domain Input	100
4.5	Summary and Discussion	103
5.	IMPLEMENTATION ISSUES	104
5.1	Tools for Representing Knowledge and Control	104
5.2	Encoding Domain Knowledge	107
5.2.1	Knowledge Representation	109
5.2.2	Issues Related to Encoding Domain Knowledge	114
5.3	Encoding Control Knowledge	120
5.4	Encoding Student Knowledge	123
5.5	Encoding Tutoring Knowledge	128
5.5.1	Tools for Reasoning about Discourse	129
5.5.2	Example of a System That Manages Discourse	133
5.5.3	Issues Related to Encoding Tutoring Knowledge	139
5.6	Encoding Communication Knowledge	146
5.6.1	The Role of Communication Knowledge	147
5.6.2	Examples of Communication Media	150
5.6.3	Issues Related to Encoding Communication Knowledge	153
5.6.4	Tutors with Natural Language Processing (NLP) Capabilities.	155

5.7 Summary	159
6. SOFTWARE AND HARDWARE CONSIDERATIONS	160
6.1 The Nature of Artificial Intelligence Programming	160
6.2 Choosing an AI Language	163
6.3 Knowledge Engineering Tools	172
6.4 Choosing Hardware	175
6.5 General Programming Tools	177
6.6 Summary and Discussion	180
7. THE FUTURE: COMPUTER PARTNERS IN EDUCATION AND INDUSTRY	181
7.1 Impact of Knowledge-based Tutors	181
7.2 Impact of Knowledge-based Technology on Education	182
7.3 Integration of Knowledge-based Tutors	185
7.4 Needed Breakthroughs	187
7.5 Impact of Technology on the Workplace	192
7.6 Living in the Knowledge-based Society	193
 APPENDICES	
A. HUMAN NETWORKING	197
B. EXAMPLE STORYBOARDS	207
BIBLIOGRAPHY	222

LIST OF FIGURES

1. Sectional View of the Recovery Boiler	21
2. Dialogue Between Tutor and Student	23
3. Tasks Performed on the Boiler	24
4. Focused View of the Fire Bed	24
5. The Complete Control Panel	25
6. Trends Selected by the Operator	26
7. Caleb: A System for Teaching Second Languages	28
8. Themes and Pieces in the Spanish Curriculum	30
9. Communication Icons in Caleb	31
10. Error Therapy for the Silent Tutor	32
11. Statics Tutor: Simulation	34
12. Thermodynamics Tutor: Simulation	35
13. Electronic Networks Connecting People with Information	39
14. Screens from Intermedia Project [Yankelovich et al., 1985]	42
15. "How the West Was Won" [Burton & Brown, 1982]	45
16. Features Behind Knowledge-Based Tutors and CAI Systems	50
17. Components of a Tutoring System	52
18. The Geometry Tutor [Anderson et al., 1985]	56
19. The Economics Tutor [Shute & Bonar, 1986]	57

20. Knowledge Engineering Tasks, adapted from Rissland and Schultz [1987]	61
21. Knowledge Engineering Tasks, Part 2	62
22. Models of Reasoning in an Intelligent Tutoring System (after Clancey[1987])	63
23. Topic Breakdown for Thermodynamics	73
24. Communication Primitives	74
25. Dimensions of a Topic to be Tutored	76
26. Stages in Building a Knowledge-Based Tutor	82
27. Detailed Stages in Building a Knowledge-Based Tutor	83
28. Tasks to be Accomplished	92
29. Tasks to be Accomplished, Continued	93
30. A Framework for Managing Tutoring Discourse	98
31. Representation and Control in a Tutoring System	105
32. Tools for Reasoning about Tutoring Knowledge	106
33. Four Models in a Tutoring System	107
34. A Portion of SCHOLAR's Semantic Net [Carbonell, 1970]	108
35. A Conversation with SCHOLAR [Carbonell, 1970]	110
36. The Curriculum Information Network [Barr et al., 1976]	110
37. Hierarchy of Frames	112
38. A Conversation with WHY [Stevens & Collins, 1977]	116
39. Reasoning About Examples	121
40. The Genetic Graph [Goldstein, 1982]	124
41. "How the West was Won" [Burton & Brown, 1982]	125
42. Reasoning about Discourse Level	130
43. <u>D</u> iscourse <u>A</u> ction <u>T</u> ransition <u>N</u> etwork: DACTN	131

44. Two Phases of the Consultant	132
45. Evaluation of a Present-Oriented Time Perspective	134
46. Dialogue about the Evaluation of Figure 45	135
47. Levels of Control in TEV	137
48. Screen for Tutoring about Force [Duckworth et al., 1987]	140
49. Plan Schema for Tutoring about Force	141
50. Goal Tree for Tutoring about Force	142
51. The Wired Society	148
52. Compact Disk Technology	152
53. Merger of Compact Disk and Knowledge-based Tutors	153
54. NLP Interface in the SOPHIE System [Burton et al., 1982]	157
55. Representation and Control in a Tutoring System	162

CHAPTER 1

CHALLENGES TO EDUCATION IN AN INFORMATION SOCIETY

1.1 The Information Society

Issac Asimov once said that the important thing to predict about an innovation is its impact on society. For technologies related to the computer, and the increased amount of information they generate, the important assessment to make is how the changing need for information impacts on the structure of society. The computer has already increased our dependence on information. Societal changes related to information are now beginning to decrease the need for conventional educational practices. This document describes the information revolution and its impact on society, particularly education. It investigates ways to restructure education in order to point the way to a fully developed information-based society.

More information is now available to more people at more locations. Increased information at home, school, work, government buildings, military sites, and academia institutions requires new learning skills. People need to know how to store, access, and reason about larger amounts of data. Nearly instantaneous and world-wide communication, via electronic networks, requires that people select from large storehouses of data, not only the appropriate data, but also an appropriate medium for communicating that data and a way to organize it.

As information becomes more prevalent, people become more independent and free to explore distant knowledge bases. Education then moves away from being a local institution where knowledge supposedly resides, and becomes instead a number of processes

which are centered on the individual. Knowledge becomes available beyond books and beyond time- and space-bound classrooms. It can be gleaned from networked humans, distant encyclopedias, remote archives, and on-line systems. Human behavior and interactions around information have already changed. In the workplace, people are already required to handle more information, faster communication, and increasingly complex syntheses about data [Zuboff, 1988].

To take advantage of increased information and to continue to be prosperous in the information age, our society needs a basic restructuring of education. Currently teachers use computers only rarely for instruction purposes and few can exploit the enormous potential offered by interactive technologies [U.S. Congress Office of Technology Assessment, 1988].

This document motivates a restructuring of education in order to bring information technology into the classroom and to look at the teacher's role in that process. A vehicle is proposed to streamline the process: knowledge-based systems. These systems are computer programs that use technology, such as artificial intelligence methodology, high-speed communication networks, and multimedia, to improve people's ability to handle and reason about large amounts of information.

Advanced technology can only be introduced into education within a rejuvenated institutional framework. This document argues that education must be restructured, and further, that the introduction of information technology in education is essential for full realization of an information-based society. This chapter focuses on societal issues that provide the intellectual opportunities and challenges. It describes factors that constrain traditional educational practices. Later chapters describe knowledge-based systems and elucidate how they might offer a solution. The final chapter returns to the societal issues raised here and offers a possible solution for restructuring education.

1.2 Challenges to Education

To achieve a greater use of technology, especially information technology, requires a state of mind that allows information to be changed into knowledge for the purpose of problem solving [Quinn Patton, 1987]. Educators and others concerned with teaching need to develop a sense of how technology impacts the classroom, how the expanded nature of communication affects the teaching of basic skills, and how the changing relationship between people and information impacts educational practices. Education has always been critical in humanity's attempt to handle change and to pass on its knowledge from one generation to another. In this age of abundant information, education plays an even more critical role as information is created, multiplied, and disseminated in far larger quantities.

Many educational issues are raised: How will information cause changes in schools? What role does technology play? How should teachers prepare for the information age?

Clearly schools need to provide responses. They need to respond to changes brought about in society and to provide students with skills that prepare them for life and, in the short term, for the workplace. For example, basic skills such as low-level arithmetic, bookkeeping, or typing might once have guaranteed a life time of employment. These same skills today are not as valuable if the applicant can not also edit, format, "publish," and communicate this information electronically. The information age demands knowledge which was not available even a single generation ago.

This document suggests that knowledge-based systems provide a way to bring the information age into classrooms. Such systems use a large internal representation of knowledge to provide assistance for users accessing and reasoning about knowledge. When specifically designed for education, such systems provide interactive environments that support individualized learning. Working with knowledge-based systems helps students master new cognitive skills. Rich in knowledge, such systems can organize and index information and support dynamic, context-sensitive selection of material. They provide

a vital link in the filtering, modeling, and sharing process whereby massive amounts of data and information, available through multimedia and electronic networks, are passed on to the next generation. They can also provide behavior approaching that of a human engaged in one-on-one tutoring and can be used to teach a new body of knowledge or to review one previously presented.

Knowledge-based systems are distinguished from conventional computer systems, including Computer-Aided Instruction (CAI), video, or CD-ROM media, which are structured for direct access. These latter systems require machine or user to search through extensive indices, menus, numbering systems, or list of topics to access information. Information may be virtually unlimited, yet the problem becomes one of appropriate selection. The average student cannot effectively use the indexing methods and cannot achieve a reasonable plan for learning [Suthers, 1989].

On the other hand, knowledge-based tutors reason about student actions, discourse, or pedagogical goals before providing a new environment, piece of data, or response. By using extensive knowledge, these systems employ more complex and fine-grained indexing mechanisms and can surpass the richness of structure available in conventional systems. They become an active as well as an interactive medium, responsible for their own decisions about how and when to propose new material for a student's consideration.

One goal of this document is to enlist more people in the process of building such systems. Obviously, the availability of more trained participants will increase production of the systems. Moreover, the very exercise of building them will enhance the use of technology and knowledge in education. Increased numbers of systems will further improve the dialogue between researchers and practitioners, computer scientists and educators, and psychologists and domain experts.

1.3 Societal Factors that Obstruct Education

The combined information and communication revolutions contribute to an obfuscation of education. Factors such as *rapid technological advances*, *discontinuous change*, *the knowledge explosion*, and *information overload* provide obstacles to achieving effective education. Conversely, these same factors provide great educational challenges and opportunities for increased human intellect. This section discusses the constraints placed on education as a result of these revolutions.

The amount of information needed by workers and students alike is increasing, and the relevance of schooling, especially with outdated curriculum, continues to decrease. Complex and abundant information threatens to make skills and knowledge out-dated before workers get a chance to use them. Even today, people need to be retrained frequently, and businesses to be restructured regularly, to take advantage of newly acquired on-line data and automated facilities.

The "knowledge explosion" is already here and with it innovations related to information storage and access, such as computer networks and multimedia facilities. Even in 1962, Fuller [1962] argued that technological change was already beyond the ability of most people to comprehend, let alone master. A *New York Times* advertisement puts it this way:

"Yesterday we called it science fiction. Today, we call it news." (*New York Times* [1988]).

Before the 20th century, the rate of change of society and technology was relatively slow and not detectable in the average lifetime [Postman & Weingartner, 1969]. Until a hundred years ago, a well-educated person could manage throughout his/her life using the skills passed down by grandparents. Rapid change, however, is a product of this century. In communication, transportation, medicine, and writing, more innovations have emerged in the last 50 years than in the whole history of each field.

For example, almost the whole history of medicine until 50 years ago is the history of the placebo effect [Postman & Weingartner, 1969]. Then 50 years ago antibiotics arrived, and more recently, open-heart surgery, and invitro fertilization. Communication and media follow in a similar way; few technological innovations existed before 200 years ago. Then the printing press came into use about 150 years ago, then the telegraph, the photograph, the telephone, rotary press, motion pictures, radio, talking pictures, television, and now computers and video.

Change is not new in our society. What is new today is the *rate of change*. There has been a “qualitative difference in the character of change” [Postman & Weingartner, 1969]. That rate of change has increased so that pre-established criteria and norms become ineffective within a single lifetime. In the past, a person could grow up in a town, be educated by local teachers, become employed in a local industry, function as an adult, and suffer no dramatic redefinition of values or knowledge during her/his lifetime. “Natural cycles,” stability and predictability, were characteristic of the time [Postman & Weingartner, 1969]. Today, however, this is frequently impossible. Easy communication, e.g., television, air travel, overnight mail, conference calls, and fax services, provide views of other possibilities and other working conditions that were not possible in the last century.

This situation has serious repercussions for education. Today’s workers could not have learned the information they needed 10 or 20 years ago. Jobs involving text processing, robotics, bio-engineering, clerical skills, molecular-biology, and automated controls, to name a few, demand skills unknown when the current workers were in school. These jobs require life-long learning. Technological advances for specific fields now need to be taught by the employer, whether in industry, military, academia, or government. This process will probably repeat itself for several generations and with greater rate of change.

Currently, information is a primary source of industry for the technologically developed world [Naisbett, 1984]. In fields susceptible to scientific inquiry, information is already a major technology and a primary industry. Creating, processing, and distributing

information "is" the job in professions such as law, engineering, architecture, journalism, library science, and medicine [Naisbitt, 1984]. Information has virtually replaced money as the new power source in our society. Unlike other resources, information is not subject to the law of conservation; it can be created, destroyed, and most importantly, it can be synergistic—that is, the whole is usually greater than the sum of its parts.

Accessing this amount of information, let alone mastering it, can be overwhelming. Just indexing it provides a challenge. At least 40,000 scientific journals are estimated to roll off presses around the world [Broad, 1987]. About 7,000 articles are written each day, and that number is expected to double every 5.5 years [Naisbitt 1984]. The knowledge we use, the context in which we learn it, and nearly every human behavior is affected by issues of information access and retrieval [Postman & Weingartner, 1969].

Uncontrolled and unorganized information is a problem of our society. Users need to locate information quickly and accurately and will pay for the privilege of doing so. The emphasis in an information society shifts *from supply to selection and reliability*. "Being without [general] computer skills is like wandering around a collection the size of the Library of Congress with all the books arranged at random with no Dewey Decimal System, no card catalogue, and of course no friendly librarian to serve your information needs" [Naisbitt, 1984, pg. 27].

The Need for Educational Change. Technology may help manage information overload in particular jobs; however, that same technology typically requires more training to enable workers just to *comprehend* the job. As sophisticated communication skills become increasingly important, our education system has increasingly turned out graduates with weaker reading and writing skills [Carnegie Council, 1979]. Teaching people to handle complex and abundant information presents difficult problems. Looking for solutions in the public elementary/secondary school system seems reasonable because schools are primary among those institutions that prepare our species for survival. Or, from another perspective, school is the one institution that is "inflicted" on everybody [Postman & Weingartner, 1969].

Unfortunately, schools do little to enhance our chances for survival. Schools have been characterized as irrelevant [McLuhan, 1973], as shielding children from reality [Weiner, 1954], as educating for obsolescence [Gardner, 1968], as being based on fear [Holt, 1964], and as designed to induce alienation [Goodman, 1970; Kozol, 1967].

A brief list of major problems in education includes written and scientific illiteracy, limited student involvement in classes, inappropriate curriculum, lack of complexity in curricula, teacher unpreparedness, and lack of customized teaching. Current levels of illiteracy and lack of skills cost business and government billions in welfare and prison costs [Naisbitt & Aburden, 1985]. Corporations pay nearly \$60 billion per year for education [Naisbitt, 1984].

How are schools solving these and other problems that result, in part, from the combined revolutions in communication and information? Several federally commissioned studies graphically illustrate our current status.

- “. . . because of deficits in our public school system, about one-third of our youth are ill-educated, ill-employed, and ill-equipped to make their way in American society.” (*The Carnegie Council of Policy Studies in Higher Education* as reported in the *Washington Post*, November 28, 1979.)
- Most Americans are moving toward “virtual scientific and technological illiteracy” (*U.S. Department of Education* and the *National Science Foundation* report, 1980 as reported in U.S. Report, the *New York Times*, October 23, 1981.)
- There exists a “rising tide of mediocrity” in our education system. (*National Commission on Excellence in Education*, in its report *A Nation at Risk*, April 1983.)

- We have raised a generation of scientific illiterates while many other nations have moved in the opposite direction. We have effectively committed technological disarmament through failure to adopt rigorous standards of science and mathematics education. (*The National Commission on Excellence in Education*, April 1983)
- The average Japanese student scores 100% better in mathematics at all levels than the top 10% of American students [Walberg, 1983]. Swedes, Australians, Britons, Canadians, and the French all do upwards of 50% better than U.S. students. The Soviets graduate six times as many engineers annually as the U.S. even though the populations of the two nations are roughly equal; the Japanese produce about twice as many scientists and engineers as the U.S. from about half the population base [Walberg, 1983].

The generation graduating from high school today is the first generation in American history to graduate less skilled than its parents [Naisbitt, 1984].

1.4 The Role of Education in an Information Society

One purpose of education is to train people to function productively in society. But, in a society undergoing rapid change, this becomes increasingly difficult. Students learn behaviors or skills with potential applicability to their work and then those skills might become irrelevant in a short time.

A fixed curriculum, commonplace in schools today, is no longer acceptable for educating people in our complex and changing society. For example, the natural science curriculum of the turn of the century is out of date; language/grammar taught from that vintage is under siege; and history and social science, as described 20 years ago, is open to serious question.

Irrelevant curriculum is most apparent in science education. Students might read and hear about advances such as quarks, quantum mechanics, test-tube babies, or black holes. Yet their basic classroom activities reflect science as it was taught decades ago. New technology and instructional innovations have upgraded physics education considerably; however, it is said that if Issac Newton (circa 1650) were to return to the physics classroom today, he would feel very much at home; indeed, the same topics are taught in the same order and accompanied by the same laboratory assignments for the past 75 years.

Elementary mathematical skills, including addition, subtraction, multiplication, and long division are handled effectively in the information age by \$5.00 calculators. Some schools systems now freely distribute calculators to all their students. Should basic arithmetic skills be routinely practiced in the classroom? Obviously, mathematical concepts should be taught, but the tools of advanced technology should also be made available so that students can focus on the principles and abstract concepts of mathematics, not the rote skills.

Old standards, such as calculus and statistics, are under fire because numerical analysis and statistics computer packages provide more complete solutions than does the closed form of calculus. For example, the National Science Foundation has awarded a grant of nearly a million dollars for the redesign of the calculus curriculum [Callahan, 1988]. As pointed out in this research proposal, the teaching of calculus has remained grounded in practices and viewpoints more than a century old despite the large impact of computer technology in areas where calculus is used, such as physical as well as biological and social sciences.

Certain new technologies threaten to change the nature of long-standing curriculum items. For example, hand-held language machines, on-line spellers, thesauruses and dictionaries, will alter how foreign languages, language composition, and language grammar are taught.

The Need for Life-long Learning. Given rapid change and unclear specifications of skills, how can schools best prepare students for the future? Primarily by moving away from concerns about *what to teach* and by focusing on *how to learn*, (see for example Toffler [1980] and Papert [1980]). Education should prepare children for life on their own as autonomous learners. No one can predict the future; rather education should become part of a process that prepares students for lifelong learning—to become lifelong consumers of education.

The Need for Distributed Education. This mandate impacts both on traditional classrooms and out-of-the-classroom education. In classroom education, the curriculum needs to be updatable, modularly replaceable, and designed so that new topics and appropriate teaching strategies can be inserted to teach new curriculum. No curriculum should be finite, bounded, and regulated in part by large publishing houses.

Out-of-the-classroom education has already begun to be distributed to other agents, such as families, communities, employers, and media. In fact, non-traditional education is growing in industry, military, and the private sector [Perelman, 1987]. For example, families within a given school community now spend two to ten times the amount of money spent by that school on computer education programs [Wakefield, 1986]. Industry has taken responsibility for education beyond that of their own employees and, in part, assumes the role as educator for future workers. For example, General Electric contributes \$1 million for Saturday tutoring sessions for secondary students and has put \$20 million into a program aimed at doubling the number of students from selected inner-city public schools entering college by the year 2000 [Teltsch, 1988]. Xerox Corp has donated \$5 million dollars to establish the Institute for Research on Learning, which researches methods by which technology can be applied to education in order to improve industrial productivity. Beginning in 1986, Exxon has spent at least \$1.5 million annually for pre-college education [Teltsch, 1988].

Current educational institutions cannot work alone. Contributions from industry, academics, and government are sorely needed. Existing problems within schools make it

all too clear that schools will change or might be eliminated. For instance, one desperate school board in Chelsea, Massachusetts, a suburb of Boston, has turned over the daily operation and long-range planning of their entire school system for 10 years to the School of Education of Boston University [*The New York Times*, 1988].

Skills Needed in an Information Society. How can skills needed in the year 2000 or beyond be known? How will technology redefine future work skills and reshape the public's need to access information? How can students prepare today to work with future technology? We do not know the answers to such questions. However, technology itself does not define nor obviate skills such as these questions suggest. In some cases, technology creates jobs; in others, it eliminates them. For example, robots and intelligent control systems require specially trained workers.

Although machine control systems may eliminate unskilled jobs, these same systems increase the demand for highly skilled maintenance and service workers, both creating or eliminating jobs at all levels of skills. Attentive and competent operators are essential for smooth operation of robots and sophisticated control systems. Information technology and office automation, including the advent of word processors and spread sheets, have already necessitated mastery of new skills. For instance, some secretaries now handle tasks previously reserved for bookkeepers, such as predictive budget planning, or for information managers, such as electronic and worldwide network communication.

Technology does not define jobs; rather, the way that technology is used determines which skills will be needed [Clarendon, 1986]. Technology reshapes the nature of the work carried out. Some businesses create new jobs by taking advantage of data unavailable a few years ago. For instance, on-line stock quotes, global market prices, and easy access to research and library material all create a need for human skills in accessing and organizing information more efficiently. Jobs that previously required rote skills now require more evaluative reasoning and more sophisticated judgement. In general, fear of job losses stemming from the introduction of new technologies to the workplace have proven ill-founded [Clarendon, 1986]. Sophisticated reading, writing, and communication skills are

clearly required in an information-based economy whereas lack of such skills seriously penalizes a worker's productivity, jeopardizing a nation's economy.

As education expands to become a life-long activity, and as community and industry contribute to provide training, the student-as-consumer becomes the center of a "school" which is no longer a building or a time period, but rather a daily process [Perelman, 1987]. Within this context, knowledge-based systems have emerged and offer the potential of supporting people in becoming autonomous lifelong learners.

1.5 The Role of Knowledge-based Tutors in an Information Society

The information society exacerbates existing education problems. This section proposes solutions to those problems that might be provided by knowledge-based tutors. Examples of such tutors are given in Chapter 2. In brief, knowledge-based tutors are proposed as platforms upon which students can explore knowledge, ask questions, generate hypotheses, become more autonomous learners, and test concepts.

1.5.1 *A Classroom Problem*

One educational problem that knowledge-based systems immediately address is the limitations placed upon learning by narrative teaching, including lecture-based classes which promote inactive and uninvolved students. In classrooms that focus on narrative teaching, topics such as arithmetic or writing are taught in an isolated, insulated, and disconnected way. In narrative teaching, curriculum is based on stopping and organizing a spontaneous process to make it palatable to students. Thus mathematics is taught separately from literature and each topic of science is disjoint from every other. Lack of student experimentation and involvement contributes to student inactivity and promotes discussions of objects and processes as if they were motionless and predictable, providing a mechanistic view of the world. For example, memorized equations and preformulated

data facilitate only rote learning; problem solving devoid of explanations and qualitative reasoning results in rote memorization of procedures and formula. This kind of narrative teaching establishes and maintains irrelevancy in the classroom and turns children into “containers” to be “filled” by teachers. The primary role of teachers within this paradigm is to try to regulate the way the world enters the student.

In fact, life flows within a creative and complex process; entities move and impact on each other. People who learn from life, rather than from classes, interact with entities and become involved in their own learning process. They can exercise native creativity and learn to transform processes around them. Many educators have suggested bringing this interactive form of learning into the classroom. It has been called variously dialectic teaching [Freire, 1982] or constructivism [Piaget, 1971] (see Section 5.6.6). This pedagogical approach encourages students to experiment with and to learn from complexity. It supports students in selecting their own information, creating new environments, generating hypotheses, and becoming involved through inquiry or discourse. A dialectic interaction asks students to examine two apparently opposing views and to openly inquire, experiment, and test in order to achieve a resolution.

The alternative approach, or narrative education, according to Freire [1982] persists because it is politically effective. In countries where poverty and oppression dominate, this form of education seems to ensure that the poor remain passive and unquestioning. Such an approach and its embodiment serve to obviate thinking: lectures, reading assignments, methods for evaluating “knowledge,” even the physical distance between teacher and pupil in the classroom (e.g., one teacher for 30 students). Neither the teacher nor the student is involved in authentic thinking, which involves questioning and problem solving, since thinking is concerned with reality and takes place mostly in communication. People perpetuate narrative teaching according to Friere because they care neither to have the world revealed nor to have the poor become powerful. Such a disjoint teaching approach makes people less human, because humanity is only defined in terms of intelligent beings interacting and transforming reality.

1.5.2 *One Promise of Knowledge-based Tutors*

Advanced technologies, proposed later in this document, offer a dialectic or constructivist approach to teaching. Admittedly, they are not intended to solve all the problems confronting education. For example, the number one health problem in the United States is mental illness; more Americans suffer from mental illness than from all other forms of illnesses combined [Postman & Weingartner, 1967]. At least 7.5 million American children suffer from mental problems severe enough to require treatment—that is 12% of all the children under 18. Suicide is the second most common cause of death among adolescents. Also, parental beating is the most common cause of infant mortality in the United States. These and other problems can only partially be dealt with through education.

However, knowledge-based tutors can address problems resulting from narrative education and accentuated by the communication and information revolution, such as information overload, complex retrieval of knowledge, and working with rapid change.

The goal of these systems, as described in the rest of this document, is to

- be nearly as effective as human tutors, which are nearly 200% more effective than classroom lecturers [Bloom, 1984];
- provide consistent, modifiable, and affordable industrial training;
- produce a clearer understanding of
 - human reasoning, learning skills, and process models;
 - tutorial strategies and principles;
 - subject area knowledge.

Progress has been made in each area. Thus, for example, the process of building knowledge-based tutors has led to a clearer understanding of human reasoning and learning skills in several domains. Anderson [1985] described his cognitive studies of geometry,

which were made during the two years before he developed the geometry tutor (see Section 2.2.1). Woolf described the task analysis that preceded building of the Recovery Boiler Tutor ([Woolf et al., 1986], see also Section 2.1.1) and the Physics Tutor ([Woolf et al., 1988], see also Section 2.1.3). Building these systems has produced a clearer elucidation of tutorial strategies and an identification of related work in instructional science. Such strategies, for example, were researched before the physics tutors were built (Section 2.1.3). Acquiring and representing subject area knowledge is a key component of building any knowledge-based system. This process, and the representation of subject area knowledge, is described in detail in Chapter 3. Many educators and instructional designers have commented that the process of building such systems has produced a clearer understanding of the subject area knowledge.

1.5.3 New Roles for Teachers

Knowledge-based systems have the potential to redefine the roles of students and teachers: students become the explorers and teachers the advisors; students present new hypotheses and teachers counsel them; and students define information for the purpose of later testing it, while teacher monitor their behavior. This new paradigm advocates a view of teaching as a collaborative exploration. Given vast amounts of available information, the new role of the teacher will be to explore, along side of the student, distant knowledge sources, remote experts, and intelligent knowledge bases. Can knowledge-based systems help classroom teachers harness the power made available through the knowledge revolution? One goal of this document is to provide guidelines for building such systems to test this proposition.

1.6 Changes to Education in the Information Age

The consequences for society are serious if innovations to improve education in the information age do not become practical in the next several decades. The change to a fully information-based society will occur. The question is: How many nations and what percent of their citizens will participate in that society and how competitive can any nation remain as the world becomes more dependent on information? For example, unless schools in the United States are radically transformed and become part of the age of technology, we will spend decades catching up with more enterprising countries, such as Japan [Perelman, 1987].

Citizens who cannot pass basic reading and writing proficiency tests at the high-school level will remain technologically backward. The basic school design needs to be reworked; education will not be improved by more "add-ons," such as the addition of a few computers or videodisks placed in selected (and likely affluent) communities, while leaving the basic structure unchanged.

One goal of this document is to describe the process of building knowledge-based systems, including the requisite knowledge, resources, personnel, and development stages. It is expected that exploration of issues around building these systems will, in part, prepare educators for entry into the information-age classroom. The intention is to provide enough material for a reader to make reasoned choices from among a variety of tools and methodologies. The document does not promote a specific design approach or implementation architecture. Rather, it prompts educators to learn about knowledge-based systems and to begin to cultivate an approach to information which renders information useful for problem-solving.

As Issac Asimov suggested, any powerful innovation will leave its mark on society. The computer and information revolutions have already impacted society. They have changed how people relate to and think about information. These same revolutions are about to change education. As concerned citizens and educators, we need to guide that impact

and to manage planning and policy decisions to assure that such changes contribute to the health and vitality of our educational system.

The remainder of this document focuses on the effect of applying knowledge-based systems to education. The next chapter describes several knowledge-based systems in detail, as examples of how information can be used for problem solving. The next four chapters then focus on development of these systems: Chapter Three looks at epistemologic issues identifying the knowledge to be encoded; Chapter Four presents the steps of the implementation processes defining the resources, personnel, and time commitments; Chapter Five describes artificial intelligence tools and methodologies; and Chapter Six examines hardware and software considerations. The seventh and final chapter summarizes some theoretical issues and makes long range predictions while returning to questions of classroom practices and the educational questions posed in this chapter.

CHAPTER 2

KNOWLEDGE-BASED TUTORS

The last chapter presented a view of the information society and highlighted a number of societal problems which exasperate an already bankrupt educational system. Scientific illiteracy, limited student involvement, irrelevant curriculum, and narrative teaching, in which predominantly lecture-style teaching produces uninvolved and powerless students, are all aggravated by the information society. This chapter explores solutions to those problems and proposes knowledge-based tutors as a way to establish a partnership among students, teachers, and machines. Systems are described that organize and disseminate information, teach a variety of subjects, and help students reason about complex issues. These systems promote an inquiry interaction in which students are prompted to propose hypotheticals, explore large knowledge bases, and become critics of their own work.

The example knowledge-based systems described here are admittedly narrow in scope. Many long- and short-term goals remain to be achieved before technology such as that described here becomes generally available (see Chapter 7). However, anecdotal evidence of systems placed in schools, industry, and military sites suggests that these systems are useful, enjoyable, and effective. Some systems have been shown to teach more effectively and efficiently, others to engage students in real-world problems, provide complex communication to networked data banks, and still others to move students toward a greater sense of competency at an earlier stage.

2.1 Three Example Systems

This section describes three knowledge-based tutors in detail. Several other systems are described later in relation to the type of educational problem they address. The first three systems include the *Recovery Boiler Tutor*, which teaches a complex industrial process; *Caleb*, which teaches a second language; and the *Physics Tutor*, which teaches statics and thermodynamics. The first system is finished and now being used in more than 60 sites throughout the United States. Because it is complete, it is presented in greater detail than the second and third systems, which have been developed only to the prototype stage.

2.1.1 *Recovery Boiler Tutor for Teaching Complex Industrial Processes*

The Recovery Boiler Tutor (RBT) [Woolf et al., 1986] trains control room operators to operate a Kraft recovery boiler, a type of boiler found in paper mills throughout the United States.¹ The goal is to challenge students to solve new problems with the boiler while monitoring and advising them. The system discriminates between optimal, less than optimal, and clearly irrelevant student actions. Students can continue freewheeling or purposeful problem-solving behavior while the tutor offers help, hints, explanations, and tutoring advice when needed or requested. Students are expected to observe the impact of their actions on the simulated boiler and to react before the tutor advises them about potential problems. Students can change setpoint controls and request information about the boiler while the tutor selectively discusses the optimality of their actions and suggests how the student might better focus his/her actions or better utilize the data.

¹RBT was built by J. H. Jansen Co., Inc., Steam and Power Engineers, Woodinville (Seattle), Washington and sponsored by The American Paper Institute, a nonprofit trade institution for the pulp, paper, and paperboard industry in the United States, Energy Materials Department, 260 Madison Ave., New York, NY, 10016.

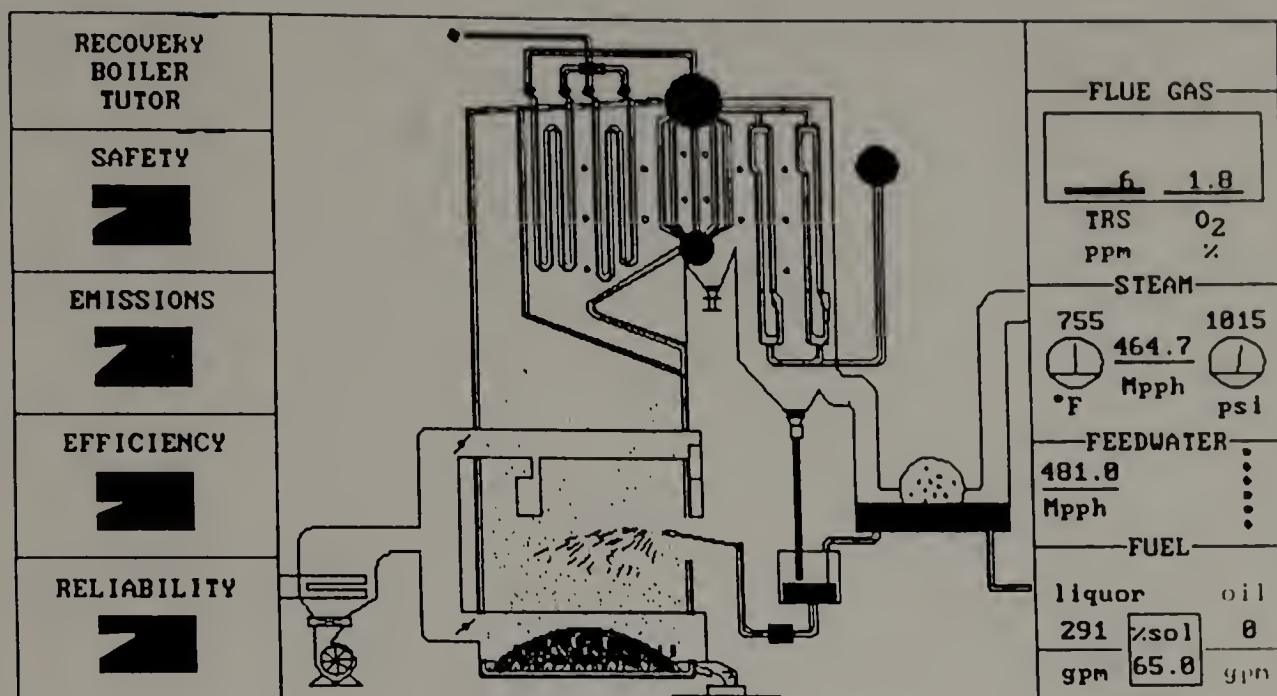


Figure 1 Sectional View of the Recovery Boiler

The tutor is based on a mathematical model of the boiler and provides an interactive simulation complete with help, hints, explanations, and tutoring customized to the individual user (Figures 1 to 6). Students can initiate any of twenty training situations, emergencies, or operating conditions, or they can ask that an emergency be chosen for them. They can also accidentally trigger an emergency as a result of their actions on the boiler. Once an emergency has been initiated, the student is encouraged to adjust meters and perform actions on the simulated boiler in order to solve it.

A sample interaction between the student and tutor is shown in Figure 2.² An important feature to note about the dialogue is that at any point during the simulated emergency there are a large number of actions an operator might take and, as the prob-

²The dialogue of Figure 2 was not actually produced in natural language; student input was handled through menus, and tutor output was produced by cutting text from emergency-specific text files loaded when the emergency was invoked.

lem worsens, an increasing number of actions that *should* be taken to correct the operating conditions. Thus, an immediate and correct response might require only one action, such as to clean the primary air ports by using a rod, but a delayed response might cause the situation to worsen and require the addition of auxiliary fuel.

Students are able to interact with the tutor through a hierarchy of menus, which allow them to check for a tube leak, clean the smelt spout, select the alarm board or control panel board, or perform other activities (see Figure 1). While the simulation of the recovery boiler is running, the operator can view the boiler from many directions and can focus on several components, such as the fire bed in Figure 4. The tutor provides assistance through visual clues, such as a darkened smelt bed, acoustic clues, ringing alarm buzzers, textual help, explanations, and dialogues, such as the one in Figure 2. The operator can request up to thirty process parameters on the complete panel board (Figure 5), view an alarm board (not shown), change twenty setpoints, and ask questions from a menu, such as "What is the problem?", "How do I get out of it?", "What caused it?", and "What can I do to prevent it?"³ The operator can request meter readings, physical and chemical reports, and dynamic trends of variables. All variables are updated in real time (every 1 or 2 seconds).

In addition to providing information about the explicit variables in the boiler, RBT provides tools designed to help students reason about implicit processes in the boiler. One such tool is composite meters (left side of Figures 1 and 4 through 6) that record the state of the boiler using synthetic measures for safety, emissions, efficiency, and reliability of the boiler. The meter readings are calculated from complex mathematical formulae that would rarely, if ever, be used by operators themselves to evaluate the boiler. For instance, the safety meter is a composition of seven independent parameters, including steam pressure, steam flow, steam temperature, feedwater flow, drum water level, firing liquor solids, and combustibles in the flue gas. Meter readings allow students to make

³RBT answers these four questions by cutting text from a file loaded with the specific emergency. The tutor does not "reason" about its own knowledge to answer these questions. The tutor does, however, have knowledge about the domain as described in [Woolf et al., 1986].

[Note: A partial blackout is in progress and it will lead to deterioration of the combustion process if no action is taken. Meters indicate that CO₂, SO₂, and O₂ readings are abnormally high.]

[An alarm buzzer sounds.]

Tutor: What action are you going to take?

Student: Check smelt spout cooling water. [Student's action does not result in a change in the boiler.]

Tutor: It looks as though the liquor isn't burning . . .

Student: Help. What is going on? [Student requests help.]

Tutor: A partial blackout is occurring and this causes TRS, CO₂, SO₂, and O₂ to increase. Steam flow and steam temperature are dropping. I think part of the bed is not burning.

Student: Rod the primary air ports. [A small improvement in the boiler readings can be observed.]

Tutor: Your action is correct; however, at this time rodding alone is not effective . . .

Student: Raise the liquor temperature. [Student's action does not reverse the emergency and a new alarm buzzer sounds.]

Tutor: Rodding the air port apparently was not enough to undo the blackout.

Student: Increase the oil fuel supply. [Student's action reverses the emergency.]

Tutor: That solved the problem alright. Good thinking.

Analysis of the Problem: You had a partial blackout caused by plugged primary air ports and a cold bed. Partial blackout can be effectively treated through a combination of rodding the primary air ports and adding more heat. The problem can be avoided by keeping the air ports clean.

Figure 2 Dialogue Between Tutor and Student

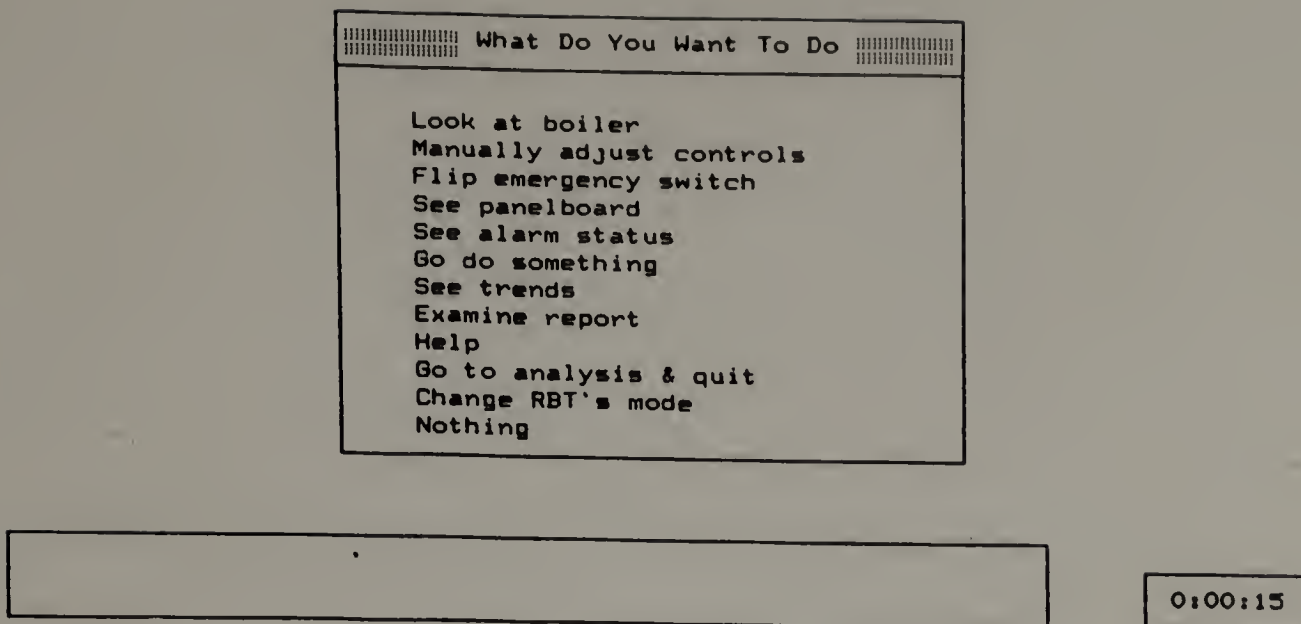


Figure 3 Tasks Performed on the Boiler

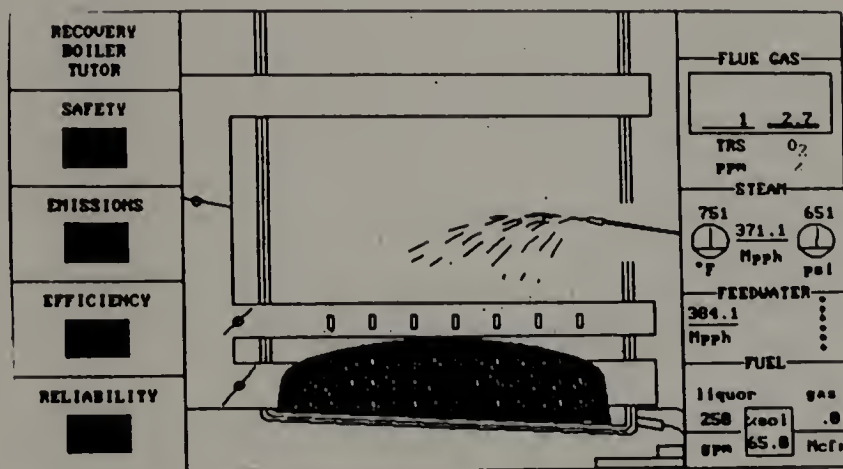


Figure 4 Focused View of the Fire Bed

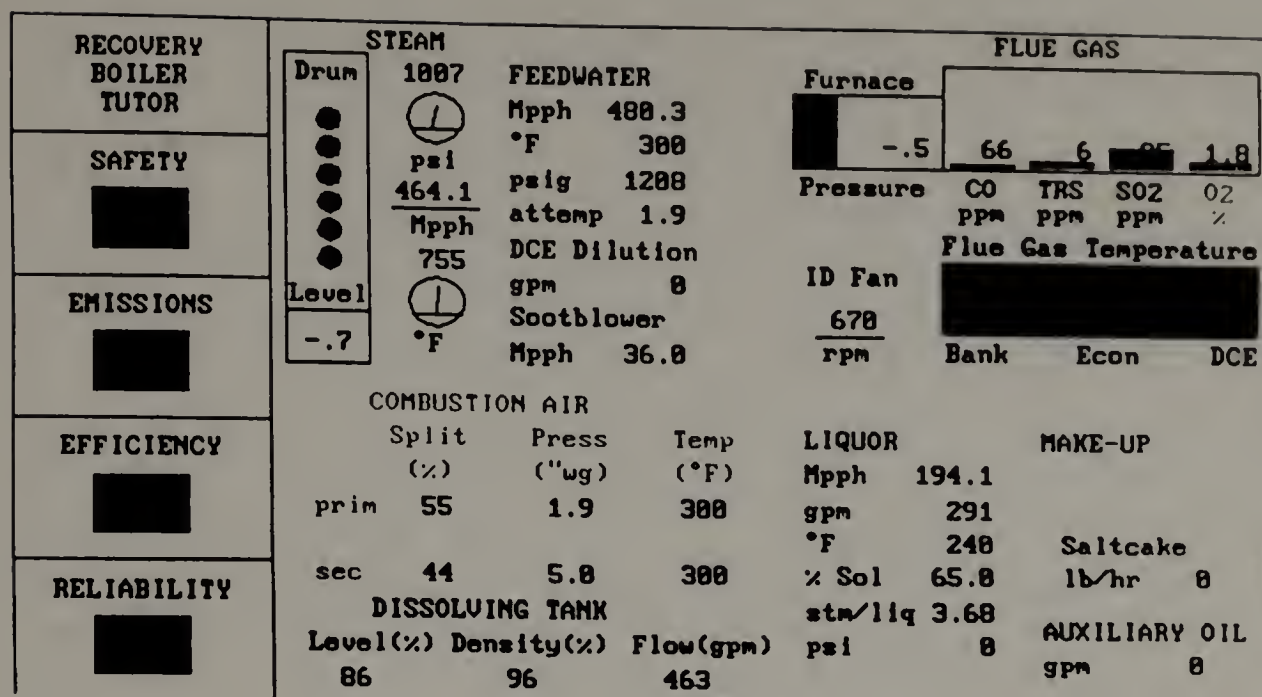


Figure 5 The Complete Control Panel

inferences about the effects of their actions on the boiler using characteristics of the running boiler.

Other reasoning tools display trend analyses (Figure 6), and animated graphics such as those shown on boiler figures. Trend analyses show how essential process variables interact in real time by allowing an operator to select up to ten variables, including "liquor" flow, oil flow, and air flow, and to plot each against the others and against time. Animated graphics provide realistic and dynamic drawings of several components of the boiler, such as steam, fire, smoke, black liquor, and fuel.

Each student action, be it a setpoint adjustment or a proposed solution, is given an accumulated response value, which reflects the operator's overall score, how successful or unsuccessful his/her actions have been, and whether the actions were performed in sequence with other relevant or irrelevant actions.⁴

⁴These meters are not yet available on existing pulp and paper mill control panels. If they prove effective as training aids, they could be added to the panels.

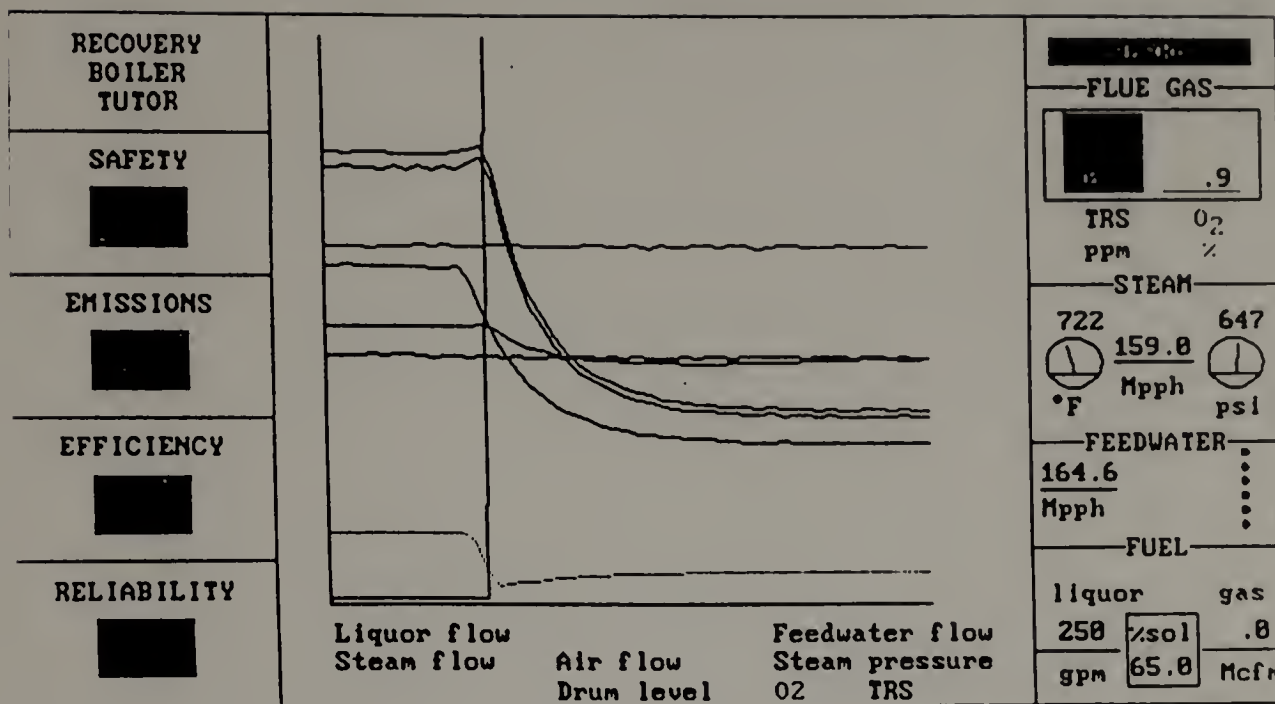


Figure 6 Trends Selected by the Operator

The overall operator's score might be used to sensitize the tutor's future responses to the student's record. For instance, if the operator has successfully solved a number of boiler emergencies, the accumulated value might be used to temper subsequent tutoring so that it is less intrusive. Similarly, if a student's past performance has been poor, the accumulated value could be used to activate more aggressive responses from the tutor.

RBT has been well-received as a training aid in the control rooms of pulp and paper mills throughout the United States.⁵ Informal evaluation suggests that operators enjoy the simulation and handle it with extreme care. They behave as they might in actual control of the pulpmill panel, slowly changing parameters, and examining several meter readings before moving on to the next action. Both experienced and novice op-

⁵RBT was developed on an IBM PC AT (512 KB RAM) with enhanced graphics and 20 MB hard disk. It uses a math co-processor, two display screens (one color), and a two-key mouse. The simulation was implemented in Fortran and took 321 KB; the tutor was implemented in C and took 100 KB.

erators engage in lively use of the system after about a half-hour introduction. When several operators interact with the tutor, they sometimes trade "war stories," advising each other about rarely seen situations. In this way, experienced operators frequently become *partners* with novice operators as they work together to simulate and solve unusual problems.

2.1.2 *Caleb for Teaching a Second Language*

Caleb is an intelligent tutoring system for teaching languages [Cunningham, 1986]. It is based on a powerful pedagogy called "The Silent Way," a method developed by Caleb Gattegno [1970] which uses nonverbal communication within a controlled environment to teach second languages. This directed discovery environment is designed to engage a student in an active learning process.

The tutor teaches Spanish by using graphic representations of Cuisenaire Rods⁶ to generate linguistic situations in which new words, nouns, verbs, and adjectives are given meaning by the actions of the rod. For example, a rod might be shown playing various roles: as an object to be given or taken by the student, or as an instrument used to brush teeth. As each new rod is presented, students theorize about what situation is encountered, type in replies to the machine tutor, and revise their hypotheses as needed.

After the computer presents a visual situation using the rod, the student types a phrase to describe the situation in the text window. For example, if a picture of a rod appears while the words "una regleta" are displayed, the student types "una regleta," as seen in Figure 7A. If the student theorizes that a newly presented word such as "blanca" describes the size of the rod, the new word can be added to the definition. Meanwhile, the student will have learned to write the word, spell it, and place it correctly in a sentence (Figure 7B). The student will also have classified the word as a descriptor and will have invented phrases using it. However, if the student's assumption is wrong and this word is not a descriptor, the tutor will correct him/her. Making and correcting hypotheses is central to language learning and many other areas. By refining the meaning of a word, the learner approaches mastery of the word by approximation.

⁶Originally developed by Gattegno for teaching arithmetic.

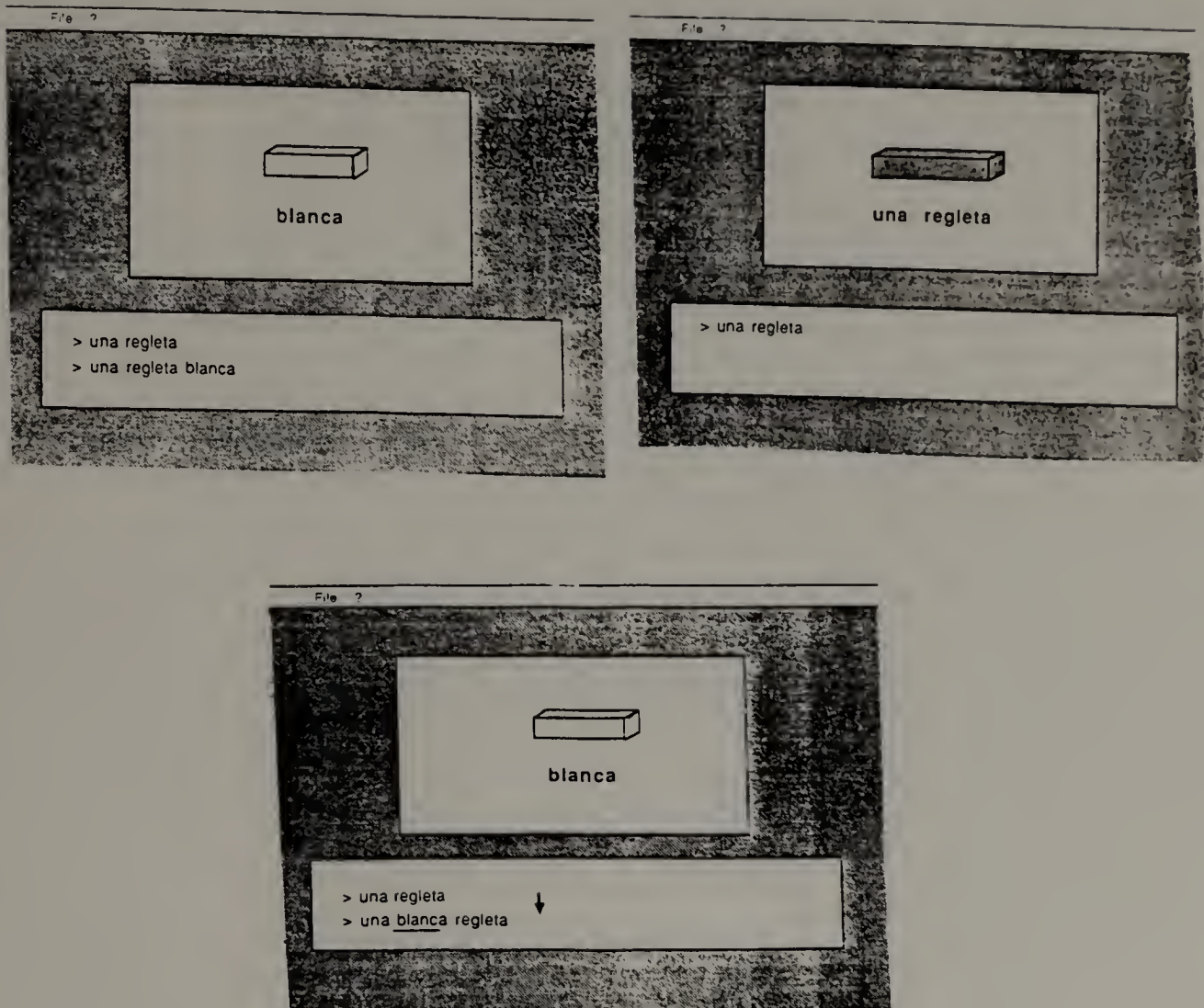


Figure 7 Caleb: A System for Teaching Second Languages

Figure 7 shows the student's interaction with the tutor. In Figure 7A, the tutor presents a new piece, a rod located in the center box. The student responds by typing the word for the new piece at the cursor. In Figure 7B the student invents a new phrase by combining old pieces with the new one. Figure 7C shows how the tutor corrects a student who places the adjective before, rather than after, the noun.

The tutor doesn't *repeat* new words over and over as in traditional language training classes. Rather, it *mentions* each new word once, and only once. It provides minimal *pieces* of the new language, a *piece* being defined as a phoneme, syllable, word or phrase (Figure 8). Pieces are aspects of the language that the student can't invent, such as vocabulary and pronunciation. The tutor communicates silently using *gestures*, edit signals, *pantomime* and the rods, only "speaking" to provide words that the student has not yet heard. Icons used to represent these gestures, edit signals, and pantomime are presented in Figure 9. These icons are used by the tutor, who plays the role of orchestrator and monitor rather than information giver.

With the introduction of verbs, action becomes possible. For example, the tutor prompts the student to ask the tutor to take the rods by indicating a hand taking two white rods in the graphics window. The student types the command, "Toma dos regletas blancas" ("Take two white rods"), and the tutor removes the rods from the screen.

The student uses both word-oriented responses typed at the keyboard and action-oriented responses performed with the mouse and pictured objects. For example, when the tutor gives the command, "Toma dos regletas blancas" ("Take two white rods"), the student responds by using the mouse to take two pictured white rods with a grasping-hand-shaped cursor.

The nonverbal communication of a human Silent Way tutor (the gestures, nods, hand signals, pantomime) are represented on the computer screen. Students learn to interact with the tutor and to recognize, for example, when it is their turn to produce a sentence, when the tutor is about to say something, when the tutor expects more than the student has produced, when an error needs correcting, or how to get help when they are stuck.

Errors are indicated as the student tests and revises his/her theories about the language. When an error does occur, the student is neither deluged with entire sentences,

<u>THEME</u>	<u>PIECES</u>	<u>POTENTIAL MISCONCEPTIONS</u>
1. noun	a rod una regleta	word order/agreement
	<i>example response: una regleta</i>	
2. adj	white blanca grey gris striped listada dotted punteada	word order/agreement
	<i>example response: una regleta blanca</i>	
3. conj	and y	word order/use in series
	<i>example response: una regleta gris y una regleta blanca</i> <i>una regleta gris, una regleta blanca, y una regleta negra</i>	
4. numbers	two dos -s three tres -s one una	word order/agreement
	<i>example response: dos regletas blancas . . . y tres regletas negras</i>	
5. noun	one blank	use in series/agreement
	deletion ones <i>example response: una regleta roja y una blanca</i> <i>dos regletas blancas y tres negras</i>	
6. verb +	take toma	word order/agreement
	direct object <i>example response: toma una regleta gris</i> <i>toma una regleta gris y tres blancas</i>	
7. verb +	give me dame	word order/pronoun use/agreement/case
	indirect object <i>example response: dame una regleta blanca</i> <i>dame tres regletas negras y dos blancas</i>	
8. pronoun	it, them la, las	word order/pronoun use/agreement/case
	<i>example response: toma una regleta blanca y dámela.</i> <i>toma tres regletas negras y dámelas.</i>	

Figure 8 Themes and Pieces in the Spanish Curriculum

Label	Idea to get across	Icon
go	it is your turn to do something	blinking cursor
wait	tutor busy, do not worry about nothing happening	Mac watch
attention	tutor about to do something new	sound
signal OK	let student know his response is OK	happy face
signal error	let student know he has error	sad face/Mr. Yuk
puzzled/ repeat	say/do again (unintelligible response)	puzzled face /again sign
more	say/do more (incomplete response)	hand pull
help	help is available	?
dictionary	list of words already covered	book shape
throw out	extra stuff, do not need	Mac trash can
stop	save and quit = good bye	hand waving bye /stop sign
pause	take a break / stop timer	coffee cup
take	icon for mouse action	grasping hand
give	icon for mouse action	open hand
pacing regulator	slow down or speed up	speedometer
frustration gauge	student emotional state	thermometer
error correcting	word processing techniques	highlight blinking cursor placement arrows fade in line

Figure 9 Communication Icons in Caleb

nor provided a correct model to imitate. Instead, the precise location of the error is pointed out, as in Figure 7C, with underlining so that the student may correct it. The goal is to allow students to develop their own sense of correctness or *inner criteria* for the new language.

The tutor monitors student input for correctness. A fault tolerant parser filters "noisy" (inconsistent or incoherent) input so that some errors are ignored and some are treated depending upon the situation. This is done to reduce the amount of corrective feedback received by the student. When the tutor treats an error, only the piece that requires correction (noun, verb, or adjective) is pointed out; secondary pieces or those pieces studied earlier are ignored. Students edit their own input. Caleb uses typical word-processing techniques such as highlighting to indicate words, syllables, and parts of syllables that need correcting. Figure 7C shows how the tutor indicates a misplaced word error; the student is then expected to move the highlighted word to the correct place.

The tutor bases its decisions about correcting student work or the order of presentation on its current goal, e.g., to teach a new piece or old theme (Figure 10). It uses five contexts to determine the number of times the student should practice the piece.

		INTRO	PRACTICE	MORE PRACTICE	REVIEW	
MISSING PIECE	cp	treat	treat	treat	treat	cp = current piece op = old piece ot = old theme
	op					
	ot					
MISPLACED PIECE	cp	treat	treat	treat	treat	
	op	treat				
	ot	ignore				
WRONG PIECE	cp	treat	treat	treat	treat	
	op	treat	treat			
	ot	ignore	ignore			
EXTRA PIECE	cp	treat	treat	treat	treat	
	op	ignore	treat			
	ot	ignore	ignore			
SPELLING	cp		treat	treat	treat	
	op	ignore	ignore	treat		
	ot		ignore	ignore		

MISSING ACTION	cp	treat	treat	treat	treat
	op				
	ot				
WRONG ACTION	cp	treat	treat	treat	treat
	op	treat			
	ot	ignore			
EXTRA ACTION	cp	treat	treat	treat	treat
	op	ignore	treat		
	ot	ignore	ignore		

Figure 10 Error Therapy for the Silent Tutor

For example, in the *intro context* the tutor simply presents the first example on the list of examples associated with each piece. When the tutor is in the *practice context*, the example source remains the same, and the tutor moves down the list in a fixed order. In the *more practice context*, examples are chosen randomly from the piece example list. In the *review context*, examples are taken from an example pool of the current theme or of old themes. In the *error context*, examples come from the list associated with the error itself.

2.1.3 Physics Tutor

Another group of tutors has been built in cooperation with the Exploring System Earth (ESE)⁷ consortium for teaching elementary physics at the high-school and college levels. These tutors are based on interactive simulations that encourage students to

⁷ESE is a partnership of academic, industrial, and government institutions dedicated to the development and dissemination of learning environments to supplement introductory science instruction in high schools, colleges and universities. The schools include the University of Massachusetts, San Francisco State University, and the University of Hawaii; the industries include Hewlett-Packard Corp.

work with elementary concepts of physics, such as mass, acceleration, and force. The goal is to help students generate hypotheses as necessary precursors to expanding their own intuitions. We want the simulations to encourage students to "listen to" their own scientific intuition and to make their own model of the physical world before the tutor advises them about the accuracy of their choices. Cognitive research results help us to identify and encode knowledge about how students work in physics. This allows the system to track the student's cognitive processes as he/she develops such hypotheses. These tutors have been described in Woolf and Cunningham [1987] and Woolf and Murray [1987].

Figure 11 shows a simulation for teaching concepts in introductory statics. In this example, students are asked to identify forces and torques on the crane boom, or horizontal bar, and to use the mouse to construct regions on the screen or force vectors on top of the boom and cable. When the boom is in static equilibrium, there will be no net force or torque on any part of it. Students are asked to solve both qualitative and quantitative word problems about several positions of the boom. For example, they are asked if the force on the cable is less than or greater than the force when the angle of beam to wall is less than 90° , 90° , or greater than 90° .

If the student had specified the forces for the crane boom shown in Figure 11A and had left out a force vector located at the wall and pointing upwards, there are many possible tutorial responses depending on the particular strategy in effect. The machine might present an explanation, a hint, provide another problem, or demonstrate that the student's analysis leads to a logical contradiction. Still another response would be to withhold explicit feedback concerning the quality of the student's answer and demonstrate the consequence of omitting the "missing" force; i.e., the end of the beam next to the wall would crash down. Such a response would show the student how his/her conceptions might be in conflict with the observable world, and it might help him/her visualize both his/her internal conceptualization and the scientific theory.

A second physics tutor is designed to improve a student's intuition about concepts such as energy, energy density, entropy, and equilibrium in thermodynamics (see Figure 12). It makes use of an over-simplified but instructive simulated world consisting of

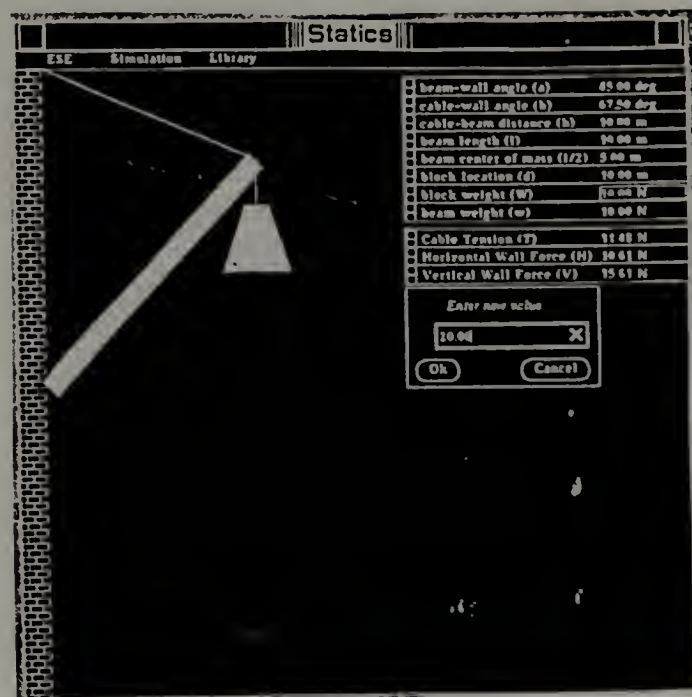
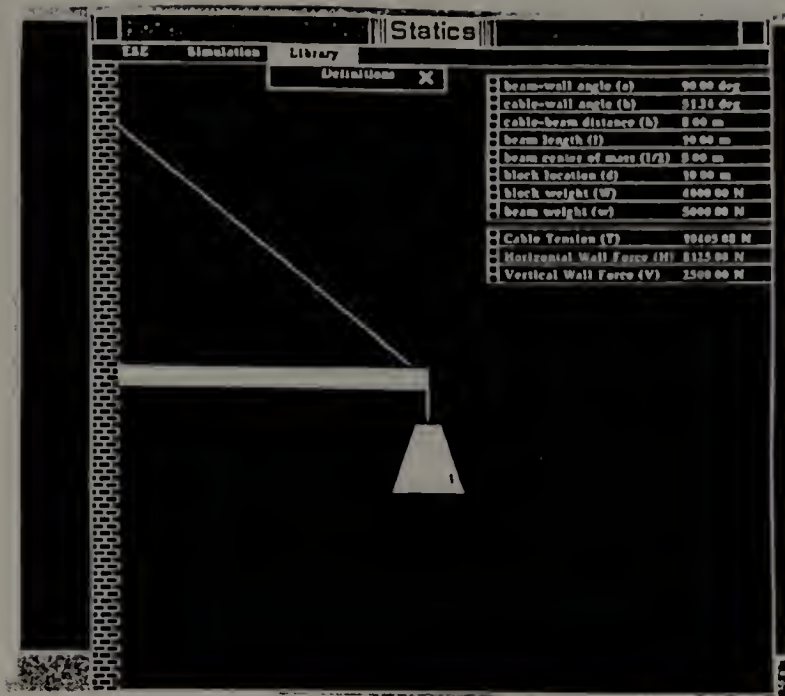


Figure 11 Statics Tutor: Simulation

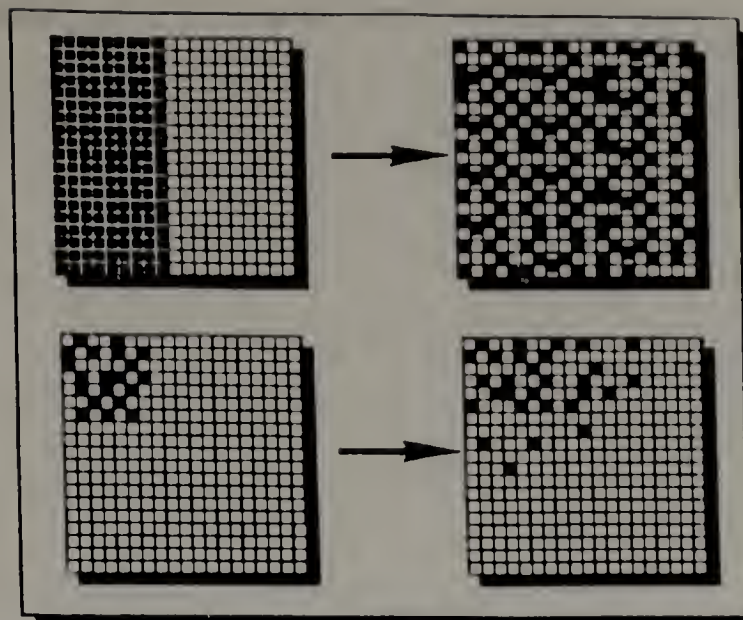


Figure 12 Thermodynamics Tutor: Simulation

a two-dimensional array of identical atoms to teach the second law of thermodynamics.⁸ This law is taught at the atomic level [Atkins, 1982] within a rich environment for observing and testing the principles of equilibrium, entropy, and thermal diffusion. The student is shown, and is also able to construct, collections of atoms that can transfer heat to one another through random collision.

Like the statics tutor, the thermodynamics tutor monitors and advises students about their activities and provides examples, analogies, or explanations. In this simplified world, the atoms have two states: grounded and excited. The excitation energy is transferred to neighboring atoms through random "collisions." Effectively, any excited atom will give its high energy to a neighboring atom if that second atom is grounded and has been selected by our random number generator routine. Students can specify initial conditions, such as which atoms will be excited and which are grounded. They can observe the exchange of excitation energy between atoms, and can monitor, via graphs and meters, the flow of energy from one part of the universe to another as the system moves toward equilibrium.

⁸The second law states that heat cannot be absorbed from a reservoir and completely converted into mechanical work.

In this way, several universes can be constructed, each with specific areas of excitation. For each system, regions can be defined and physical qualities such as energy density or entropy can be plotted as functions of time. Various shaped regions within each universe can be analyzed and monitored. Several universes can be constructed, each with specific areas of high energy and associated observation regions. For each observation region, concepts such as temperature, energy, and energy density, can be plotted against each other and against time. Thermodynamic principles can be observed in action; for example, heat transfer can be observed through random collision, and entropy can be observed as a function of initial system organization.

At any time the student can modify the temperature of the system, the number of collisions per unit time, and the shape of the observation regions. These parameter changes cause changes in the system. All student activities, including questions, responses, and requests, are used by the tutor to formulate its next teaching goal and activity. Each student action is used by the tutor to reason about whether to show an extreme example or a near-miss one, or to give an analogy or ask a question.

2.2 Additional Examples

Other knowledge-based tutors, in addition to those described above, have been brought into classrooms and training sites. They have been shown to teach more effectively and efficiently than traditional lecture-style classes [Anderson & Reiser, 1985; Woolf et al., 1987], to engage students in real-world problems, to provide complex communication to other networked students and data, and to move students toward greater competency at an earlier stage in their education [Pollack, 1987]. Each type of result is detailed below. A variety of knowledge-based systems are described providing evidence for these results. Additional systems are described in Chapter 5 in more detail in connection with a specific tool used or issue addressed. Although each system was tested with only a small sample and used for limited time, results appear promising.

2.2.1 *More Effective and Efficient Teaching*

A programming course at Carnegie-Mellon University used an intelligent tutor and showed a 43% improvement in test performance at the end of the semester and 30% reduction in learning time [Anderson et al., 1984; Anderson et al., 1985; Anderson & Reiser, 1985]. Using traditional lectures, students spent about forty hours covering the first six lessons of a LISP course; whereas, with the intelligent tutoring system and the lectures, students were able to complete the same lessons in only 15 hours.

SOPHIE, a **S**ophisticated **I**nstructional **E**nvironment for electronic troubleshooting, was able to reason about whether a student's solution was appropriate, given previous information [Burton & Brown, 1982]. SOPHIE reasoned about a student's attempts to debug a simulated electrical circuit. The system used a complex simulation of an electronic circuit to test hypotheses. For example, it could test a student's conjecture about the cause of a problem and "refuse" to carry out probes that the student proposed tests that were unimportant to the solution of the problem (see Section 2.6.2 for more discussion of SOPHIE).

As part of the Athena project at M.I.T., language instruction combines video and computers to provide more efficient teaching, by using foreign language as it was intended, i.e., to solve problems connected with real world activities [Pollack, 1987]. For example, one program places a student in a simulated walk through Bogota, Columbia accompanying a Spanish-speaking scientist. The scientist has lost his memory and is trying to discover where he left a vial of virus that imperils Latin America. Success in finding the vial is directly related to the student's understanding of the scientist's conversation and that of other people he/she may encounter. The students ask questions in Spanish by typing sentences into the computer. The particular video segment shown depends on the instructions and questions typed by the student. The program is based on goal-directed language learning and experiential training and encourages students to experiment, hypothesize, and employ language to solve real problems faster and earlier than they would otherwise in the classroom [Pollack, 1987].

2.2.2 *Engaging Students in Real-World Problems*

Other knowledge based systems, though not “intelligent” in the sense that they don’t employ AI principles, still provide the student with complex real-world problems and tools. Students have access to complex scientific data or engage in complex activities communicating through an electronic medium. The goal is to improve a student’s reasoning ability by placing him/her in relevant and interesting situations. For example, one system at Stanford University enables students of French history to become involved in social issues and actions of the 17th century [Pollack, 1987]. The program, developed by Carolyn Lougee, Professor of French history, uses rich simulations to provide a taste of life in the time of Louis XIV. Students play the role of male landowners and attempt to increase their wealth and status by making proper investments and by properly choosing a woman to court. The program helps them visualize the fact that life during that time revolved around the annual harvest. Such an issue rarely comes across in textbooks, but in the simulation, students can see that good harvests and clever politics will affect a person’s wealth, prestige, and ability to propose marriage.

Another system provides a rich framework for solution of physics problems [Bork, 1987]. The simulations and computer-derived questions serve as the basis of an entire introductory physics course. This course has been chosen over a traditional lecture-style course by every 3 out of 4 students for their first-year physics course at the University of California, Irvine [Bork, 1987]. The course provides simulations of physics topics, demonstrating, for example, how “thrown” bodies behave under the action of various force laws. Students are free to change the initial conditions, constants in the force law, or scale of plotting. These simulations are supplemented with associated material that gently guides the student to explore different simulated situations. “Bare” simulations do not seem to work well with many students, particularly unmotivated ones.

The course is organized around a set of on-line exams with problems generated at random or selected from a pool of stored items. Problems are generated to address the difficulty a student might be experiencing. The generator produces a widely varying collection of problems of a given type. Multiple choice is never used, and the same test is never given twice under any circumstance. If a student has trouble with a problem, a detailed learning aid is presented. Although teachers like to think that students use tests

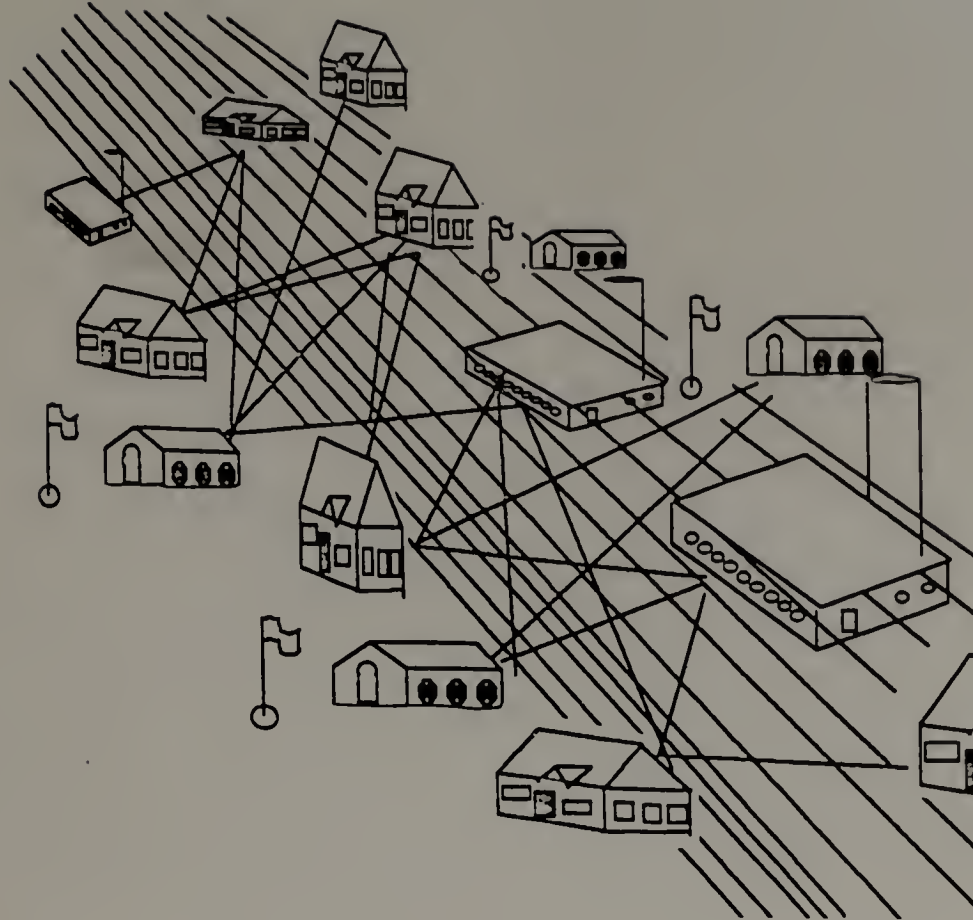


Figure 13 Electronic Networks Connecting People with Information

as a learning experience, few students would agree. However, in evaluative studies of this course, 80% to 90% of the students identified the tests as the *major* source of (though perhaps not motivation for) learning.

2.2.3 Complex Communication Networks

Electronic networks place students in long-distance communication with other students who cooperate on real problems (see Figure 13). Computer networks are an important educational technology which enable scholars from around the world to communicate with others and to access information from vast computer data banks. Networks provide a model of how science can be accomplished in the real world by enabling teams of people to share results and to critique the work of others. Such systems do not necessarily use Artificial Intelligence, yet provide a powerful complement to systems that do.

Teams of experts in widely scattered locations already use electronic networks to work closely together on similar problems. Papers are co-authored globally, conferences planned, and proposals submitted—all without the authors leaving home. At least one programming “language” and one book, *Common Lisp* by Guy Steele [1984], have been designed and written entirely on an electronic network in which more than 50 people

participated. Except for two one-day face-to-face meetings, all the discussions and designs about the Common Lisp language and the book over more than two years were done through the computer network. Authors sent files and discussions to each other for editing or critique. Each author had access to shared references and clerical and stenographic services. Automatic storage of the discussion produced more than 1100 pages which proved invaluable in the preparation of the book.

A national program called KIDNETWORK establishes networks between schools and enables children to produce and communicate significant scientific results [Tinker, 1987]. Using networked microcomputers, students from nearly 300 high-school classes make scientific measurements that are evaluated, synthesized, and reported nationally. One system measures acid rain. Each classroom, on a prearranged date, makes external measurements of rainfall. Data is directly entered into a microcomputer and then into a central data-bank. The results are then tabulated and returned to the students, who can view their own and national results through color graphics. In this example, students perform a real activity with important scientific results. Their statistics provide better data than currently available through the Department of Environmental Health as a result of the vast number of sites. This project is sponsored by the National Geographic Society and Apple Computer Corporation.

Electronic networks allow students to have a real audience for their work, namely other students from other regions and other countries. Southworth [1988] has set up communication between students in Hawaii, California, and the Soviet Union. In one exchange, students described their family, schools, and pets. Telecommunication capability coupled with local area networks allows students to extend their influence and "voice" across significantly large distances.

Other sources of information are available through networks. Recorded knowledge from libraries and museums can be transferred to more universally accessible machine-readable data-banks. The contents of central files are retrievable by anyone at any time. These files contain data such as daily stock-market closings, or production figures for tin in Bolivia. Home computers, including video and acoustical two-way electronic equipment, already place people in contact with research results, translations, and facsimiles of current literature such as newspapers, advertisements, and library material. Through

computer networks, the isolated student or the small-college student can access the same material as that available to students at larger universities.

2.2.4 *Complex Information Management*

Several computer features facilitate a student's ability to think about complexity. One example is hypermedia, which allows students to wander through massive amounts of electronically stored information to retrieve complete texts, stories, biographies, graphics, animation, sound, movies, motion video, or audio as needed and to arrange them in terms of their own priority (Figure 14, also see Section 5.6.2). The complex hypermedia system at Brown University shows that words and pictures need not be organized sequentially into hierarchies [Yankelovich et al., 1985]. The system, called Intermedia, is used to teach literature, biology and other topics. Students retrieve complete text or graphics as needed and arrange them on the screen in terms of their own priority. Documents have arbitrary beginnings and endings and can be explored rather than read sequentially. Such systems provide for non-sequential reading and writing and enable users to browse through networks of information, to sample bite-sized pieces of information, and to add to this living data-base by inserting their own information and links. Any document can be annotated in this way and will contain programmable links to other documents or files. Links can lead to pictures, video sequences, or music. In a system produced at Harvard University and Boston University, Greek classics are stored on computers along with English translations, commentaries, lexicons, and illustrations [Pollack, 1987]. A student coming across the name of an unfamiliar character or god in the "Iliad," for instance, can immediately jump to biographical information about the character or to a sketch of that character. The system has already been used in teaching part of a course at Harvard. Such systems provide virtually instant access to all kinds of data- historical papers, museum archives, reference books, business data-bases, and on-line educational resources. New documents can be created by chaining existing ones together. Data and graphics can be programmed to allow users to set their own course through islands of information. A good hypermedia system encourages browsing and hunting, rather than reading from beginning to end. Several dozen hypermedia systems can be purchased such as Hypercard for the Mac and the Intermedia system for any Unix-based Macintosh.

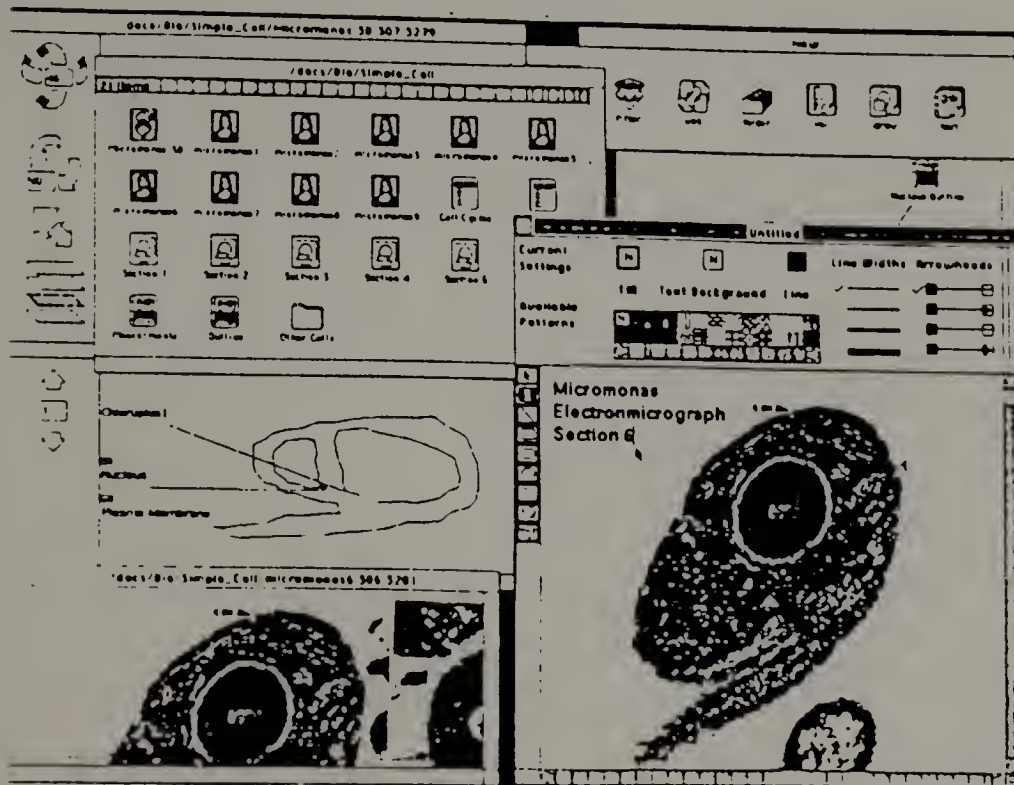
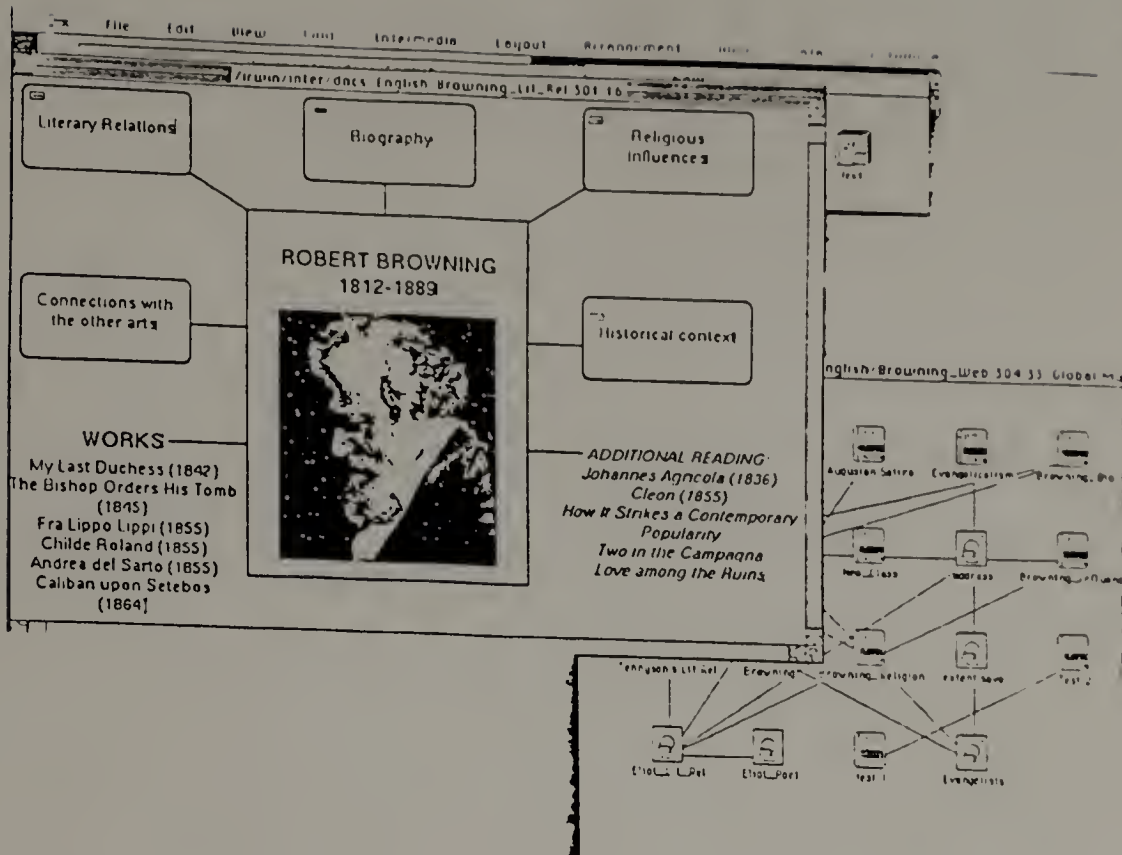


Figure 14 Screens from Intermedia Project [Yankelovich et al., 1985]

Studies report that using a computer to store and retrieve information influences younger students to reason about complexity. For example, the learning and thinking of eighth-graders were tested after they wrote a science term-paper in which half the students used regular print media as source documents and half used computerized encyclopedias which could be searched superficially across several topics by simply entering additional key words [Krendl & Fredin, 1984]. The study predicted that students provided with computerized information would improve their "horizontal knowledge" (knowing a little bit about a wide variety of topics) over those students using only print media. The control students, it was suggested, would have higher levels of "vertical knowledge" (knowing a great deal about a few topics) since they were more likely to focus in-depth on a few topics and to read one complete article before retrieving a new volume from the shelf. Surprisingly, students who used the electronic encyclopedia scored significantly higher on measures of *both* horizontal and vertical knowledge. This result is explained in part by the availability of printers which recorded electronic encyclopedia entries and allowed students to read complete entries in depth. The students noted that the electronic encyclopedia and the printer eased the process of information gathering and the process of writing the paper.

Other programs allow students to visualize complex phenomena that would be physically impossible or prohibitively expensive to experience outside of the classroom. A series of physics simulations gives students an appreciation of topics such as relativity, electromagnetic theory, and waves [Carbrera, 1987]. Students can use simulations to "experience" travel in a spaceship at the speed of light or to see the path of electrons responding to different forces.

GUIDON made complex medical knowledge accessible to a medical student [Clancey, 1982; Clancey, 1986b; Clancey, 1987]. It used a mixed-initiative dialogue and a case-method paradigm to tutor information from an expert system. The primary expert system used was MYCIN [Shortliffe, 1974], a rule-based system which contains approximately 1,000 rules for solving medical problems about infectious disease, diagnosis, and therapy. NEOMYCIN, a later version of GUIDON, was based on making some of the design changes discussed in Section 4.3.1 [Clancey & Letsinger, 1982].

2.2.5 Individualized Teaching

Several systems monitor a student's behavior and advise him/her about possible misconceptions and common errors. Systems at Yale University can "look over the shoulder" of a novice student-programmer, somewhat understanding his/her intentions and providing tutorial advice [Johnson & Soloway, 1984]. The system detects semantic errors in a programming assignment and understands between 70% and 80% of all student programs written to solve a particular simple problem. Semantic errors are mistakes in the code which cause inconsistent behavior of the running program, but are not severe enough (i.e., not syntactic errors) to keep the program from working. The running program is inconsistent with the student's intentions. To understand the student's intentions, the system first identifies the goals of a given assignment and then determines how the student achieved each goal. Then it determines how the student solved, or failed to solve, each of the goals required by the assignment. The system's identification of bugs is derived from a process model of programming.

In another system, BUGGY replicates the errors individual students make on subtraction problems. It demonstrates that subtraction errors are explicit, systematic deviations from correct procedures [Brown & Burton, 1978]. The system was not built to teach, rather to train teachers to recognize and classify individual subtraction errors. It encoded both correct and incorrect processes of simple arithmetic in a procedural network and could automatically produce 330 "bugs" for subtraction.

BIP was able to individualize its advice to students of elementary programming and to provide custom-tailored exercises at a level appropriate to the individual programmer [Barr et al., 1976]. The system inferred the student's ability by testing him/her, evaluating the written programs, and selecting new exercises consistent with a model of the student's presumed skill.

WEST provided an individualized coaching environment for a game which exercised elementary-level arithmetic skills (see Figure 15) [Burton & Brown, 1982]. The object

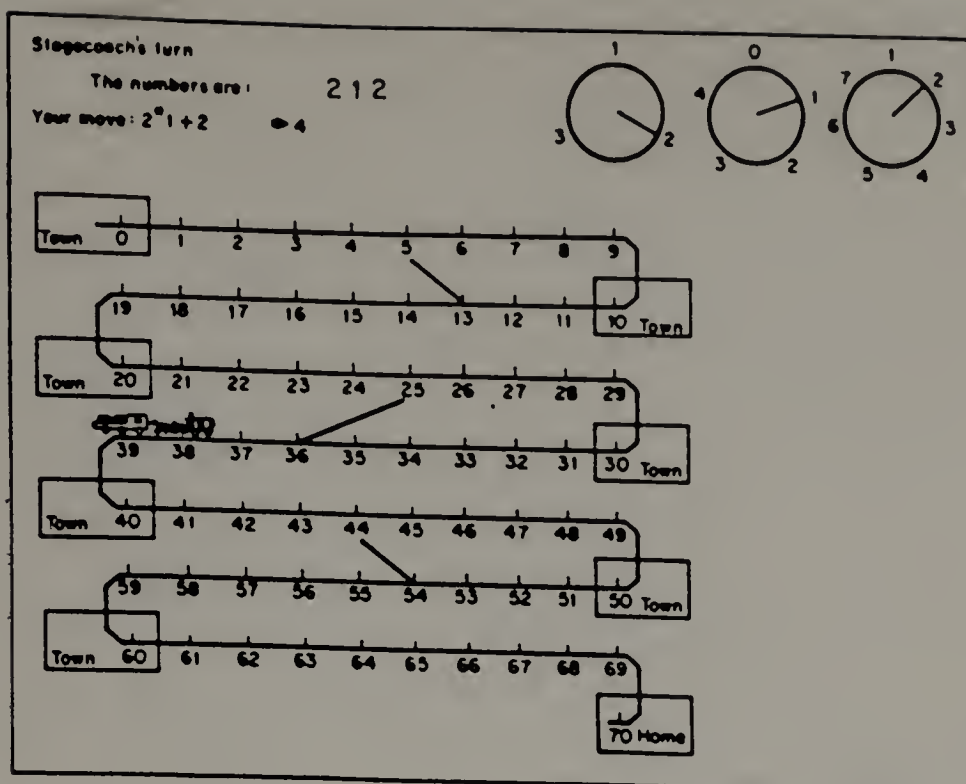


Figure 15 "How the West Was Won" [Burton & Brown, 1982]

of the game was to move a player across an electronic gameboard by an amount equal to the value of an algebra expression that the student constructed from values produced randomly by three dials on the screen. The coach individualized its description of better moves and missing skills based on a model of the student's skills.

2.3 Knowledge-Based Tutors versus Computer-Aided Instructional Systems

The knowledge-based systems described above, which utilize Artificial Intelligence methodologies, are intended to "understand" *what*, *whom*, and *how* they teach and then to tailor their content and method to the individual student. Teaching material does not consist of a repertoire of prespecified responses; rather, the system reasons about the student, the complexity and size of the information, and then it determines its response. In part, a good tutor, whether machine or human, engages a student in communication either for the purpose of presenting new material or for clarifying a body of knowledge to which the student has already been exposed. To do this effectively, a system must probe a student's knowledge, understand difficult conceptual issues, and know what mis-

conceptions might exist. Such systems require artificial intelligence (AI) techniques. In this section, we briefly describe these techniques and clarify how these systems can be distinguished from traditional CAI, or Computer-Aided Instruction.

The basic role of a knowledge-based system is to work with a student to solve real problems. Ideally, the system should

- answer hypothetical questions posed by students;
- recognize a student's problem-solving methods;
- comment on the closeness of a match between the student's solutions and that of the expert;
- provide assistance during a student's possibly incomplete problem-solving activities; and
- explain an expert's solutions and inferences.

In directing technology toward such ends, the first priority must be to build practical problem-solving knowledge into the tutor. This priority precedes that of generating fancy graphics or natural language discourse. Advances in the interface, feedback, discourse management, and curriculum design can follow once problem-solving has become a clear focus.

Knowledge-based tutors are distinguished from typical written, printed, video, and even CD-ROM educational media in that the latter are structured for direct access and thus require the machine or the user to search through an extensive table of contents, index, numbering system, or list of references, indices, numbering systems, etc. [Suthers, 1989]. In traditionally written, printed, video or CD-ROM media, the amount of information may be virtually unlimited, and the problem becomes one of selection rather than one of storage. The average student can not effectively use the indexing methods and cannot achieve a reasonable plan for learning the information. Similarly, the presentation style for existing systems is fixed. Once a sequence of material is created, the emphasis, choice of viewpoint, approach, and terminology used are unchangeable by the student. Knowledge-based tutors, however, surpass the richness of structure available in these

other media. They have the potential for dynamic, context sensitive organization and selection of material [Suthers, 1989]. They use more complex and fine-grained indexing mechanisms and can be active as well as interactive.

The most salient structural difference between knowledge-based systems and more traditional teaching machines is that the latter encode an expert's decision about how to respond to a student, whereas AI systems typically encode the reasoning that allowed the teacher or domain expert to make the decisions in the first place. Thus an AI system encodes knowledge which allows it to dynamically generate new decisions about problem-solving or pedagogy; it reasons about its decision process rather than using predefined actions. AI-based tutors contains knowledge about how and when to use domain, student, or pedagogical models, but do not contain the explicit decision within its data-base.

One impact of the shift from CAI to knowledge-based systems lies in what is transferred from expert to system [Wenger, 1987].⁹ While traditional systems encode actions and the resulting system displays those actions, AI workers transfer knowledge about how to teach and the resulting system generates exercises, responses, and examples—dynamically adapting its actions based on the encoded knowledge.

Recent systems can make decisions not anticipated by experts, and, like other artificial intelligence systems, may differ greatly from their human counterparts and may eventually outperform them in certain respects [Wenger, 1987]. After all, books have outperformed people for centuries within the narrow perspective of their ability to record information precisely and permanently. The ideal intelligent tutoring system has the potential to perform entirely autonomous reasoning. For example, such systems might engage the student in a discussion to find out what he/she knows and how he/she reasons about that knowledge. The ideal tutor does not appear to be impossible to create. Limitations stem from the state of the art in artificial intelligence and holes in our own knowledge about teaching, learning and communication (see Chapter 7), not from any basic limitations in the conceptual design.

⁹At this point only a one-way transfer (from human expert to system) is being considered. Sometime in the future we might begin to train teachers and domain experts to learn about the wealth of knowledge in an intelligent tutoring system.

An ideal tutor, whether human or machine, should understand both the knowledge to be taught and how a student might learn that knowledge. Traditional Computer-Aided Instruction (CAI) systems do not reason about the student or the domain. Typically, they cannot solve the problem given to the student; rather, they check a student's answer against a stored response and produce a "canned" statement. Some authors argue that there is a continuum of activities, such that adding "intelligence" to an existing CAI system can make it intelligent. This is not true as the following section indicates.

2.3.1 Comparing Two Instructional Systems

Compare an imaginary CAI and a knowledge-based system designed to teach about oceanography. The goal might be to reproduce a learner's visual experience and opportunity for scientific experimentation provided by a visit to the ocean itself. Obviously, a visitor face-to-face with the ocean is at a distinct advantage over either a student using one or the other system; the visitor can jump waves, experience the turbulence of the water, and run in the sand. Neither system will reproduce the sensations and tactile experiences of a day at the beach. However, the ideal system should enable students to simulate and surpass some of the learning opportunities, for example, to control water, wind, and temperature in order to test dependent parameters, such as sand and waves, and to observe how independent parameters affect dependent ones.

A knowledge-based system might allow the student to vary and control multiple wave patterns superimposed on a video sequence of waves. By using a video disk, it might make the motion of the wave and the contours of the coastline available for the student to test. Computer graphics defined by natural language might enable the student to propose hypothetical wave configurations and complex interactions between the water and the sand. Meanwhile the system might observe the student's interactions, recognize mistakes in his/her responses, and suggest repairs to possible misconceptions.

Traditional CAI systems offer less interaction and less reasoned intervention because they typically present static scenes, be it ocean or beach, and preprogrammed facts and formulas about the relationship between parameters, such as the effect of temperature and wind on waves. They typically ask questions and expect precise and inflexible answers. Even with interactive video or speech synthesis, a traditional CAI system does

not respond to the idiosyncracies of the student, allowing him/her to define scenarios, generate hypotheses, or to engage in dialogue. Such systems typically present a narrative or a sequence of videos using pre-established data and do not allow a student to reason about the unpredictability and changeability of a situation.

2.3.2 *The Technology Behind Knowledge-based Tutors and CAI systems*

Because intelligent tutors make inferences and ask questions before generating their own response, they are not "mechanistic" in the sense of CAI systems. Rather, they represent a variety of information and reason about that information. Figure 16 provides a simple comparison at the structural level between knowledge-based tutors and traditional computer-aided instructional systems (adapted from Wolfgram et al. [1987]). Obviously only a stereotypical view of each system is presented, especially since many of the features listed for knowledge-based tutors are not yet fully developed.

CAI systems often require huge static data-bases and define planned excursions through such a curriculum. Canned comments or stored tasks are often explicitly encoded and triggered by explicitly anticipated student answers. They respond the same way, whether the student is knowledgeable or confused. Simulated environments, such as microworlds, might allow a variety of student behavior, permitting student experimentation and exploration. However, such simulations are often unmonitored and thus not very effective as teachers or as vehicles for knowledge transfer to other domains. At worst, CAI systems are "electronic page turners." At best, they are precursors of knowledge-based tutors. Because they do not recognize anything beyond the "expected" student action, their repertoire of responses is often rigid, shortsighted, and tedious. Two things are clearly missing: knowledge about the student's ability and the ability to enact good, flexible pedagogy.

CAI systems are similar to books in that ideas and concepts acquired from experts are pre-organized by the author and written down explicitly for presentation to the student [Wenger, 1987]. Like books, they cannot dynamically access an expert's knowledge about the domain, nor do they "know" how to teach independent of that explicit data. For instance, they cannot answer unexpected questions from students, draw inferences about a student's knowledge, nor dynamically modify their own presentations.

KNOWLEDGE-BASED TUTORS

COMPUTER-AIDED INSTRUCTION

Data Characteristics

Uncertain and incomplete data
Dynamic and static variables

Exact and factual data
Static variables

Reasoning

Heuristic
Inferencing
Predictive
Dynamic
Bottom-up/data driven
Multiple solutions
Symbolic manipulation

Uncertain reasoning
Extensive search techniques

Mechanistic, monotonic

Simple if-then statements
Static
Control driven
Single solution
Numeric and alphabetic
manipulation
Yes/No decisions
Little search

User Interface

(Possible) Natural language dialogue	Menu/Command interface/Multiple choice
Quantitative and qualitative discussion	Quantitative discussion
Maintained by knowledge engineer	Maintained by programmer

Figure 16 Features Behind Knowledge-Based Tutors and CAI Systems

Researchers in CAI, like authors of good textbooks, produce new course material by placing new topics within a predetermined framework. They transfer branching decisions and actions from one domain to another. Under such a paradigm, software tools such as "authoring systems" are appropriate both to build and to use. There have been good CAI solutions which introduced sophisticated graphics (including PLATO [Bitzer, 1961] and TICCET, [MITRE, 1972]), employed a variety of interface mechanisms (e.g., touch-sensitive screens), and taught a variety of domains (e.g., elementary arithmetic, reading readiness, chemistry, history, and language).

Jaime Carbonell built SCHOLAR in the late 1960s, thus making the first application of AI technology to teaching systems [Carbonell, 1982]. His system is considered the forerunner of modern knowledge-based tutors. It distinguished between knowledge of the subject matter and knowledge of teaching (see Section 5.2.1) and reasoned about the student, responded opportunistically, and, for the first time, was able to parse and answered student questions. It provided the first example of a "mixed-initiative dialogue" in which either the student or machine could initiate the interaction.

Solid results have accumulated since the late 1960s when researchers were content to build illustrations that showed ideas at work on toy domains, such as geography. Slowly these ideas were shown to be powerful enough to handle practical teaching problems (e.g., tutors in electronics [Brown et al., 1982] and Pascal Programming [Johnson & Soloway, 1985]). Now researchers are building systems in part as experiments to answer truly difficult questions about cognitive processes and learning (e.g., Anderson's LISP and Geometry tutors were built in part to test his cognitive theory of learning, ACT [Anderson et al., 1984; Anderson et al., 1985], see Section 2.2.1 above).

2.4 Components of a Knowledge-based Tutor

Figure 17 illustrates the components typically associated with a knowledge-based tutor: domain model (or expert knowledge), student model (also including misconception knowledge), tutoring model (with teaching strategies), and the environment and interface (providing an envelope through which the student interacts with the system). These models interact very closely with each other in a working system, as will be described in

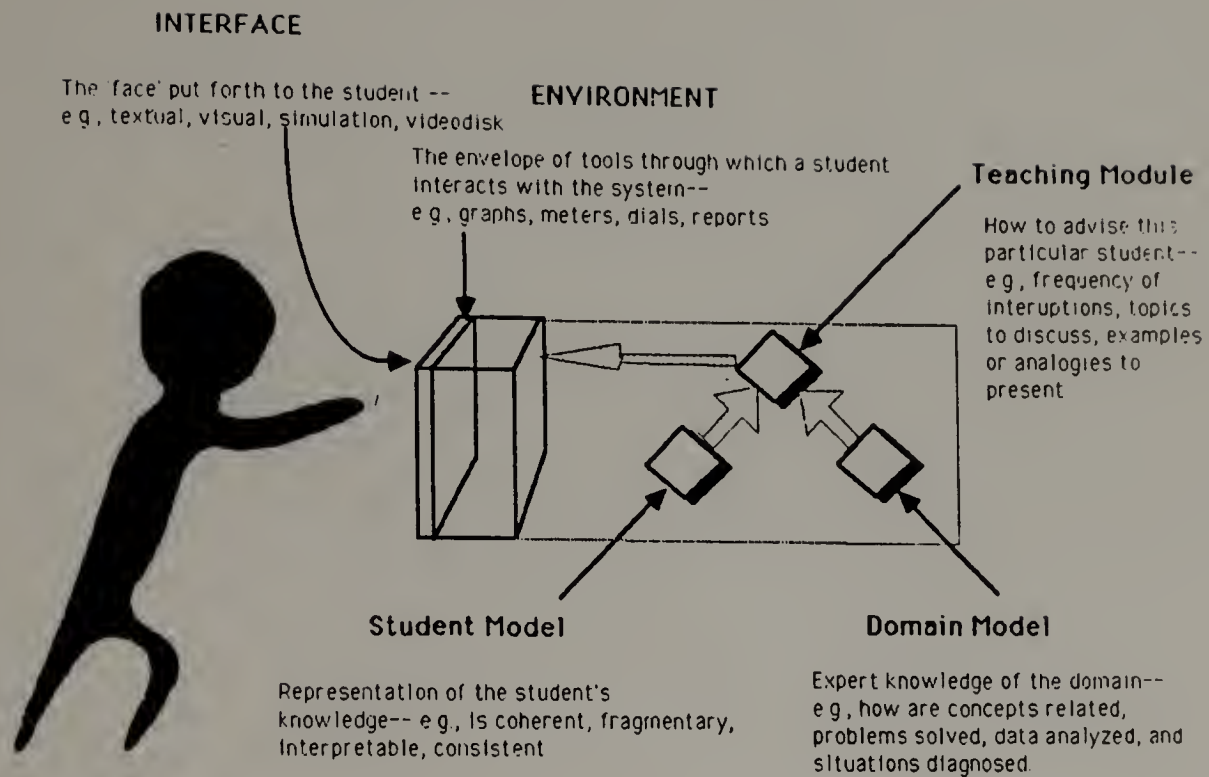


Figure 17 Components of a Tutoring System

the next three chapters. This section serves to introduce them and to describe only the environment and interface models. The other models (student, domain, and tutoring) are described in Chapters 3, 4 and 5 along with a treatment of some Artificial Intelligence techniques and knowledge engineering methodologies needed for their implementation.

Interface. The interface of a knowledge-based system provides the “face” that is put forward to the student. It provides the visual, acoustical, or textual mode of the tools that will be used in dialogue with the student. It instantiates the conversation—whether providing help, assistance, coaching, or tutoring.

For example, the primary interface mode in SOPHIE [Brown, Burton & de Kleer, 1982] was textual conversations (see Section 5.6.4). The nature of the communication was “reactive tutoring” or challenging the student through hypothesis evaluation. Providing a coaching interface for an experiment in electronics implies that the tutor provides tools for the student to perform the measurements in electronics. This was accomplished in SOPHIE through a mathematics simulation of an electronic circuit. It also implies that the system makes some assumptions about the student’s presumed knowledge and informs the interface, which can then provide coaching advice as needed.

A variety of interface types are available to the developer of knowledge-based systems. These are mentioned here briefly as an indication of the diverse ways in which a system might communicate with the student. Interface types can be used separately or in conjunction with other interfaces, e.g., a combined simulation and tutor.

Types of interfaces are as follow:

- Modeling
- Simulation
- Reaction
- Tutoring
- Coaching

Interface considerations can be divided into two categories: the mode and nature of the communication [Burton, 1987]. Under mode of communication, hardware-related issues,

such as the use of video disk, speech understanding, or natural language are considered (see Section 5.6). Under the nature of the communication, tutoring and coaching might be discussed.

The Environment. The environment provides tools and operators that the student uses while solving a problem or engaging in learning activities. For example, the student activity supported in SOPHIE was to find a fault in a piece of electronic equipment (see Sections 5.2.2 and 5.6.4) [Brown et al., 1982]. The primary tools available were textual—the student asked for measurements and hypothesized faults. The environment that supported these activities provided a simulation of the circuit, a limited natural language process, and routines to set-up contexts, keep histories, etc. The environment does not necessarily include forms of help that one would classify as intelligent.

As stated above, all components of a tutoring system interact strongly with each other. This is especially true for the environment. For example, if a system asks a student to record measurements for an experiment in optics, the interface certainly should supply measuring devices so the student may make such measurements. Environments provide a wide variety of tools and activities. Several are described here.

The Historian's Microworld provides students with a chance to discover what a historian does [Copeland, 1984]. The system is used by teams of students who are trying to find answers to perplexing historical situations. The teams brainstorm to arrive at hypotheses for historical action based on data provided by the computer based on key-word analysis students' questions. They use data from the system to reject, refine, or expand possible hypotheses. Ultimately, they "publish" their results so each team can see what data was used and what conclusions were arrived at. Below is a typical example provided by Burton [1987] and paraphrased from Copeland [1984]:

From 1565 until 1769, the "Manilla Galleon," laden with rich cargo, sailed from Manilla to Acapulco. Prevailing winds forced the ship to sail north, contact the California coast north of San Francisco and then sail down the coast to Acapulco. Because of the great distance traveled and the poor weather conditions, this nine-month voyage was very difficult. For more than 200 years, with passengers and crew weak or dying from starvation and vitamin deficiency, the galleons on this route did not stop but sailed past what is today one of the most fertile and inviting coastlines in the world. Why?

In this case, teams of students suggested reasons such as fog, hostile natives, and rocky coast.

Another example environment is provided in Anderson's Geometry tutor [Anderson et al., 1985; Anderson et al., 1981]. It makes explicit several properties of geometry (see Figure 18) and enables a student to visualize three features of the problem-solving domain that are left implicit in traditional textbooks: graphic effects the figure that the proof is referencing; the tree-structured nature of geometric proof; and movement between two possible problem-solving strategies, forward and backward reasoning.

Traditional text-bound geometry problem-solving treats theorems as verbal and logical chains from premise to conclusion. Typically, the proof is non-graphic, involving a sequence of textual statements starting with premises and using theorems and applications of modus ponens on previous statements. Traditionally, graphic effects of the proof are "left up to the reader". This approach hides the fact that the goal of a geometry proof is to act on and transform a geometric figure. Geometric reasoning is often not a simple linear logical chain, but rather a bushy tree, including possible, and frequently not optimal logical paths. Students sometimes work from the goal backwards and at other times from the premises forward. These alternative paths are clearly articulated in Anderson's geometry tutor.

In the Smithtown Economics Tutor [Shute & Bonar, 1986] (Figure 19), students are provided with scientific inquiry tools that enable them to collect, organize, and understand data in the domain of economics. These data should allow the student to explicitly state such laws as those of supply and demand. The environment allows a student to set

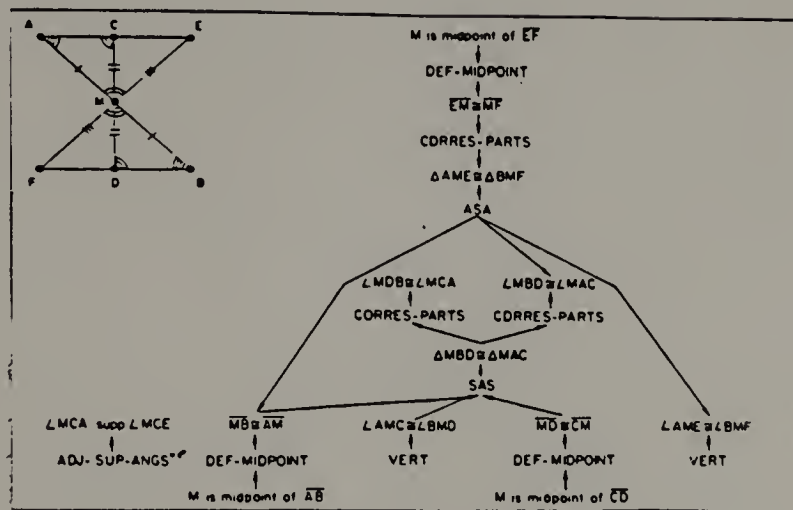
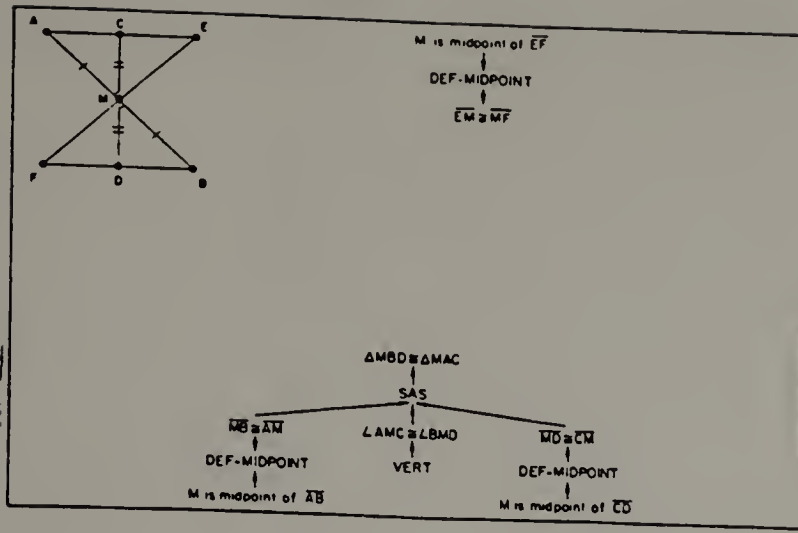


Figure 18 The Geometry Tutor [Anderson et al., 1985]

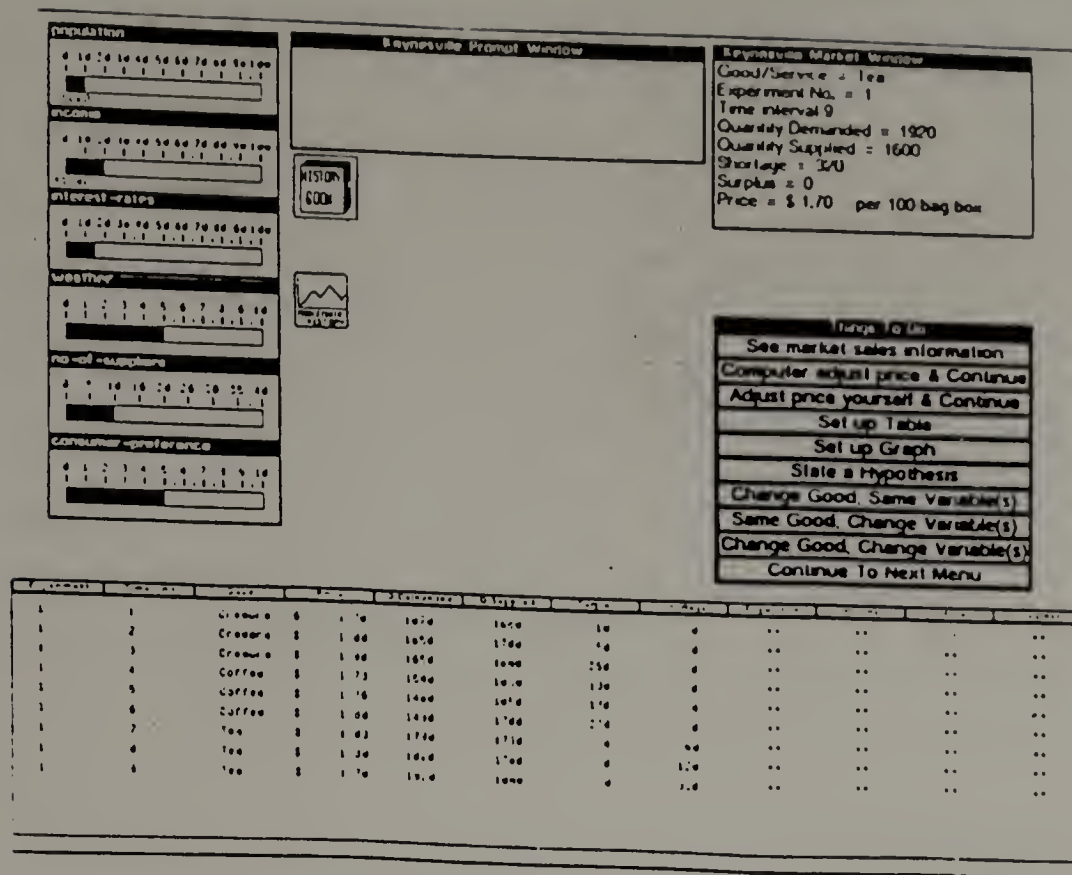


Figure 19 The Economics Tutor [Shute & Bonar, 1986]

values such as population and income, to choose goods and services, such as tea, coffee, or creamola, and to make predictions about changes in the entities such as prices, supply, and demand as a result of changes in such entities as the available quantity or number of outlets.

The coach “suggests” strategies that enable the student to be more effective in his/her exploration of the factors which influence economic theory. The coach evaluates the number and amount of variables modified and compares them with other variables. The student might have generalized a concept across goods by changing the commodity, e.g., move from coffee to tea, and keeping the same independent variables, e.g., population and number of outlets. The student explicitly expresses his/her hypothesis and then rejects or modifies it on the basis of additional data.

Several themes are apparent in the design of knowledge-based tutoring environments [Burton, 1987]. One is that, in most cases, development of the environment resulted from extensive research into the cognitive nature of the task (see, for example, Anderson [1981] for research leading up to the geometry tutor, and Woolf [1986] for research into the knowledge of a boiler operator). A second theme is that the environment was based on isolation of key “tools” required for attaining expertise in the domain. Thus, the economics tutor fosters experimentation and scientific inquiry and the geometry tutor,

through visualization, fosters both forward and backward reasoning in the development of geometry proofs. Thus, each environment would be a valuable aid to motivated learning, even without help from any on-line tutor. A third theme is fidelity to the world that is modeled [Hollan et al., 1984]. Fidelity is a measure of how closely the simulated environment matches the real world. High fidelity means the situation is almost indistinguishable from the actual environment.

2.5 Summary and Discussion

This chapter presented several knowledge-based tutors as examples of systems that addressed some educational issues presented in Chapter 1, including scientific illiteracy, limited student involvement in classes, inability of education to respond to rapid change, and the need for the acquisition and organization of complex data. Preliminary tests with knowledge-based tutors reveal that students find them effective and enjoyable. They make teaching more efficient, are able to relate to real-world problems, allow more relevance in the curriculum, and show some hope for improving science literacy. Clearly a lot remains to be done. These systems do not provide a panacea for education, even if they were easy to build and generally available, which they are not. Clearly, education has not been turned upside down. However, results are promising and when methodologies and tools such as described in Chapters 3, 4 and 5 become plentiful and inexpensive, evaluation of these systems will become possible to either support or refute the claims being made here. The distinction between knowledge-based tutors and computer-aided instructional systems was also described in terms of the information, control, and knowledge needed for each system. The flexibility of each system was compared.

The next few chapters describe epistemological issues and artificial intelligence (AI) principles underlying the building of such systems. These chapters might require more technical background; however, new terms are explained at length and references for both AI concepts and knowledge-based systems techniques are given at the end of the document. Case studies are included to illustrate AI principles in use.

CHAPTER 3

EPISTEMOLOGICAL ISSUES

3.1 Introduction

In order that computers be of real benefit to education, they should solve real problems, comment upon a student's ability to solve problems, and respond to student inquiries. This requires a large amount of knowledge which is difficult both to acquire and to encode in a machine. Issues of assessment, design, and implementation need to be addressed. This chapter describes how to represent the needed knowledge. It provides models of reasoning for using tutoring, domain, and cognitive knowledge along with examples of knowledge acquisition techniques for encoding concepts, procedures, and problem-solving heuristics in the subject of physics.

A structural description of a knowledge acquisition interviewing process is presented, based on the need to listen to how domain and teaching experts actually help students to understand difficult concepts. A framework is presented which constrains the acquisition processes and provides a structural design for acquiring the knowledge. This work contributes toward the ultimate goal of developing an authoring system for knowledge-based tutoring systems.

The next chapter shifts the focus to describe processes and stages involved in actually building the systems. Chapter 5 describes tools and methodologies that might be used for implementing the system. These three chapters are distinguished by whether the reader is interested in evaluating the knowledge needed (epistemological issues), has already accumulated the knowledge and needs an organizational plan (development is-

sues), or has begun to use tools and methodologies and is exploring alternative designs (implementation issues).

3.2 Knowledge Engineering

At the beginning and for some time into development of a knowledge-based tutoring system, knowledge engineering is the cornerstone of the process. Knowledge engineering involves identifying and encoding that knowledge used by the machine to reason the subject matter and to make inferences. It involves extracting, documenting, and analyzing knowledge as well as determining how that knowledge will be encoded in the system. This process includes several other activities, such as designing and analyzing code, and organizing and coordinating large numbers of people who need to transfer their knowledge in a consistent and organized manner. Obviously then the knowledge engineer, who directs this process, needs to have experience in working with people. He/she should be a good student as well as a talented teacher in order to first understand the domain and then transmit it. Such multifaceted people are difficult to find and hold onto in a long-term project. Thus, we propose that four specialists be enlisted to support the knowledge engineer and share the workload (see Chapter 4). The tasks of these four specialists are delineated in the next chapter. In addition, we suggest that the information to be encoded by these specialists be clearly identified and teased out to make the process of knowledge engineering more clearly defined. The remainder of this chapter offers such an identification and clarification of the requisite knowledge.

A framework for extracting knowledge is suggested in Section 3.3. Primitives are identified along with specific information that would then be tailored for inclusion in each tutor module. This chapter explicitly enumerates questions to ask and provides some clarification of the expected answers. The next two chapters clarify how to encode the resultant information. Each primitive identifies information for a particular module of the system. For example, the student model might contain data about how students learn

Domain Primitives

Information breakdown:	Generate map of the (sub)topics of the domain.
Information assumed known:	Identify topics that students are assumed to know.
Information type:	Fact, process, system, descriptive, or prescriptive; first principles or meta-knowledge.
Relationship between primitives:	Prerequisite, generalization, specialization, or analogy of each primitive.
Things to do:	Things students can manipulate in the environment (e.g., moving a fulcrum, redistributing mass).
Heuristics:	Rules of thumb about solving a problem in the domain (e.g., examine an extreme case; look at the simplest case, break problem into parts . . .).

Tutoring Primitives

Domain examples:	Easy ("start-up") examples, standard textbook ("reference") examples, counter-examples; strange, hard-to-grasp anomalies; general fill-in-the-blanks template-like examples.
students can ask:	Identify the qualitative (e.g., What is the direction of movement of the particle?) and quantitative (e.g., What is the mass of the object?) questions. Indicate the complexity of the question—number of variables, math level, and relevance to the domain (e.g., irrelevant).
Questions tutor can ask:	Same as "Questions students can ask" above, but for the tutoring system. Tutor responses (e.g., definitions, descriptions, hints, congratulations).
Instructional design:	Teach descriptive knowledge and explicit procedures for interpreting concepts. Teach self-diagnosis and self-correction. Teach how to interpret knowledge in a variety of special cases.

Figure 20 Knowledge Engineering Tasks, adapted from Rissland and Schultz [1987]

Cognitive Primitives

Underlying student view:	Indicate how the student might view the domain (e.g., as fragmentary or coherent, interpretable, etc.).
Problem-solving knowledge:	Indicate whether novices can process this knowledge on their own or do they typically ask for definitional knowledge.
Ideal behavior:	Indicate steps students should take for each task.
Knowledge state:	Describe various student states to make tutoring response decisions (e.g., confused, bored, knowledgeable).
Self-diagnosis:	Do students recognize their own cognitive difficulties in this domain or are they typically insecure or uncertain about how to apply knowledge?
Common errors and misconceptions:	Identify errors and ways to diagnose them. Identify misconceptions and ways to correct them.

Communication Primitives

Interactive graphics:	List system components that the student can manipulate, such as simulations or animations.
Screen elements	Describe icons, menus, windows available to student, mouse activations, tools available (e.g., clocks, calculators, help tools, dictionaries).
Input/output conventions	Describe methods used for gaining input from student and output from system

Figure 21 Knowledge Engineering Tasks, Part 2

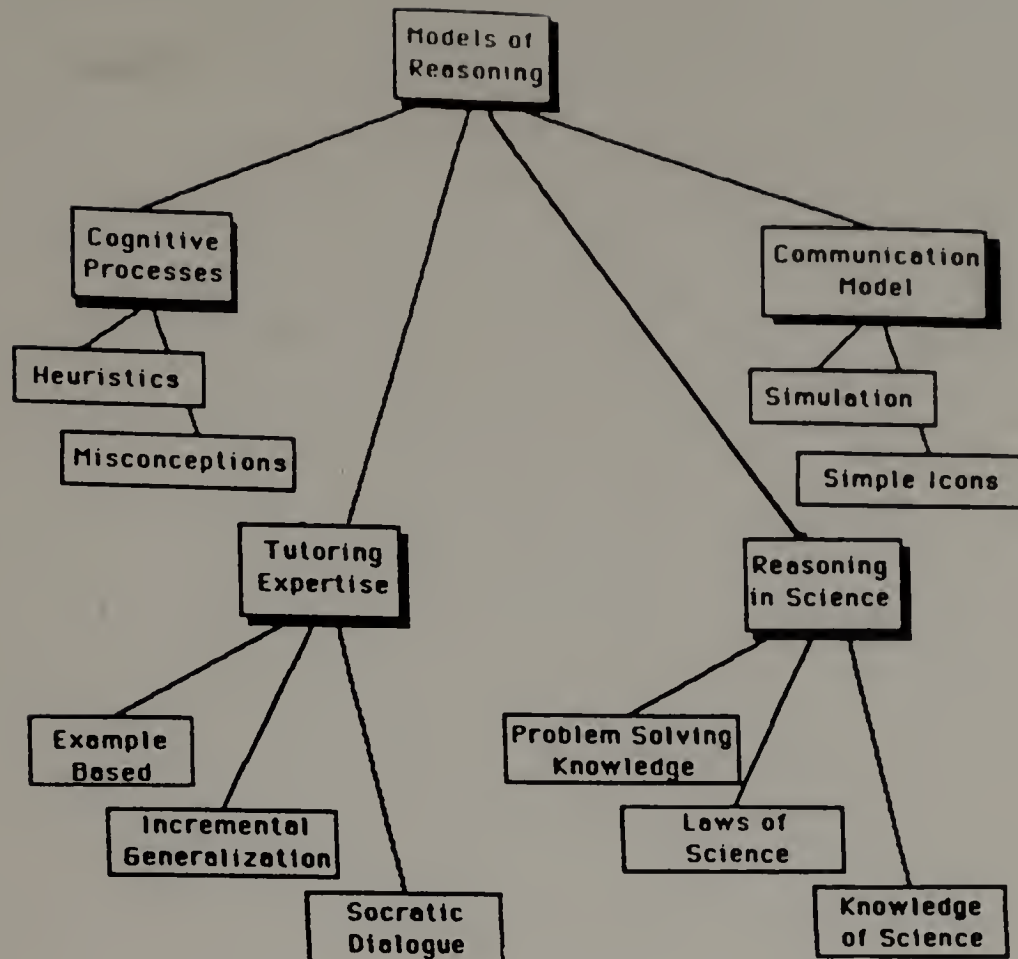


Figure 22 Models of Reasoning in an Intelligent Tutoring System (after Clancey[1987])

and differences between novice and expert problem solvers. Building the student model might require that a system make assumptions about the student, hold beliefs about his/her knowledge, and impart propaedeutic principles (the knowledge needed before learning about a domain, such as learning vector analysis in order to study mechanics) [Halff, 1988].

3.3 Tutoring as Qualitative Modeling

Clancey has suggested that intelligence is a process of modeling reasoning [Clancey, 1986]. This implies that building an intelligent tutor requires modeling reasoning about a domain, about learning, and about the act of tutoring. Figure 22 shows some of the requisite models. To build each model requires probing the relevant experts with specially

designed questions. In this section, we focus on how to elicit that knowledge for three of these models: cognitive processes, domain (in this case science), and tutoring knowledge. We describe techniques used to build the tutors described in Section 1.1.3 and indicate how successful we have or have not been working with experts to transfer their knowledge.

As an illustration of the kind of primitive knowledge we try to elicit from research literature and experts, we list some components of cognitive processes that we hope to acquire:

Student's Underlying Knowledge:

Is the domain seen by the student as fragmentary, coherent, or interpretable?

Is the student confused, bored, or knowledgeable?

Do novices ask for definitional information or can they process the task using their own knowledge?

Heuristics used by a Student:

Rules of thumb used to solve problems in the domain
(e.g., examine an extreme case; look at a simple case).

Ideal Behavior:

Steps taken by an expert to solve each problem for each task.

Self-diagnosis:

Does student recognize his/her cognitive difficulties?

Is student insecure or uncertain about how to apply knowledge?

Common Errors and Misconceptions:

Identify ways to correct misconceptions.

Identify errors and ways to diagnose them.

Identify misconceptions and ways to tease misconceptions apart.

For instance, a knowledge-based tutor ought to be able to infer goals and intentions of the student from observed behavior. For tutoring especially, it is important that an evolving "mental state" be made explicit during the course of the dialogue. Thus a representation of the student should contain not only explicit recorded knowledge, but also implicitly updated knowledge as the dialogue continues [Kass & Finin, 1987]. This kind of dynamic modeling might be less necessary in other intelligent computer systems, such as an expert system.

AI modeling methodology allows us to identify primitives for reasoning about tutoring and student knowledge (see also Chapter 5). Control and knowledge structures need to be made **explicit**, **reusable** and **generative** [Clancey, 1987]. They need to be general at the domain level; i.e., structures should be usable for several tutors and transferable to new domains. They need to be generative in that they can show generality at the case level; i.e., procedures can be used again and again to solve new problems in a single domain. The resulting mechanisms should also **promote experimentation** in that continued testing should allow the author to learn more about the applicability of an individual model and about how to fine-tune its reasoning mechanism.

The models we describe below fit into a framework which begins with primitive information extracted from experts. For each model, we identify questions whose answers fit specific areas of our system's knowledge or control mechanism. In this section we enumerate those questions and provide some clarification of the expected answers. In later chapters, we indicate how and where we stored the answers (see Chapter 5).

The description in this section is concerned with physics teachers and our attempt to encode their knowledge about teaching and learning physics. The operational knowledge of the field includes many principles, rules, and formulas used to analyze problems in physics. However, these principles do not completely define the problem-solving knowledge in the domain. Rather, new conceptual issues need to be addressed such as how to classify problems by their deep structure and which principles apply and when. Some experts, when questioned about how they solve statics problems for instance, regurgitated formulas. Yet, we need to uncover the rules behind the formulas. We need information about how to identify variables, how to choose equations, and how to reason about situations.

We suggest that modeling the problem-solving processes requires focusing on how and why a scientist asks questions, proposes experiments, or uses heuristics solving problems. Acquiring such knowledge is not easy. For example, in building the statics and thermodynamics tutors (Section 2.1.3) we worked with cognitive scientists, physicists, as-

tronomers, and potential users of the system, such as high-school and college teachers, for more than 18 months. We produced over 100 pages describing processes, screen designs (including help activities about physics), and cognitive studies (identifying educational goals, potential errors, and misconceptions) before any code was built [Rappleyea, 1987].

3.3.1 *Modeling Cognitive Processes*

We prefer that cognitive research results from previous studies be made available to us before we design intelligent tutors. These studies require years of effort from psychologists and domain experts. They provide information about how students learn in the domain, the differences between novice and expert problem-solvers, and key parameters of the student model (e.g., Anderson [1981] and Woolf et al. [1986]). For example, before the statics tutor (Section 2.1.3) was built, we perused more than a decade of research into physics misconceptions in mechanics (e.g., McDermott [1984] and Clement [1982]). This knowledge was reviewed and some of it included in the tutor described in this section.

Identifying Heuristics. Cognitive knowledge is not easily identified, nor quickly provided by experts. For instance, we asked physics experts to provide heuristics for solving physics problems. Heuristics are not the actual steps used for solving problems.¹ Rather, heuristics are the “rules of thumb” that any person, expert or novice, uses to approach a problem. In the simulation environment, heuristics include “first try a simpler version” and “relate to your everyday experience.” Such rules are not guaranteed to work; they are only intended to reduce the search for a correct solution. For a novice the search might include testing a vast number of equations. For an expert, heuristics and clustering of problem types reduces the number of viable equations.

¹For example, to solve a typical problem in statics, one identifies forces acting in the situation, represents these in vector diagrams, writes algebraic expressions based on these diagrams and the physics principles (e.g., vector sums of the forces and the torques are each zero), and then solves the equations for the unknown quantities.

To facilitate teaching these heuristics, the machine tutors allow the learner to:

1. *inspect* systems in action;
2. *manipulate* parameters, such as size of universe, density of excited states vs. those at ground state;
3. *answer qualitative questions* using
 - (a) *prediction*
 - (b) *comparison*
 - (c) *modification*;
4. *focus* on one topic at a time, figure out which parameters to change and what to look for;
5. *incrementally* change the system to compare cases that are similar in all but one respect.

The system encourages use of such heuristics by allowing students to manipulate elements of the simulation. In the thermodynamics simulation (Section 2.1.3), for instance, students can manipulate the size of the universe, the number of atoms at excited or grounded state, and the number of observation windows. In the statics tutor, they can manipulate elements such as the size of the boom, the mass of the weight, and the angle of the boom.

The tutor can lead the learner through tactical steps to experiment with these parameters or it can engage him/her in a kind of "Socratic" dialogue. A Socratic dialogue might allow a student to continue an errorful problem-solving approach, such as using overgeneralization, until he/she reveals the error by making obviously unreasonable conclusions. Regardless of whether the initiative rests with the tutor or the learner, such heuristic capabilities should be explicitly built into the tutoring system.

Student Misconceptions. Research into cognition, particularly into learning physical concepts, has yielded much information about misconceptions [McDermott, 1984; Clement, 1982, 1983]. "Misconceptions" is the technical term used to describe student conceptions rooted in students' intuitions and everyday experiences, but which are at variance with standard current understanding of science. Many such misconceptions are very persistent even in the face of very good and very focused teaching of the correct conceptions and are very common even in students who achieve well in standard science courses. Fortunately, these common and "deep" misconceptions are finite in number. Research has begun to identify ways to help students overcome them [Clement & Brau, 1984; Murray et al., in press]. One domain in which much work has been done is classical mechanics, of which statics is a part. We have used these research results in the design of our statics tutor.

When a student makes an error, this need not be due to a misconception; errors may be due to oversight or failure of memory, lack of necessary information, failure in the mechanics of equation-solving, uncertainty in how to apply "known" principles in the particular example, as well as to genuine misconceptions. An intelligent tutor should make suppositions as to the cause of an error, test these, and respond accordingly. Some examples from the statics or crane boom problem.

- A student may fail to include, in a listing or diagram of forces acting on the boom, the force exerted by the wall on the boom. This may be an indication of a common misconception, the belief that static, rigid objects such as walls cannot exert contact forces—they can only "be in the way" and prevent the other object from moving.
- A student may include a force at the point where the boom touches the wall, but draw the force vector *into* the wall rather than away from the wall. This may indicate a confusion between forces exerted *on* the body in question and forces exerted *by* it.

- A student may correctly include the force of gravity on the boom, but place the effective point where this force acts not at the center of mass but at the end of the boom. This may indicate that the student lacks knowledge about the effective point of action of gravity on an extended body, or that the perceptual salience of the end of the boom is strong enough to overcome what he/she has “learned” in a general way about gravitational forces.

3.3.2 *Modeling Tutoring Expertise*

A system's evaluation of student behavior, common errors, and plausible misconceptions precedes and informs its generation of appropriate responses—be that giving correct answers, elaborating on a student's answer, or providing new information. Rules that enable the system to mimic conventional classroom teaching strategies are not necessarily appropriate. For instance, classroom teachers often avoid talking about student misconceptions. However, misconceptions should be dealt with in knowledge-based tutors. Rich example-based simulations might engage misconceptions as active concepts to be expressed and acknowledged by the learner. These misconceptions can be discussed along with more formal physics concepts being taught.

One goal might be to encourage students to entertain as “felt conflicts” the disparity between their conceptualizations and the more formal science [Claxton, 1985]. Machine responses might be geared toward supporting students who are engaged in conflict resolution. In fact, rather than provide correct answers, such a tutor should increase the student's articulation, exploration, and expression of alternative conceptions, so that the disparity and overlap between the two concepts becomes clear enough for the student to resolve. Examples, questions, and consequences should support this process and show students how their conceptions are in conflict with the observable world in terms of descriptions, predictions, actions, and explanations.

Toward this end, we asked experts to provide teaching primitives that would facilitate satisfactory resolution of conflict. A teaching primitive is an action or presentation provided by a machine in response to a student. How and when to use each primitive is determined by mechanisms based on common teaching strategies (see Section 5.5). Such mechanisms might include an example generator [Woolf et al., 1989] or discourse network [Woolf & Murray, 1987]. The teaching primitives provided by our experts are repeated from Figure 20.

Questions:

Qualitative and quantitative questions asked of the student or the system.
 (e.g., What is the direction of movement of the particle?
 What is the mass of an object? Identify the variables, math level,
 relevance of each question.)

Examples:

Easy ("start-up") examples.
 Standard textbook ("reference") examples.
 Counter-examples, strange, hard-to-grasp anomalies.
 General fill-in-the-blanks template-like examples.

Presentations:

Provide explicit procedures for explaining topics.
 Support self-diagnosis and self-correction.
 Present descriptive knowledge.

How and when each primitive is invoked is described in Section 5.5. A few such techniques are outlined here.

If a student's error is suggestive of a deep, yet imprecisely identified misconception, a general approach is to teach by demonstrating the consequences of his/her misconception. For example, the system might simulate the results of missing forces on the crane boom. If the specific misconception is known and multiple evidence manifested, then the tutor might propose a new example. The generated response might be a simple anchor, or extreme example. If the error is considered a simple oversight, as in the case when a similar question was handled correctly in previous examples, then the machine might teach by guidance and perhaps provide a hint.

These general rules apply to interactive simulations for any topic. Specific rules for the statics tutor are outlined below [Woolf et al., to appear].

- The statics tutor demonstrates the consequences of incorrect action in several ways; it might ask him/her to consider the resulting equations (which would show a contradiction), or ask a qualitative question about the balance of force or torque vectors (which could also show a contradiction), or cause the physical system to move according to the forces indicated by the student.
- The tutor asks the student about analogous cases, simple cases in which the student's intuitions are valid, or extreme cases in which an unphysical outcome is clear on qualitative grounds. The tutor follows up by asking the student to describe similarities and differences between analogous examples.
- The tutor leads the student through a kind of mental check-list, e.g., by asking the student, for each force indicated, what body exerts that force, or by asking the student to list each of the bodies in contact with the boom.
- The tutor asks leading questions, or gives a more-or-less direct hints.
- The tutor informs or reminds the student of a relevant fact or principle and asks him/her how that fact applies in the present case.
- The tutor simply informs the student of the right answer or method and then proceeds.

3.3.3 *Modeling Reasoning in Science*

In this section we highlight the extent to which experts provided topics and domain knowledge consistent with building knowledge-based tutors. We asked experts to provide the following:

A map of the (sub)topics of the domain.

Concepts and topics assumed known by students.

Type of knowledge for each topic:

Facts, processes, system, descriptive, or prescriptive knowledge.

First principles or meta-knowledge.

Relationship between topics:

Prerequisite, generalization, specialization, or analogy knowledge.

For example, the topics shown in Figure 23 were used to build the knowledge base for the thermodynamics tutor. However, having elicited this knowledge from the expert, we recognized that we needed additional attributes about topics. Thus, each of the following attributes for each of the topics was also requested and represented in the arcs of the semantic network:

importance of the topic

complexity of the topic

prerequisite topics

supports provided for other topics

causes other topics

temporal relations to other topics

evidence for the existence of other topics

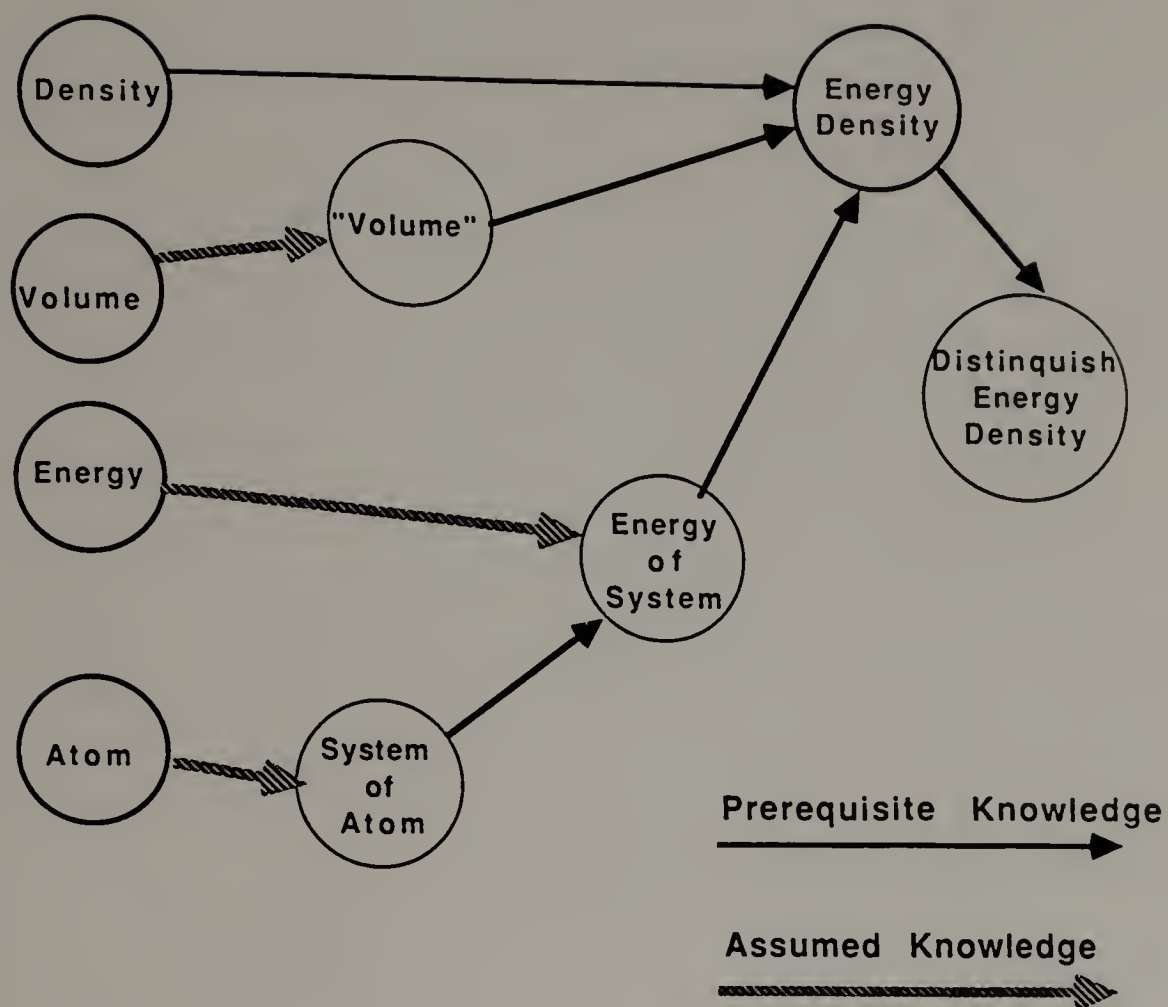


Figure 23 Topic Breakdown for Thermodynamics

Interactive Graphics:

Components a student can manipulate: e.g., dials, meters, icons.
Available simulations and animations.

Things a Student Can Do:

Modify objects such as a fulcrum, a valve, a mass.

Screen Elements:

Menus, windows available to student.

Mouse activation.

Available Tools; such as clocks, calculators, help tools, dictionaries.

Input/Output Conventions:

Methods by student to input response or action: menu, icon, text, speech.

Methods used by system to output response or action: menu, icon,
text, speech.

Figure 24 Communication Primitives

3.3.4 *Modeling Communication Knowledge*

A student's ability to interact with the system is determined in part by the communication facilities provided. The kind of issues to be resolved here are listed in Figure 24. In addition, issues about the inclusion of multi-media—video, sound, movies, and images should be discussed and choices made. Such issues and decisions are discussed in Section 5.6.

3.4 Indexing Information to be Taught

The previous section described epistemological issues to be resolved after a domain focus has been chosen. However, selecting the domain focus and indexing it presents an additional set of epistemological constraints. This section discusses some of those issues and suggests how to refine a choice once the basic domain has been selected.

The process of selecting a focus is fraught with hidden problems and considerations. At first, the primary problem appears to select a set of topics. However, very soon the problem becomes one of distilling the proposed set into a small enough subset to enable the large amount of knowledge to be encoded. Information to be taught can range from qualitative statements such as "The force exerted by gravity is inward," to more factual data such as " $F = -mg$." It can range from meta-knowledge about processes, such as "Use gravitational forces only when describing an earth-bound body," to synthetic knowledge about how to put information together such as "Add force lines after recording weight and distances of the structure." Section 5.2 provides a detailed discussion about how to represent this information, while this section describes several ways to refine and classify the proposed domain in order to 1) pare down expectations of what a machine can teach and 2) utilize tools and methodologies developed by other researchers in similar domains.

The knowledge to be taught might be organized by type and use. For instance, it might include:

- Facts: structural relations and recurrent process (e.g., geography facts, laws of weather);
- Procedures: how to operate a device (e.g., steam engine, recovery boiler tutor);
- Systems: structure of large systems (e.g., electronic circuit, economic system);
- Meta-knowledge: how to learn the domain knowledge (e.g., principles for solving algebra word problems);
- Diagnostic problem-solving knowledge: hypotheses to consider and data to clarify before making a diagnosis (e.g., which lab evidence to explore when diagnosing a patient's disease); and
- Formal reasoning: axioms, inference rules, and derivation methods (e.g. mathematics, logic, programming).

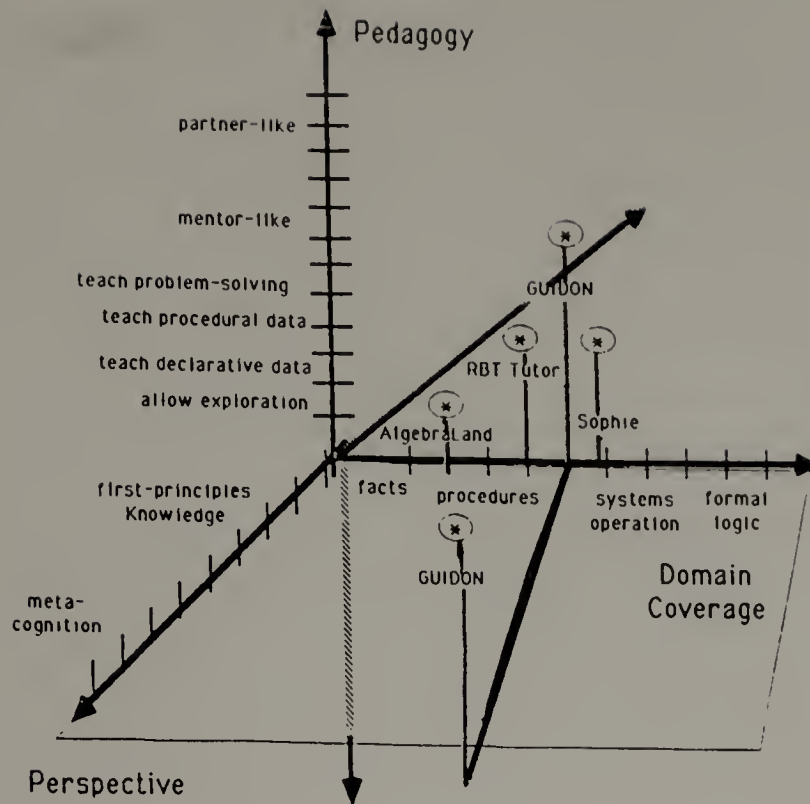


Figure 25 Dimensions of a Topic to be Tutored

Current tutoring systems cannot teach certain information and thus the newcomer to this field should avoid certain approaches. The fringe of what current systems can tutor is currently:

- Naive Science: commonly understood yet typically mistaken laws of science (e.g., elastics can be used for pulling, but not for pushing);
- Causal Modeling: effect of actions on objects (e.g., heat causes moist air to release moisture as rain);
- Envisionment: simulating an image or effect before it happens;
- Analogical problem-solving: intuitive understanding of analogous events.

Figure 25 shows three dimensions or indexes along which to consider the information to be tutored. The first index (Domain Coverage) describes the extension of the information and identifies whether the tutor will examine an entire system (ecological

or economic system), or an individual topic (force or acceleration). The second index (pedagogy) indicates the nature of assistance to be provided and can include providing declarative (descriptive and factual) data, procedural (prescriptive) data or complex problem-solving methods. Teaching declarative knowledge usually involves providing descriptions or classifications of objects, whereas procedural knowledge (prescriptive) usually includes providing rules or features to observe while problem solving in the domain. The third index (perspective) describes the level at which the domain information will be communicated. This varies from discussing primitive first principles knowledge, or the underlying rules and laws responsible for the domain, to meta-cognitive knowledge, or descriptions of how to organize and learn material in the domain. It is instructive to place the proposed domain along the dimensions of this figure to both identify its indices and to relate the proposed system to existing tutors built to teach comparable information.

Example topics which can fit along the indices include:

Individual Topics: (e.g., force, acceleration, emergency shutdown procedure

- **Declarative first principles knowledge**, e.g., gravity exerts a force towards the center of the earth.
- **Declarative meta-knowledge**, e.g., calculate acceleration as a function of velocity.
- **Procedural first principles knowledge**, e.g., begin an emergency shutdown procedure by first draining the super heaters.
- **Procedural meta-knowledge**, e.g., if you need to calculate force on a body, first draw the vector diagram and then write the equations.

Entire System: (e.g., electronic, economic, or boiler systems)

- **Declarative first principles knowledge**, e.g., a break in an electrical circuit interrupts the flow in the circuit. Parallel electronic circuits exhibit characteristics similar to those in simple circuits.
- **Declarative meta-knowledge**, e.g., parallel electronic circuits can be viewed as circuits with separate energy sources.
- **Procedural first principles knowledge**, e.g., the law of supply and demand emerges from variable and competitive pricing of commodities.
- **Procedural meta-knowledge**, e.g., to observe the interaction of steam pressure and steam flow in a boiler, plot the variation of each parameter against time and against each other.

Additional Knowledge to be Represented. In addition to the above information, an intelligent tutor often represents other information not included in the domain. Thus, a tutor might need to know about mathematics in order to solve certain problems. For instance, to perform medical diagnoses the expert system MYCIN relies on principles of integral and differential calculus. This extra information serves as foundational and primitive information separate from the specific problem being solved [Wolgram et al., 1987].

Domain-independent knowledge needed to teach physics problem-solving provides some difficulty for a designer, because though such additional information must be included, it is difficult to know how much or how little the student can be presumed to know. Thus, physics might be taught assuming or not assuming prior knowledge of calculus. Certainly, the ability to manipulate algebraic expressions might be assumed to be known by a student, yet this might also, during the tutoring session, be shown to be false.

3.5 Summary

This chapter began with a practical assessment of epistemological issues to be resolved in building a knowledge-based tutor. It suggested ways to question experts in order to elicit such knowledge from them and focused on the nature of the knowledge to be identified, specifications to consider, and methods to use for acquiring knowledge. The knowledge includes how a tutor comprehends a student's solution of problems, tutoring strategies, and how a knowledge engineer builds domain, tutoring, and student models.

CHAPTER 4

DEVELOPMENT ISSUES

4.1 Introduction

This chapter provides a second practical view of how to implement a knowledge-based tutor. Whereas Chapter 3 provided a general view of the knowledge needed, this chapter describes the implementation process from an organizational perspective delineating the people, places, and tasks needed to encode knowledge. Goals, priorities, stages, and time commitments are defined with special attention paid to the work of the knowledge engineer, whose effort, as shown earlier, precedes and in some cases defines the work of other specialists.

A framework is proposed for uncovering such knowledge from experts. This framework assumes collaborative efforts among several specialists, including computer scientists, domain experts, cognitive scientists, and educators. The need for several people to work on intelligent systems has been well documented [Bobrow et al., 1986; Mittal & Dym, 1985]. Multiple experts are needed because much of the knowledge is subjective, unorganized, and misunderstood. No one person, whether computer scientist or teaching specialist, can supply the wide variety of knowledge required. In building the framework, the strengths, weaknesses, and communication style of each expert should be considered. (The specific responsibilities of each are described in Section 4.4.) However, since specialists often view the world differently, computer scientists should be trained in educational theory and practice, and educators trained in computer science and Artificial Intelligence, etc.

This document is directed at training specialists in the methodologies of computer science and AI.

Anecdotal evidence is presented here about collaborations toward building such systems. We have been involved in four ongoing projects: (1) Exploring System Earth project [Duckworth, 1987]; (2) a tutor based on the Silent Way [Cunningham et al., 1986]; (3) various projects pursued by the Knowledge Communication System Group at the University of Massachusetts; and (4) the Recovery Boiler Tutor [Woolf et al., 1986]. In each case, educators, computer scientists, and domain experts collaborated over not less than 18 months to design and build intelligent tutors. Anecdotal reports from these projects suggests that formal organizing mechanisms are helpful for learning to avoid obstacles and stumbling blocks.

Further, given the complex and heterogeneous nature of the knowledge to be encoded, tools that transfer teaching, learning, and domain knowledge to a system should be very helpful. Currently, few such tools exist, but where they do, they ought to be considered for inclusion during the building process. Such tools are described later in Chapter 5.

4.2 Organizing the Development Process

Building an intelligent tutor requires six stages as outlined in Figure 26 and elaborated in Figure 27. These stages are similar to those required for development of any expert system [Wolfgram et al., 1987], yet they show additional remodeling and redefinition steps beyond those needed for a traditional software engineering project. Whereas the latter typically has clearly defined program specifications, an AI system is often defined in terms of “reasonable” and “intelligent” behavior. That is, a knowledge-based tutor

FEASIBILITY STAGES:

1: Problem Identification and Definition

FORMULATION STAGES:

2: Paper Design and Prototype Development

REFORMULATION STAGES:

3: Implementation

4: Test, Evaluate, and Revise Cycles

MAINTENANCE STAGES:

5: Integration

6: Evaluation and Maintenance

Figure 26 Stages in Building a Knowledge-Based Tutor

might be specified to generate sensitive and custom-tailored responses. Such behavioral specifications are not easily achieved. Several cycles of redesign and revision are clearly needed to produce such results; thus the life-cycle of a knowledge-based tutoring project includes several cycles of evaluation and remodeling.

The building process also requires a lengthy preparation stage before construction can actually begin. The initial stage is jointly directed by the knowledge and domain and tutoring experts. Its aim is to synthesize and prepare information for inclusion into the system. Middle stages are characterized by the actions of teachers and programmers who help design and reconfigure a prototype system before molding it into an integrated system. The final stages require collaboration of evaluators, teachers and programmers who redefine and build the system readying it for inclusion in an educational community. Also, students should be involved early on and periodically invited to use, evaluate, and contribute to the system.

Leadership and Management Issues. In addition to organizing development of the system, several issues related to managing this process should be addressed at the outset. These issues include, but are not limited to:

FEASIBILITY STAGES

Stage 1: Problem Identification and Definition

Principal participants: knowledge engineer, teaching and domain expert

Key Tasks: outline teaching goals and key topics, identify questions student can ask of system

Duration: 4-6 months, overlaps with Stage 2

FORMULATION STAGES

Stage 2: Paper Design and Prototype Development

Principal Participants: knowledge engineer, programmer, teachers and students

Key Tasks: design prototype knowledge base, inference engine, teaching strategies

Duration: 2-4 months, overlaps with Stages 1 and 3

REFORMULATION STAGES

Stage 3: Implementation

Principal Participants: programmer, teaching expert

Key Tasks: Enlarge and refine knowledge base, elaborate tutoring questions and responses

Duration: 6 months, overlaps with Stages 2 & 4

Stage 4: Test, Evaluate, and Revise Cycles

Principal Participants: domain and teaching experts

Key Tasks: tape record students using system, evaluate teacher's use of system, integrate new results into system's design

Duration: 4 months, overlaps with Stage 3

MAINTENANCE STAGES

Stage 5: Integration

Principal Participants: teachers and programmers

Key Tasks: redesign curriculum and teaching activities to include in system further results of Stage 4

Duration: 6 months, overlaps with Stage 6

Stage 6: Evaluation and Maintenance

Principal Participants: teachers and students, programmers

Key Tasks: Implement additional versions based on formal evaluation of system

Duration: ongoing

Figure 27 Detailed Stages in Building a Knowledge-Based Tutor

- **Curricular Focus** should initially be on several small topics, processes, or laws to tutor. The system should be modular so that new topics can be added during the construction stage (see Section 3.4).
- **A large number of potential users** should exist, i.e., an ongoing need should exist for training in the chosen area. Ideally, the system should be useful for both initial training to explain and teach new information, and for ongoing training and retraining. In addition, there should be a real need for the training, possibly because of a large turnover in students, a requirement for repeated on-going training, or a failure of existing training.
- **The effort required** should be realistically assessed. A great deal of time, money, and effort will be required to assemble experts, perform the knowledge acquisition tasks, and build and ultimately test the system. The duration of project effort should be realistically measured in person-years, not in person-months [Bobrow et al., 1986].
- **Hardware and software cost estimations** should be made. Cost, availability of equipment, and local programming expertise often determine the particular mix of hardware and software chosen for the job (see Chapter 6). Acquiring these resources might require several months lead time. For these reasons, the selection process should be started early during the feasibility or formulation stages of the project (see Figure 27).
- **A team of experts** should be assembled. At least four experts should be available to the project: a computer scientist, a domain expert, a teaching expert, and a cognitive expert (see Section 4.4 below). Long-distance experts, linked by phone, network mail, and jet plane, are acceptable. Each of these people should be invited to participate and be evaluated as to ability, availability, and commitment.
- **Resources** should be committed. A knowledge-based tutor requires the serious commitment of management or institutional leaders. Appropriate time and re-

sources must be allocated and reallocated during the life of the project. For many companies or universities, building a knowledge-based tutor is a separate research and development project. As such, it must be properly approved and integrated into the direction and focus of the organization.

- **The resulting product** will need to be integrated into existing training or teaching practices. Thus separate workshops might be required to facilitate the teachers, administrators, and students in using the new product. This integration period might begin before the final product is complete, using only a prototype system.

4.2.1 Knowledge Engineering

At the beginning, and for some time into development of the tutor, the knowledge engineer is the key member of the development team and must perform the lion's share of the work. For that reason, the role of the knowledge engineer should be discussed separately.

Knowledge engineers need to be multi-talented individuals— "Jacks/Jills of all trades." They are responsible for eliciting the knowledge used by experts and for identifying the sources of knowledge primitives, or actions to be performed by the tutor. They work with people, taped protocols, questionnaires, and reference literature. They perform extraction, documentation, and analysis of information and determine how it will be encoded in the system. This process of defining and acquiring knowledge is called "knowledge engineering."

Knowledge engineers should be capable workers in several disciplines. Since they often implement code, they should be *designers* and *analysts* trained in computer science and programming. They should also be good *organizers* and *coordinators* since their job is to co-ordinate many people and to transfer their knowledge in a consistent and organized manner. They should be good *students* as well as talented *teachers* in order to first understand the domain and then transmit it. Finally, it is helpful if they have

experience or training as *psychologists* so that they can manage the large number of experts. Obviously, such multi-talented people are hard to find. Therefore, we propose the use of a team of four people who support the knowledge engineer and help perform all these tasks (see Section 4.4 below).

After a team has been found, knowledge engineering begins. This phase is very time consuming. For example, in building the statics and thermodynamics tutors (Section 2.1.3) we worked with cognitive scientists, physicists, astronomers, and potential users of the system, such as high-school and college teachers and students, for more than 18 months. The group produced over 100 pages of rules, processes, screen designs (including help activities about physics), and cognitive analysis (identifying educational goals, potential errors, and misconceptions) before any code was written.

The knowledge engineer should document the reasoning processes included in the domain and teaching modules. She/he should keep a record of how key concepts, sub-problems, examples, and questions are mapped out. Such data are then made consistent for inclusion in the chosen knowledge representation. The knowledge specification document becomes a reference for researchers as well as a guide for prospective users and workers.

During the organization and compilation of data, several problems might arise. For example, domain specialists frequently resist attempts by computer scientists to clarify algorithms and rules that they use. This is because they have "compiled" their knowledge and it has become virtually unconscious. Often the specialist becomes overwhelmed by the difficulty of teasing apart his/her own knowledge; typically a specialist has difficulty analyzing his/her own knowledge when the complexity of that information becomes computationally dense. At such a time, several solutions are available; for example, the domain specialist might read literature based on a different approach to the problem. Researchers from one discipline might investigate another view of the problem; thus, a physicist might investigate how psychologists study physics learning or a cognitive scientist might be shown research into the structure of the domain from a physicist's viewpoint.

Such collaboration helps correct the academic “myopia” of some experts. Working alone, for example, one expert might not be able to crystallize the utility of a contribution from another domain because he/she is unaware of the efforts made in that other field. Thus a computer scientist working alone might not be aware of complementary research performed by cognitive scientists to understand how to teach programming concepts. Information about the categorization of topics or the availability of a wealth of tutoring strategies in each domain is vital to making reasoned judgements about the design of a tutoring system.

4.3 Stages of Development

This section describes the stages required to design and build a knowledge-based tutor. The process was outlined in Figure 27. That figure will be briefly described here and pointers given to other sections where a more complete treatment of the requisite activities can be found. These activities and processes are further elaborated in Figures 28 and 29.

4.3.1 *Stage 1: Problem Identification and Definition*

The first stage is assessment of the scope of the project and identification of the primitives involved in representing that knowledge. This task requires a domain expert, or expert in the field who can participate fully in the project for about a year. At this time, the focus is on a topic or set of topics to be taught by the tutor (see Section 3.4).

Building a Storyboard. After the topics and general tutoring strategies have been identified, the domain expert, working with the knowledge engineer, should produce a 10-ply storyboard [Rissland & Schultz, 1987]. (Ten-ply is defined as 10 interactions between user and machine, or one response from the system, one from the user, etc., until 10 exchanges have occurred. Ten is chosen so as to force the expert to explore an extended

range of possible machine/user interactions.) This process allows the experts to judge the dimensions of the dialogue as well as the need for flexibility and responsiveness of the system. Eleven sheets in all (one for what the system looks like initially and ten for user/system exchanges) should be produced. Example storyboards and design documents are provided in Appendix B.

A simulation or animated drawing will add power to the system. The student should be able to change several components of the figure. Such simulations and animations should be described in the storyboard along with all available menus, windows, buttons, and icons be illustrated. Options and parameters that are available to the student should also be outlined at this time (see Section 5.6 and the examples storyboards in Appendix B).

4.3.2 Stage 2: Paper Design and Prototype Development

Once the knowledge primitives and storyboards have been resolved, prototype development can begin. This stage includes building a knowledge base, tutoring manager, primitive student model, and simple simulation. Knowledge-based programming is distinguished from a more traditional "straightline" program in that knowledge is represented and an inference engine¹ is built to pass through that knowledge (see Chapter 5). The specific path taken by the system through the knowledge is not predefined; rather the control structure, often represented as "if/then" rules, determines how and in what order the system will use the encoded knowledge. Knowledge representation is the first step in this prototype development stage. Knowledge can be represented as semantic networks, frames, scripts, if-then rules, and other available representation (see Section 5.2).

¹An inference engine is a function which searches through a knowledge base and makes inferences about that knowledge. An inference engine can come to believe new facts on the basis of other information. It can draw inferences.

Since knowledge-based programming often involves making machines do things that they have done before (such as understanding written documents or coordinating movement of cooperating robots), AI programmers are not always certain how to achieve these results. In fact, many knowledge-based programs are experiments in design that evolve only as the programmers' ideas evolve.

In a completed AI system, a control structure, typically in the form of antecedents and consequents, might trigger and branch in ways unanticipated by the programmer. This is not true for traditional software projects in which the goal, specifications, and machine decisions are clearly indicated before coding begins. Rewriting code in traditional software engineering is often done while removing bugs or making the system perform more exactly according to specifications. Rewriting code in knowledge-based programming often reflects an attempt to refine system behavior which was not explicitly stated or to specialize knowledge so that the system can make more highly reasoned decisions. To do this, AI programming requires new tools, languages, and programming skills (see Chapter 6). Thus the term knowledge-based programming refers more to an approach to code production, a methodology for encoding knowledge and control, and a set of tools to expedite the process. It does not refer to a particular language or subject matter.

4.3.3 Stage 3: Implementation

Once a prototype system is built to demonstrate the reasoning power of the proposed tutor, the system's knowledge and control structures should be firmed up and made more robust. During this stage, the prototype is made suitable for use by students and the experts who focus on elaborating and completing each system component. Detailed knowledge about topics to be taught, the cognitive model, tutoring actions, and screen design are implemented; the prototype can be discarded as "throw-away code." The most successful features of the prototype will be recoded in the final system. Additional knowledge at this time may indicate a need to change data structures and control strategies.

Additional data gathering is needed to dramatically expand the knowledge base and complete the user interfaces. A special knowledge acquisition interface might be built to allow teachers and other workers to contribute to and modify information in the knowledge base. A knowledge-acquisition interface is a computer screen that requests components of the knowledge base and inference rules, but does not require the author to work in a low-level programming language.

Domain Model. At this time, additional knowledge of topics, concepts, processes, and rules of inference is defined and entered into the knowledge base. There may be hundreds of these concepts or topics (see Chapter 2).

Cognitive Model. At this time, a refined model of the student's knowledge in the domain is encoded in the system. This model represents whether the student sees the domain as coherent or fragmentary and whether the knowledge is interpretable or definitional (see Section 3.4.1). The knowledge base is expanded to include more errors and misconceptions and diagnostic procedures that might enable the system to both recognize those errors and clear up misconceptions. The parameters of the cognitive model are elaborated and finalized.

Communications Interface. The interface must be engineered and made suitable for use by a variety of students. A system should not fail when a student produces unexpected input. Rather, it should gracefully acknowledge the input and recover. Critics should be asked to offer design suggestions for menus, buttons, and graphic elements (see Section 5.6).

Teaching Strategies. At this time all tutoring actions and rules should be refined, teased apart, and elaborated. The questions, answers, analogies, and examples should have been tested by students and refined during earlier stages (see Section 5.5).

4.3.4 *Stage 4: Test, Evaluate, and Revise Cycles*

At this stage the knowledge-based system is ready to be tested by students. The system should be evaluated and modifications made based on these studies. This stage is time-consuming and may result in development time equivalent to the first three stages. Based on results found here, workers may completely redefine and rebuild portions of every module.

Ideally, the system should be subjected to extensive testing by both students and expert teachers. One way to do this is to give problems to both the system and a teacher working with groups of students. Results of the two groups should be compared. The responses provided by the knowledge-based tutor should largely agree with that given by the expert.

4.3.5 *Stage 5: Integration*

An intelligent tutoring system should be integrated into the training or schooling site. This might include changing classroom procedures, curriculum, and protocol for all members of the educational community. Potential resistance to the system might need to be handled. Some of these issues, including evolutionary change in the classroom that might occur, are described in Section 7.2.

4.3.6 *Stage 6: Evaluation and Maintenance*

The environment in which the system is placed is dynamic: curriculum changes, teachers and students change. To remain useful and usable in a dynamic setting, the system must be flexible. Thus the system should be updated when new concepts, curriculum items, and related issues arise. Evaluation and maintenance contribute to this process.

FEASIBILITY STAGES

Establish a knowledge engineering capability (see Section 4.2.0) (Time Required: 4-6 months, dependent upon availability and skill of knowledge engineer and experts).

A. Identify team members to conduct research: computer scientist/knowledge engineer, teachers/trainers, domain expert, and cognitive scientist. Anticipate the need for replacements and dynamic regroupings: find a solid core of principles and visiting participants (see Section 4.3).

B. Define the problems, topics, knowledge, and dimensions of the knowledge to be taught (see Section 3.5). After the knowledge has been defined, *reduce* the scope of it by the approach below:

1. Define the subproblems.
2. Define a prototype.
3. Estimate the complexity/feasibility of delivering such a prototype.
4. *Stop* if the feasibility of a prototype is only *slightly* out of reach or *within* reach.
5. Divide the subproblem into smaller problems and continue with 2 above.

C. Describe a communications environment/interface. Identify the available hardware and software systems (Chapters 3 and 5). Indicate options available to the student and responses/monitoring provided by the system.

D. Define constraints. Consider hardware/software constraints, development issues, staffing, management, and technological considerations.

FORMULATION STAGES

Identify research and development resources and build prototype (see Chapters 3 and 6). (Time Required: approximately 3-6 months, carried on in parallel with Feasibility Stages).

A. Pedagogical and Cognitive Research. Acquire results of earlier pedagogical and cognitive research into the domain. Explore the availability of experts who might participate, on-going studies which might be adopted, and on-line knowledge bases. Avoid using text books—they typically describe a static and deterministic way to communicate information. Determine concepts, rules, and heuristics to be taught (see Section 3.5). Identify teaching strategies (see Section 3.4.2).

B. Support Experts: Plan regular and structured interviews with domain experts. Establish interviewing procedures and ways to reward the whole team. Document this process, with notes and written summaries, for use during the next development project.

C. Leadership and Management. Determine who is in charge of what; who reports to whom. Identify individuals responsible for knowledge acquisition, implementation, evaluation, redesign. Assign task leaders and communicate expected results.

D. Build a prototype.

Figure 28 Tasks to be Accomplished

REFORMULATION STAGES

Reimplement and refine prototype using enhanced hardware and software tools (see Chapter 6). (Time Required: Approximately 6-12 months, depending on availability of equipment; carried on with Formulation Stages.)

A. Reevaluate software/hardware issues (see Chapter 6). Consider a second implementation. Again avoid sequential languages (e.g., COBOL and BASIC), standard software design methodologies, and restrictive or strict type definitions (e.g., C, and FORTRAN). Consider selection issues: portability, existing community, degree of support, ease of learning, size of language. Identify machine issues: keyboard, mice, displays, windows, editors, bundles, knowledge engineering shells.

B. Evaluate hardware issues: single user operators, manipulation of objects, optimization for function calling. Consider dedicated machines: companies, price-performance statistics. Consider conventional machines: processors and microprocessors. Consider specialized processors: parallel processors.

C. Explore and implement Artificial Intelligence techniques including, e.g., rules, frames, forward and backward chaining, hierarchical frame structures, procedural attachments, multi-legend and blackboard architectures (see Chapters 5 and 6). Work with knowledge representation scheme.

D. Demonstrate new proof of concept system and accept feedback.

E. Continue to integrate information from multiple-experts-based proof of concept system. Increase production of rules, screen designs, and information produced by team members, teachers/trainers, cognitive scientists and domain experts.

F. Stabilize the iteration process. Release a version of the system for testing with students/trainees.

MAINTENANCE STAGES

Test and evaluate the system. (Time Required: allow a minimum of 9 months.)

A. Integrate the system into classroom, training site, or community. Use test results in the next design, implementation and release of the system.

B. Continue interviews with teachers/trainers and domain experts to clarify impact of system. Encode additional cases, concepts, rules, and heuristics as needed. Continue refinement of the learning model and elaborate encoded teaching strategies.

Figure 29 Tasks to be Accomplished, Continued

4.4 Gathering a Team

An important part of building a tutoring system is developing a "community memory" in which multiple experts contribute their knowledge of teaching in the domain. Building such a community memory requires realization of the fact that knowledge is often distributed, incomplete, and acquired incrementally [Bobrow et al., 1986]. Thus, part of the administrative problem is to generate a team of persons dedicated to completing the project.

This is especially true in tutoring systems where the domain expert, cognitive scientist, and teaching expert are typically not the same person. Experience with commercially successful expert systems, such as R1 [McDermott, 1982] and the Dipmeter Advisor [Smith, 1984], suggests that using knowledge from only a single expert can result in systems that are foreign to other users or ones that contain conceptual holes. In the case of the Dipmeter Advisor, the first expert solved problems in an uncommon way, creating blind spots in the knowledge base [Bobrow, 1986].

In order to develop a community memory for tutoring systems, a framework should first be created for recording teaching experience and domain knowledge. The discussion in this section elaborates the process of building such a framework and provides general criteria for developing tools to use in the framework. Complex and diverse tools are needed. For example, expert system shells might provide a framework for building expert systems, since they store concepts and rules for making inferences about those concepts [Anderson, 1985; Streibel et al., 1987]. However, such tools are often based on production rules and are limited in their ability to represent the history and dependency of the tutoring interaction [Woolf & Murray, 1987]. They are limited in many ways; they do not adequately represent multiple antecedents or consequences for an action; they fail to describe a logical stream of prior activities for a given state; they are not able to represent complex tutoring and misconception knowledge, such as reasoning about teaching strategies, selecting paths through domain concepts, and validating and remediating misconceptions. Ideal tools to compile expert knowledge for building knowledge-based tutors have not yet been developed. This section looks at the responsibilities and contributions of each expert, in order to learn how to define and develop such tools.

4.4.1 *Environmental Input*

The computer scientist builds the envelope within which a student interacts with the system (see Section 2.4). This we call the "environment." Here tools and operations specific to solving problems or performing activities in the domain are encoded. Often most of the tutor's memory is used for the environment [Bobrow et al., 1986]. Obviously, the contributions of the teaching, cognitive, and domain experts must interact with the work of the computer scientist. For example, if the system asks a student to record the incident and resulting angles for light rays in an optics experiment, one would assume that the environment supplies appropriate tools for setting up an optics experiment and measuring angles.

Existing environments suggest many desiderata for development of effective interfaces. Several are listed below:

1. The environment should be intuitive, obvious, and enjoyable. The student's energy should be spent learning the material, not learning how to use the environment [Cunningham, 1986]. For example, the visual activities of the second language tutor (see Figure 2.7) mimic ways the human teacher uses gestures, mime, facial expressions, hand signals, and rods to indicate errors, express feelings, or convey meaning. Each icon is designed to be clear and unambiguous in order to make use of the student's intelligence, experience, and resourcefulness.

2. The environment should record not only what the student actually did, but what the student intended to do (the goal), might have forgotten to do, and was unable to do (Burton, 1988). The environment should provide a wide bandwidth within which multiple student activities can be analyzed. For example, the Pascal tutor developed by Johnson and Soloway [1984] analyzed an entire student program and diagnosed possible student misconceptions before it offered advice. This ability is contrasted with tutors that can only record and analyze individual keystrokes.

3. The environment should be grounded in teaching and cognitive knowledge about how experts perform tasks in the domain. For example, Anderson [1981] conducted extensive research with geometry students before developing his geometry tutor's interface, and Woolf et al. [1986] incorporated knowledge from experts with more than 30 years of experience working with boiler operations before building the RBT environment.

4. The environment should isolate key "tools" for attaining expertise in the domain. For example, the economics tutor [Schute & Bonar, 1986] (see Section 2.4) monitored the student's ability to set parameters such as supply, demand, price, and distribution centers in order to observe the effect of economic principles. The RBT tutor provided graphs (trends) of process parameters over time (Figure 6) and abstract meters (left side of screens in Section 2.1.1) to facilitate an operator's ability to reason about complex processes and to allow him/her to make inferences about the effects of actions taken.

5. The environment must approximate physical fidelity² to the world that is modeled [Hollan et al., 1984]. The RBT tutor presented a mathematically exact duplicate of the industrial process. It modeled and updated over 100 parameters every two seconds. Visual components of the industrial process, such as an alarm board, control panel, dials, and reports were replicated from the actual control room. In the physics tutors (Section 2.1.3), the student can test activities such as random collision in the physical world.

6. An environment should be *responsive*, *permissive*, and *consistent* [Apple, 1985]; it should apply skills that people already have, such as moving a cursor, rather than requiring people to learn new commands. *Responsive* means that the student's actions have direct results; the student should not need to perform a lengthy set of actions in a rigid and specified order before achieving a goal. *Permissive* means that the student should be allowed to do anything reasonable and that there should be multiple ways to achieve actions. *Consistent* means that moving from one application to another, e.g., from editing text to developing graphics, should not require learning a different interface. All tools should be based on similar interface actions, such as pull down menus and single- or double-mouse clicks.

²Fidelity is a measure of how closely the simulated environment matches the real world. This measurement can be in terms of output of mathematical model or visual representation. High fidelity means that the system is almost indistinguishable from the real world.

No one environment is appropriate for every domain: each domain must be analyzed to determine how experts function in that domain, how novices might behave differently (see for example Larkin et al. [1980] and Chase and Simon [1973]), and what tools might help novices to attain expert behavior. Not all systems are aimed at novices and thus each system should be designed for and tested with its particular audience.

4.4.2 *Teaching Input*

Acquiring sufficient teaching expertise to build a tutoring system is a long-term process. In many domains, cognitive research has just begun which will explain how people teach and learn. Refinement of teaching knowledge in such machines has been made possible only recently. Decision logic and rules to direct a tutor's intervention are just now being represented and must be cautiously tested and modified. For example, the framework in Figure 30 was developed for managing discourse in an intelligent tutor [Woolf & Murray, 1987]. It dynamically reasons about which discourse choice to make and provides custom-tailored feedback to a student in the form of examples, analogies, and simulations. This framework (described in Section 5.2.1) is currently being refined to improve a physics tutor's ability to respond to idiosyncratic student behavior. The structure is designed to be rebuilt. A graphics editor can be used to modify response decisions. Appropriate machine responses can be assessed and, through the editor, continuously improved.

No single teaching strategy is appropriate for every domain; for example Anderson et al. [1984, 1985] built geometry and Lisp tutors that responded immediately to a student's incorrect student answer, be it a small misplaced comma or an incorrectly spelled command. These authors argued that immediate computer feedback was needed because erroneous solution paths in geometry and Lisp might be so ambiguous and confusing that the student would not recognize delayed notification of an error. Thus the tutor intervened frequently to avoid fruitless student effort.

However, the industrial and language tutors described earlier in Sections 2.1.1 and 2.1.2 were passive, not intrusive advisors. Their strategy was designed to subordinate teaching to learning [Gattegno, 1970] and to allow the student to experiment while developing hypotheses about the domain. These two tutors encouraged students to develop

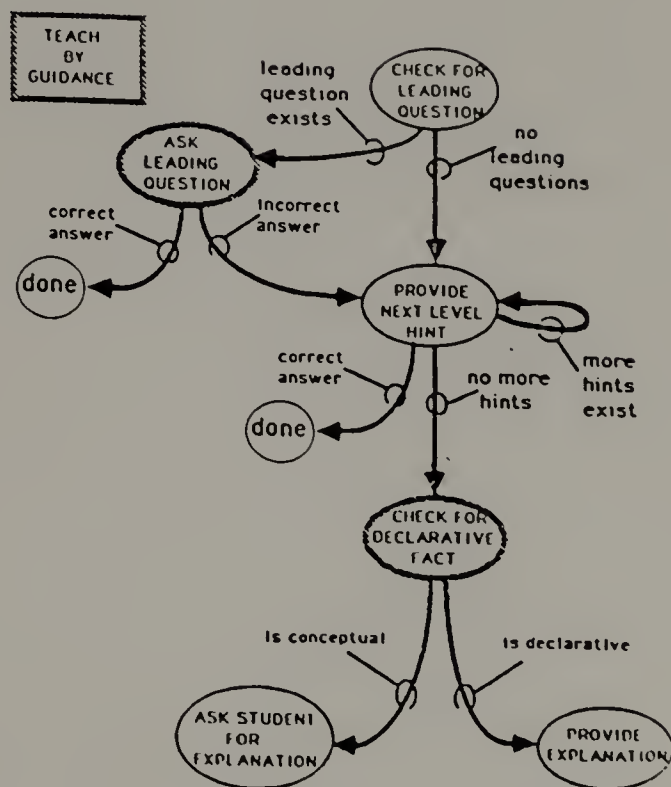
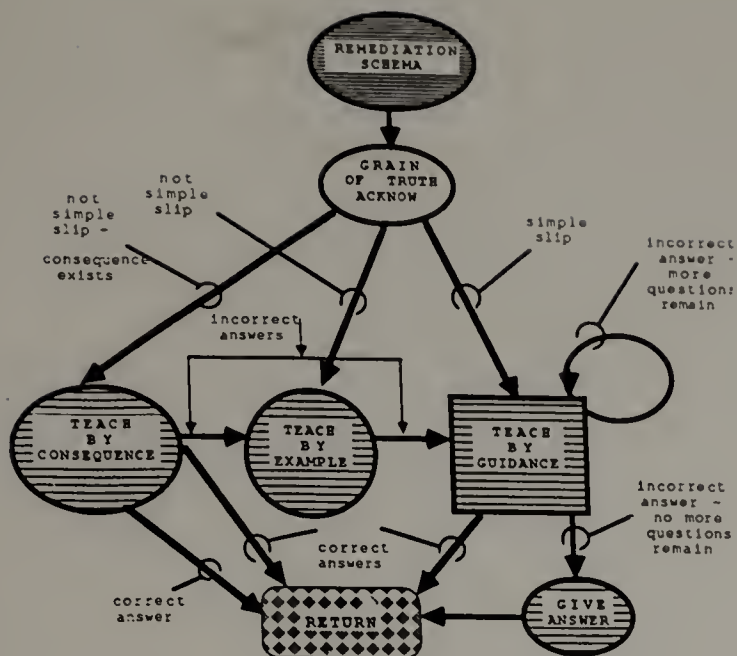


Figure 30 A Framework for Managing Tutoring Discourse

their own intuitions and did not correct them as long as their performances appeared to be moving closer to a precise goal.

In an industrial setting, trainees must learn to generate multiple hypotheses and to evaluate their own performance based on the effects of their actions on the industrial process. No tutor is available during working hours. Thus, the teaching strategy in the industrial tutor was to encourage students to trust their own observations and to learn through animated simulations of the process, trend analyses, and meters dynamically updated in real time. The textual dialogue provided added assurance that operators would extract as much information as possible from the data, and the system provided a mechanism to redirect them if they did not.

4.4.3 *Cognitive Input*

Cognitive science research provides the basis for selecting instructional strategies. One goal is to find out how people learn in a given domain. For example, cognitive science research is beginning to uncover common errors, probable misconceptions, and effective remediation in a number of domains. The importance of addressing common errors and misconceptions in physics is well-documented [McDermott, 1984; Clement, 1982], and the intelligence of a physics tutor depends greatly on making that knowledge explicit.

A tutoring system should allow students to generate hypotheses necessary for expanding their intuitions. Students must use these hypotheses to develop their own models of the physical world and to uncover or listen to their own scientific intuitions.

Cognitive scientists have had an implicit mandate to determine this knowledge. They now research how students reason about qualitative processes, how teachers convey prerequisite knowledge needed for learning in the domain [Halff, 1988], how predictions can be made about bug stability [van Lehn, 1988], and what tools are used by experts working in the field. For example, to build the thermodynamics tutor (see Section 2.1.3) required (1) investigation of the tools currently used by physicists, (2) studies focused on cognitive processes used by novices and experts to understand thermodynamics, and (3) research into novice physics problem-solving as compared with that of experts.

Cognitive science research elucidates actions taken by experts to make measurements or to perform transformations in the domain. This is called "heuristic knowledge" and is defined as knowledge about *how to solve problems*. (See Section 3.3.1 for a discussion about how to elicit this knowledge from experts.) This knowledge differs from procedural knowledge in that it adds neither content nor concepts to the domain, but rather describes actions taken by experts to use conceptual and procedural knowledge. This knowledge has rarely been included in tutoring systems, but must be included if tutors are to monitor their students' problem-solving activities and experiential knowledge about how to work in a field.

The Recovery Boiler tutor (Section 2.1.1) begins to articulate this kind of knowledge by recording explicitly the operations performed by trainees to solve emergencies. It shows students their false paths and gives some of the reasons behind rule-of-thumb knowledge used to solve problems. A tutor might provide a student with a variety of examples from which she/he can explore a large space of problem-solving activities. Such tutors show students their own path, a preferred path, and perhaps, in time, their own underlying solution processes. Simply elucidating these operational components of problem-solving in a domain and the rules that apply is obviously not sufficient to understand how a person learns in a new domain. However, such traces can begin to help a student *learn how to learn* and help to clarify the processes behind problem-solving behavior.

4.4.4 Domain Input

An "in-house" domain expert is critical to building an intelligent tutoring system. By in-house, we mean that the domain expert must be part of the project team (available at least weekly) for anywhere from six months to several years while domain knowledge is acquired. Less commitment or a less active role would provide a less than adequate transfer of domain knowledge.

In the tutors described in Sections 2.1.1–2.1.3 domain experts were integral in the programming effort. In RBT, the programmer, project manager, and director of the project were themselves chemical engineers. More than 30 years total of theoretical and

practical knowledge about boiler design and teaching strategies were incorporated into the system. Development time for this project would have been much longer than the actual 18 months if these experts had not already identified the chemical, physical, and thermodynamic characteristics of the boiler and collected examples of successful teaching activities.

The second language tutor was developed by a person who holds a graduate degree in teaching English as a second language and who has spent more than seven years using the Silent Way to teach intensive language courses for people living in foreign countries. This programmer was able to perform knowledge engineering on herself to extract and encode the tutoring knowledge.

The physics tutor was being built after months of part-time work with ESE Consortium members who were physicists and astronomers. Potential users of the systems, high-school and college physics teachers, contributed teaching and environment expertise. From this effort, more than 100 pages of rules, processes, screen designs (including help activities about physics), and cognitive studies (identifying educational goals, potential errors and misconceptions) were produced before any code was written.

Sometimes domain knowledge can be encoded through the use of expert shells which frequently use rule-based systems to record topics and rules that impact on those topics in the domain knowledge base. Anderson et al. [1985] used GRAPES [Sauers & Farrell, 1982] to represent the rules programmers use for solving problems, to describe LISP functions, and to represent higher level programming goals. He used *buggy* rules to represent misconceptions that novice programmers often develop during learning. Streibel et al., [1987] used OPS5 to write rules for genetic problem-solving and to encode teaching strategies.

Desiderata for acquiring domain knowledge include the following:

1. Domain experts should be very good and, if possible, the best in the field [Bobrow et al., 1986]. For example, Dendral [Lindsay et al., 1985], an expert system for the generation and testing of hypotheses about chemical structures and spectroscopic data, was built with a team that included Joshua Lederberg, a Nobel-prize winning geneticist, Carl Djerassi, a world-class expert on mass spectral analysis, and several professional

chemists and computer scientists. Perhaps coordinating with a university professor, even via long distance, is a viable option.

2. Domain experts are expensive. Using knowledgeable people in the domain is expensive and time-consuming. However, their willingness and availability to participate is critical to the knowledge engineering process. Assigning the task to a person of lesser ability, or worse, to a person "with time on their hands," might doom the project to failure. Enthusiastic support from funders and supervisors, including sufficient allocation of resources, human and otherwise, is a prerequisite to success of the project.

3. Domain experts might have incomplete knowledge or conceptual holes in their knowledge. For this reason, several experts are needed to test and modify domain knowledge throughout the life of the tutor.

4. Similarly, domain knowledge might be distributed [Bobrow et al., 1986; Mittal & Dym, 1985]. This means that knowledge can be spread so diffusely among different research projects and experts as to leave any system unfinishable by a single expert, or sometimes even several experts. Thus, domain knowledge must be acquired incrementally and must be prototyped, refined, augmented and reimplemented. This means that the time needed to build a tutoring system "should be measured in years, not months, and in tens of worker-years, not worker-months" [Bobrow et al., 1986].

5. Domain knowledge as found in textbooks is typically incomplete and idealized [Bobrow et al., 1986]. Thus, textbooks might be inappropriate as a primary source for either domain or teaching knowledge in a knowledge-based tutor. They rarely contain the common-sense knowledge that expert tutors or professionals in the field use to choose the next teaching strategy or to solve a difficult problem. Books tend to present clean, uncomplicated concepts and results. To solve or to teach real-world problems, a tutor must know messy but necessary details of real or perceived links between concepts and unpublished rules of teaching and learning.

4.5 Summary and Discussion

Construction of a knowledge-based tutor requires a good deal of planning, organizing, and management. The process is typically long and complex. A developmental methodology was presented here which included task lists, questions to be asked, activities to be performed, and issues to be considered. This chapter suggested that a community memory be developed to encode and organize such teaching and learning information. The process of gathering this information from teaching, learning, domain and computer specialists is currently hampered by lack of a common vocabulary. This chapter provided initial steps toward establishing such a vocabulary by outlining the criteria experts should use in making their contributions. The community memory itself would provide a focus for articulating such knowledge about thinking, teaching, and learning and thus would contribute to communication among experts. The chapter also dealt with the logistics and pragmatics of constructing a large knowledge-based tutoring system. It suggested specific prerequisites for each of the four major players: domain, teaching, cognitive, and communication expert. Goals, tasks, and criteria upon which job completion is defined were listed. An administrative time table presented goals for each expert in terms of feasibility studies, formulations, and maintenance of the system.

An extended methodology for developing these tutors began in Chapter 3 which provided a classification of the knowledge needed to be encoded. This chapter then described some of the administrative issues to consider and the following chapter will continue with a view of available artificial intelligence tools to facilitate building these systems.

CHAPTER 5

IMPLEMENTATION ISSUES

5.1 Tools for Representing Knowledge and Control

Authors of knowledge-based tutors might some day purchase off-the-shelf Artificial Intelligence (AI) tools to develop their systems. They might choose from semantic networks, natural language interfaces, planners or explanation systems. Today, however, few such AI tools exist outside of laboratories. When they can be purchased, they are often incompatible with the language, software package, or hardware already in use.

Chapter 2 outlined an epistemology for identifying the knowledge needed in a knowledge-based tutor. This chapter focuses on how to encode that knowledge. It suggests tools and methodologies that might become available and evaluates advantages and disadvantages of each. Thanks to active researchers and a wealth of recently formed companies, we expect that such software as described here will in fact become available in the short term. The next chapter describes issues about software and hardware selection, such as alternative programming languages, software tools, knowledge engineering tools, and hardware choices.

Cycle of Development in Artificial Intelligence Systems. Development of intelligent tutors, like development of any AI system, requires several iteration cycles: computer scientists and instructional designers first collaborate on the design; a prototype is implemented and evaluated; and the prototype is modified and refined based on information gained through testing (see Chapter 4). This cycle is repeated as time permits. AI programming frequently involves creating machine behavior that has not been seen be-

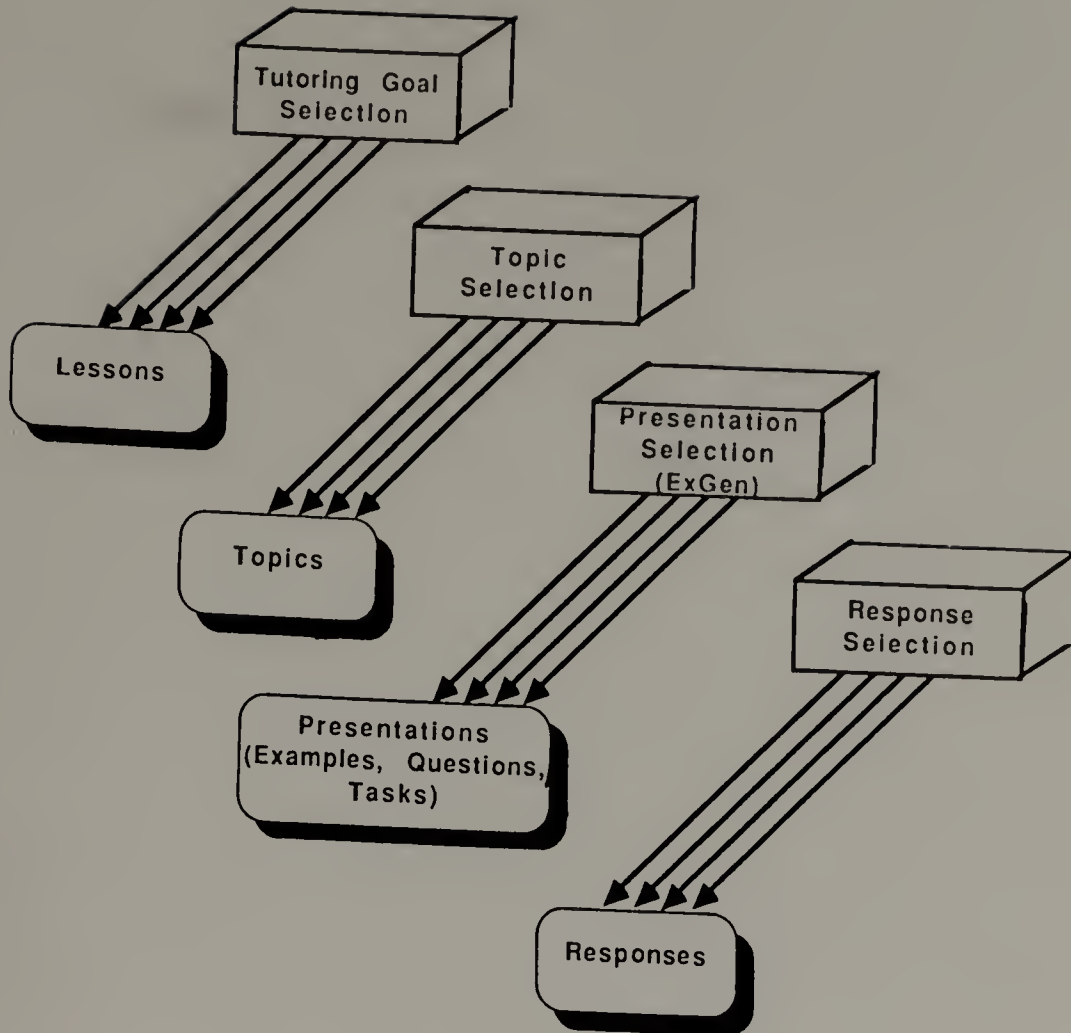


Figure 31 Representation and Control in a Tutoring System

fore; programmers need to experiment with designs that evolve only as their ideas and experience evolve [Brattle Corp., 1984]. Thus AI programming requires skills and tools that allow a developer massage a program until it exhibits the sought-after behavior (see Chapter 6).

Representation and Control. Artificial Intelligence programs require a representation of knowledge and control structures which define the way an interpreter traverses that knowledge (see Figure 31. Knowledge representation refers to how knowledge is stored and may include knowledge bases to hold concepts, activities, relations between topics, a variety of the lessons, topics, presentations, and response selections available to the tutor. Control refers to selection of appropriate pieces of knowledge for making a diagnosis, a prediction, or an evaluation. For tutoring, control structures might be specified at the

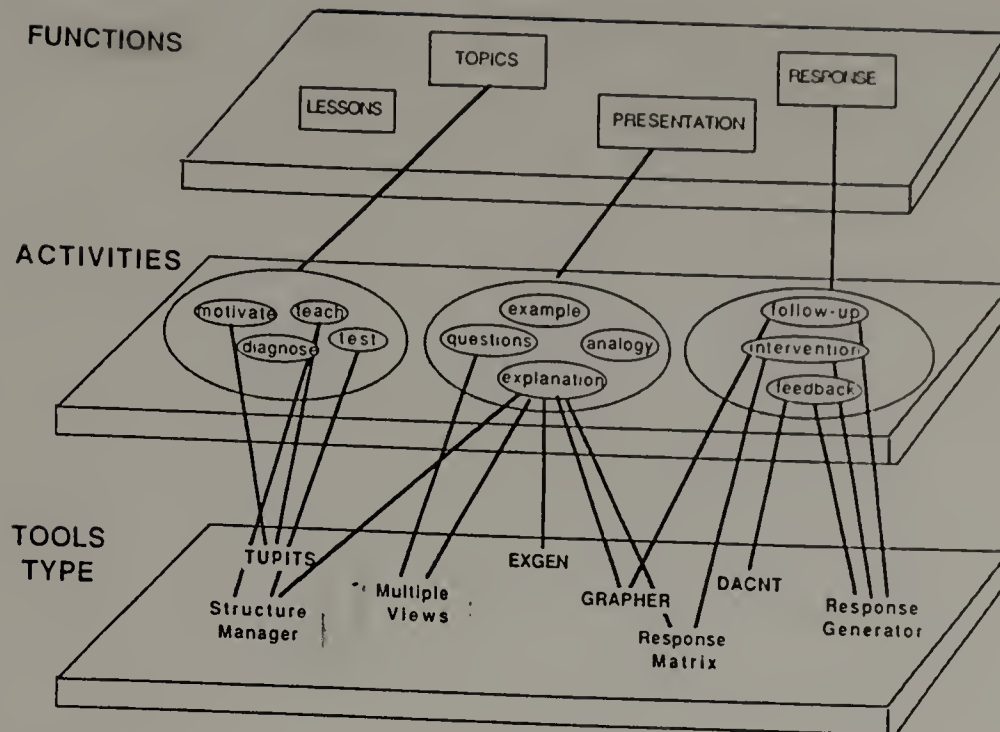


Figure 32 Tools for Reasoning about Tutoring Knowledge

four levels indicated in Figure 31, separately defining control for selection of lesson, topic, presentation, and response selection. Control structures might be motivated by specific instructional and diagnostic goals; thus, for example, one control structure might produce a predominantly Socratic interaction or another might produce incrementally generalized new problems to solve or concepts to explain. Control structures are specific to a particular level of representation and uniquely define the reasoning to be used for that knowledge base.

Encoding this large amount of knowledge is difficult and time consuming. The tools described in this chapter facilitate representation of and reasoning about such knowledge (see Figure 32). For each knowledge base shown in the figure (lessons, topics, presentation, or response), tools facilitate reasoning about or representing that knowledge. For example, if the tutor is about to motivate or teach a topic (from the TOPICS knowledge base), it can choose to provide examples or questions (from the PRESENTATION knowledge base). Several tools are available at each step. Only a few will be described

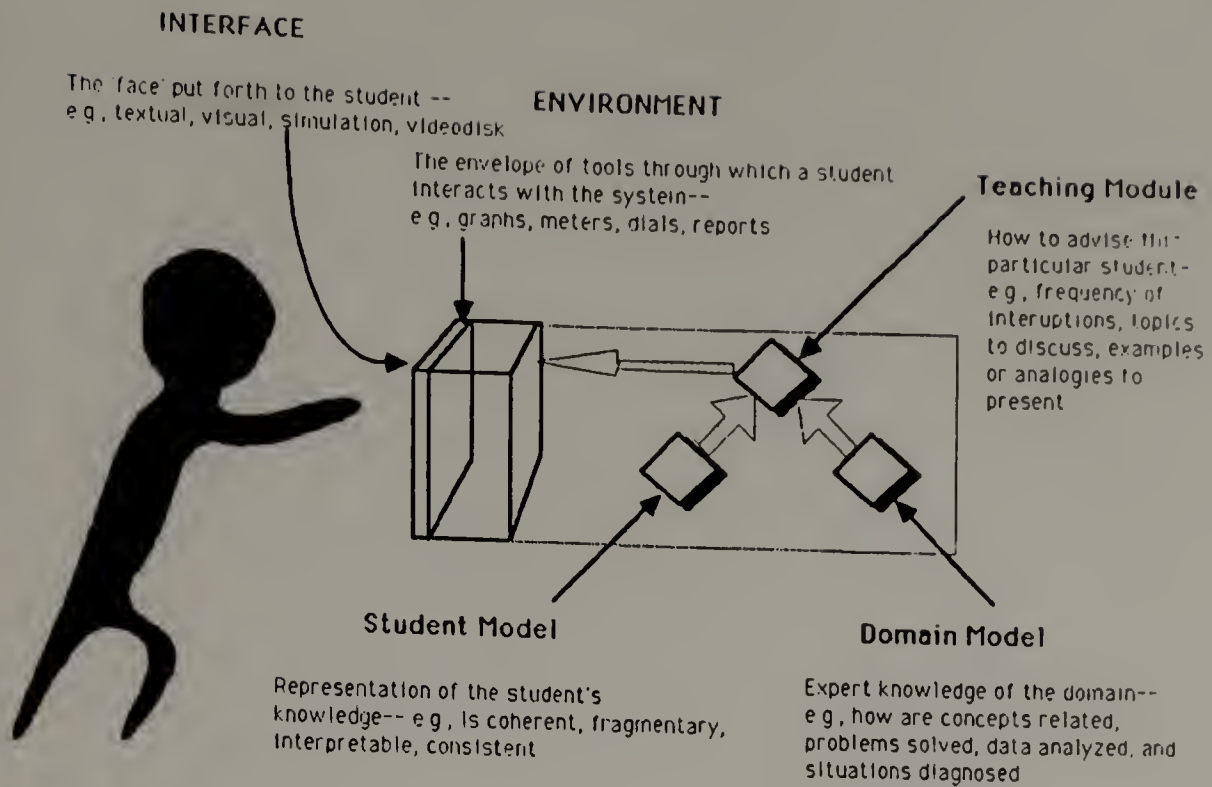


Figure 33 Four Models in a Tutoring System

here, namely TUPITS, Exgen, Response Matrix, and DACTN. We divide the discussion into two parts, separately describing tools for representing tutoring knowledge and those for representing control of search within that knowledge.

Tools are described along with a system that initially demonstrated the power of these tools and the implementation issues raised. Authors of these systems should consider similar issues before selecting a tool for a specific implementation. An attempt is made to define criteria by which such tools might be judged. The knowledge to be encoded is discussed in four sections, detailing domain, student, tutoring and communication knowledge as shown in Figure 31 and described in Section 2.4.

5.2 Encoding Domain Knowledge

One of the first tasks facing the knowledge engineer is to represent domain knowledge, including that knowledge used by the system to perform problem solving in the domain

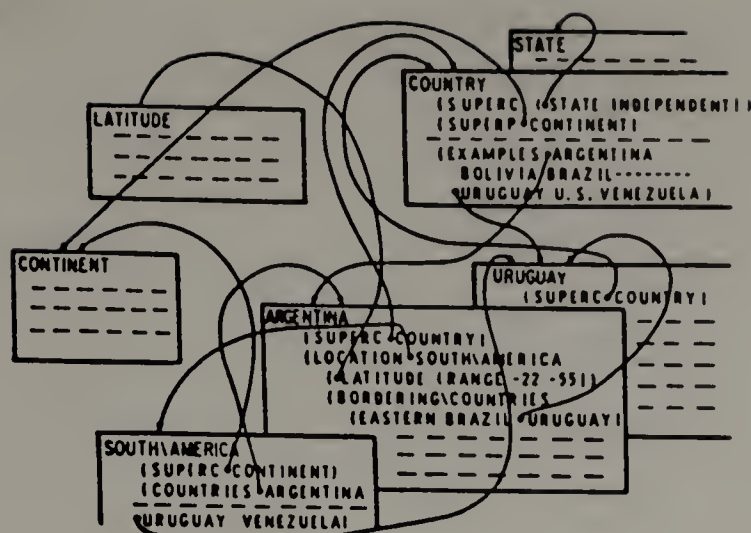


Figure 34 A Portion of SCHOLAR's Semantic Net [Carbonell, 1970]

and to evaluate the student's solutions to the same problems. In some cases domain knowledge encodes problems for the student to solve or topics to discuss; sometimes it is responsible for solving those problems (e.g., SOPHIE [Brown et al., 1982]), or contains the processes that first translate a student's input into usable form for evaluation against system knowledge. Sometimes, as a result of adding heuristics to this component, a system is able to explain itself and to talk about its own problem-solving reasoning (e.g., GUIDON [Clancey, 1982], SOPHIE [Brown et al., 1982]). In such a case, a system is said to be "aware of its own knowledge." Heuristics enable the system to comprehend a broader range of input (e.g., WHY [Burton et al., 1982], SCHOLAR [Carbonell, 1970]). It results in flexibility in the system's understanding of a student's response and its ability to interpret the nontraditional response, i.e., responses couched in actions or words that differ from the norm. It also enhances the system's ability to express its own knowledge of the domain.

5.2.1 Knowledge Representation

Knowledge representation, as will be demonstrated in the several examples in this section, is critical in any tutoring system. Defining a knowledge representation scheme means establishing symbols and defining how these symbols will be arranged to reason about and produce meaningful descriptions of particular concepts. The representation itself is simply a set of syntactic and semantic conventions or symbols that make descriptions of elements or procedures possible. It is a stylized version of the world [Cherniak & McDermott, 1985]. Experience with AI systems has shown that designing a good representation is often the key to transforming a hard problem into a simple one. The existence of an underlying or internal representation provides a template upon which the AI system is built.

Networks that Represent Topics. One popular way to represent knowledge is through semantic networks, as did Carbonell [1970] in his SCHOLAR program. He suggested that semantic nets were a feasible model of the way people store and access information and assumed that systems could hold mixed-initiative dialogues (i.e., interactions that might be initiated by either the student or the system) by traversing such a network and asking students questions about the information. Objects and concepts were represented as nodes in a semantic network (see Figure 34), and SCHOLAR discussed the subject of South American geography (see Figure 35) by traversing the network to answer the student's questions. The system used hierarchical links to define super concepts and could perform simple inferencing through propagation of inherited properties. Thus, the system knew that Santiago is in South America since it is in Chile and Chile is in South America.

Networks that Represent Curriculum. Semantic networks can also represent a curriculum, or the tasks and skills to be taught. By continually refining the network path in response to deficiencies, the system can tailor each teaching sequence to an individual student. Such a network can continually undergo refinement and testing as the instruc-

SCHOLAR: Approx what is the area of Brazil.
 Student: 2500000 square miles
 SCHOLAR: Wrong. Please indicate if the following statement is correct: the area of Paraguay is approx 47432 square miles.
 Student: What is the area of Brazil?
 SCHOLAR: The area of Brazil is approx 3600000 square miles. Now answer the question you didn't answer before.
 Student: False.
 SCHOLAR: Rephrase your statement. I can't understand it.
 Student: Incorrect.
 SCHOLAR: Very good. The area of Paraguay is approx 157000 square miles. The language in Brazil is French. Right or wrong.
 Student: Wrong.
 SCHOLAR: You are correct. The language in Brazil is Portuguese. Is it true that the capital in Brazil is Brazilie?
 Student: Yes.
 SCHOLAR: That's fine. Approx what is the population in Brazil?
 Student: 60000000.
 SCHOLAR: Approximately true, but a more exact value is 70000000. What is Brazil?
 Student: Tell me about Peru
 SCHOLAR: Wait a minute. I am checking my files. Let's proceed now. Peru is a country. It is in South America. The population is approx 7000000 people. The capital is Lima. Now answer the question you didn't answer before.
 Student: Q/A.
 SCHOLAR: Do you want to change the general mode of the interaction to Q/A? Please answer Y or N.
 Student: Y
 SCHOLAR: OK, Mr. Jones. Let's go ahead.
 Student: Please, tell me more about Peru.
 SCHOLAR: The area of Peru is approx 480000 square miles. The language is Spanish.
 Student: How many ...
 :

Figure 35 A Conversation with SCHOLAR [Carbonell, 1970]

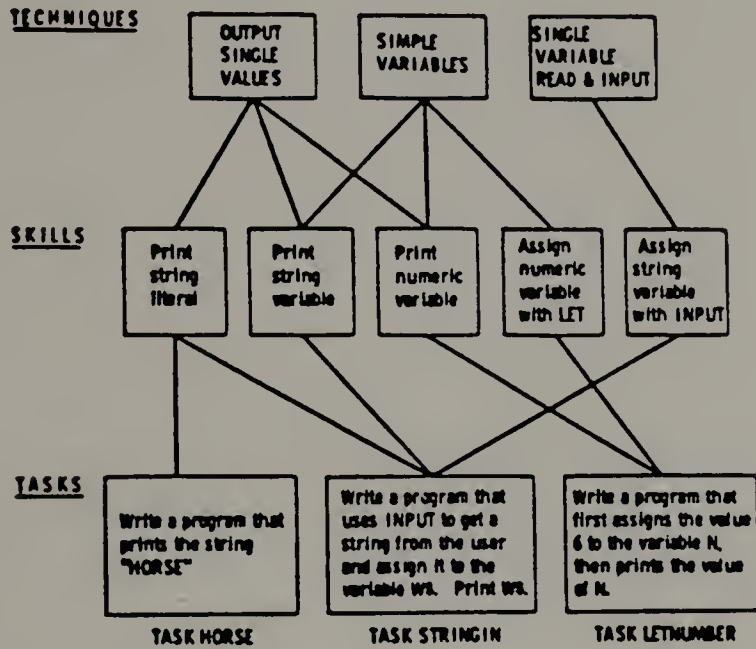


Figure 36 The Curriculum Information Network [Barr et al., 1976]

tional design research proceeds. BIP (Basic Instructional Program) was an early example of such a system teaching introductory programming in Basic [Barr et al., 1975]. It selected tasks from a semantic network and related tasks in the curriculum to issues to be taught Figure 36. In this system, curriculum was represented as techniques, skills, and tasks to be presented to the student. The network representation allowed the system to teach about complex hierarchies and from non-linear traversal between issues. In determining the next curriculum topic, the tutor constructed two sets of skills: those needing to be exercised and those that were sufficiently mastered to be assumed in a new exercise. The objective of this tutor was to select problems that would train the student in new skills without including any skills beyond the student's reach. In teaching a new skill, the tutor would recognize known materials as skills that were sufficiently mastered. A second version of BIP [Wescourt et al., 1977] ordered each skill as a network. Skills were represented along with requisite relations and pedagogical information, such an analogy to present to the student.

BIP did not have much information to relay to the student beyond the brief exercise stored in the semantic network. It did not have as one of its goals to understand Basic programming beyond the originally stored tasks, and it could not offer feedback beyond the simple hints stored with each task. Thus, it was unable to diagnose students' logical errors, to troubleshoot programming designs, or to suggest new solution plans. The system simply advised the student about programming constructs that should or should not be used.

Tutoring Primitives. We define **tutoring primitives** as those elements needed for tutoring, such as topics to be taught, specific tutoring responses, and possible student errors. Thus a domain knowledge base might hold a variety of examples, types of knowledge, tasks to be given to the student, and discourse states describing various human-machine interactions. For example, we used a network of Knowledge Unit Frames to represent the topics, examples, explanations, and possible misconceptions used in the tutors described in Section 2.1.3. The frames explicitly expressed relationships between topics such as

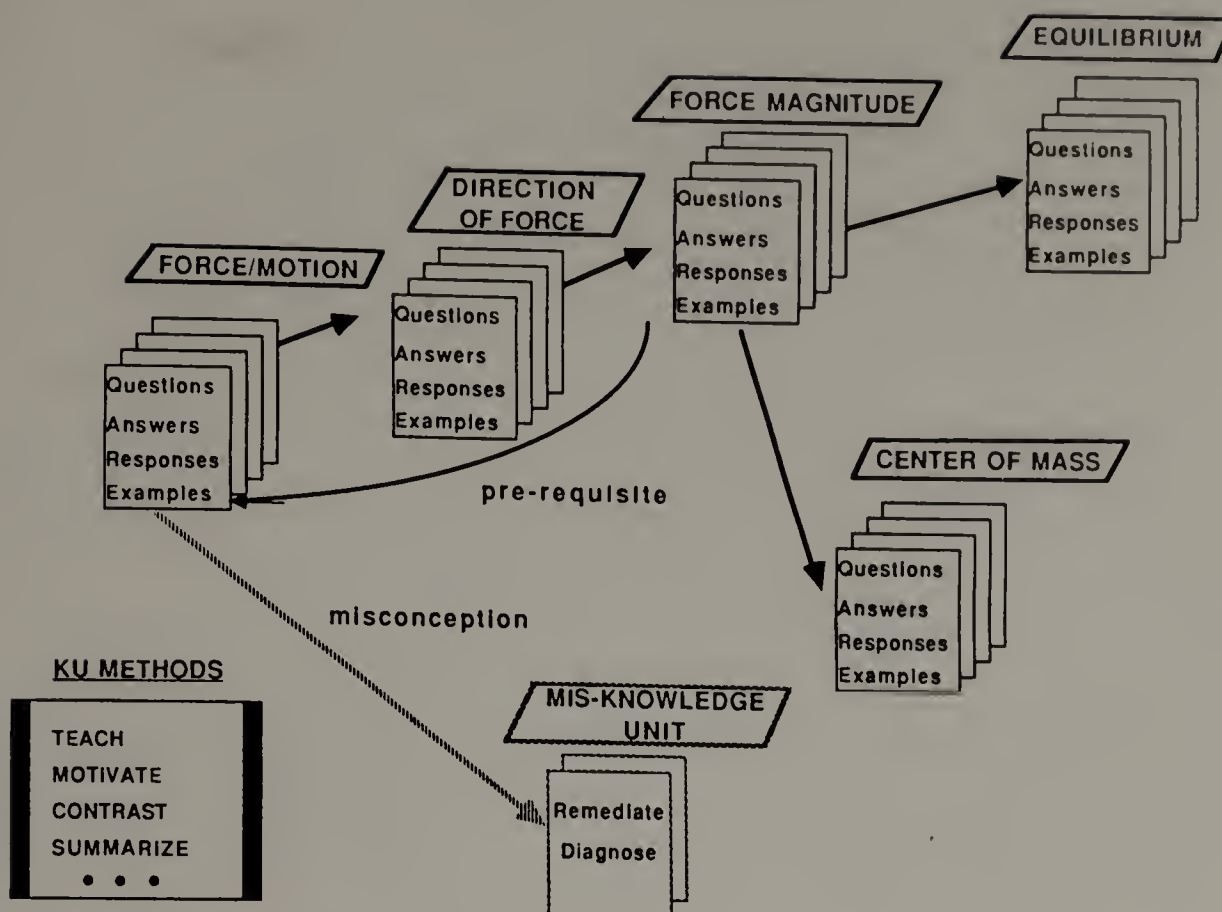


Figure 37 Hierarchy of Frames

prerequisites, corequisites, and related misconceptions (Figure 37). An important notion about the network is that is declarative—it contains a structured space of concepts, but does not mandate any particular order for traversal of this space. The network describes tutorial strategies in terms of a vocabulary of primitive discourse moves. It is implemented in a language called TUPITS¹ which was used to build both the tutors described in Section 2.1.3. It is object-oriented and provides a framework that the tutor uses to reason about its next action.

As shown in Figure 37, each object in TUPITS is represented as a frame and each frame is linked with other frames representing prerequisites, co-requisites, or triggered misconceptions. The primary objects in TUPITS are:

¹TUPITS (Tutorial discourse Primitives for Intelligent Tutoring Systems) was developed by Tom Murray and runs on a Hewlett-Packard Bobcat and an Apple Macintosh II.

- Lessons which define high-level goals and constraints for each tutoring session (see [Woolf et al., 1988]);
- Knowledge Units (KUs);
- MIS-KUs, which represent common misconceptions, wrong facts or procedures, and other types of “buggy” knowledge;
- Examples, which specify parameters that configure an example, diagram, or simulation to be presented to the student;
- Questions, which define tasks for the student and how the student’s behavior during the task might be evaluated; and
- Presentations, which bind an example and a question together.

MIS-KUs, or “Mis-Knowledge Units,” represent common misconceptions or knowledge “bugs” and ways to remediate them. These are inserted opportunistically into the discourse. The tutoring strategy parameterizes this aspect of Knowledge Unit selection by indicating whether such remediation should occur as soon as the misconception is suspected, or wait until the current Knowledge Unit has been completed.

Acquiring Tutoring Primitives Knowledge. Knowledge acquisition means accessing and encoding the questions, examples, analogies, and explanations that an expert might use to tutor a particular domain, as well as the reasoning he/she might use to decide how and when to use tutoring primitives. The TUPITS system has a graphical editor which is used by the instructional designer to encode and modify both primitives and the reasons why one primitive might be used over another. The graphical editor allows a teacher to generate and modify primitives without working in a programming language. The system lists a series of primitives; the user chooses a primitive object, bringing it into an edit window, from which he/she can modify it or build new ones.

5.2.2 *Issues Related to Encoding Domain Knowledge*

As shown by the examples above, researchers have used a variety of knowledge representations, such as semantic networks and frame-based systems, to encode domain knowledge. The choice between these and other representations should be based on reasoning about several issues. Several issues are presented below.

Grain Size. A knowledge representation "divides up the world" according to a metric known as grain size, often measured along an epistemological continuum beginning with "bits-and-pieces" and ending with "chunked elements." At the bits-and-pieces extreme, distinct and unconnected elements are used to represent elements in the subject area (as in WHY [Burton et al., 1982]), whereas at the chunked extreme, relations and morphisms between elements indicate temporal, logical, or pedagogic connections between the elements (as in GUIDON [Clancey et al., 1982]). The chunked representation uses connectedness or layers of importance and logical precedence of elements to help create groupings for tutoring the knowledge.

The grain size measurement indicates more than a passive concern for epistemology or implementation style. It seems to establish a limit on the flexibility of the system's ability to teach, in that the bits-and-pieces approach can only teach elements independently of each other, whereas the chunked approach might use the clustering of domain connections to structure its tutoring. Moreover, the chunked approach, when modeling the abstractions and relations uncovered by cognitive scientist concerned with how a human learns the subject area, is better suited to capture generalizations and strategies of learning.

For example, psychological studies suggest that experts chunk their knowledge in a number of fields: chess [Chase & Simon, 1981], story understanding [Bower et al., 1979], physics [Larkin et al., 1980], and programming [Gerhart, 1975; Soloway et al., 1983; Bonar, 1985]. These studies suggest that experts use plans, abstractions or templates to understand a domain. Computational chunking of mathematical knowledge has been de-

scribed [Rissland, 1978], and Minsky has suggested that the computer's ability to "learn" and "understand" knowledge is ultimately connected with its ability to "chunk" knowledge [1983]. Minsky recognizes the pedagogical importance of connectivity information:

I think it's bad psychology when teachers shape our children's mathematics into long, thin, fragile definition tower-chains instead of robust, cross-connected webs. Those chains break at their weakest links, those towers topple at the slightest shove. . . . Perhaps, instead, we ought to help them build more robust networks in their heads. In real life, the best ideas are as cross-connected as can be.

The secret of what something means lies in the ways it connects to all the other things we know. The more such links, the more a thing will mean to us. . . . Rich, multiply-connected networks provide enough different ways to use knowledge that when one way doesn't work, you can try to figure out why. When there are many meanings in a network, you can turn things around in your mind and look at them from different perspectives; when you get stuck, you can try another view. That's what we mean by thinking [Minsky, 1983].

In terms of implementation, the bits-and-pieces approach implies, but is not limited to, a small number of loosely connected nodes or production rules which are then sparsely interwoven. A chunked system, on the other hand, is often implemented in frames, scripts, or layered and richly linked semantic networks. Most tutoring systems lie somewhere between the two extremes. Some even combine the two approaches. For example, GUIDON used both approaches. Originally, it was built from hundreds of not too well-connected production rules (see next section), and later was reconstructed as a richly connected network system modeling the abstractions and generalizations used by medical interns. It performed better using the hybrid system.

Example Grain Sizes. The WHY system [Stevens & Collins, 1977] is an example of a bits-and-pieces representation. It used a semantic network of sparsely connected nodes to

⋮

WHY: Do you think the Amazon jungle has heavy rainfall or light rainfall?
(Rule 1: Ask for a prediction about a particular case.)

Student: Heavy rainfall

WHY: Why does the Amazon have heavy rainfall?
(Rule 2: Ask for prior cause.)

Student: Because it's near the Atlantic

WHY: Yee, the Amazon jungle is near the equatorial Atlantic. How does that affect rainfall in the Amazon jungle?
(Rule 3: Ask for intermediate cause.)

Student: The water in the current is warm

⋮

WHY: Do you think that any place with mountains has heavy rainfall?
(Rule 6: Form a general rule for an insufficient cause.)

Student: Yee

WHY: Southern California has mountains. Why doesn't Southern California have heavy rainfall?
(Rule 7: Pick a counter-example for an insufficient cause, and Rule 2: Ask for prior causes)

⋮

Figure 38 A Conversation with WHY [Stevens & Collins, 1977]

describe information about rainfall. Based on a Socratic teaching paradigm, it explained data at the point in the knowledge base where the student's error could be mapped and then moved to the point where the correct knowledge was located, eventually questioning the student about every intervening topic. However, because the system was inflexible about its responses, it tended to badger a student about each detail of the intervening topics when it could have generalized over incorrect or missed topics. By assuming that tutoring can begin at any point in a network and can continue to any connected point, the WHY system attempted to "feed" data to students in a piecemeal fashion.

The WHY system could have chunked its domain knowledge. In fact, psychological studies accompanying the implementation effort uncovered extensive data on how people think about rainfall and how common misconceptions tend to cloud people's reasoning. If these data had been included, the system might have predicted student behavior or avoided repetitious questioning by, for instance, exploring the student's knowledge of factors such as ocean currents or wind direction and predicting from that his/her knowledge about rainfall. The system might gloss over, or briefly mention, additional remaining factors and not ask the student to elaborate so much detail.

When data are multiply connected, common variables can be taught as units, allowing them to be abstracted and explained in terms of their common causal factors. This dependency would improve the student's ability to deal in generalities and help focus the machine dialogue on key concepts.

GUIDON [Clancey, 1979] is an example of a large teaching system in which the bits-and-pieces approach failed and a later implementation, based on chunked knowledge, succeeded. MYCIN, the large medical expert system from which GUIDON taught, is now used by doctors to diagnose and treat infectious diseases. However, doctors work at the so-called "compiled" level where they use rules stripped of the causal reasoning and cross-links needed by a student to learn the same rules. In order to teach from these rules, GUIDON had to "decompile" and cross-index the stripped down rules and provide the student with generalizations and references between the rules.

GUIDON was originally implemented by "reversing" the rules of MYCIN. The original system failed, Clancey said, because medical diagnosis is not taught "cook-book" style [Clancey, 1979a]. That is, medical practitioners do not diagnose diseases using perfect recall on a huge number of medical facts and rules. They use common variables to abstract their knowledge. For example, "yellow coloring is suggestive of liver disorders." To quote Clancey and Letsinger [1981]:

A psychological model of diagnostic thinking cannot be represented by simply rewriting MYCIN's rules. Instead, the representation and interpreter must be augmented, and the rules organized by multiple, orthogonal structures.

For example, a simple interpreter change is to allow incoming data to cause new subgoals to be set up and pursued. When a physician hears that the patient has a stiff neck, the association to meningitis might come to mind, prompting him to determine if the patient has a headache as well. To bring

about this effect in NEOMYCIN, a new type of antecedent rule had to be allowed, and a local change made to the . . . control structure.

Clancey's experience suggests that the original GUIDON system was ineffective exactly because inferences between possibly connected elements of the domain were not made by the system. After he added layers of meta-knowledge to include generalizations and common variables on which the system could draw, the system became more effective.

Multiple Representations. A second issue in the design of domain knowledge is the choice of representations. At least two roles are played by the chosen representation. One is to solve the problem given to the student (e.g., SOPHIE and GUIDON). A second is to use the representation to communicate with the student. The first role suggests that the representation should be powerful for problem-solving (e.g., predicate calculus or production rules). The second suggests that the representation should be useful for description and explanation of problem solving knowledge. Possibly it should contain a subset of terms he/she uses to think about problems, e.g., spatial or temporal relations between topics should be expressible in the representation. For instance, if a system solved problems in predicate calculus, it should also represent its solutions in a language other than predicate calculus in order that the solution be communicated to the student. Most systems fail to do both; their representation either solves the problem or explains it, but not both. Goldstein [1977] used the term **opaque experts** to refer to systems that cannot explain their reasoning and contrasts these with **articulate experts**, which can explain their problem-solving activities.

The two roles of representation raise some complex issues. Can the same representation be used for both problem-solving and communication? Conversely, could a representation that constitutes an expressive language in which to talk to the student have enough logical consistency for problem-solving behavior? What is to be done with rules and concepts that are expressible only in the more efficient problem-solving repre-

sentation and are not directly translatable into a language accessible to the student? How important is it to make the mechanics of problem-solving at least theoretically accessible to the student? For instance, if a system used powerful inference routines to represent domain knowledge, should these routines be explainable to the student? How can the inferencing mechanism and the rules be conveyed to the student in terms of his /her own language? These issues remain active as the development of tutoring systems continue.

Example Domain Representations. SOPHIE was a landmark effort in the development of domain representations [Brown & Bell, 1977]. It used multiple representations of knowledge to reason about its domain of electronic circuits and to discourse with the student about circuits. It reasoned about the student's input by using syntactically meaningful categories such as "resistors," "transistors," and "measurements," which were associated with grammar rules used to parse the student's input. Each category also specified the appropriate electronics rule to use, e.g, Kirchoff's Law, which gives the value of the current component when other terminal component values are known. SOPHIE could answer the student's hypothetical questions about circuit values or generate explanations about possible faults in the circuit. SOPHIE had knowledge about the rules it used to solve problems. It described this knowledge to the student and reasoned about a student's partial solutions. It was able to suggest that a student's hypothetical test on the circuit was superfluous in light of existing data that the student had about the circuit. By careful translation of its reasoning into natural language explanations, SOPHIE produced information that might otherwise have been unavailable to the student because of the discrepancy between its problem-solving and its communication representation, see Section 5.6.4.

WEST [Burton & Brown, 1982] used an embellished rule-base language to reason about a student's skill acquisition in a game playing environment. It coached the student by modeling pieces of his/her skills and expressing these, along with explanations and suggestions about the "optimal" move, given the conditions of the game.

GUIDON [Clancey, 1982] provided an example of the tension produced by the two roles of the representation chosen for the domain knowledge. GUIDON consisted of nearly 1,000 rules to diagnose diseases, yet this representation alone was **not** adequate to use in conversation with the student. The rules were sufficient for problem solving and enabled the diagnosis and suggestion of a treatment for diseases. However, the student's needs, as a learner, required a more complex representation, one which included logical relations, generalizations, and associations that the original rule set was missing. This discrepancy led to a reconfiguration of the domain knowledge into a new system, called NEOMYCIN [Clancey & Letsinger, 1981].

5.3 Encoding Control Knowledge

In an AI system control defines search through the knowledge structures in order to select appropriate information for making a diagnosis, a prediction, or an evaluation. In knowledge-based tutor systems, control enables the tutor to select responses and to tailor them for the individual. This section describes two such control structures. The first has been built in conjunction with the TUPITS system described in Section 5.2.1 above, and the second facilitates generation of examples from a "seed" example base.

Control in the TUPITS system. Control in TUPITS (see Section 5.2.1), is achieved through information associated with each object which allows the system to respond dynamically to new tutoring situations. For instance, Knowledge Units, or topics represented as objects, have procedural "methods" associated with them that:

- teach their own topic interactively;
- explain knowledge didactically;
- teach their own prerequisites;
- test students for knowledge of that topic;

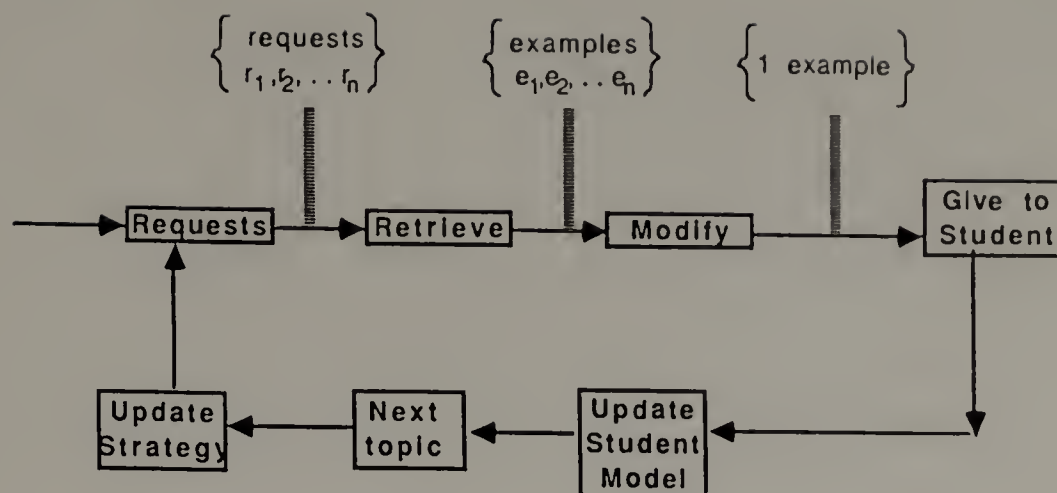


Figure 39 Reasoning About Examples

- summarize themselves;
- provide examples of their knowledge (an instantiation of a procedure or concept);
- provide motivation for a student learning the topic; and
- compare this knowledge with that of other Knowledge Units.

A specific tutoring strategy manifests itself by parameterizing the algorithm used to traverse the knowledge primitives network based on classifications of and relations between knowledge units. Several major strategies have been implemented thus far. For example, the tutor might always teach prerequisites before teaching the goal topic. Alternatively, it might provide a diagnostic probe to see if the student knows a topic. Prerequisites might be presented if the student doesn't exhibit enough knowledge on the probe. These prerequisites may be reached in various ways, such as depth-first and breadth-first traversal. An intermediate strategy is to specialize the prerequisite relation into "hard" prerequisites, which are always covered before the goal topic, and "soft" prerequisites, taught only when the student displays a deficiency.

Control and Reasoning about Examples. Another example of reasoning about tutoring primitives is shown by the actions of ExGen [Suthers & Rissland, 1988; Woolf et al., to appear]. ExGen takes requests from the various components of the tutor and produces an example, question, or description of the concept being taught. A "seed" example base contains prototypical presentations of each type. ExGen's modification routine expands this into a much larger virtual space of presentations as needed. The goal is to enable the tutor to have flexibility in its presentation of examples and questions/tasks that accompany those examples, without needing to represent all possible presentations explicitly.

Requests given to ExGen are expressed as weighted constraints called *requests* (see Figure 39). The constraints are written in a language which describes logical combinations of the desired attributes of the example, and the weights on them represent the relative importance of each of these attributes. The returned example generally meets as many of the constraints as possible in the priority indicated by the weights.

ExGen is driven by *example generation specialists*, or knowledge sources, each of which examines the current discourse and student models and produces requests (weighted constraints) to be given to ExGen. These example generation specialists may be thought of as tutoring rules, encoding such general prescriptives as "when starting a new topic, give a start-up example," or "ask questions requiring a qualitative response before those involving quantities."

The tutoring strategy impacts on this layer of presentation selection by prioritizing the relative importance of the recommendations produced by each of the example generation specialists. Within a strategy, each specialist has a weight multiplied by the weight of the requests produced by the specialists. Altering the behavior of the presentation control involves changing the weights on the specialists by selecting a new strategy.

For instance, one specialist requests that presentations describing the current Knowledge Unit be given, and another requests that the student be questioned. These com-

peting requests are ordered by the current tutoring strategy. We are also examining strategies for temporal ordering of the presentation of examples, such as Bridging Analogies [Clement & Brown, 1984; Murray et al., to appear] and Incremental Generalization.

5.4 Encoding Student Knowledge

A good human teacher knows how a student organizes his/her information, recognizes common learning problems, and distinguishes a student who organizes data in an unusual way yet has the correct information. Such a teacher is familiar with behavior that demonstrates both correct and incorrect rules.

A machine tutor should also have this kind of knowledge. It should reason about a student who solves a problem incorrectly and should gracefully try to remediate misconceptions. Machine reasoning about a student's thinking resides in the "student model." Early systems had no student model. At best, they used a stereotypic representation of domain knowledge tagged with those topics presumed known and unknown. Few effective student models exist today. This section describes several student model types, including *overlay*, *skill*, and *bug modeler*. An overlay modeler is a knowledge base which is a subset of the domain model; a skill modeler is an overlay modeler in which the domain knowledge is represented as skills; and a bug modeler is a knowledge base (not necessarily a skill modeler) in which student errors and misconceptions are represented.

Overlay and Skill Modelers. The WUSOR student model was designed as an axiomatic base of rules or topics overlaid on the domain knowledge [Carr & Goldstein, 1977]. The student's knowledge was seen as a subset or overlay of domain knowledge; items were tagged as "known," "unknown," or "insufficient data to know." Other systems used the overlay model and represented domain knowledge as skills or a proficiency gained through experience [e.g., BIP], rules [e.g., WEST], or preferred tutoring styles [e.g., WHUMPUS]. When skills are used to model domain knowledge, learning is mea-

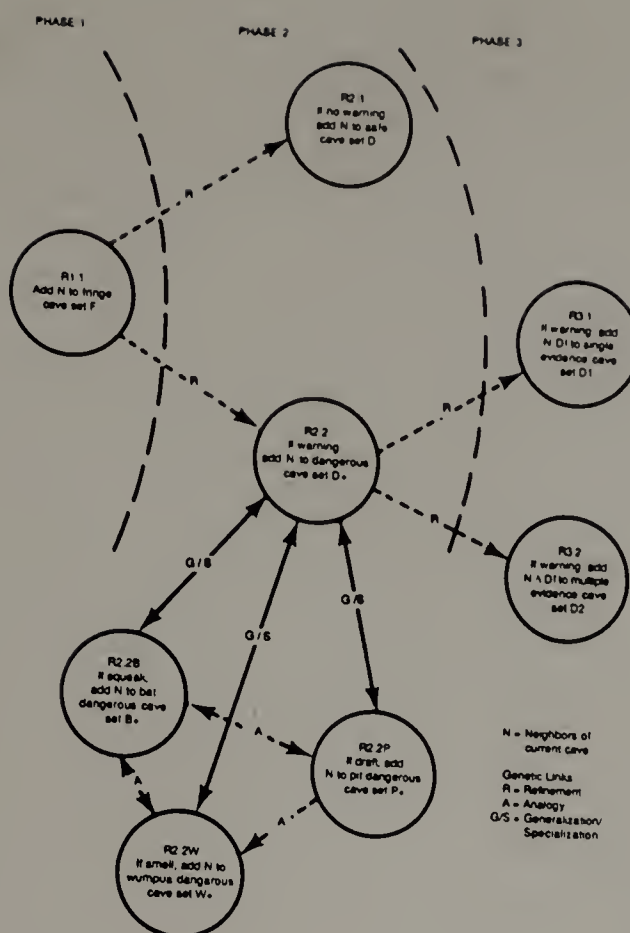


Figure 40 The Genetic Graph [Goldstein, 1982]

sured in terms of the student's ability to use the skill appropriately [e.g., BIP and WEST] or to explore the skill. The BUGGY research project [Burton, 1982] provides evidence that disputes the validity of using overlay knowledge to represent student knowledge.

WUMPUS was a skill modeler, which originally used a bits-and-pieces representation to model its domain, a simulated warren of 20 or more randomly connected caves, replete with dangers and warnings of a dragon [Goldstein, 1982]. The coach monitored a user's attempt to slay the dragon and advised about its location based on reasoning about bats, pits, and smells. A Genetic Graph mapped out procedural knowledge in the form of reasoning about analogy, specialization, generalization, of information gleaned by the user (see Figure 40). This evolutionary epistemology divided the set of rules into phases of increasing skills. It indicated the relation between current skills and prior ones and provided an indication of the kind of explanation to provide. The coach advised about

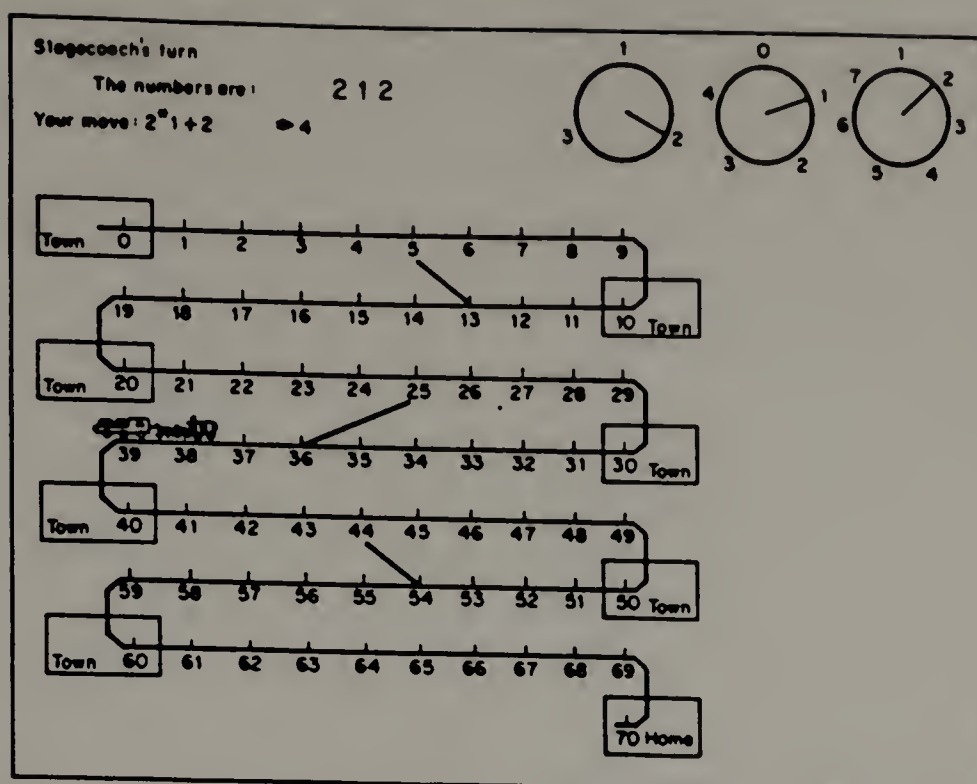


Figure 41 "How the West was Won" [Burton & Brown, 1982]

these procedural skills and recognized missing skills, such as a user's failure to generalize from multiple evidence about the existence of a pit. The tutor customized its advice based on whether the user failed to generalize, specialize, or use an analogy. However, the tutor failed to recognize the relative difficulty of particular skills and thus was rather insensitive to the player's ability to comprehend its advice based on present knowledge. It had no mechanism for mixed-initiative dialogue.

BIP [Barr et al., 1976] was an example of a tutor that used both an overlay and a skill modeler to represent student knowledge. It represented the skills of Basic programming in nodes of a branching tree, called the "Curriculum Information Network" (see Figure 36). Each node contained the skill, the exercises needed for testing the achievement of that skill, and correct and incorrect example programs tested a student's skill at each node. Additional information was represented, such as fine-grained knowledge about the evolution of the skill, such as analogies to the skill, generalizations from and specializations of the skill, and relative difficulty of learning the skill. The tutor searched

the curriculum network, through links that expressed the relationship between the skills and was enumerated by the nodes. It used inferences to move the tutor through the syllabus, making assumptions and evaluations about the student's weaknesses or strategies by evaluating his ability to perform the exercises. One limitation of BIP was that its skill modeler was used as the basis of all its tutoring activities: it could neither understand nor engage in activities beyond those represented on the nodes of its syllabus. It could not, for instance, reason about a misconception which arose from a student overgeneralizing about a programming concept or attributing power to a construct that it did not have.

WEST was another example of a tutor that used an overlay modeler. It demonstrated how the overlay methodology can break down. WEST coached students who solved algebra problems and the object of the game was to move a player across an electronic gameboard, see Figure 41. Movement of the player was geared to the largest algebraic value a student could construct from the value randomly provided by three electronic dice on the screen. The system moved the student's pieces a distance equivalent to the value constructed. A simple overlay modeler was used to project the student's recent choice of algebra operators on top of those that would have been chosen by an expert in the same situation. The tutor recognized the student's moves or missing skills and described these to him/her along with an example of the optimal strategy.

However, the system often misunderstood the student's problem-solving steps and reasoned that an error had been made, when in fact the student chose a move which didn't lead to a win. For example, the students often did not take advantage of features that might improve their performance. Often they might not use special features that would lead them to gain advantages such as "bumps" to remove an opponent's icon and place it several positions behind and "shortcuts" to reduce the normal pathway to the goal. Also, because the student enjoyed watching the icon move more slowly across the board, he/she might not take advantage of these moves. Yet, the system evaluated failures to use "bumps" as a lack of knowledge on the student's part, then "missing bumps" were incorporated into the student model in terms of missing skills. The system recorded that

the student was not capable of writing the preferred algebraic equation, when, in fact, the student's failure to do so was predicated on other goals. The original WEST system did not understand this situation, though a later implementation did.

WEST could have either included the bug in its student model or could have questioned the student about his motives before accusing him of not being able to write the requisite equation. However, the system was designed to entertain as well as to educate and the designers chose not to question the student's goal. Its methodology was to remain unobtrusive and to interrupt as little as possible.

Bug Modelers. Though the systems described thus far could present a variety of information to a student, they could not deal effectively with student errors. In order to do so, the system would need to include bugs in its student model and would need to bias the tutor's knowledge toward recognizing inconsistencies in the student's reasoning. Several systems exist which are able to recognize a subset of errors [e.g., BUGGY]. Such systems are called bug modelers.

BUGGY was a system that resulted from an extensive study of student errors in the area of simple arithmetic exercises [Brown and Burton, 1978]. BUGGY represented both correct and incorrect procedures of simple exercises and developed a methodology for reproducing the errors people make in a procedural skill. For instance, the system could produce 330 "bugs" for subtraction. The correct or incorrect procedures of the same skill were represented and applied for solving subtraction problems. Passage through the procedures resulted in the application of correct procedures; insertion of incorrect procedures in the network led to an incorrect solution.

BUGGY was not built to teach students; it was designed to illustrate the power a system would gain if it could automatically generate all possible incorrect student answers. But the implications of BUGGY are great. It provides computational evidence that student behavior is not a subset, nor a simplification, of expert behavior. Rather, errors

can be explicit and systematic deviations from correct procedures. As such, BUGGY provided evidence against the use of an overlay model of student knowledge.

5.5 Encoding Tutoring Knowledge

The tutoring component contains strategies, rules, and processes to govern the system's instructional interactions with a student. It holds knowledge about **how** to teach and determines, for instance, what instructional tool will be tried, when, and how often (e.g., provide a hint or ask a question). Some of the reasonableness or intrusiveness of a system is determined by this component.

Decisions made here are informed by reasoning done in the domain and student models. This component is not responsible for language processing, discourse management, or input-output behavior of the system. Communication activities, similar to those required of any interactive discourse system, rightly belong in the communication model, which determines the syntactic and rhetorical features of the interaction. The tutoring component itself handles only how to act based on the tutoring objectives of the system. It makes decisions about which goal to pursue, problems to present, questions to ask, hints to provide, and how to further interact with the student.

For example, GUIDON had a most sophisticated tutoring and dialogue model (see Section 5.2.2 above and [Clancey, 1982]). It included knowledge of discourse patterns and the means for varying strategies that the tutor used to guide the dialogue. Tutoring rules decided **when** a remark might be appropriate, **whether** to take the option, and **what** to say. One rule, for instance, used "entrapment" to force the student to make subsequent answers that will reveal some aspect of his understanding (or misunderstanding). Other rules explicitly defined how a subgoal would be discussed; e.g., suggest a subgoal, then discuss the goal, and finally, wrap up the discussion of the rule being considered. The

system was able to carry on a flexible dialogue by switching its discourse at any time to portions of an AND/OR tree, which represents the domain knowledge. It used state variables in addition to the rules and rule sets, to keep track of context. Thus GUIDON had dynamically updated the view of the student and the discourse history.

Tutors can represent and reason about alternative responses to the student. Choices being made are concerned with how much information to give and what motivational comments to make. For instance, the machine must decide whether or not to:

- talk *about* the student's response;
- provide hints, leading questions, or counter-suggestions;
- provide *motivational* feedback about the student's learning process;
- say *whether* an approach is appropriate, inappropriate, correct, or incorrect, and say *what* a correct response is.

Motivational feedback may include asking questions about the student's interest in continuing or providing encouragement, congratulations, challenges, and other statements with affective or prelocutionary content. Control is modulated by which tutoring strategy is in effect, which in turn places constraints on what feedback or follow-up to generate. The strategy may also specify that system action be predicated on whether the student's response was correct, or even that no response is to be given.

5.5.1 *Tools for Reasoning about Discourse*

This section describes several tools which enable a tutor to reason about and generate tutoring response. For example, Figure 42 displays one such tool which guides a system's ability to elaborate, give reasons and congratulates the student. The tool advises about strategies such as Socratic tutoring which would include being brief and not giving away the answers. For each approach primitive responses are available in TUPITS for the

AFFECTIVE ACTION	TUTORING STRATEGY					
	Informative	non-intrusive	directive	concise	coy	encouraging
Feedback	encourage					✓
	give-away			✓	X	✓
	congratulate	✓	X		X	X
	give-reason	✓			X	X
Follow up	challenge		X	X		X
	hints				✓	
	elaborate	✓			X	
	reaction				X	

Figure 42 Reasoning about Discourse Level

machine to perform the requested tactic. The tool defines a priority ordering for selecting each response tactic; thus to be Socratic, the machine must place highest priority on the tactic called "coy" and a secondary priority on the tactic "be informative." If there is a conflict between tactics, the one with the highest priority will win.

We realize that a more flexible and responsive discourse management technique than that shown in Figure 42 is critical to a tutoring or consultant system. By discourse management, we mean the system's ability to maintain interactive discourse with the user and to tailor its responses beyond the generalized discourse level suggested above. Ideally, the system should customize its response to the idiosyncracies of a particular user.

Ideally, the system should ensure that an intervention relates directly to an individual's personal history, learning style, and on-line experience with the system. It should dynamically reason about a user's actions, the curriculum, and the discourse history. In doing this, the tutor should make each user feel that his/her unique situation has been

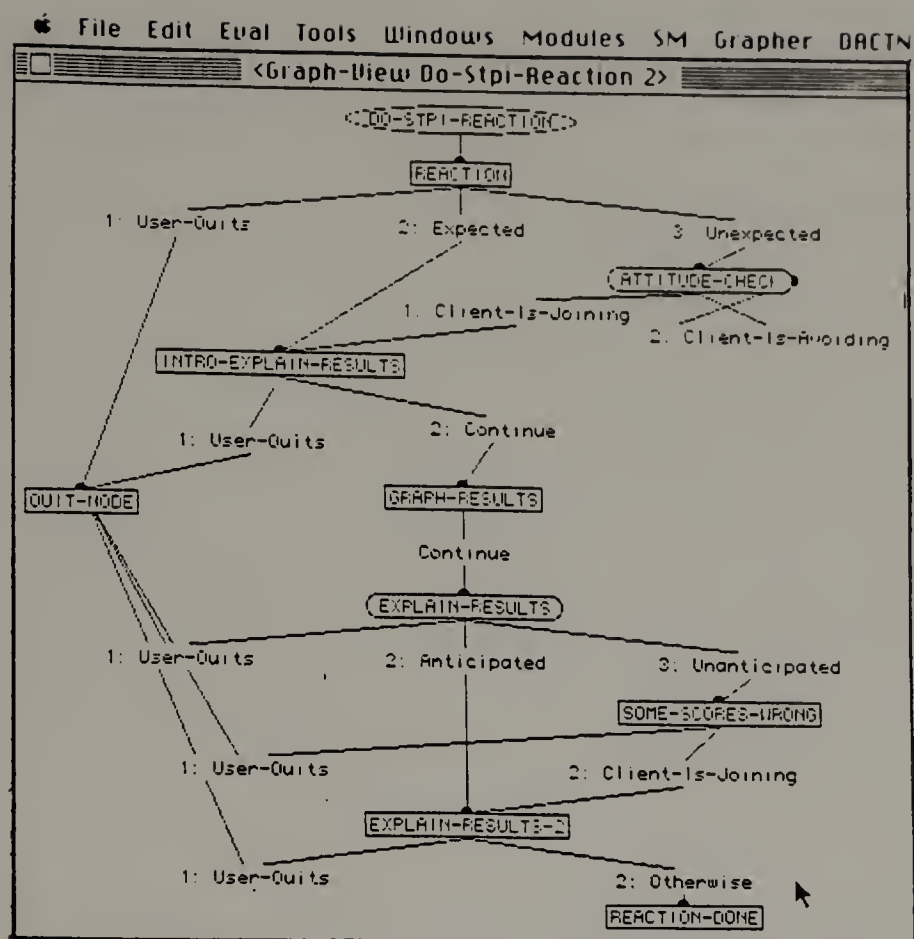


Figure 43 Discourse Action Transition Network: DACTN

responded to, appropriately and sensitively, i.e., it should simulate one-on-one human tutoring behavior.

The mechanism we use to do this is called a DACTN, *Discourse ACTION Transition Network*,² which represents primitives in human-machine dialog. Figure 43 is a DACTN for responding to a user about the inventory test of questions that he/she took in the system described in Section 5.2 below. This graphic is taken directly off the screen of that system. Sometimes the intervention steps designated by a DACTN are based on a taxonomy of frequently observed discourse sequences which provide default responses for the tutor [Woolf & Murray, 1987]. The discourse manager might also reason about local context when making discourse decisions, where local context is taken to be an aggregate of the user profile and response history.

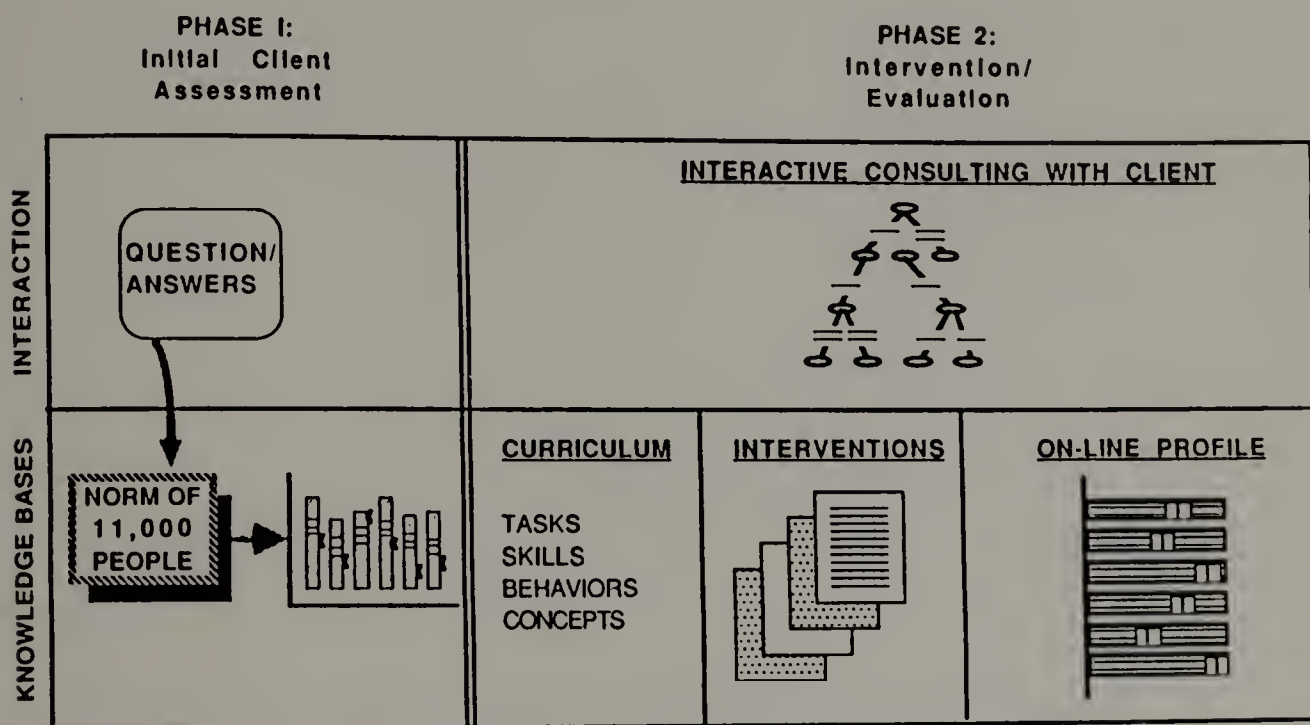


Figure 44 Two Phases of the Consultant

A DACTN represents the space of possible discourse situations: Arcs track the state of the conversation and are defined as predicate sets while nodes provide actions for the tutor. The discourse manager first accesses the situation indicated by the arcs, resolving

²Rhymes with ACT-IN.

any conflicts between multiply-satisfied predicate sets, and then initiates the action indicated by the node at the termination of the satisfied arc.

Arcs represent discourse situations defined by sets of predicates over the client profile and the state of the system. For instance, the value of the arc "CLIENT-IS-AVOIDING" (top-half of Figure 43) is determined by making inferences over the current state of the profile and recent client responses. Placing actions at the nodes rather than on the arcs, as was done in the ATN, allows nodes to represent abstract actions which can be expanded into concrete substeps when and if the node is reached during execution of the DACTN. For example, the node "EXPLAIN RESULTS" (middle of Figure 43) expands into yet another complete DACTN (recursively) to be executed if this node is evaluated in the course of the intervention.

5.5.2 *Example of a System That Manages Discourse*

The discourse action network presented in the previous section is part of a consultant tutor [Slovin & Woolf, 1988] which first tests a user's knowledge and skills in the area of time management and then enters into a discussion about techniques to promote awareness of a variety of time perspectives. Each user response causes the user model, or in this case the personality profile, to be updated, which in turn affects the interpretation of the current discourse situation. DACTNs allow discourse control decisions to be based on a dynamic interpretation of the situation. In this way the mechanism remains flexible, domain-independent, and able to be dynamically rebuilt—decision points and machine actions are modifiable through a graphical editor, as explained in this section. DACTNs have been implemented in two domains, one of which is described below.

TEV (Time, Energy, and Vision) presents interventions directed at improving an individual's personal time perspective [Slovin & Woolf, 1988; Blau et al., to appear]. The system moves through two phases which model the human-to-human consultation process: 1) Initial Client Assessment, and 2) Intervention/Evaluation (see Figure 44).

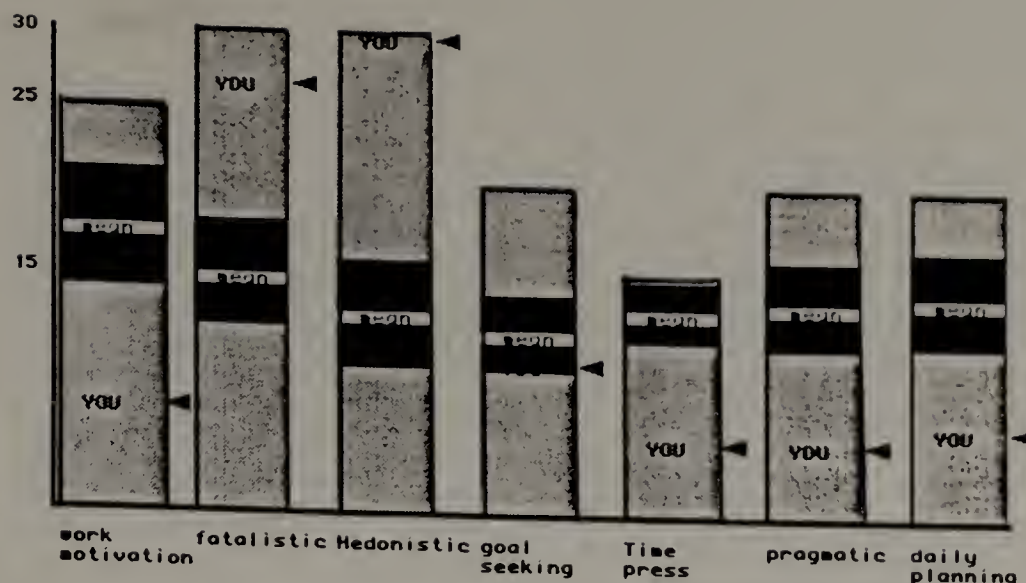


Figure 45 Evaluation of a Present-Oriented Time Perspective

Phase I: Initial Client Assessment. During the first phase, TEV gathers information about the person's attitudes, knowledge, and skills using an assessment instrument which, in a non-computerized version, has been tested with 11,000 individuals. The client is presented with a series of statements which focus on attitudes, knowledge, and skills related to time perspective (Stanford Time Perspectives Inventory [Gonzales & Zimbardo, 1985]). For each statement, the client is given a choice of five ratings ranging from very characteristic to very uncharacteristic and is asked to indicate "How characteristic is this of you?" Example statements include the following:

- Thinking about the future is pleasant to me.
- If things don't get done on time, I don't worry about them.
- I do things impulsively, making decisions on the spur of the moment.
- I try to live one day at a time.

Responses are evaluated according to the established interpretation of the instrument and then used in two ways. First they are shown to the client in a graph form based on

the client's rating among his age-group peers (see Figure 45). Then an explanation of the results is provided along with an opportunity for the client to react (see Figure 46). The evaluation and the client's response are used as the initial database to form an internal client profile. The profile is used to help determine the future direction of the consultation process and to generate expectations about client responses. It becomes the primary determining knowledge base for designing the consultation and is constantly updated to provide a moment-to-moment view of the client.

Tutor: Your scores indicate that you have a basically present oriented time perspective. Do you agree?

Person A: No, I think I also consider the future in my life.

Tutor: Well, let's look more closely at your scores, and see where you think they are wrong.

Person A: OK.

Tutor: Your score on the hedonism scale indicates that you may be extremely similar to people who are primarily interested in having a good time, taking risks, and who often act impulsively. Does that sound like you ?

Person A: Mostly, but it sounds a little extreme.

Tutor: OK, let's look now at the fatalism scale; it indicates that you may be very similar to people who live one day at a time in order to avoid planning for a future they believe is determined by fate. Is that what you do?

Person A: Yes, I guess so.

Tutor: Let's look at one more scale, Goal-seeking. Your score here shows that you may not be similar to people who plan for the future and get satisfaction from setting and achieving their goals. Do you agree?

Person A: It is true that I don't do much goal setting.

Tutor: It was mainly your scores on these scales that led me to view you as basically a present-oriented person. Since you disagree with the hedonism score slightly, why don't we say slightly present-oriented ?

Person A: Yes, that sounds more like me.³

Figure 46 Dialogue about the Evaluation of Figure 45

Phase II: Intervention/Evaluation. The second phase uses the profile to begin the consultation process. Here the goal is to create an ongoing dialog which leads the client to form and hold a more balanced time perspective. A second goal is to improve the client's repertoire of useful time management behaviors. These goals are achieved

²The dialogue is not generated through natural language processing, rather through menu input and canned text output. This might be a limitation of the system, however, prior work in the area of computer-based therapy [Servan-Schreiber, 1986] shows that clinically significant therapy can be achieved in spite of this limitation.

through dialogue consultation, specifically through the use of interventions, which are exercises or presentations designed to facilitate awareness of time perspectives or to provide learning or practice of skills. Example interventions are "Learning to Say No," "Life Goals," and "Time Wasters." Dialogue strategies are derived from a large repertoire of similar activities used in one-on-one and group counseling over the last 15 years by experts in clinical psychology. These strategies and interventions have proven effective in improving time management skills for a large number of people. TEV's orientation as a consultant tutor has led to a view of interventions as dialogs. Each intervention is seen as a distinct segment of an ongoing dialogue between TEV and the client which is extended by presentation of the next intervention. The consultation experience for each client is uniquely defined by the composite of high-level interventions and low-level discourse actions resulting from his/her responses to the system.

Representing Discourse as Alternative Plans. Discourse knowledge is represented as alternative plans. Knowledge of alternative curriculum activities is stored as predefined plans and alternative discourse moves are stored as different plan contingencies in these prestored plans (see Figure 47). The consultant has limited planning ability to manage these plans and plan contingencies. Pedagogical activities and discourse knowledge have been articulated by a clinical psychologist and are used to generate the lesson plan in response to client input during the lesson. The DACTN described in the previous section manifests one characteristic aspect of the computational model of didactics. It is referred to as the *plan of action* or lesson plan that enacts didactic operations [Wenger, 1988]. The *local context* in which a particular plan of action is triggered was described in the previous section. The plan of action is a unit of decision in the didactic process that manages knowledge about the curriculum, the available teaching resources, and the client's needs. In the case of a consultant, the curriculum consists of a prioritized overlay of skills, behaviors, and concepts which the client should be able to understand, demonstrate, and integrate into his/her life. (Example skills the tutor might discuss are how to keep a "to-do" list and how to state priorities for the next month.) The plan of action is controlled by

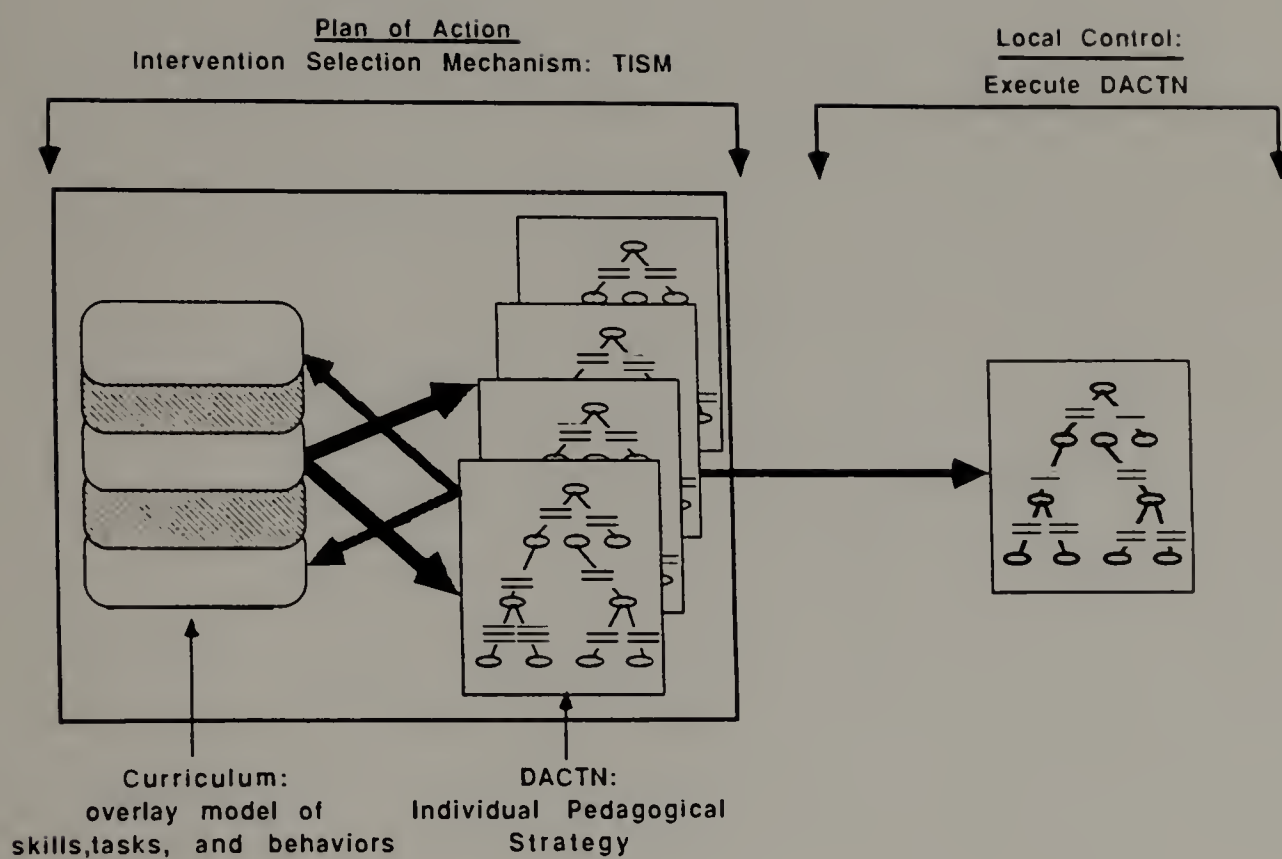


Figure 47 Levels of Control in TEV

the TISM (Tev's Intervention Selection Mechanism) which models an expert's ability to select appropriate interventions for a specific student. For each instructional objective, several pedagogical approaches (DACTNS) are indicated as being able to achieve the chosen objective (see Figure 47). Alternatively, for each pedagogical approach, or single DACTN, several curriculum objectives might be achieved. Our experts have developed a library of resources to teach alternative curriculum items, such as identifying time-wasters. During a one-on-one consulting session, TISM chooses among these resources based on an understanding of the needs and learning style of the client.

These resources are represented in the consultant in the form of interventions. The system reasons about the current context in generating the next step in its plan of action. It is constrained by the client assessment, a record of the client's state of knowledge, and system history. The TISM is responsible for establishing a globally coherent instructional objective and for ensuring that curricula items follow each other in a way that matches the client's needs.

Acquiring Discourse Knowledge. Knowledge acquisition for discourse knowledge involves encoding the reasons why an instructor makes decisions and how he/she decides when such interventions will take place. The TEV system facilitates knowledge acquisition by use of a graphical editor in which the instructor selects interventions and modifies the dialogue "on-line." The editor facilitates piecewise development and evaluation of the system, thus providing an opportunity for a wide circle of people, including psychologists, teachers, curriculum developers, and instructional scientists, to participate in the process of system implementation.

Because DACTNs provide a structured framework for representing dialogs, we have been able to develop a visual dialogue editor which allows an expert to create new interventions graphically. These interventions are automatically translated into LISP code. This allows the experts to work on knowledge acquisition without having to work with knowledge engineers. Thus we continue to elicit new interventions from our experts even as development and evaluation of TEV proceeds. By adding interventions to the library

and by linking them to the curriculum we expand TEV's repertoire without reworking the entire system. The dialogue editor allows an expert to directly manipulate a graph of the dialogue where each question, statement, or action is represented in an editable node, and each arc (also editable) represents a discourse situation that could result from the client's response. The expert adds a new question or statement and is led through a series of prompts designed to elicit the possible client responses. Each response has associated with it two pieces of information: a classification of the response, which is based on the current user profile, and the profile updates related to the choice of this response. Using a small set of classifications, i.e., EXPECTED, INDICATES-CONFUSION, AVOIDANCE, etc., the expert indicates his/her understanding of the meaning of this response. These classifications may depend on the current user profile, since this provides an indication of context. The profile modifications may include both updates based on the classification of the response and updates specific to this question and response. As each question is added the graph is updated so the expert always has a view of the current state of the intervention. The underlying DACTN is created dynamically so that at any point in the editing it can be executed against default profiles, allowing the expert to check the appropriateness of the machine's responses.

5.5.3 *Issues Related to Encoding Tutoring Knowledge*

Tools for encoding tutoring strategies are being developed currently in research laboratories, see for example [McCalla et al., 1988; MacMillan & Sleeman, 1987; Woolf et al., 1988]. Few tools are available today for use in application systems. Several issues remain to be addressed before such tools become generally available, some of which are discussed below.

Plan Recognition and Planning Systems. Recognizing the intentions of a student is very important to management of a tutoring interaction. A system should attribute to the student goals, planning abilities, and knowledge that a teacher might automatically notice, and it should match a student's observable activities with its own stored plans.

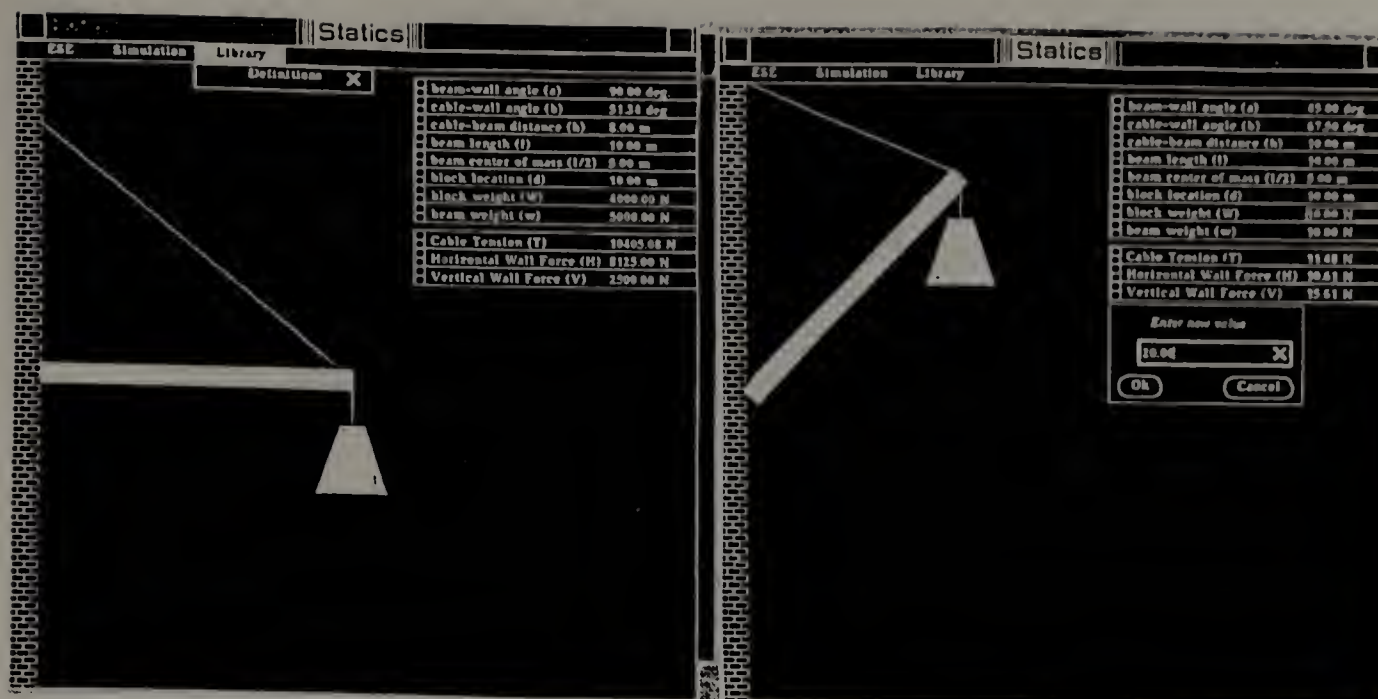


Figure 48 Screen for Tutoring about Force [Duckworth et al., 1987]

Where exceptions do occur, the system should generate plausible explanations of the student's intentions.

When a student's actions are observed, the machine might reason, "Given this response, what belief or goal could it be in the service of?" This inverse of the planning problem, called plan recognition, involves observing the low-level responses and inferring the high-level belief or intention. The system might store plans of student actions and presumed goals used in problem solving, e.g., Johnson and Soloway [1984].

Additionally, a tutoring system usually has its own goal to perform (e.g., test a student's knowledge of passive forces) and it should direct its actions to be in service of this goal (e.g., provide a graphic of a table with books on it and ask the student questions about the existent forces).

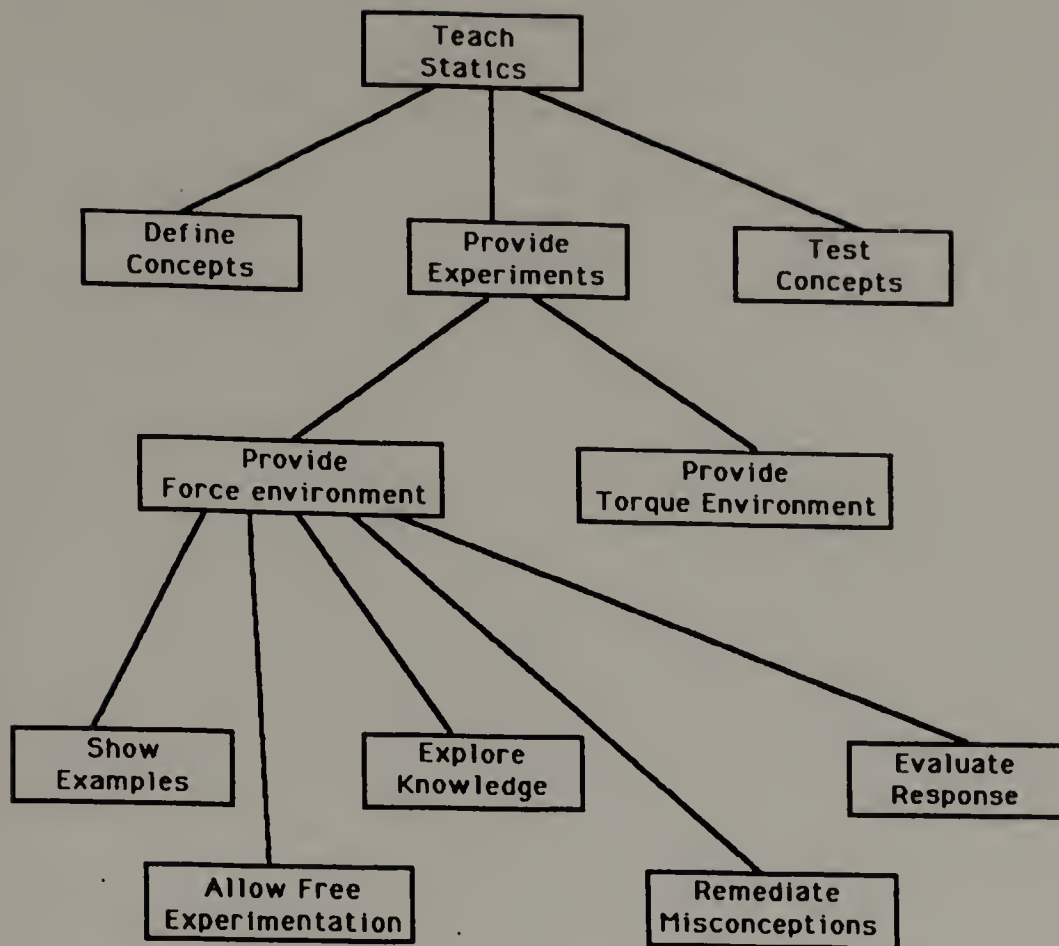


Figure 50 Goal Tree for Tutoring about Force

Figures 49 and 50 suggest a plan for teaching statics. They show the fundamentals of planning and plan recognition notation, Figure 49 defines a set of actions taken by the tutor such as "present-graphic" or "question-task," and Figure 50 defines a set of higher-level explanatory goals such as "provide force environment" or "explore knowledge." Clearly the actions of the tutor provide a way of achieving tutoring goals. Thus, to go from observed action "question-task" to its abducted goal, we assume that the action was a way to achieve the goal, such as "explore a student's knowledge." The action can be a way to do the goal, such as "question-task" or it could be a *step* in doing the goal, such as "ask for force vectors."

Thus, "ask question" is both an item to be explained within an action, explainable by the goal to "explore a student's knowledge," and it can be the explanation of the step "ask for force vectors." This is possible because explanation is an iterative process: once an explanation is found, a deeper explanation for that explanation can be found.

Using the plan scheme from Figure 49 we infer which choices have been made by the person teaching statics. The set of all plan selections gives rise to a goal tree such as shown in Figure 50. The nodes correspond to plans, broken down into steps, each of which must be executed.

For example:

```
(and(event provide_graphic)
      (event allow_experimentation)
      (event explore_student's knowledge)
      (event evaluate_response)
      (event remediate))
```

The planner's execution of the goal "provide experiments for force" explains the execution of the action "provide_graphics," "allow_experimentation," etc. A planner might start at the top of the goal tree (Figure 50) and work its way down, initiating actions to achieve a goal. A plan recognition system on the other hand starts from the bottom (from an

observable action) and works its way up the tree. So, if a student is involved in an experimental environment that allows him/her to manipulate elements of torque, the tutor might infer that the student is trying to learn statics. At each step, the system looks for a goal action such that the observed action is either a way of accomplishing the goal or a step of the plan for doing the goal. A plan recognition system works its way up the plan-schema tree until it gets to an action that is recognized as being the final (or highest) explanation provided.

Learning Styles. A tutoring system has at its foundation a model of human learning such that design of the tutor is generally in service to a particular view of learning. In this section, we sketch three possible philosophies of human learning and the impact each has on the definition of rules and strategies for a tutoring model. Human learning is complex and as yet incompletely understood. Clearly it will not be reduced to the simple learning stereotypes or behavior patterns suggested below. Nevertheless, stereotypes are helpful in defining a way to model learning and in providing a minimal basis for a learning philosophy.

Behaviorist Learning. Behaviorist learning, perhaps best epitomized by rote learning, suggests that people learn best by repetition and strong reinforcement [Pope, 1982]. According to this approach, bugs play an inconspicuous role; indeed, they are problems to be eliminated. Little time is spent understanding or repairing them.

Most pre-AI systems utilized a variation on this approach for their tutoring (e.g., PLATO [Bitzer et al., 1961]). BIP [Wescourt et al., 1977] assumed that an error indicated that the user needed more practice, and provided new problems to solve. It ignored student errors. However, it did provide a sophisticated variation to drill and practice: when the student's answers were wrong, it gave a more refined problem that exercised the presumed errorful skill. The system was not too effective, possibly because a student needs explanations and rich, supporting data to weave new knowledge into an existing and possibly errorful knowledge structure.

Constructivist Learning. The constructivist philosophy of learning suggests that humans learn by assimilation, accommodation, and equilibration [Piaget, 1971]. Assimilation is the process whereby an internal structure seeks activity by incorporating to itself some environmental data. Accommodation is the process whereby that internal structure is applied to a particular external situation. Since all external situations will contain some element of newness, accommodation leads to the differentiation of a previous structure and thus to the emergence of a new structure. Equilibration is the regulatory factor that unifies development and evolution. Intelligence makes explicit the regulations inherent in an organization. As a state, it is a continual balancing of active compensations. The implication of this philosophy is that a teacher should provide a rich "environment" and a wealth of extra information, including alternative views for the student to assimilate and accommodate. In order to learn, however, the student needs to constructively interact with the environment. He/she is fully responsible for the learning. He/she might assimilate information that is consistent with existing structures of knowledge and when the new knowledge is inconsistent with existing structures, accommodation might occur resulting in a restructuring of the knowledge.

According to this philosophy, the onus to learn is upon the student. The teacher might facilitate the process by probing weak areas or by clarifying confusing concepts. A Socratic dialogue, for instance, is an approach consistent with the constructivist philosophy [Plato, 1922; Collins, Warnock & Passafiume, 1975]. It uses techniques like overgeneralization of a student's error or applying his/her results to illogical consequences in order to reveal the error in reasoning.

Bugs play a central role in developing a tutoring approach based on this learning philosophy. Bugs identify the site of an error and make both the site and the nature of the knowledge around it explicit. They provide a window into the student's beliefs and a way to begin the tutoring process. (See Sussman [1973] and Austin [1976] for a computational assessment of the importance of bugs in learning.) A teacher, whether human or machine, sometimes cannot easily locate student bugs. Nevertheless, one can

try to make bugs explicit and present enough information to repair and rebuild knowledge around them.

WEST [Burton & Brown, 1982] attempted to identify errors in a student's game of arithmetic skills and to reveal them to the student (see Section 2.2.5). It described the issues and missing concepts and provided a concrete example of their correct use. The tutoring strategy was aimed at providing enough information for the student to construct his/her own knowledge.

Imitative Sponge Learning. The sponge approach to learning lies close to and perhaps overlaps the behaviorist learning approach. It is based on two assumptions: (a) the teacher has the requisite knowledge and (b) the student is both prepared and capable of "absorbing" that knowledge—like a sponge—in much the same form as the teacher has structured it. This view of learning implies that tutoring includes, and might be limited to, correctly displaying knowledge to a student.

A system that defines and essentially tutors from explicitly organized domain knowledge has as its premise that learning consists of imitating the teacher. However, psychological studies suggest that experts and novices structure knowledge of the same subject area differently (see, for instance, Chase and Simon [1981]; Larkin et al. [1980]; Soloway et al. [1983]). Variations in the way a novice structures knowledge as compared to the way the expert does it must be addressed by a tutor, human or machine. Few machines can do this.

5.6 Encoding Communication Knowledge

The communication model provides the interface between human and machine. Its primary activity is to converse with the student. This does not mean through natural language processing; it can mean one of several forms of communication such as menus or graphics, some of which are discussed in this section. Tutors exist that parse natural

language input and generate natural language output. However, such natural language processing does not guarantee that either a student's meaning or a machine's objective will be understood.

New media technologies provide a wealth of possibilities for this model. Technologies such as Compact-Disk Interactive (CD-I), Compact-Disk Read Only Memory (CD-ROM), and hypertext provide new ways to communicate knowledge to the user. Possibilities presented by this new media include combining a knowledge-based tutor with television, audio, images, or film to illustrate new information. New methods of communication have already emerged as machines offer operations such as browse, annotate, link, elaborate, explore and integrate information.

Knowledge-based tutors might some day act as gateways to encyclopedia-type clearinghouses of knowledge, made possible by advanced media technology. Tutors might act like intelligent agents which learn a user's preferences and prior knowledge. Intelligent technology such as described in this document, melded with new media technologies will enable people to access information easily in remote libraries, museums, data bases, or institutional archives. Given innovations in both knowledge-based and media technologies, the student will become an active learner, with the ability to manage, access, and manipulate vast quantities of information. Providing a wealth of communication materials requires computer-controlled videodisk and/or a CD-ROM (Compact Read Only Memory) as discussed in this section.

5.6.1 The Role of Communication Knowledge

Communication knowledge should enable a machine both to unambiguously receive human input and to unambiguously express the system's intent. As the planet comes on line [CasaBianca, 1988] and vast amounts of knowledge become available, the computer should be able to communicate more sensitively and to reason about the user as an

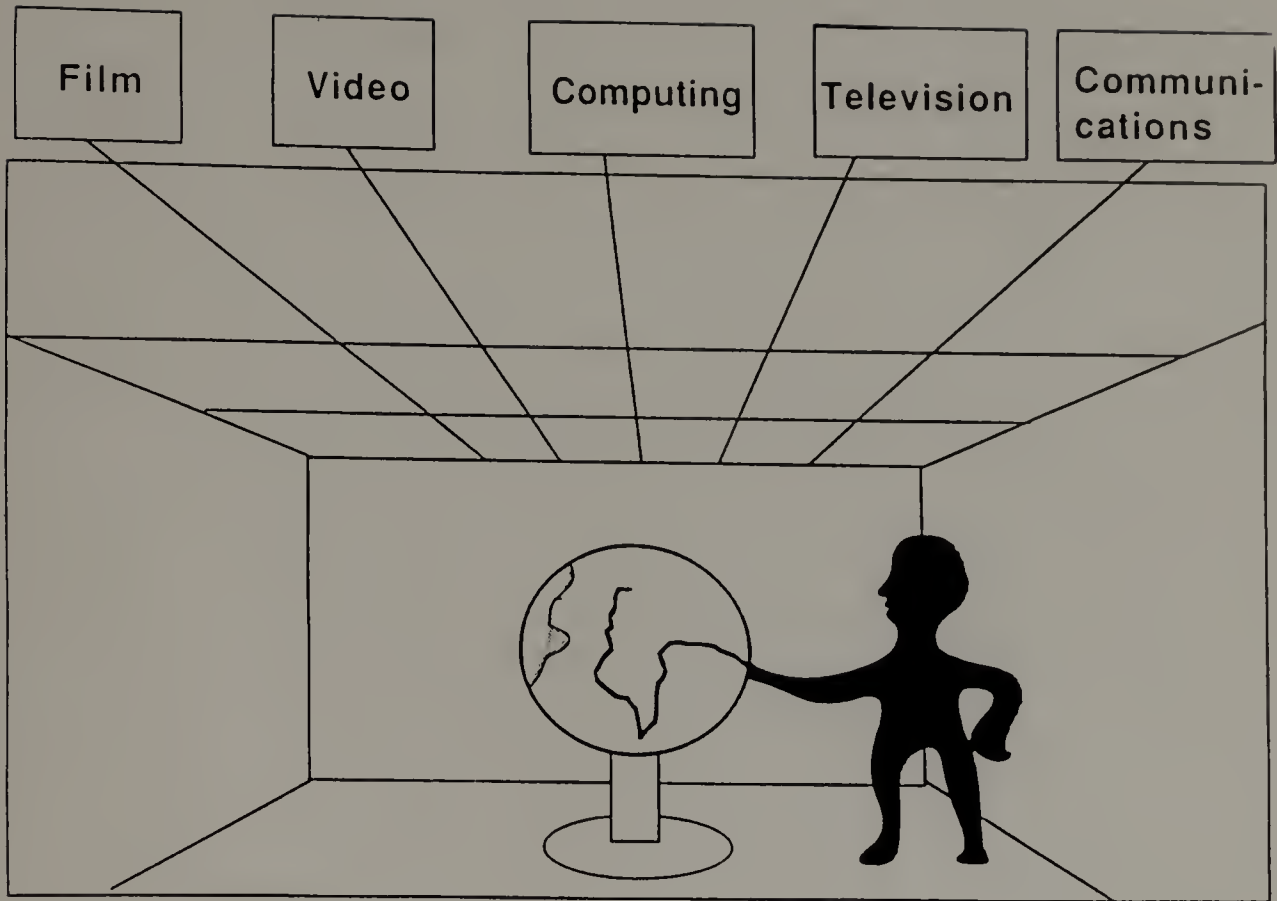


Figure 51 The Wired Society

intelligent tutor might reason about a student. Through personal computers vast amounts of information will become available, and the individual will become a node of a large electronic global communication network. However, the problem becomes one of indexing and searching large amounts of knowledge.

A student—even more than a typical computer user—cannot express what he/she does not know. It is counter-productive to ask a student what he/she would like to learn next. The student might not have a clear idea of the current topic or the prerequisite knowledge. Thus, a tutoring system—even more than a discourse system—must be equipped to recognize deficiencies in the student's interaction with the knowledge and his/her articulation of knowledge. Intelligent information resources, such as AI-based tutors, might provide the framework for a new global "wired society" in which the student gains access to knowledge and to a variety of media forms (see Figure 51).

The communication model is informed by the discourse model (which analyzes student input), the tutoring model (which reasons about an appropriate tutoring strategy), and the student model (which analyzes unexpressed student beliefs and intent). Historically, this component has been the last to benefit from sophisticated A.I. techniques; today it increasingly includes AI heuristics and techniques to enable the system to manage the dialogue intelligently.

For our purposes, communication amounts to "understanding the student's deeper meaning." The problem of understanding becomes acute if the student's knowledge is organized in a way different from that stored by the system. In such a case, it is difficult for a system understand the student, which, in part, requires the ability to ask appropriate questions and focus on relevant issues.

The communication model, for instance, is responsible for managing *mixed initiative* dialogue that allows either the student or the system to ask the next question. This kind of interaction is responsive on a local level; the student might be allowed to ask questions but eventually the system will take control and resume its topic (see Section 5.2 above

and Figure 35). Production rules are often used in this component to allow the student to pursue some subgoal while enabling the system to regain global control after a fixed number of interactions, or when the student relinquishes control (e.g., as in GUIDON).

5.6.2 Examples of Communication Media

This section describes recent innovations in the area of communication media for tutoring systems. It provides example systems that employ multi-media for training and teaching. Examples are drawn from projects in Compact Disk Interactive (CD-I), Compact-Disk Read Only Memory (CD-ROM), hypertext, and intelligent tutoring systems. The next section describes some issues in the development of a communication model.

Hardware and software innovations offer real-time digital, audio, and video education to schools, offices, and homes. The digitization of information is the driving force behind the merger of media and information. Compact disk and laser technology has enabled the digitization of sound, video, and 3-D graphics. Ultimately, CD and computer systems will interact by way of digital networks and fiber optic telephone systems that will be in place throughout the United States possibly by the mid-1990s.

Compact Disk Interactive (CD-I). Compact Disk Interactive (CD-I) technology allows a user to interactively direct the future visual sequences, sound, stills, and animation in a programmed disk. User interaction produces both the next sequence in a movie or the next visual screen and the next sound. Products that incorporate full CD-I have been released by companies such as AIM in 1989 [CD-ROM Conference, 1988].

The visual capability of CD-I is very high, including video resolution stills that are equal in quality to a TV studio picture. Red/green/blue computer graphics are available in various combinations of resolution and color depth. Audio stored on a CD-I disk is virtually indistinguishable from full CD digital audio.

An example of this technology is Dark Castle, an emerging CD-I product [Brewer, 1988].³ The product provides highly animated, full movie-like screens, believable sound, and an internal computer that allows the user to generate random sequences of audio, video, and combinations therein.

The product is based on a highly interactive game by the same name which now runs on a Macintosh. Dark Castle allows the user to direct the travels of an adventurer hero who fights off dragons, monsters, and other enemies based on the directives given to the system by the user. In the CD-I version, the user clicks a mouse to move various objects on the screen. Thus the hero can be manipulated to perform a variety of actions, such as enter a room or climb a ladder. A simple hypertext script (see below) ties together objects, such as a desk, a ladder, and a stone, as well as text on the screen, spoken narration, and visual illustrations. Objects or words are linked with corresponding audio or animated actions. The user causes the scene, the hero, and the situation to change based on his/her choice of the next scene or activity.

The disadvantage of the present crop of emerging CD-I products is that there is no intelligence in the computer's actions, no reasoning about the user's activities, or ability on the computer's part to problem solve. This will be handled when AI techniques are incorporated into CD-I (see Section 5.6.3).

Compact Disk Read Only Memory (CD-ROM). A CD-ROM is a compact disk used as a computer storage medium. It stores data and other mixed media on a disk about the size of a traditional 5-inch floppy (see Figure 52). The first CD-ROM product released for mass consumption was the Grolier Electronic Encyclopedia, which is a complete text of a 20 volume encyclopedia (with no pictures). Computer searches through the CD-ROM allow intersection of multiple words as well as use of AND, OR, and NOT operators. One problem with this technology is that other applications are not accessible to the user while he/she is in the middle of a CD-ROM product. Other CD-ROM products include

³Dark Castle is being produced at America Interactive Media (AIM), Los Angeles, CA.

A 12 cm laserdisc

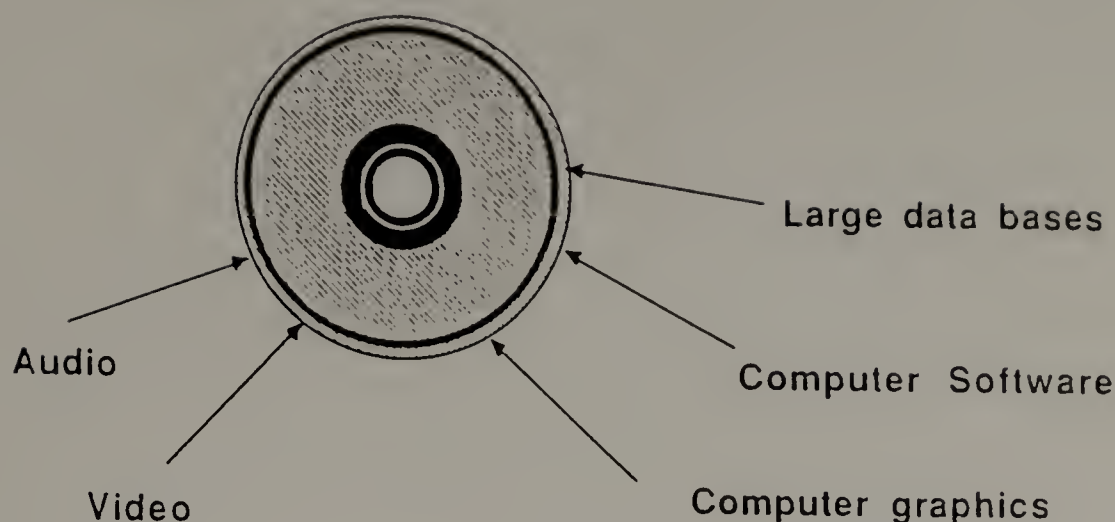


Figure 52 Compact Disk Technology

textual compendiums such as a dictionary, almanac, zip code listings, Bartlett's familiar quotations, and the World Almanac.

Hypertext.

"The written word has been sequential for the past 3,000 years. Suddenly we find that it doesn't have to be that way" (Ted Nelson, as quoted in CasaBianca [1988b]).

Figure 2.14 is from the Intermedia Hypertext System developed at Brown University [Yankelovich et al., 1985]. Such systems allow students to retrieve complete texts, stories, biographies, graphics, animation, sound, movies, motion video, microscope pictures, or audio as needed and to arrange them in terms of the students' priorities. Intermedia shows that words and pictures need not organize into hierarchies. Documents can have arbitrary beginnings and endings and can be explored rather than read sequentially. Computers and rational databases give us this power, by permitting zig-zag motion based on information being examined along hyperpaths. Hypertext provides for non-sequential reading and writing. Users can browse through networks of information, sample bite-sized

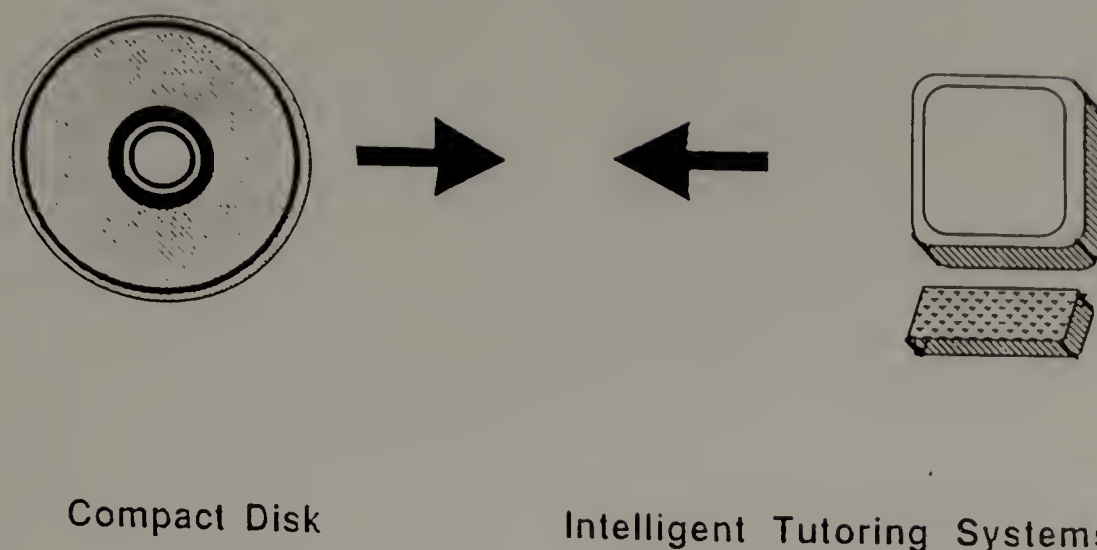


Figure 53 Merger of Compact Disk and Knowledge-based Tutors

pieces of information, and add to this living database by inserting their own links and information. Any document can be annotated in this way and will contain programmable links to other documents or files. The links can also lead to pictures, video sequences, or music.

Such systems provide virtually instant access to all kinds of data—historical papers, museum archives, reference books, business data-bases, and on-line educational resources. New documents can be created by chaining existing ones together. Multi-layered planes contain nodes of text. Data and graphics can be programmed to allow users to set their own course through islands of information. A good hypertext system encourages browsing and hunting, rather than reading from beginning to end. Several dozen hypertext systems can be purchased. Hypercard, by Apple Computer Company, is packaged with every Macintosh II and is a most popular first generation implementation of this technology.

5.6.3 *Issues Related to Encoding Communication Knowledge*

A variety of communication interfaces can be used as communication modules for knowledge-based tutors. Several issues, some of which are described below, should be considered before a particular one is chosen. For example, communication media such

as described above are limited in their effectiveness when used without human teachers. Video training systems have been shown to be rather ineffective without a teacher who provides suggestions about paths to take through the material and questions to ask. A primary restriction of current media systems is their lack of smooth integration; systems which utilize CD-I, hypertext, or artificial intelligence techniques seem to do so to the exclusion of other techniques. Integration of all these techniques will produce a truly compelling tutor. Currently, systems exist which demonstrate portions, but not the totality of such a merger.

For instance, Grolier's CD-I Dictionary takes advantage of hypertext and hypermedia capability within a CD-I environment. It is one of the world's first CD-I products and allows multiple access, browsing, and increasingly interactive journeys through the dictionary material. It does not have intelligence and thus cannot make decisions about what to present and how to best present it. It also has no graphics, which is a great loss given the power of current technology.

Another system, being built by the United Nations for delivery to over 150 nations, involves both CD-ROM and hypercard production to teach about pest tracking, quarantine, and controls. Through donated computer systems, some of the poorest, most illiterate farmers in the world will be provided with specific agricultural strategies and knowledge and have access to the latest and best available medical and agricultural advice.

The merger of artificial intelligence techniques and interactive video would allow a user to determine and guide his/her own learning. The following scenario will be possible:

A learner faces a video screen, holding a pointed device. A video and audio sequence begins. From now to the end of the sequence, the learner can interrupt the program and enter into a discussion, asking such questions as, "What is actually happening now?" "Why was x before y?" "Tell me about . . ." The user can also point to objects and say "What does x stand for?"

"Tell me more about x," "Show me some examples of x being used." [Parkes & Self, 1988]

A truly interactive and intelligent video system will respond to student questions by providing additional stills, movies, natural language explanations, generated graphic overlays, or audio; the system will decide which presentation is the most appropriate and how to respond to machine-perceived student misconceptions.

All the standard concerns of intelligent tutoring systems have to be re-interpreted within video-based intelligent tutoring systems [Parkes & Self, 1988]. For example, a video sequence can be the subject of a tutorial discussion only if the tutor has access to symbolic descriptions of the video content. Such descriptions are totally lacking in ordinary interactive video.

5.6.4 *Tutors with Natural Language Processing (NLP) Capabilities.*

For the most part, researchers in intelligent tutoring systems have avoided the use of natural language interfaces. Rather, they have relied on menu or multiple choice input and canned or cut-and-paste output to provide communication interfaces. The reasons for this are many. Few natural language tools have become available as a result of slow research progress in both language understanding and language generation. In addition, intelligent interfaces, which employ menus, hypertext, and multiple windows provide enough variety and depth of communication to approximate that offered by a natural language interface [see Clancey, 1986].

One outstanding exception to this is the work done on a natural language interface for SOPHIE [Brown & Bell, 1982; Brown et al., 1982] (see Section 5.2.2 above). Their system used a semantic grammar to parse input and a context model to perform language

comprehension. The tutor was able to handle nearly all reasonable sentences generated by users (see Figure 54). This was because, as in most tutors, interactions between student and tutor were kept within a restricted domain; thus, the communication model only had to discuss a small and well-defined set of terms and concepts on the general topic of electric circuits. SOPHIE's front end was unusual in that it was both efficient and effective. Its responses were versatile, being sufficient for a large range of questions and allowing for a wide range of possible student input. The semantic grammar in SOPHIE was designed to decompose sentences into their semantic, more than syntactic constituents. Thus, instead of parsing an input sentence into its syntactic grammatical components such as noun, verb, etc., it decomposed sentences into semantic categories associated with key words. For example, a sentence containing the word "voltage" was be parsed according to whether a voltage reading was proposed, asked for, measured, or provided as a hypothesis. Thus, the grammar represented the different ways elementary concepts could be expressed by the student. The grammar also took advantage of the fact that students typically interact with the system in two ways. Students ask for existent measurements or they propose hypothetical scenarios. In each case, the student's discourse with the tutor involved giving instructions to the system, represented internally as operations (e.g., take a measurement) with specific arguments (e.g., voltage reading at location V22). The parser allowed words to be skipped if they did not fit into the stored pattern.

SOPHIE handled anaphoric references and ellipses, as shown in the second input sentence in Figure 54: "What is *it* in V22?" The *it* in this input sentence is *not* resolved by the *parser*, which handles only the current input sentence. Rather, a context model of prior sentences of the dialogue holds a number of semantically compatible values that can be substituted for missing functions and arguments in current sentences. Thus, the use of "voltage" in the previous sentence and syntactic parallelism in the first two sentences are used to recognize the "it" in sentence two as referring to voltage in the first sentence. Though the authors make no claim to cognitive validity for the grammar, the system

The instrument has a fault and its present control settings are:

CC = 1, CR = HIGH, LOAD = 20, VC = 1, VR = HIGH

← Front panel settings

>>What is the output?

The output voltage is 11.7 volts.

uses model of dialog for reasoning pronouns

>>What is it in a working instrument?

In a working circuit the output voltage is 19.9 volts.

>>What is the voltage between R8 and Q1?

The voltage at N1 is 47.6 volts.

>>What is the base emitter voltage of the voltage limiting transistor?

= transistor

The base emitter voltage of Q5 is -.037 millivolts.

← hypothesis evaluation

>>Is it possible that Q6 is open?

Is the complete fault mode EB/EC/BC = OP/OK/OP what you intended?

>>Sure

constructs counter-example

When the settings were:

CC-1.0, CR = HIGH, LOAD = 20, VC = 1.0, VR = HIGH

That fault conflicts with the following measurement. You observed that the output voltage was 11.7 volts. If Q6 had fault mode EB/EC/BC = OP/OK/OP It would be 23.9 volts. In a working circuit it is 19.9 volts.

That fault does not explain any of the symptoms you observed but it is consistent with the values for the base emitter voltage of Q5 and the voltage at N1.

← but also determines what is "good" with the hypothesis.

Figure 54 NLP Interface in the SOPHIE System [Burton et al., 1982]

does seem to take advantage of the fact that humans bring to any text a large amount of domain specific knowledge.

Other early systems used a multitude of devices to handle communication. For example, WHY [Stephens & Collins, 1977] had a strong communication model and accepted natural language input (see Figure 38). However, because the student model was so weak, it could only mildly understand the student's intentions and couldn't use the communications model to focus on a specific topic within a particular subgoal.

In the Genetic Graph (GG) (Section 5.4, Figure 40) Goldstein proposed modifications to the original WUMPUS coach which represented relations between skills of a game and thus produced more sophisticated utterances about skills which a student was learning to use [Goldstein, 1982] (see Figure 40). The GG encoded generalizations, analogies, deviations, and simplifications of each skill in a modified semantic network and guided the coach through these skills and relations between skills. First the GG suggested which skills to discuss, namely those on the frontier of the player's knowledge. Then it supplied advice about expressing that skill in a natural language utterance, perhaps as an analogous instance of a previously learned skill (e.g., "Oh, Mary, you remember we had the same situation when you were in Cave 15. . . ") or as a generalization of an earlier skill (e.g., "Mary, since you have two warnings about Cave 15, you can infer that it is more dangerous to enter Cave 15 than to enter another one with only a single warning. . . "). The GG provided knowledge about how to discuss each skill and provided insight about which skills were premature to discuss given the player's knowledge as represented in the GG.

5.7 Summary

This chapter provided a view of the implementation issues involved in building a knowledge-based tutor. It suggested tools and methodologies available (or nearly available) for authors of such systems. The tools were divided according to the four sources of knowledge in a tutor: domain, student, tutoring, and communication models. The tools included semantic networks, planning and plan recognition systems, multi-media, and natural language processing systems.

CHAPTER 6¹

SOFTWARE AND HARDWARE CONSIDERATIONS

6.1 The Nature of Artificial Intelligence Programming

This chapter describes hardware and software considerations to be made before embarking on your project. We assumed throughout the document that the reader had a low-level understanding of the field of Artificial Intelligence. However, at this time, it is appropriate to explicitly define the field in preparation for clarifying these hardware and software considerations.

Artificial Intelligence is the study of intelligent behavior and its replication in a computer.

The field of Artificial Intelligence (AI) attempts to develop intelligent machine behavior. AI programmers frequently find themselves in the position of creating behavior that has never been seen before. Unique demands are placed on them as they represent large amounts of knowledge and generate clever ways to search through that knowledge. Frequently, AI programmers don't know exactly how to generate more intelligent behavior

¹Much of this section is based on an excellent, albeit now outdated, monogram called *Artificial Intelligence Computers and Software: Technology and Market Trends* written by David D. McDonald and John Clippinger, published by Brattle Research Corp., 215 First Street, Cambridge, MA 02142, 1984.

and need to experiment with designs that evolve only as their ideas and experience evolve. Thus, new tools, languages, and programming skills are required.

As compared with conventional software systems, AI systems present many new problems. Experimentation with programming design and implementation is possible only with a supportive programming environment. In conventional software engineering systems goals and specifications are written out in great detail before coding begins. Conventional projects involve explicit tasks, such as "update personnel records" or "analyze data according to these functions." Debugging such a system consists of refining the code to ensure that it achieves the stated goal. However, in AI programs, the goal is to generate more intelligent behavior. For tutoring systems, this might mean generate more sensitive or more responsive one-on-one tutoring; it might also mean generate machine inferences about student actions, skill level, and possible misunderstandings.

As described in Chapter 5, building a tutoring system requires representing knowledge and then building functions to traverse that knowledge. Knowledge representation refers to how knowledge is stored and how it models the domain, human thinking, learning processes, and tutoring strategies, as shown in Figure 31. Knowledge bases might store concepts, activities, and relations between topics. Or they might store a variety of lessons, topics, presentations, and response selections. Control structures might be motivated by specific instructional and diagnostic goals, e.g., one control structure might produce a predominantly Socratic interaction or an incrementally generalized new problem for a student to solve. Control structures might be specific to a particular level of representation and uniquely define the reasoning to be used for that knowledge base.

AI programming, then, refers to an approach to representing knowledge and control structures to traverse that knowledge. It also refers to an approach for achieving code production and a set of tools to expedite that process. It does not refer to a particular subject matter, programming language, or type of hardware. Good tools and languages developed for AI should place a minimum of constraints on a programmer's imagination and should provide a supportive environment for handling large amounts of knowledge.

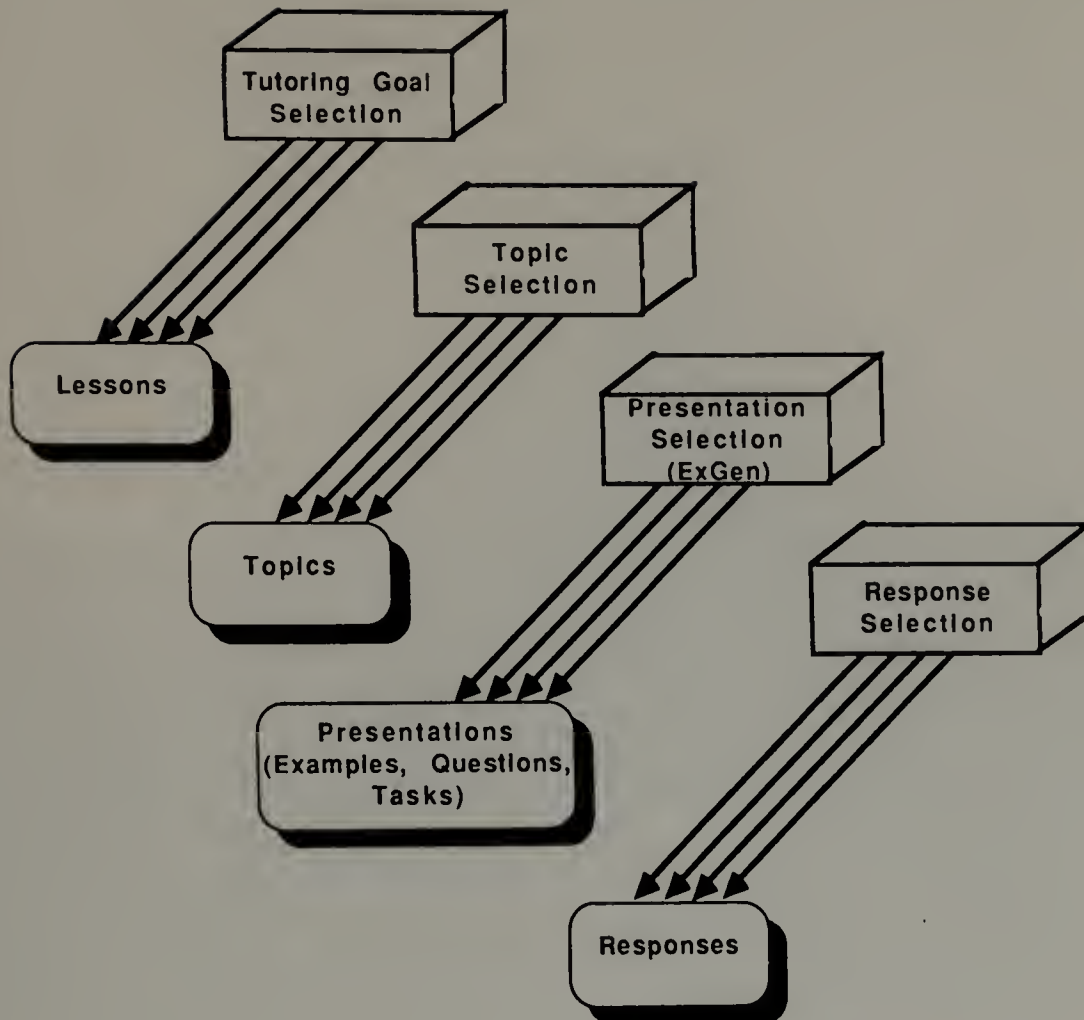


Figure 55 Representation and Control in a Tutoring System

Such tools and languages should allow programming operations to be tested more rapidly and more easily than has been possible with other languages, such as those with fixed data structures (e.g., Pascal), fixed control structures (e.g., Fortran), or those which are compiled (e.g., Fortran, Cobol, Forth). Trying to capture human behavior within a machine involves solving intensively demanding problems, both intellectually and computationally. These demands result in strong language requirements to hide mechanical details (e.g., as does A.P.L. for mathematical tails). The required mechanisms should be assumed automatically by the system, thereby allowing researchers to devote intellectual time to the problem itself. Such mechanisms are described in this chapter, along with sample software and hardware innovations that provide some solutions.

AI programming is unique and has given rise to numerous tools which have had an enormous impact on conventional computing. Software and hardware that have been developed and nurtured in AI laboratories have become part of conventional computing environments, and now can be found in the marketplace. Example products include on-line traces, debuggers, bit-mapped displays, mice, windows, icons, and object-oriented programming.

6.2 Choosing an AI Language

A variety of languages can be used for AI programming. However, few are specially equipped with the language facilities required for AI programming. These facilities are described in this section along with details from several programming languages. AI "packages," including expert systems shells, are described in the next section. Neither this section nor the next will provide an exhaustive list of languages; rather, each provides enough descriptive material about language features to advise the reader about how to select an AI language or package.

LISP. LISP is the traditional language of AI. Nearly 95% of the best-known AI programs written in the United States were built in LISP [Charniak & McDermott, 1985]. Nearly all the knowledge-based tutors described in Chapters 2 and 5 were built in LISP. It is the oldest programming language next to FORTRAN in active use and was developed in 1957.² Its use today is growing thanks to the advent of a standard framework (CommonLISP) and general purpose hardware that provide excellent LISP environments (e.g., Apple Macintosh II, Sun, IBM-RT, and Hewlett-Packard Bobcat Series). Four key properties of LISP set it apart from other programming languages:

1. few syntactic conventions;
2. programs treated as data;
3. details handled automatically; and
4. facility for working with character data.

In LISP, all data is defined as either an atom (a number or a symbol) or a list (a collection of data surrounded by parentheses). All programs can be treated as functions and yet manipulated as data. Apart from these and a few other conventions, LISP has few strict syntactic rules.

Many details are handled automatically. For example, a programmer can construct new data structures, such as arrays, and, should the structure outlive its usefulness, it will be taken apart and the memory used in another structure. This is called "dynamic storage allocation." Construction and destruction of a data structure is done without programmer intervention. In addition, LISP programmers may change the structure of the language to suit themselves. If they don't like a construct, they can invent a new one. Thus, a new command or data structure might be defined through a macro to be used

²LISP, *List Structure Programming Languages*, was developed by Prof. John McCarthy, in 1957, as a procedural formalism to express the newly evolving ideas of Artificial Intelligence.

only for a given program. This new construct might simply change some feature of an existing construct, such as to evaluate arguments during run time. For languages such as Pascal, Logo, and Fortran, such freedom does not exist. Another example of construction freedom is in the use of arguments in functions. Most languages expect that functions will take arguments and will return exactly one value. However, in LISP a programmer can specify zero or more arguments, can indicate when they will be evaluated (e.g., at run time), can use any structure (data, functions, or programs) as arguments, and can define side effects that result in place of a returned result.

LISP is also ideal for AI programming because it provides automatic facilities for associating information with alphanumeric characters. Instead of being oriented toward manipulation of numbers, as are APL, BASIC, C, Forth, and FORTRAN, LISP easily associates character strings, such as "elephant" with other symbols, such as "large animal." LISP is primarily an *interpreted language*, rather than a compiled language. (In this respect it is like APL.) Rather than translating an entire body of code into machine language and then running it, LISP looks at a written line of code and executes it line-by-line. It also comes complete with an environment which performs debugging and editing. To use LISP you must use the entire environment. (LISP is similar to APL and Smalltalk in this regard.) One detail that is handled automatically is called dynamic memory management (see below).

A particularly good example of LISP code which is easy to read and also hides implementation details is provided in Brattle [1984]. This code is excerpted from an animation program and deliberately selected for its clarity.

```
(define character Cinderella
  (process initial description
    (physical (and beautiful shabby))
    (personality (and good friendly hard-working shy))
    (role-in-story most important)))
```


This code hides procedural and data structure details needed to implement the larger function "define character," a function which was defined earlier in terms of its three attributes, "physical," "personality," and "role-in-story," and connected with an appropriate piece of animation. By having set aside features of processes appropriate for each of its attributes, the associated features can be encapsulated in a construct, "define character," which is placed at the same level, semantically (evaluated at nearly the same time), as the original attributes. The AI community believes that such code is easy to augment, modify, and debug because it is easy to read and understand (see section below on "Learning LISP").

A second mechanical detail that is hidden in LISP code is memory management, achieved through dynamic storage allocation. Data structures are freely created in LISP and can be ignored when no longer used. The underlying mechanism of the LISP environment handles all the decisions of allocating enough storage for new data structures, creating pointers to them, and later reclaiming their space when the program no longer refers to them. This process is handled by invisible (and ideally rapid) "garbage collection" mechanisms.

LISP programmers can develop programming aids that help them conceptualize and design large systems. Programming tools (such as the editors and debuggers described in Section 6.5 below) make LISP highly productive and capable of prototyping very large systems rapidly. According to a study completed by the MIT Air Transportation Laboratory on the cost of developing software for aerospace applications, LISP had a productivity score, based on useful code produced per unit time, of nearly twice that of its nearest competitor, PL/1, Multex, and nearly thirty times that of Cobol [Brattle, 1984].

LISP can communicate to "foreign" languages such as FORTRAN, C, or Pascal. Extensive provisions are made for calling functions and passing arguments on general purpose machines such as Macintosh II, Sun, IBM-RT, and Hewlett-Packard Bobcats. Being able to call foreign code is a great convenience, allowing a programmer to draw

on established libraries of special purpose routines. These facilities also enable a user to develop interfaces to complex peripherals or to special shared-system services such as printers or massive external databases.

Foreign language calls are also important because LISP is not committed to machine-level details (as are languages such as C or FORTRAN) and as a result, its code for simulations or graphics is less efficient, i.e., slower. Where LISP can call and pass arguments to foreign code modules, a common practice is to use LISP to handle high-level decision making, such as described in Chapters 3 and 5, and to call C or FORTRAN routines for rapid production of graphics and control of I/O devices.

LISP and related tools reduce the programmer's need to remember an enormous amount of details by taking on the predictable and mundane activities themselves. Three types of software development tools available in typical LISP systems are described in Section 6.5 below. The tools handle code and environment management, debugging, and analysis.

These tools, derived from LISP software originally developed on special purpose LISP machine hardware, are available on general purpose hardware. Software is the key source of productivity gains, and new generations of programming power tools have continued to evolve. Popular vendors for Common LISP are Lucid at the high (expensive) end, GoldHill at the mid-range, including Allegro for the Macintosh II, and Texas Instruments at the low end for an IBM-XT.

Learning LISP. LISP is not a hard programming language to learn. For instance, Scheme, a streamlined version of LISP, is taught as a first programming language at M.I.T., UC, Berkeley, and other colleges. One reason why LISP is taught as a first language, especially on special purpose hardware such as the TI explorer or Symbolics LISP machine, is because of the uniformity of conventions. All of the different subsystems—editor, mailer, font editor, debugger—are controlled in analogous ways, meaning that once you have learned it for one subsystem, you have (almost) learned it for all. Each

subsystem presents a visual appearance that is similar to every other; use of highlighting, techniques, and organization of mouse menus and click conventions are the same because of the uniformity of environment conventions used. However, general purpose machines such as Apple Macintosh II, Sun, IBM-RT, or Hewlett-Packard Bobcats do not traditionally have the environmental uniformity described above. LISP itself is also easy to learn because of its exceptionally clean semantics (as discussed above). It is particularly good for teaching fundamental concepts such as variable binding and scope, functions, control and data structures, etc.

However, as a result of the large amount of available facilities and specialized code, becoming a good LISP programmer takes time. The "learning curve" for producing genuine results in LISP is steep (requires much time and attention). Both broad and deep knowledge of numerous LISP commands, the richness of the LISP development environment, the numerous control options, and the multiple representation paradigms must be mastered before significant results can be produced. Even the best programmers typically require several months of intense work to get high enough on the curve to employ all of a good LISP's capability.

LISP as a Social Phenomenon. The unique ability of LISP to undergo language extension and development makes it something of a "social" phenomenon. Since it can be transparently extended upon itself by the creation of new data and control structures, some LISP users become writers of "systems" code for the rest of the community to use. This "user-and-developer" individual is very important to the community. If he/she is sufficiently unhappy with the performance or style of any part of the LISP system, he/she is likely to design and build a better facility. When done, the facility will be usable by the rest of the community and can be ported to other communities, given a common hardware and software base. The use of CommonLISP has lessened this phenomenon somewhat.

CommonLisp. CommonLISP is arguably the standard LISP today. It can be moved between hardware systems and will run consistently on all of them. It is not a dialect of

LISP, per se, but rather a specification of certain core functions and data types of LISP that should be included in any LISP running on any machine, along with a statement of how these functions should behave. Thus, there is no definitive implementation of CommonLISP. Rather, the intent of the standardization is that CommonLISP provided by any hardware vendor is restricted to facilities that are a part of the standard and will run on that hardware, as well as any other hardware produced by any other vendor. Vendors have agreed to adhere to the CommonLISP definition of these core facilities, adding to them those facilities that may be unique to a particular site or hardware. The addition of facilities is especially true for window-based display packages. The core of standard functions is quite large: several hundred functions. Nearly all the knowledge-based systems built since 1987 are built in CommonLISP, C, or Prolog. CommonLISP encompasses all of the LISP facilities that have been tested for several years. The intent is that as other facilities become less experimental, i.e., a uniform sense of how they should be used emerges, they will be added to this commonly agreed upon standard. The standard is not intended to remain fixed for all time; changes and extensions are expected as the thinking of the LISP community continues to evolve.

In sum, LISP is quite popular for development of knowledge-based tutoring systems. It provides large functionality, flexibility, expressibility, and is designed for symbolic reasoning. On the other hand, its very large repertoire of useful functions and auxiliary features makes it time-consuming to learn and a difficult language in which to gain proficiency.

Prolog. Prolog is a high-level language specifically for defining relationships and their implications. In this sense, it is like an expert systems package: It allows a statement of facts that will be the basis of its inferencing, but does not allow the ready encoding of information about how reasoning should be carried out. LISP, on the other hand, is a systems development language and is intended to allow the rapid development and redesign of task-oriented languages such as those at the level of Prolog. Thus Prolog should be compared to other very high-level languages or expert systems shells that

hide control structures, such as OPS5 (see Section 6.3 below). It is a good choice for programming a task if the task fits into its framework as explained below.

Prolog is the preferred language for development of AI systems in Japan and England. Most Prolog systems remain committed to a single control structure, a few data structures, and weak expressions of algorithms. The language performs a blind search via back-tracking through all conceivable candidate solutions until it finds one that works. Prolog is best suited for first-order relationships, where the programmer selects a set of names for the relationships and individuals and then uses depth-first search to satisfy a given request. Once all the facts of the problem are stated, Prolog's implicit control structure can be called to consider successive possible values for the variable and to test the data base for consistency against original facts, deriving new, intermediate facts where necessary. Prolog is derived from the technique of resolution theorem proving, which in the past has been known for its exceptional slowness. Yet, Prolog has returned to favor partly as a result of the increased speed of today's computers. It typically supports only depth-first search with strictly chronological backup, and an exceptionally direct and uncomplicated control structure which can be implemented very efficiently.

Where it is appropriate, Prolog will find its solution, will hide details and will let the programmer focus on the information he/she is trying to encode. If Prolog is used for something for which it was not designed, it can be exceptionally obscure and difficult to read. For example, a Prolog system without arrays, records, or data clustering conventions remains quite deficient by comparison with LISP systems. By restricting the complexity and variety of choices in control and data structures, Prolog is able to provide extra optimization of the restricted set and thus make use of the speed of today's computers. On the other hand, inappropriate use of Prolog results when one tries to enlarge the depth-first-with-backtracking control structure. Prolog typically provides no provisions for extending the notation of the language.

Popular vendors for Prolog are Quintus at the high end, Arity Prolog at the mid-range, and Barland's Turbo Prolog at the low end.

Smalltalk. Smalltalk is based on a metaphor of objects and methods to activate those objects. It passes "messages" between "objects." An object can be an independent process that acts whenever it is sent a message. Each process is defined in terms of a set of procedure definitions that specify the messages it can receive and, thereby, determine the actions it can carry out. A Smalltalk program consists of a set of class definitions organized in an inheritance network, similar to the semantic network described in Section 5.2. It has a primitive ability to create procedures by "instantiating" classes, or generating examples of an object, and then activating these examples by sending one of them a message. The class definitions are a set of messages and associated procedures.

Smalltalk was a very exciting language when first introduced by Alan Kay at Xerox Parc around 1974. Smalltalk originally introduced a set array of "power tools" that are considered standard today. As implemented earlier on the Alto, Smalltalk sported the first widely used, large bitmapped screen, the first mouse, and was connected via the first local area Ethernet. Work done on the original LISP machines of Symbolics and Texas Instruments intended to pick up on the software and hardware innovations on the Alto and build a vehicle for LISP rather than Smalltalk. These Smalltalk ideas have ultimately found their way onto smaller general purpose machines and have become more readily available on machines such as Tektronics, Macintosh, or IBM-AT, where its compact encoding scheme can be very effective. Xerox's Smalltalk 80 is currently very popular for people who want versions that run on IBM- PCs and on Macintoshes.

However, there are some disadvantages to using Smalltalk. The worst is that it is an all or nothing language/environment – it requires the entire machine for itself. This characteristic may change as the language itself evolves. As currently implemented, it has total control of the display and the environment. Programmers cannot mix-in code written in other high-level languages or in assembly languages. This means that the programmer cannot call on code written in other languages, such as C or FORTRAN.

For many programmers, the advantages of a uniform object-orientated language outweigh the disadvantages. In addition to its unique way of organizing computations, it

allows programs to be built, used, and modified through a visual interface. The programmer can refer to objects by selecting them from a visual display with a mouse rather than having to write an ad-hoc access program. He/she can keep both this editor and interaction stream present at the same time on the screen because I/O was organized into separate windows. This relieves the programmer of memory load and time lost due to switching contexts; the display screen becomes a kind of spatially organized memory with objects and the dynamic state of a program displayed as an iconic picture or a strategically placed string of text.

Variant AI Languages. In addition to CommonLISP as the most available language for expert system development, several object-oriented languages, are now available, such as C++ [Harmon, 1987]. Object-oriented languages allow the user to develop good interfaces (from high-level primitives for constructing interface elements such as windows and pop down menus), generate rapid prototyping (through facilities which allow a system to be rapidly implemented and tested), and to develop reusable code (modules of code which can be built and reused in different projects).

Object-oriented C is offered by Productivity Products International's Objective-C and Bell Lab's C++. Kyoto LISP, which is written in C, is an interesting language for expert systems development. The object-oriented standard for CommonLISP has resulted in more attention paid to object-oriented programming [Harmon, 1987]. The trend to C and Unix based machines is indicative of the shift of expert systems development to more conventional general purpose hardware and software.

6.3 Knowledge Engineering Tools

Today one does not try to quickly build a new AI system in unaugmented Lisp because although it hides uninteresting implementation details, it still requires a great deal of low-level programming. Instead, one might use one of the several "packages" that

have been developed specifically for expert systems: OPS5, KEE, ART, POPLOG. Many of these systems were written in LISP or C or Prolog and many can run "on top of" LISP, meaning that while using them, one can freely draw on the tools of the LISP development environment (see Section 6.5 below). Thus, LISP has become a systems programming language in which to write AI languages such as OPS5, but is not necessarily a good first language in which to quickly develop a powerful and fast solution to an AI problem. On the other hand, such packages are not used extensively by research labs; indeed some of the most innovative users of expert systems still prefer to use an unaugmented LISP [Feigenbaum, 1989].

Knowledge engineering (KE) packages are representation systems that help capture and represent an expert's knowledge. In as much as their methods and approaches are appropriate to the task, they save time and effort. The disadvantage of buying a particular AI package is that it brings with it a set of unexpected and possibly inappropriate methodological assumptions that may be quite different from what a specific application requires. This may be especially true for the inexpensive (about \$500) AI development software. If a specific application is similar to a program already developed, then there may be little problem. For example, you might obtain the Empty MYCIN (EMYCIN) knowledge base for a medical diagnosis problem. However, if your application is in a domain which has not yet been used in an AI system, then the problem is greater, and one might consider building the system from scratch; a competent AI programmer could reproduce these packages in about three to six months' time.

Shells. Expert systems "shells" provide a quick way to enter knowledge into a knowledge base and make inferences about it. Such systems typically provide a single fixed-knowledge representation, e.g., a framework of rules, and a few control structures, e.g., forward and backward chaining through those rules. Shells have become available at a variety of prices and for a variety of machines. Shells are being purchased by operational and development departments of corporations and research labs for general purpose machines and for direct application. For knowledge-based systems, if the project

programmers can acquire LISP know-how, the project will be more productive using a primitive symbolic language. However, if such expertise is not available, then one might begin with a package.

The commercial expert systems market has changed rapidly and is now so active that there is no question about its future [Harmon, 1987]. There are so many expert systems-building packages that choosing the appropriate package is frustrating. Several questions need to be answered by buyers of such systems, the first being about programming language. Packages built in a language that is designed for symbolic processing (such as LISP or Prolog) are more flexible, often providing better editing environments. Users who are doing both research and development and who are building large complex stand-alone systems are advised to select symbolic language knowledge engineering packages.

On the other hand, packages built in conventional languages, e.g., FORTRAN, Pascal, or C, can be run on conventional hardware and can pass data to and from conventional programs and databases. Users from traditional management and information departments building direct applications that will interact with mainframe programs and databases need to either select these conventionally based packages, which may be less efficient, or move their entire operation into the more symbolic languages [Harmon, 1987].

Knowledge engineering systems can be purchased for various hardware systems, including specialized LISP machines, mainframes, Unix workstations, and personal computers. Several products continue to sell well in this volatile market [Harmon, 1987]. Neuron Data puts out an expert package that runs on all major microcomputers, including Sun, Macintosh, Hewlett-Packard, and IBM-AT. S1 by Tecknowledge and KEE by Intellicorp are fighting for control of the high-end LISP machine market. Aion is the most visible product for mainframes, with IBM's Expert System Environment/VM a possible contender. Aion is the only tool with both IBM-PC and mainframe versions that are completely compatible, making cost-effective development a real selling point. Systems for the AT market abound: Personal Consultant Plus, by Texas Instruments, KES, Guru, M1, Object N Expert, and Acorn, etc. Other knowledge engineering systems have been

designed for the low end of the PC market, such as Personal Consultant Easy, Exsys, Insight 2++, 1st-Class, and VP Expert. The high-end market has moved from LISP machines to Unix workstations and/or the 386 machine, and that has caused a redistribution of vendor attention. OPS5 is a stable and robust shell for higher-end machines.

Domain-specific tools have emerged tailored for specific applications, for example, developing process control systems. This type of tool should increase as more effective methods are developed for designing and building them.

6.4 Choosing Hardware

Knowledge-based tutors can most profitably be built on general purpose computers with symbolic programming capabilities. Development and delivery of tutors today does not require acquisition of dedicated LISP machines. Conventional hardware, such as DecVax and Mini-Vax machines, Unix workstations (especially SUN), and personal computers built around an Intel 80286 chip have been suitable hosts for AI development. The most active area for hardware is in LISP chips, of which Texas Instruments and Symbolics have been first vendors. The hardware market is rich. Both the TI Explorer and the Symbolics Lisp machines are available on a general purpose microcomputer through a board that can be inserted into an Apple Macintosh II. These upgraded systems are called MicroExplorer (a TI Explorer on a Mac) and MacIvory (a Symbolics LISP machine on a Mac), respectively.

The market for expert systems on general purpose computers has exceeded most expectations. The primary reason for this is the ability of vendors such as Coral (now owned by Apple), Lucid, and Goldhill to produce LISP on conventional hardware to run as fast, or nearly as fast, as that on LISP machines.

Mature LISP environments, whether on LISP or general purpose machines, provide rich, high quality hardware such as peripherals and interfaces that AI programmers have

come to expect as a part of the total development environment. These tools include a very high quality keyboard, with fast and definite action. The larger the display the better, with a high resolution and a bit-map display screen. This translates to at least a 60 Mhz monochrome monitor of about 800x1200 pixels. One expects that the bit-map can be used as a vehicle for the display of information that can be used as a "handle" for selecting a program object and manipulating it.

An added feature of this large bit-map screen is the increased output of information and communication. As screens have become larger and redisplay faster, the need for hard-copy printouts of programs has rapidly decreased. Much time and effort is saved by not having to continually print out fresh copies of a rapidly changing program. Given a bit-map display, the programmer can print a screen of information to a laser printer or can copy it to a file. This increases the directness and convenience of communicating information about complex program situations among members of a laboratory.

Another desideratum of an AI environment is the ability to rapidly generate and modify windows. This provision for elaboration and novel redesign distinguishes good window systems from mediocre ones—it is not sufficient to have sophisticated window management facility. The best window interface might have a large number of primitive capabilities and be organized hierarchically as a message-passing system. Given a good window system, even novice programmers can bring up new configurations of windows and mouse-object interactions in a very short time—between one day and one week. Ease of use depends on two factors: well thought-out primitives (e.g., commands such as "open window"), and intermediate-level constructs (e.g., such as "insert function output into window"). Given such commands, only new parts of the user's design need to be changed by using message passing. Standards in window systems have been achieved such that good window systems provide much of the following capabilities:

1. Every window is a separate, freely changeable entity with all its attributes readily available to the programmer, e.g., size, color, font, set, borders, labels, and relative position on the screen, and relative to other windows.
2. Windows are independent entities whose display and input buffers can be dynamically associated and reassociated with multiple active processes under program and end-user control.
3. Windows can display text, icons, or arbitrary bit patterns at any window—relative or absolute position—and can interpret keyboard and pointer mouse input with equal flexibility.

In sum, a window package must 1) provide the programmer/designer with the capability to tailor windows to new applications, 2) must be in the same programming language as, and totally integrated with, the rest of the program development facilities, and 3) should be built around a detail-encapsulating device such as an object-oriented class or “flavor system,” i.e., a subset of functions that facilitate the building of modularized objects, based on message-passing.

Mice are considered standard as a means of directly selecting or “pointing” to positions on the screen. A mouse should provide stability while being clicked, accurate vernier movements, and ease of rapid movement, for which roller-based mice are presently best. Voice entry has made a slow and less than winning entrance into the marketplace. Other hardware innovations are related to the production and integration of multi-media with computers, e.g., video, CD-ROM (Compact Disk - Read Only Memory), television, and audio. These latter devices are discussed in Section 5.6.2.

6.5 General Programming Tools

In previous sections we talked about software and hardware tools that facilitated AI programming. The value of these tools depends in part on the success of general

programming tools, such as described in this section. General programming tools are designed to facilitate the examination, management, and debugging of code in general. Both AI specific and general programming tools should work together to allow a user to lift up the level of development activities from underlying programming languages to high-level designs. In this section we discuss general programming tools that facilitate code development.

General programming tools typically work at one of three levels: code and environment management, debugging, and analysis. These tools are designed for originating, examining, managing, and debugging code. They reduce the memory load and drudgework performed by taking on the predictable and conventional activities, thus relieving programmers of having to carry out details and allowing them to focus on matters of design and on managing the unexpected. Program development tools are applicable to programming tasks of all sorts; they facilitate a programmer's movement from the well-understood to the experimental.

Editors. A good editor supports modification and manipulation of both code structure and text. It "knows" LISP and supports organization through automatic indentation, module balancing, and syntax checking, i.e., parentheses balancing. It also allows the manipulation of program text as simple characters and lines. The editor should be on-line and should share the display space with the program being tested so that the code and the behavior it produces are visible simultaneously. The editor (and the compiler) should be in the same virtual address space as the rest of the development system. It should be programmed in the same language as the rest of the system. The file system, which manages code and other data structures when they are not actively loaded and able to run, should be as transparent as possible. The access, printing, and automatic restorage of files after modification can all be left to the editor.

Debugging. Good environments typically provide sophisticated debugging facilities. A program will be placed in an interactive loop at the point where an error occurs and allow direct inspection and modification of variable values, examination of the stack of

pending calls at various levels of detail, and simultaneous access to the editor. Facilities are provided for deliberate returns from the point of error, with a user-supplied return value so that the program can continue on as though the error had not happened. There is also a facility for "backing up" to a higher point in the calling sequence and restarting the process after interactively editing and reloading the module that caused the error. Without such a facility, a great deal of programming time is lost due to having to restart long programs from the beginning in order to continue testing. In mature development environments one expects to find:

1. facilities for tracing calls to functions with a "breakloop" on specified input or output conditions,
2. ability to "single-step" any program, function call by function call; and
3. an "inspection" facility for displaying complex data-structures in a readable way and interactively examine their parts.

Assistance in Analyzing Code. Writing a large amount of code requires analysis and organizational tools to lessen the programmer's memory load and assure the burden of producing the "obvious" detail code from specifications. Such tools are only beginning to be available through "programmers" assistants, or systems that write code from detailed specifications. Facilities that are available and that one should expect are:

1. Cross-indexing and cataloging of program objects that define variables within a program definition, the global variables referenced, and the functions it calls and what functions call it.
2. Integration of cross-index catalogs with the editor so that one can, for example, change all names of calls of "process-by-months" to "process-by-weeks," by carrying out the editing, reloading, and refiling automatically.

“Programmers assistants,” when available, will provide automatic examination of code to check consistency and to offer corrections. Prototype assistants can now translate specifications into LISP Code [Waters, 1982] or advise a programmer about rewriting existing code that will be more efficient [Fischer, 1987]. The problem is that if one does not program in the style these facilities are tuned for, the “assistant” might convert correct code into incorrect code. Since assistants typically cannot be turned off, considerable ingenuity may be required to get around them. In general, the area of programming apprentice systems is a vigorous research area, and one should expect such systems to be included as parts of LISP system “bundles” sometime in the future.

6.6 Summary and Discussion

This chapter explored the availability of software and hardware systems for building knowledge based tutoring systems. Languages such as LISP, Prolog, and Smalltalk were described along with features that an author should assess before choosing a system. Knowledge engineering shells, such as KEE and Personal Consultant, were described along with general programming tools, such as editors and debuggers. Hardware, including special purpose LISP machines and general purpose machines, were discussed including specific features, such as memory size, screen and keyboard design, and the existence of bitmap display and windowing.

CHAPTER 7

THE FUTURE: COMPUTER PARTNERS IN EDUCATION AND INDUSTRY

This document presented a guide for development of knowledge-based tutors. It described a number of problems and issues to be addressed and provided guidelines for educators involved in the development process. Examples showed how the computer can be used as a "trusted consultant," "benevolent mentor," "cognizant tool," and "problem-solving partner" (Peelle & Riseman, 1975; Slovin & Woolf, 1989). In this chapter, predictions are made about future uses of such systems in education and industry as computers and humans begin to cooperate. This chapter also discusses current barriers that make building these systems difficult. It defines the type of breakthroughs needed in psychology, education, and computer science to achieve partner-like computer systems. Ethical and moral issues related to the impact of technology on daily life are acknowledged, since they will play an increasingly important role in the responsible design, implementation, and use of computer power.

7.1 Impact of Knowledge-based Tutors

Implications of this work go beyond the possibility of producing a few more knowledge-based tutors. The expectation is that as the process of building knowledge-based systems becomes clearer to a wider and more varied audience, additional development activity will stimulate the advance of information technology into the classroom. Not only will a larger corpus of authors, including instructional designers, teachers, administrators, and

psychologists help produce a wider variety of tutors, but just the exercise of building such systems will generally enhance the communication of knowledge between domain, computer, instructional, and cognitive experts.

Development of knowledge-based tutors will also contribute to several areas of Artificial Intelligence. For example, prototype tutors might demonstrate how a machine can reason about a user's knowledge and how it might flexibly communicate with him/her. Currently, AI systems are intolerant of their users. Even expert systems have little understanding of the user's knowledges, little expectation about how s/he communicates, and only a weak or stereotypic model of the user. Such systems cannot explain their own reasoning and cannot use discursive acts, such as questions and answers, to clarify the user's current needs. Development of knowledge-based tutors should help researchers in AI define how to build a user model, generate machine explanation, engage in question/answering, tailor discourse to an individual user, build large-scale machine-mediated communication systems, and encode more about human learning for use in long term interactive human/machine projects, such as process control.

A final implication of this work is the realization that any system which communicates information to a user must have an AI model of that user, or an active agent that keeps track of the user's knowledge, makes inferences about his/her goals, and considers which style of communication or sequence of discursive topics is appropriate.

7.2 Impact of Knowledge-based Technology on Education

We recognize that education is critical in a society's attempt to increase and pass on knowledge from one generation to another. Intelligent tools are seen as vital—filtering, modeling, and sharing massive amounts of data and information that will become available through multi-media and electronic networks. One might wonder how the existence of such intelligent tools will impact on education.

Any new technology passes through several phases as it impacts society [Dede, 1988]. These phases, for example, are visible in the case of the automobile. After a 50-year development period, the car has finally arrived, having generated societal changes that are now more consequential than the changes brought on by the invention of the car itself. These changes can be seen in roads, cities, and parking places (or lack thereof). Once the automobile fully entered the society, commuting distances were scaled up to take advantage of rapid, fairly safe and reliable transportation. The car, and later jet travel, provides a new model of reasonable travel for family life, e.g., commuter marriage became possible along with international tourism and multi-national companies.

The advent of knowledge-based systems in education might generate a similar abundance of auxiliary societal and educational changes [Dede, 1988]. Below we suggest four phases of evolutionary changes that might occur as a result of the introduction of knowledge-based systems:

- **Phase One:** Knowledge-based systems, such as intelligent tutors, are adopted by affluent schools and training sites. These systems carry on limited one-on-one tutoring sessions with a limited number of students. This phase is currently underway, e.g., see Anderson [1985], Johnson and Soloway [1985], Woolf et al. [1987], and Psotka [1988].
- **Phase Two:** Schools and training sites begin to change internally to take better advantage of these tools. Knowledge-based tutors might become stand-alone teaching modules for small groups of students, thus reducing crowded classrooms and improving individualized teaching. Networked systems, distributed at a distance, learning, and non-school sites also become basic to schools. This phase has also begun, e.g., Tinker [1987] and Southworth [1988].
- **Phase Three:** Schools develop new functions and activities enabled by knowledge-based systems. For example, the number of lectures and their length might be reduced considerably, opening the way for teachers to assume the role of consultant

and advisor in conjunction with classroom computers. This has happened already in some "computer classes." Educational resources beyond schools, such as communities, families, industries, and the military will begin to assume educational leadership roles, possibly opening up competitive relationships between alternative institutions. This has already happened in industry training. For educational institutions, this phase is about five years away, and thus planning and policy analysis should begin now.

- **Phase Four:** The original role of schools may become radically transformed, displaced, or obsolete as goals, such as marketplace success of software, dominate more traditional educational goals. This phase is possibly a decade away, and again planning should begin now.

Planning and policy analysis will help assure that the continuing implementation of these four phases is not dominated by the self-interest of a few, as suggested in Phase Four above. The destructive side-effects of each phase should be carefully monitored and contained. For example, as computer systems play a more central role in education, government agencies and (software) publishing houses might seek greater control over what information is communicated.

Knowledge-based tools might bring other changes to education [Dede, 1988]. For example,

- **Administrative changes.** More data about students, classes, and teacher evaluation will become available, allowing more careful analysis of classroom activities. Expert systems might be used to offer diagnosis and evaluation. Middle management and administrative assistant roles could erode if based primarily on information filtering and simple statistical tasks.
- **Reduced class size.** Smaller class sizes could result from use of knowledge-based tutors and non-human instructional agents. As more computers are introduced,

students will begin working in small groups. Education will become more distributed.

- Improved teacher status. Teachers will require more and different training as they assume a more personal relationship with students. Learning to be a counselor and advisor, and learning to work alongside machines with encyclopedic knowledge will become more challenging and more humbling. Only the best and most qualified teachers will survive. This should provide an impetus to upgrade the role of teacher in society.
- Increased numbers of students. As more people assume the role of student in academia, industry, and government, the total amount of "adult education" will increase.
- Increased educational equity. Since the entire economic society will be dependent on and benefit from education, each member of society would have a strong self-interest in promoting optimal educational opportunities by all learners.

Machine-based tutors and coaches might be responsible for communicating basic concepts while human teachers act as counselors and focus on higher order skills and "complex occupational, citizenship, and ethical knowledge" [Dede, 1988]. Both teachers and students will learn from knowledge-based tutors, each pursuing independent instructional goals. This change in teacher role would result in lessening the monotony of rote teaching.

7.3 Integration of Knowledge-based Tutors

This document proposed that knowledge-based tutors be used as tools to help adopt a technology-intensive approach to classroom teaching. However, as shown above, the integration of technology in society will itself promote substantial changes in society. A technology does not determine its own effect on society. Rather, the form of its

implementation and integration provides a powerful societal impact. Thus, knowledge-based tutors might produce undesirable results if their implementation is not carefully monitored. For example, an author of these systems might be interested in gaining control over the knowledge communicated. S/he might limit the system so that the curriculum can not be modified or refined once installed. Such systems would contribute to a centralization versus a decentralization of education. Other less stringent systems might facilitate "show-horning" multiple domains into a single framework, and might restrict the number of tools and interfaces available for teaching. Such a technology would serve to homogenize rather than diversify education. It would restrict thinking and serve to further trivialize the role played by students and teachers.

The technology in this document has been described in terms of a decentralized implementation. It has been proposed as a way to tailor a variety of curricula to an individual and to using multimedia (video disk, audio, video, and film) to enrich and enliven the presentation.

Current educational structure is based on centralized learning and a graded track system which reflects the industrial view of human society. Technology described in this document can facilitate a departure from this approach and can deliver decentralized education, achieved through distributed communication and knowledge-based systems. Once artificial intelligence techniques for education are integrated with multi-media and hypertext systems, humanity might be poised to establish a global infrastructure of knowledge-based systems with the individual at its center (see Section 5.6). Achieving this multi-media communication network implies the ability to connect students to:

- stores of widely available encyclopedias of information accessible through networks;
- problem-solving expert systems designed for use by people or other machines; and
- computational agents or interfaces that facilitate human-machine communication.

Once knowledge-based systems employing multi-media hardware and software are generally available and easy to build, artificial intelligence as discussed in this document will emerge as a core technology for educational multimedia.

One potentially significant impact of intelligent and multi-media machines is to transform education from a "push" to a "pull," in which people eagerly choose to work with machine tutors. For example, operators who used the Recovery Boiler Tutor (Section 2.1.1), which used only simple computer graphics, reported working on it up to 76 hours in the first three months. We didn't ask the operators to work that many hours, they just enjoyed playing with the system. Imagine what would happen if the tutor had employed multi-media! Teaching systems that attract people have an obvious and immediate advantage over other non-attracting teaching tools. It is this kind of attraction between students and systems that we encourage in the design of knowledge-based multi-media systems. The challenge comes in focusing on a new definition of intelligence, which is not limited to information storage and retrieval but is defined by the use of cognitive skills and problem-solving methods.

7.4 Needed Breakthroughs

This section describes some specific breakthroughs required in the technology and in education before full integration of knowledge-based systems can be realized. It also looks at how knowledge-based systems might provide more interesting jobs in the workplace. We separate conceptual and physical material breakthroughs into two categories: those related to hardware/software changes, and those related to educational changes.

Hardware/Software Breakthroughs. Breakthroughs and continuing developments in hardware and software will enable deployment of knowledge-based educational systems [Dede, 1988]. These include the following:

- The *memory and speed* of computer systems should continue to increase, and their *size and cost* should continue to dramatically decrease. Processing speed of micro-computers should increase by more than two orders of magnitude (to the equivalent of current supercomputers), and conventional micro-computers should be outfitted with workstation-like quality graphics (e.g., flexible window managers, buttonable icons, menus, etc.) and be capable of networking. Powerful systems should be reduced to desk or lap-top size, and their cost per student for ten hours/week usage reduced to around \$1,000 per year by 1995.
- Rapid advances should continue in the development of machine responses, not including natural language responses, to include improving a machine's explanatory capabilities, its use of felicity conditions, and its theories of hints [van Lehn, 1983]. (Felicity conditions are those principles about pedagogy used by teachers and expected by students. For example, a teacher will typically introduce a single topic, focus on it for a while, and then summarize it before moving on to a new topic.)
- Ability to represent *qualitative causal reasoning* in domain knowledge should continue and ultimately be expressible in clear and simple terms. Qualitative reasoning includes a machine's ability to use non-numeric measurements (e.g., time and physical characteristics) to model a domain and make decisions. Causal reasoning refers to a machine's ability to represent and make decisions based on reasoning about cause and effect rules.
- *Cognitive processes* of teaching and learning should continue to be made expressible, and ultimately representable, in the student model in terms of common bugs, possible misconceptions, and differential and perturbation models of errors.

Artificial Intelligence Issues. The technology of artificial intelligence is instrumental to the development of knowledge-based systems. However, AI currently has many limitations. In particular, current AI programming languages are inadequate for expressing knowledge, and control structures are limited in their ability to handle complex

contingencies of machine-person interaction. Better languages are needed for expressing conceptual, procedural, heuristic, and simulation knowledge. Although software and hardware results are impressive, some AI problems have not moved ahead and do not seem solvable in the next two decades. For example, natural language processing systems have made little progress in the last 5 years. AI systems that incorporate common sense, peripheral "real world" understanding are still a long way off. However, machine learning has just begun to emerge as a potentially viable AI technology and recent advances in connectionist models of visual recognition and learning are very promising.

Long-term AI research issues require a substantial effort before knowledge-based technologies become generalizable and well-established. Researchers need to find a way to reduce the amount of time required to produce knowledge-based systems. Currently, tutoring systems require a sizable investment of time, much more than the 200 hours suggested for building a CAI system. Existing software support systems, such as shells, enable us to move more rapidly toward development of these systems (see Section 6.3); however, they are not entirely adequate given the requirements of a knowledge representation systems, as discussed in Section 5.2.1.

Other short-term AI goals include improving communication between workers in AI and education. For example, researchers in artificial intelligence should work with people in training and education to develop training systems. Hopefully the best and brightest people in teaching should be financed by government and private foundations to build knowledge-based tools for education.

Knowledge Engineering Issues. Researchers must improve the process of transfer of expertise from humans to computers. Human knowledge is often distributed, incomplete, and acquired incrementally [Bobrow et al., 1986]. Therefore, part of the knowledge engineering effort should go toward creation of a "community memory" in which multiple experts contribute knowledge of teaching and learning into a central repository. This repository might contain all the topics, responses, presentations, analogies, and strategies for teaching specific curriculum. The path of the computer through the repository would

not be prespecified, as discussed in Section 5.1. Such a community memory would be very large, such as the knowledge base being built by Doug Lenat at MCC, Austin, Texas [Lenat, 1988]. In this 10-year project, researchers aim to encode "all" the knowledge held by an encyclopedia. It is estimated that it will hold half a million nodes when complete. Once such a framework is built, it should be applicable to many topics and many domains. A community of experts is obviously required here because a single expert might create a knowledge base that is foreign to others, has conceptual holes, or solves problems in an uncommon way due to blind spots in its knowledge base. Historically, where additional experts have been included in the knowledge engineering process, the resulting system is more successful (see Section 4.2).

Basic cognitive research into teaching and learning must be developed alongside the building of knowledge-based tutors. For example, builders of tutors need to know whether a student's view of that domain is interpretable, complete, or stored as "knowledge in pieces" [di Sessa, 1984]. Knowledge about what motivates a student and what is known by him/her should be included. Knowledge engineering cannot be successfully performed until we identify such features.

For each new domain, tutoring primitives have to be reconceptualized (including the generation of topics, strategies, misconceptions, etc.). Knowledge and heuristics used in each domain have to be made explicit. This amount of formalization of problem-solving knowledge is not usually available. Textbook knowledge cannot supply it; formulas provided in books are much too sanitized, neat, and incomplete.

The technology of building knowledge-based tutoring systems is often driven by our assessment of how we teach. Currently we don't know much about how teachers make decisions in organizing or communicating knowledge. Work in medicine [Clancey & Letsinger, 1981] reveals that trained physicians and teachers of medicine often don't discriminate precisely and consistently between cause and effect or between substances and processes. Without such clear distinctions, the physician's most basic experiential knowledge about how to solve problems can't be formalized in a computer.

Educational Breakthroughs. Several breakthroughs in education are needed to fully implement the technology described in this document:

- **Distributed Teaching:** Education should continue to be moved out of the classroom. It should be disseminated equally in school, home, community, and workplace. For example, families in an average community spend two to ten times the amount of money spent by schools in that community on computer education programs [Wakefield, 1986]. Many industries now contribute large sums for teacher training and materials for primary and secondary education (see Section 1.4).
- **Cooperative Teaching:** Education should involve groups of people working on problems in concert with tutoring systems. It should focus on a dialectic form of teaching that stresses an individual's progress through self-study, with help from machines, peers, parents, and teacher counselors. Learning should involve use of video tapes, computer databases, machine courseware, reference libraries, and neighborhood resources.
- **Constructivist Education:** Both students and teachers need to learn by doing. The constructivist teaching paradigm suggests that students need to make hypotheses and evaluate their own internal model of knowledge. Section 5.5.3 provided reasons to use the constructivist philosophy in developing a machine tutors and Section 3.3. provided examples of how to elicit constructivist reasoning about tutoring from experts. An information-based society is complex and requires the addition of powerful learning and teaching strategies to the current educational system. Constructivist strategies show evidence of being powerful and active students show evidence of learning more effectively than passive ones.

Students should tutor each other, teachers should consult with students and machines, and each should participate in collective problem-solving.

A teacher is an expert on something, a guide and often a hero. To a six-year old, such qualities may be found completely in a 10 year old . . . and often with greater motivation than if supplied by a 28 year old [Bete, 1969].

7.5 Impact of Technology on the Workplace

A new role for humans is evolving in the workplace; humans and machines are beginning to work as partners. Machines can already replace people in complex but well-structured tasks, such as factory scheduling, monitoring, diagnosing, and summarizing events in an electric power plant [Bruno et al., 1986], trouble shooting electronic equipment [Brown et al., 1982], and designing new copier machines [Talukdar et al., 1986; Mittal et al., 1986]. On the other hand, people are more adept at recognizing and learning from analogical situations, solving unusual problems, and reasoning from incomplete and imperfect data. Using these complementary intellectual strengths, both computers and humans could work together in a partnership which emphasizes the strength of each participant [Peelle & Riseman, 1975].

A knowledge-based workplace, where humans and computers share the learning and performance tasks, might require *more* human-worker intelligence. Humans will need to use machine intelligence to augment their own thinking, yet the job might become more complex as a result. Complex jobs require both structured and unstructured decision-making. Humans still supply sophisticated reasoning such as creativity, flexibility, decision-making, evaluation synthesis, and holistic thinking. The cooperation of humans and machines in the workplace requires that machine intelligence use models of skilled activities, intelligent tutors, and expert decision-aids to communicate with humans.

Basic cognitive skills such as computation, pattern matching, designing, and planning are becoming well-understood and being implemented into AI tools. As this occurs,

education should shift from teaching lower-level skills such as the steps for double-entry bookkeeping and move toward higher-order cognitive skills such as training for creativity and decision making. To continue to train people on lower-level skills would be as effective as "grooming John Henry to compete with the steam engine" [Dede, 1988]. People will need a foundation of lower order concepts, i.e., steps necessary for long division, but will not need to be drilled on lower computational skills, which can be better performed by calculators and computers. Training should evolve toward helping humans understand how sophisticated problems are solved, how unusual cases are recognized, and how to communicate, either with humans or machines, to access information. Thus, educational assessment should shift from evaluation of a student's ability to memorize topics and define concepts to evaluation of their higher-order cognitive skills.

Workers will also have to be educated in affective abilities such as cooperation, compromise, and group decision-making. This is necessary because industries that become decentralized and democratic as a result of knowledge-based tools require that humans communicate more often and in more depth [Dede, 1988]. Affective and interpersonal abilities will become an important measure of educational effectiveness in a future where person/machine partnerships dominate. Teaching affective skills requires altering classroom structures and goals.

7.6 Living in the Knowledge-based Society

The knowledge-based society, as described in Chapter 1, has already come to pass; i.e., we now live in a world where access to information is a prerequisite to increased knowledge, power, and wealth. Humanity needs to acquire the requisite long-range, global attitudes about using this information, communicating it, and distributing it in order to advance the economy. Appropriate use of such knowledge is vital to survival of a global economy. No country can remain prosperous in the current era by clinging to an industrial base when the global marketplace has become information-based [Dede,

1988]. Additionally, an economy based on partnership between workers and intelligent tools which does not also provide an adequate educational basis is 'skating on thin ice.' Such an economy does not provide an infrastructure needed for the weight that will be placed upon it. Trying to remain prosperous and democratic for two more decades using the traditional educational paradigm is bankrupt.

Ethical and moral issues should also be considered as they relate to the impact of technology on daily life. We need to acknowledge and become responsible for the design, implementation, and use of computer power [Weizenbaum, 1976]. This section discusses some of these issues.

Socio-educational Issues. We realize that an information society requires AI innovations, such as described in Section 7.4 above. Realization of a fully knowledge-based society will be slowed down if recent advances along the lines suggested above bog down or if barriers to improving the power/cost ratio are not overcome (physical constraints associated with quantum mechanical effects and the speed of light). Even if AI proceeds rapidly and physical limitations are overcome, the information society may not completely emerge if the development of knowledge-based systems stagnates due to insufficient funding, lack of skilled human resources, or a failure to implement research initiatives.

Another possible deterrent, in addition to the overarching problem of cost, is rejection by the educational community, including schools, universities, parents, and communities. Inertia, self-interest, and resistance to change have been known to undermine educational reforms in the past. To create a shift to a new instructional mode requires support from administrators and governing officials, extensive teacher training, and community awareness programs. The present dissention about proper goals, methods, responsibilities, and funding for education makes such a coordinated transformation very difficult [Dede, 1988].

Moral Issues. Ethical and moral issues must be considered as they relate to the use of technology in education, the private consulting room, and the workplace. Concern for the responsible use of "computer power" should play a significant role in the development of the scenarios presented here [Weizenbaum, 1976; Slovin & Woolf, 1989]. Discussions of the social and ethical responsibilities encumbered upon individuals involved in the construction of high-impact technology should be encouraged. Questions of goals, motives, purposes and values are embedded in the design of knowledge-based tutors and should be made explicit by knowledge engineers and domain experts. Perhaps some statement regarding ethical considerations in design should be required of developers.

Weizenbaum has been outspoken about the matter:

The point is . . . that there are some human functions for which computers ought not to be substituted. It has nothing to do with what computers can or cannot be made to do. Respect, understanding, and love are not technical problems. . . . Scientists and technologists have, because of their power, an especially heavy responsibility, one that is not to be sluffed off behind a facade of slogans such as that of technological inevitability. [Weizenbaum, 1976]

Winograd and Flores offer the following cultural perspective:

Computers, like every technology, are vehicles for the transformation of tradition. . . . We can let our awareness of the potentials for transformation guide our actions in creating and applying technology. In ontological designing, we are doing more than asking what can be built. We are engaging in a philosophical discourse about the self—about what we can do and what we can be. Tools are fundamental to action, and through our actions we generate the world. [Winograd & Flores, 1986]

Future collaboration with knowledge-based tutors will generate new problems and new possibilities in education, training, and consulting. We need to continually clarify

the roles of both humans and machines and improve the competencies of each. Such an unfolding process provides for further innovation in the design and development of a promising partnership.

APPENDIX A: HUMAN NETWORKING

The following is a partial list of journals, periodicals, newsletters, conferences and workshops that might appeal to the reader. Some periodical listings include descriptive material as well as resource people to contact for more information. Publications are first categorized by focus and approach and then alphabetically.

Summary of Publications by Focus

Artificial Intelligence

AI Magazine

AI Expert

IEEE Expert

International Journal of Expert Systems: Research and Applications

IEEE Transactions on Systems, Man, and Cybernetics

Artificial Intelligence and Education

Intelligent Tutoring Systems

Interactive Learning Environments

Journal of Artificial Intelligence in Education

Computer Science

Byte,

Communications of the Association for Computing Machinery

IEEE Computer

Cognitive Science

Cognition and Instruction

Cognitive Psychology

Cognitive Science

Proceedings of the Cognitive Science Society

Computers and Education
Academic Computing
Hands On!
Journal of Educational Computing Research
Journal of Computing in Higher Education
Machine-Mediated Learning

Human-Computer Interface Issues
Human-Computer Interaction
International Journal of Man-Machine Studies
IEEE Transactions on Systems, Man, and Cybernetics
SIGCHI Bulletin: Special Interest Group in Computer Human Interaction

Education
Educational Researcher Harvard Educational Review
Technology in Education
Journal of Educational Technology Systems
T.H.E. Journal: Technological Horizons in Education
Technology and Learning

Publications

Academic Computing

Focus: Computers and Education

A free journal covering computer use in higher education.

Kolbensvik, J. (Ed.),

Academic Computing Publications,

PO Box 804

McKinney, TX 75069.

Contact: Joel Kolbensvik (214) 548-2101.

AI Magazine

Focus: Artificial Intelligence

Published quarterly, AI Magazine is the official publication of the American Association of Artificial Intelligence. Its purpose is to disseminate timely and informative articles that represent the current state of the art in artificial intelligence.

Contact:

American Association for Artificial Intelligence

445 Burgess Drive

Menlo Park, CA 94025-3496

415 328 2123

AI Expert

Focus: Artificial Intelligence

Byte,

Focus: Computer Science

Cognition and Instruction

Focus: Cognitive Science

Lawrence Erlbaum Assoc.

365 Broadway

Hillsdale, New Jersey 07642

Cognitive Psychology

Focus: Cognitive Science

Academic Press

Cognitive Science

Focus: Cognitive Science

Ablex Publishing Corp.,

355 Chestnut Street

Norwood, N.J., 07648

Proceedings of the Cognitive Science Society

Focus: Cognitive Science

Communications of the Association for Computing Machinery

Focus: Computer Science

One of the oldest publications for the general computing community. It publishes refereed articles of substance emphasizing concepts and principles, as well as notices, general news, and conference information.

Association for Computing Machinery

11 West 42nd Street

New York, N.Y., 10036

212 869 7440

IEEE Computer

Focus: Computer Science

IEEE Computer Science

10662 Los Vaqueros Circle

Los Alamitos, CA. 90720

714 821 8380

Educational Researcher

Focus: Education

Published nine times a year; contains news and features of general significance in education research.

American Educational Research Association

1230 Seventeenth Street, Northwest

Washington, D.C., 20036

202 223 9485

Contact:

Susan Wantland

IEEE Expert

Focus: Artificial Intelligence

The Computer Society of the IEEE

10662 Los Vaqueros Circle

Los Alamitos, CA. 90720

714 821 8380

Hands On!

Focus: Computers and Education

A newsletter describing innovative uses of computers, especially micro-based teaching, for grade school. Published by TERC, Technical Education Research Center, Inc.,

Contact:

Robert Tinker

TERC

1696 Massachusetts Avenue
Cambridge, MA 02138

Harvard Educational Review

Focus: Education

A journal of opinion and research in the field of education. Published by an editorial board of graduate students at Harvard University.

Editorial and Business Office

Gutman Library Suite 349

6 Appian Way

Cambridge, MA 02138-3752

Contact:

Karen Maloney

617 495 3432

Human-Computer Interaction

Focus: Human-Computer Interaction Issues

Intelligent Tutoring Systems

Focus: Artificial Intelligence and Education

Learned Information (Europe) LTD

Woodside, Kinksey Hill

Oxford OX15AU

United Kingdom

Tele: Oxford + 865 730275

Contact:

Masoud Yazdani

Department of Computer Science

University of Exeter

Prince of Wales Road

Exeter EX 44PT

United Kingdom

Interactive Learning Environments

Focus: Artificial Intelligence and Education

Ablex Publishing Corporation

355 Chestnut Street

Norwood, New Jersey 07648

201 767 8450

Contact:

Elliot Soloway

Department of Electrical Engineering and Computer Science

University of Michigan

1101 Beal Avenue
Ann Arbor, MI. 48109-2110

Editors:

Eliot Soloway, Kurt vanLehn, William Clancey, Roy Pea, Tim O'Shea,

International Journal of Man-Machine Studies

Focus: Human-Computer Interaction Issues

Describes human-computer interface issues and artificial intelligence approaches to the development of computing systems.

Academic Press
24-28 Oval Road,
London, NW1 7DX,
United Kingdom.

Contact:

B. Gaines (Ed.)
Department of Computer Science, The University
Calgary, Alberta, Canada T2N 1N4

International Journal of Expert Systems: Research and Applications

Focus: Artificial Intelligence

Provides indepth reviews of expert systems applications.

JAI Press, Inc.
55 Old Post Road-No 2.,
P.O. Box 1678
Greenwich, Connecticut 06836-1678

Journal of Artificial Intelligence in Education

Focus: Artificial Intelligence and Education

Publishes articles that advance knowledge and theory on how intelligent computer technologies can be used in education to enhance learning and teaching. Published quarterly for researchers, teacher educators, curriculum/product developers, etc.

Association for the Advancement of Computing in Education
P.O. Box 60730
Phoenix, AZ85082
602 952 2712

Journal of Computing in Higher Education

Focus: Computers and Education

Scholarly essays, reviews, reports, and research articles that contribute to understanding the issues, problems, and research associated with instructional technology and educational management information systems. Articles represent all aspects of academic and administrative computing.

Paideia Publishers
P.O. Box 343

Ashfield, Ma 01330

Contact:

Carol B. MacKnight

University of Massachusetts

Journal of Educational Technology Systems

Focus: Technology in Education

Investigates and reports on actual classroom experience in the use of technology - video disks, closed circuit television, audio and audiovisual programs and programmed instruction. Purpose is to guide educators in the use and full range of available technology for the classroom

Baywood Publishing Company, Inc.

120 Marine Street

Box D.

Farmingdale, N.Y. 11735

Contact:

Dr. Thomas T. Liao

Department of Technology and Society

State University of New York

Stonybrook, New York.

Journal of Educational Computing Research

Focus: Computers and Education

Describes research on the application, effects, and implications of computer-based education. Contains critical analyses, reports on research in progress, as well as design and development studies.

Baywood publishing Company, Inc.

120 Marine Street

Box D.

Farmingdale, N.Y. 11735

Contact:

Dr. Robert Seidman

New Hampshire College

Graduate School

2500 North River Road

Manchester, New Hampshire, 03104

603 485 8415

Machine-Mediated Learning

Focus: Computers and Education

Taylor & Francis, Ltd.,

London, England.

Contact:

Friedman, E., & Resnikoff, H. (Eds.),

T.H.E. Journal: Technological Horizons in Education

Focus: Technology in Education

A free journal covering the uses of technology to improve education

Information Synergy, Inc.,

2626 South Pullman,

Santa Ana, Ca. 92705

Contact: Sylvia Charp

39 Maple St.

Upper Darby, Pa. 17082

Technology and Learning

Focus: Technology and Education

A newsletter describing advanced technology in education and training. Published by Lawrence Erlbaum Publishers.

Contact:

Hollis Heimbouch

365 Broadway

Hillsdale, NJ 07642

201 798 1913

IEEE Transactions on Systems, Man, and Cybernetics

Focus: Human-Computer Interaction Issues

SIGART Newsletter: Special Interest Group in Artificial Intelligence

Focus: Artificial Intelligence

Published by SIGART

Association for Computing Machinery

11 West 42nd Street

New York, N.Y., 10036

212 869 7440

SIGCHI Bulletin: Special Interest Group in Computer Human Interaction

Focus: Human-Computer Interaction Issues

Published by SIGCHI

Association for Computing Machinery

11 West 42nd Street

New York, N.Y., 10036

212 869 7440

Conferences

AAAI, American Association for Artificial Intelligence

Contact:

Claudia Mazzetti
AAAI Office
445 Burgess Drive
Menlo Park, CA 94025

ADCIS, Association for the Development of Computers in Schools

Contact: Tom Reeves

AERA, Association of Educational Researchers

Has dedicated several sessions to research in Artificial Intelligence and Education.

Contact:

Wally Feurzeig
Bolt, Beranek, and Newman, Inc.
10 Moulton St.
Cambridge, MA 02238
617 837 3448

EDUCOM

P.O. Box 364
Princeton, NJ 08540
(609) 520-3355

Attracts around 2,000 educators, officers, software developers, networking specialists and industry, government and research laboratory representatives from 20 countries. Explores research issues such as academic computing, coordinating libraries, and building computer networks.

Conferences of the Cognitive Science Society

Contact:

John Anderson
Psychology Dept.
Carnegie-Mellon University
Proceedings distributed by Lawrence Erlbaum Associates, Inc., NJ 07642

International Conferences on Artificial Education and Education

Contact:

A1-ED-89 Secretary—telephone (31) 20-525-2073

SWI, University of Amsterdam
Herengracht 196
1016 BS Amsterdam
The Netherlands

Covers recent work in architectures, cognitive research, domain representation, teaching strategies, student modeling and diagnosis, modeling worlds, AI-language learning.

International Conferences on Intelligent Tutoring Systems

Contact:

Marlene Jones, Telephone: (403) 297-2666
Alberta Research Council

IJCAI, International Joint Conference on Artificial Intelligence

Contact:

Wolfgang Bibel, C.S.
University of British Columbia
6359 Agricultural Road
Vancouver, BC
V6T 1W5

ICCAL, International Conferences on Computer-Assisted Learning

Contact:

Janet Harris
Center for Continuing Education
University of Texas at Dallas
P.O. Box 830688, MS CN 1.1
Richardson, TX 75083-0688

NECC, National Educational Computing Conference

Contact:

NECC '89
International Council for Computers in Education
University of Oregon
1787 Agate Street
Eugene, OR 97403-9905

APPENDIX B: EXAMPLE STORYBOARDS

Designs and Storyboards

Example 1: Design for a Chemistry Tutor, by Patricia Corradino,
Phyllis Eisenberg, Donna Lalonde, James Scott.
Funded by the National Science Foundation
Summer, 1988, University of Massachusetts

CHEMISTRY TUTOR FINAL REPORT
"Understanding Dynamic Chemical Equilibrium"

BY: Patricia Corradino
Phyllis Eisenberg
Donna LaLonde
James Scott

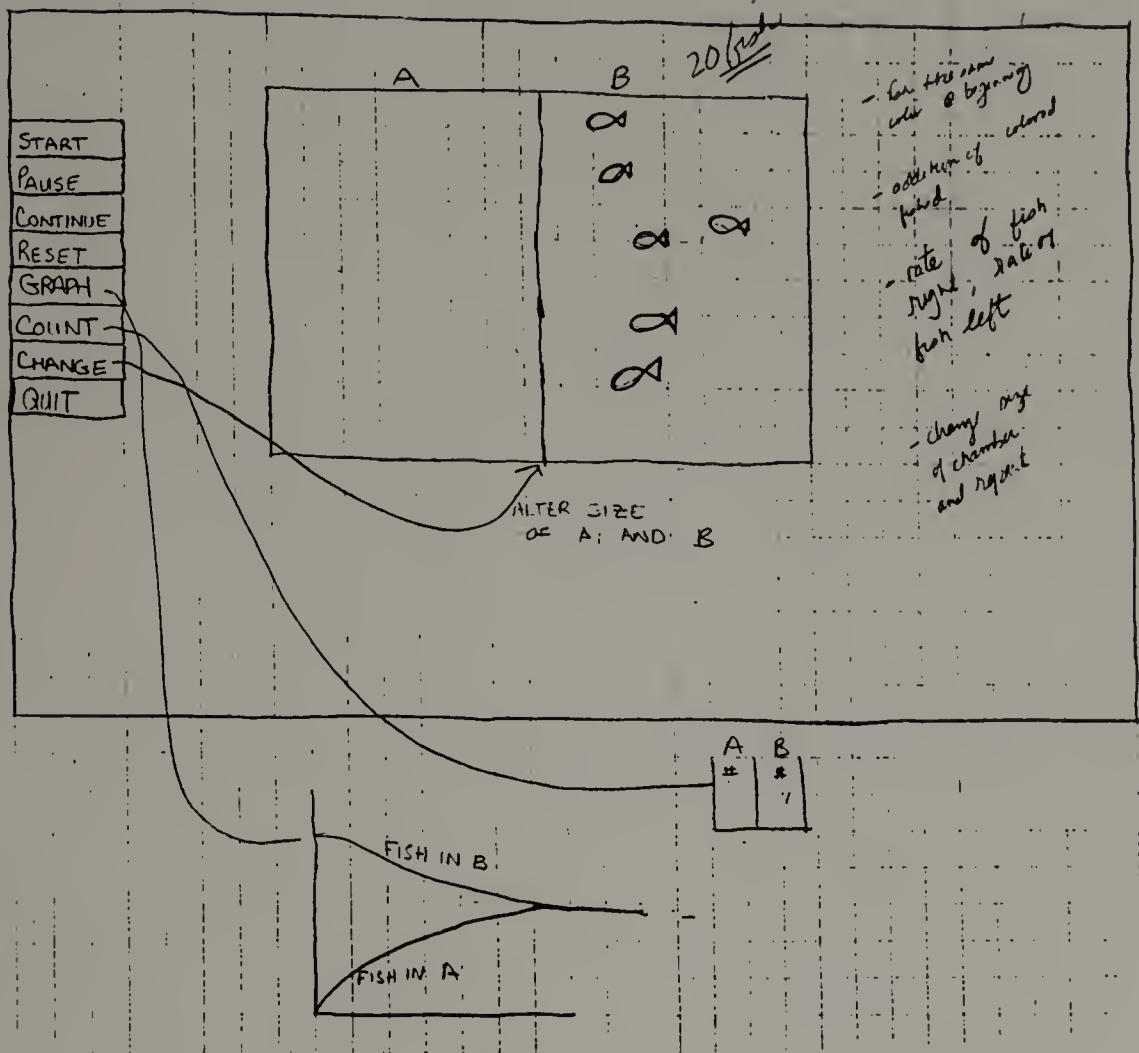
Summer 1988

OVERVIEW of STRATEGY

- I. Introduce concept of equilibrium with simulation of gold fish in a tank. Explore the students understanding of equilibrium by asking user to experiment with the simulation.
- II. Series of demonstration simulations developed to make the user familiar with the environment and to ascertain the level of understanding of auxiliary components i.e. graphs.
- III. Present the user with microworld he/she can experiment with to develop a theory about systems at equilibrium. The microworld is similar to the demonstration so the user is familiar with the devices that are available for investigation.
- IV. Address any misconceptions that are uncovered with particular examples and help.

Overview of Program

Design for a Chemistry Tutor (Cont)



Equilibrium Simulation Screen

Basic Model

Design for a Chemistry Tutor (Cont)

SCRIPT

NOTE: THE DESCRIPTION OF WHAT THE STUDENT SEES IS IN PARENTHESES. EVERYTHING ELSE IS THE TEXT THAT APPEARS ON THE SCREEN OF THE MONITOR.

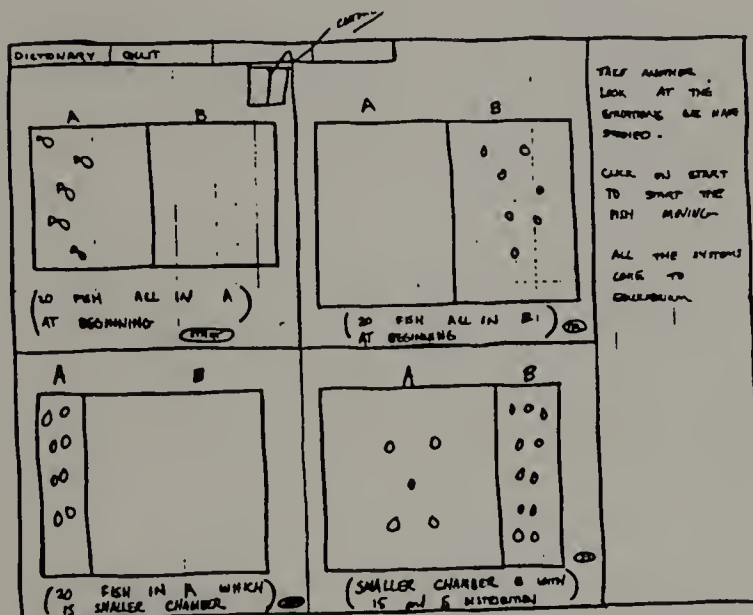
(The screen shows a tank with two labeled compartments that are equal in size. There are 20 fish in compartment B and 0 fish in compartment A.)

We're going to show you a model of a dynamic system that will reach a state of equilibrium. Our model is this closed tank in which fish can migrate from one compartment in the tank to the other. This is a closed system; the total number of fish within the tank will remain the same.

Count the number of fish in the tank. When you examine this system, you see that there are 20 fish in compartment B and 0 fish in compartment A.

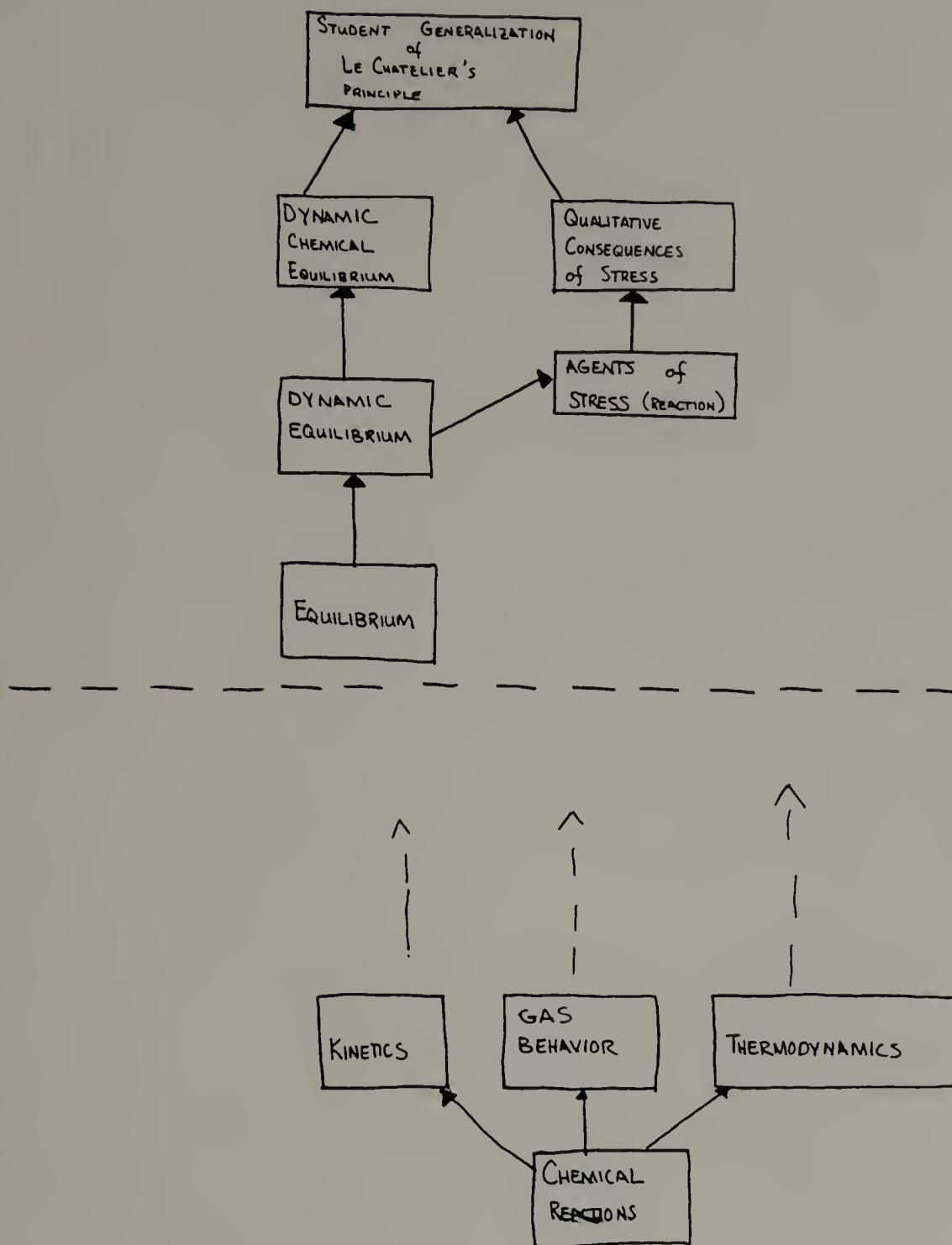
This system isn't dynamic right now. There is no motion. When you click the mouse on START, the fish will begin to move. Observe their random motion. Once this moving system reaches equilibrium, we'll ask you some questions about the conditions that exist in a state of DYNAMIC EQUILIBRIUM. You'll probably be able to answer these questions based on your observations of the movement of the fish.

Equilibrium Script



Equilibrium Simulation Screen
Four Simultaneous Models

Design for a Chemistry Tutor (Cont)



Topics Covered in the Equilibrium Simulation

Design for a Chemistry Tutor (Cont)

Click on START to begin.

(When the student clicks on START, the solid partition dissolves to a broken line, motion begins, and a counter appears on the monitor's screen that records the number of fish in compartment A and the number of fish in compartment B. These numbers do not disappear from the screen but are recorded automatically at x second intervals in a table. A second labeled counter setup that might look like two speedometers is also on the screen. These record the number of fish that cross the barrier in each direction per minute. These numbers are updated at the same intervals as the numbers of fish in each compartment are recorded.)

While the fish are moving, you can click on PAUSE to temporarily freeze the motion if you want to take a closer look at the system. Try this and count the number of fish in each compartment and the number of fish that are crossing the barrier in each direction. To return to the dynamic state, click on CONTINUE.

(The changing of numbers continues until the equilibrium state is reached.)

Notice that the numbers being recorded are no longer changing although the fish continue to move. Since the numbers are no longer changing, the system has reached a state of DYNAMIC EQUILIBRIUM. At equilibrium, the number of fish in each compartment is constant; it no longer changes even though motion continues. Look at the speedometers. The number of fish moving across the barrier in the FORWARD direction is the same as the number of fish moving across the barrier in the REVERSE direction.

Let's see if you can answer some questions about the conditions that exist in a system that is at a state of DYNAMIC EQUILIBRIUM. Be sure that the system is dynamic. If you have put the system on PAUSE, click on RESUME to return to motion.

Are the particles of this system in motion at equilibrium? (In our model the particles are the fish.) Click on YES or NO.

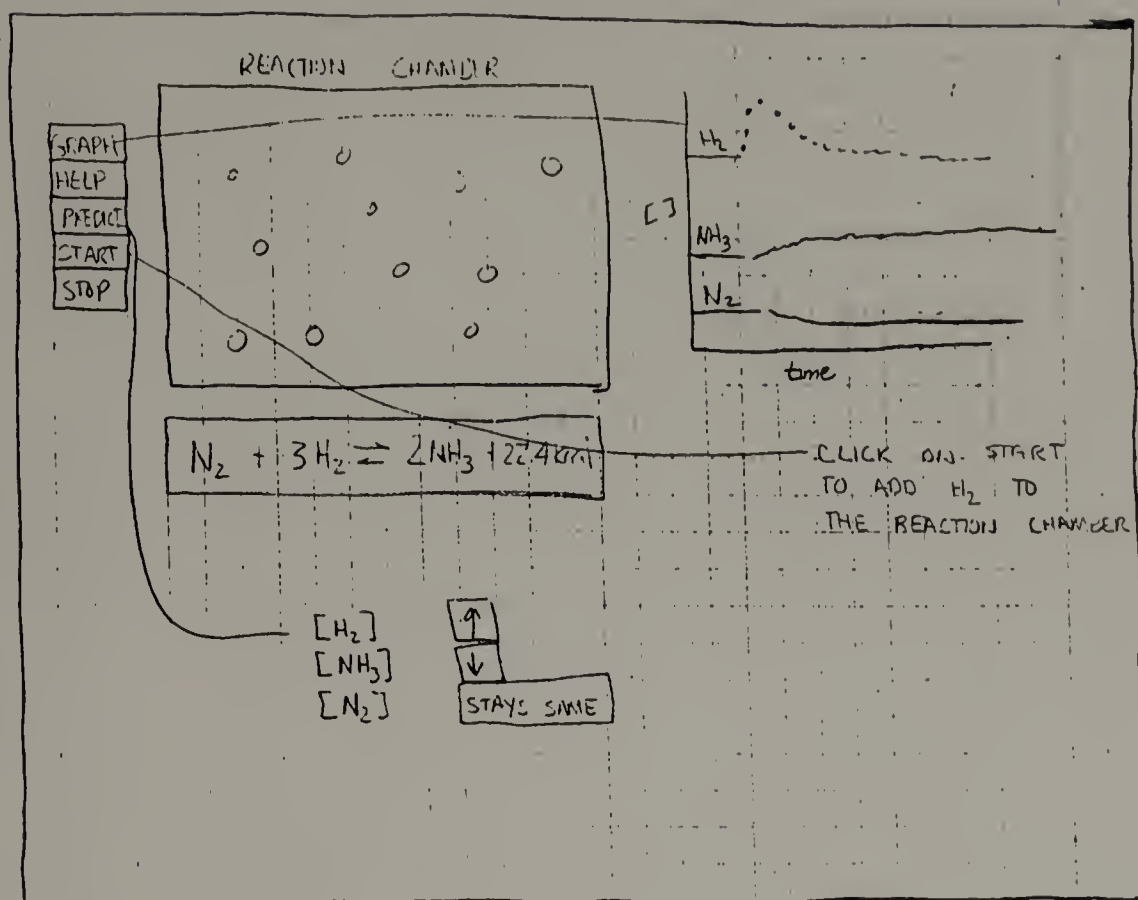
(If the student clicks on NO then ---)
Look again! Those fish are moving!

(If the student clicks on YES or if the student has clicked on NO and gone through the NO text then ---)
When a system is in a state of DYNAMIC EQUILIBRIUM, the particles are always moving.

Are the number of particles in each chamber changing at equilibrium? Click on YES or NO.

Student Actions for the Equilibrium Simulation

Design for a Chemistry Tutor (Cont)



SUMMARY DIALOG SHOULD INCLUDE WORK STRESS ASSOCIATED WITH CHANGE IN $[H_2]$

Screen from Chemistry Tutor

Design for a Chemistry Tutor (Cont)

Click on CONTINUE to advance to the next screen.

We have observed a model of a dynamic system that reached a state of equilibrium from a variety of starting conditions. This model has shown that systems at equilibrium exhibit certain characteristics. Some of these characteristics are:

1. The equilibrium state is dynamic not static. At equilibrium something is still happening. Our model showed that at equilibrium the fish continued to move.
2. The equilibrium state can be approached from a variety of starting conditions. Our model showed that the system proceeded to equilibrium regardless of the conditions within the tank initially.
3. Equilibrium can be approached from opposite directions. Equilibrium was achieved within the tank regardless of whether all the fish were initially in compartment A or compartment B.
4. The system spontaneously proceeds to equilibrium. Once started, our model showed that the system proceeded to equilibrium on its own.
5. At equilibrium reversible opposing motions are occurring at equal rates. Our model showed that at equilibrium the rate of the forward motion and the rate of the reverse motion were the same.
6. All components in a closed system are in contact and eventually reach a state of equilibrium. In our systems you observed that all fish were moving throughout the tank in both directions and eventually reached a state of equilibrium.
7. At equilibrium, the concentration of the substances in the system is constant. When you looked at the fish model at equilibrium, you saw that the number of fish in each compartment of the tank remained the same.

Click on CONTINUE to advance to the next screen. If you would like to repeat this introduction, click on RESTART.

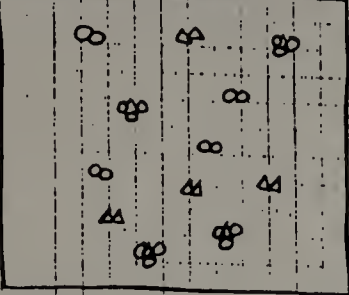
TRANSITION TO DEMONSTRATION IN WHICH STUDENT WILL MANIPULATE VARIABLES THEREBY APPLYING STRESS TO SYSTEMS AT EQUILIBRIUM AND NOTE THE CONSEQUENCES.

(A screen appears in which three different types of balls are randomly moving and colliding inside a chamber. The

Discussion about the Chemistry Tutor

Design for a Chemistry Tutor (Cont)

REACTION CHAMBER



CONCENTRATION
PRESSURE
TEMPERATURE
CATALYST

YOU CAN AFFECT A DYNAMIC EQUILIBRIUM BY CHANGING CONCENTRATION, PRESSURE, TEMPERATURE.

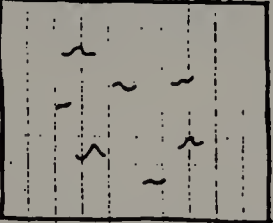
CLICK ON ONE OF THESE TO EXPLORE ITS EFFECT.

TO EXPLORE THE EFFECT OF CONCENTRATION YOU MUST CHANGE EITHER [REACTANT] OR [PRODUCT].

CLICK ON THE ONE YOU WISH TO CHANGE.

$$\text{N}_2(\text{g}) + 3\text{H}_2(\text{g}) \rightleftharpoons 2\text{NH}_3(\text{g}) + 22.4\text{kcal}$$

REACTION CHAMBER



CLICK ON THE REACTANT WHOSE CONCENTRATION YOU WILL CHANGE.

YOU HAVE SELECTED HYDROGEN, CLICK ON OK TO CONTINUE.

DO YOU WANT TO ↑ OR ↓ THE [H₂]?
CLICK ON YOUR CHOICE.

CLICK ON OK TO CONTINUE.

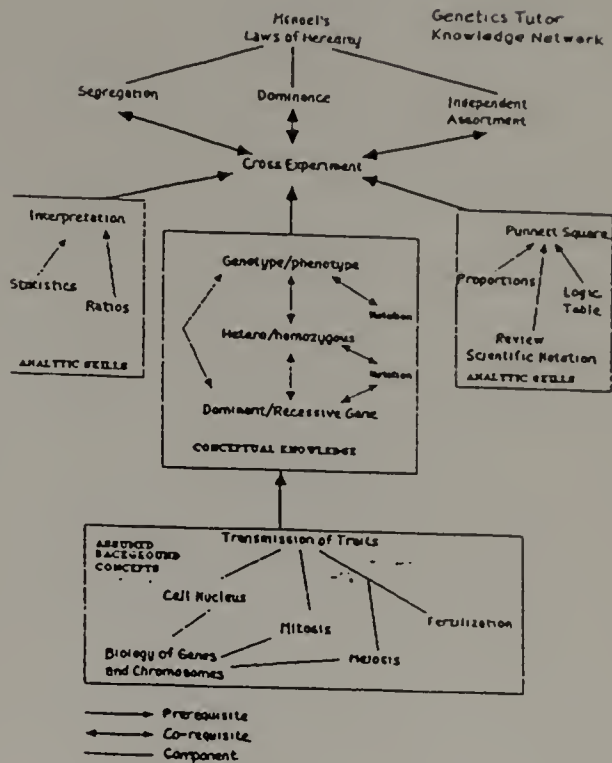
$$\text{N}_2 + 3\text{H}_2 \rightleftharpoons 2\text{NH}_3 + 22.4\text{kcal}$$

Additional Screens from the Chemistry Tutor

Example 2: Design for a Genetics Tutor
 by Cassandra Bacote-Capiga, Susan Haines, and Ralph Morelli
 Funded by the National Science Foundation
 Summer, 1988, University of Massachusetts

GENETICS
 KNOWLEDGE BASE DESIGN
 VERSION 1

Cassandra Bacote-Capiga
 Susan S. Haines
 Ralph Morelli



Topics covered in the Genetics Simulation

PRESENTATION OUTLINES FOR KNOWLEDGE UNITS

1. Dominant/Recessive KU
2. Hetero/Homozygous KU
3. Genotype/Phenotype KU
4. Punnett Square

Knowledge Engineering
 for the Genetics
 Simulation (an outline)

Design for a Genetics Tutor (Cont)

KU: 1 - DOMINANT/RECESSIVE

VOCABULARY/TERMINOLOGY

The Dominant gene manifests itself. The recessive gene is masked when combined with a dominant gene. For example, in terms of specific traits: Tall peas, brown eyes, etc.

- Possible Question/Problems:
- The gene that manifests itself is called _____.
 - Something has a dominant and a recessive gene, what does it look like?
 - A creature's mother has brown X and his father has blue X. When brown X is dominant.....etc.

NOTATION

Capital letters are used to represent dominant genes and lower case for recessive. The dominant trait is always written first.

- Examples: An individual with both genes dominant for tallness would be written TT.
An individual with both genes recessive for tallness would be written tt.

Possible Problem: How would you represent an individual with one dominant and one recessive gene for tallness?

If student answers tT, see misconception #2.

APPLICATION

Explanation: How the concepts are used; why?
-to predict offspring
-to predict genotype (genetic makeup of offspring)

Examples: Tt(mother) tt(father)
 Tt (baby)

Problem: A baby inherits the following genes from its parents when T stands for Tallness. Would the baby be tall?

Knowledge Unit
for Teaching Concepts
"Dominant and Recessive"

KU: 2 - HETEROZYGOUS/HOMOZYGOUS

VOCABULARY/TERMINOLOGY

Tell what hetero and homozygous mean. An individual that has two like genes for a trait is homozygous. An individual that has one dominant and one recessive gene for a trait is heterozygous.

Also introduce pure and hybrid as synonyms.

Examples: Tall Tall, short short, Tall short.

Problem/Question: Slot machine example. Pull lever, state whether resulting combination is hetero or homozygous (pure or hybrid).

NOTATION

Description: homozygous - tt or TT
heterozygous - Tt

Examples: As above

Questions: If tallness is dominant, how do you represent a heterozygous X?
(Answer: Tt)

APPLICATION

Description: Same as dominant/recessive with new terminology.

Examples -

Question: Same problem as dominant/recessive but using new vocabulary.

Knowledge Unit
for Teaching the Concepts
"Heterozygous and Homozygous"

Design for a Genetics Tutor (Cont)

KU: 3 - GENOTYPE/PHENOTYPE

VOCABULARY/TERMINOLOGY

Explain meaning of terms: phenotype - appearance, etc.
genotype - gene combination

Examples: Pictures of individuals with visible genetic characteristics and descriptions of the genotype and phenotype of each. Compare and contrast terms.

Problems: Use pictures used in examples.

NOTATION

Explanation: Genotype represented by genetic symbols, i.e. Tt or TtBB

Examples: See above

Problems: Give a representation.....
Describe the genotype of TtBB.
Describe the phenotype of TtBB.

APPLICATION

Explanation:

Examples:

Problems: Same as dominant/recessive and hetero/homozygous. Add new factor introduced here.

Knowledge Unit for Teaching the Concepts
Genotype and Phenotype

KU: 4 - PUNNETT SQUARE

VOCABULARY/DESCRIPTION

Explanation: The punnett square is a chart/tool used to demonstrate and analyze genetic crosses between parents. It shows the possible genotypes of the offspring and proportions.

Example: (Show this example in words and symbols on the chart and include interpretation.)

TT x Tt = 3/4 Tt, 1/4 Tt

	T	t
T	TT	Tt
t	Tt	tt

Problems: As above.

NOTATIONAPPLICATION

Describe how to use the punnett and the purpose for which it is used.

Examples:

Stock of problems that can be used as both examples and problems.

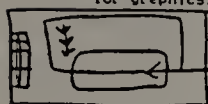
Problems:

Knowledge Unit for Teaching How to
Use a Punnett Square

Design for a Genetics Tutor (Cont)

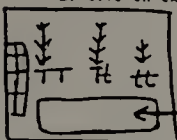
Procedural Presentation on Notation - Dominant/Recessive

1. Explain the notation using most current example for graphics.



TEXT: Scientists use symbols to represent dominant and recessive traits: Capital for dominant, lower case for recessive. The dominant symbol is always written first.

2. Give an example.



TEXT: Here are some examples. The first plant has two dominant genes for tallness, represented by TT. The second plant..... The third plant.....

3. Give a problem.



TEXT: How would we represent the genetic makeup (genotype) for the guinea pig which has one dominant and one recessive gene (hybrid)?

Here's a table you can use:

gp	brown	2
pea	tall	1



Wrong-----?-----Correct



Summary of Presentation for Dominant/Recessive Notation:

1. Explain using text for misconception Presentation: Order Doesn't Matter
2. Give Example
3. Give Problem/Question: D-R-N #1

Various Screens for Teaching About Dominant and Recessive Genes

Dominant/Recessive - Presentations #1

Declarative Presentation

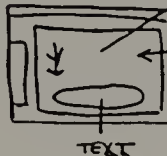
1. Give a definition



HYPERTEXT
The dominant gene manifests itself. The recessive gene is masked...

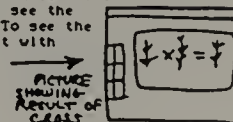
2. Give an example from a collection of appropriate examples.

PICTURE

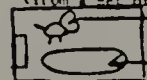


Text: For example, in pea plants tallness is the dominant and shortness is the recessive.

Here is a tall pea which contains both genes. You can't see the recessive character. To see the you'd have to cross it with another.....



3. Ask a question that probes whether they understand the vocabulary. (from a set of examples/problems)

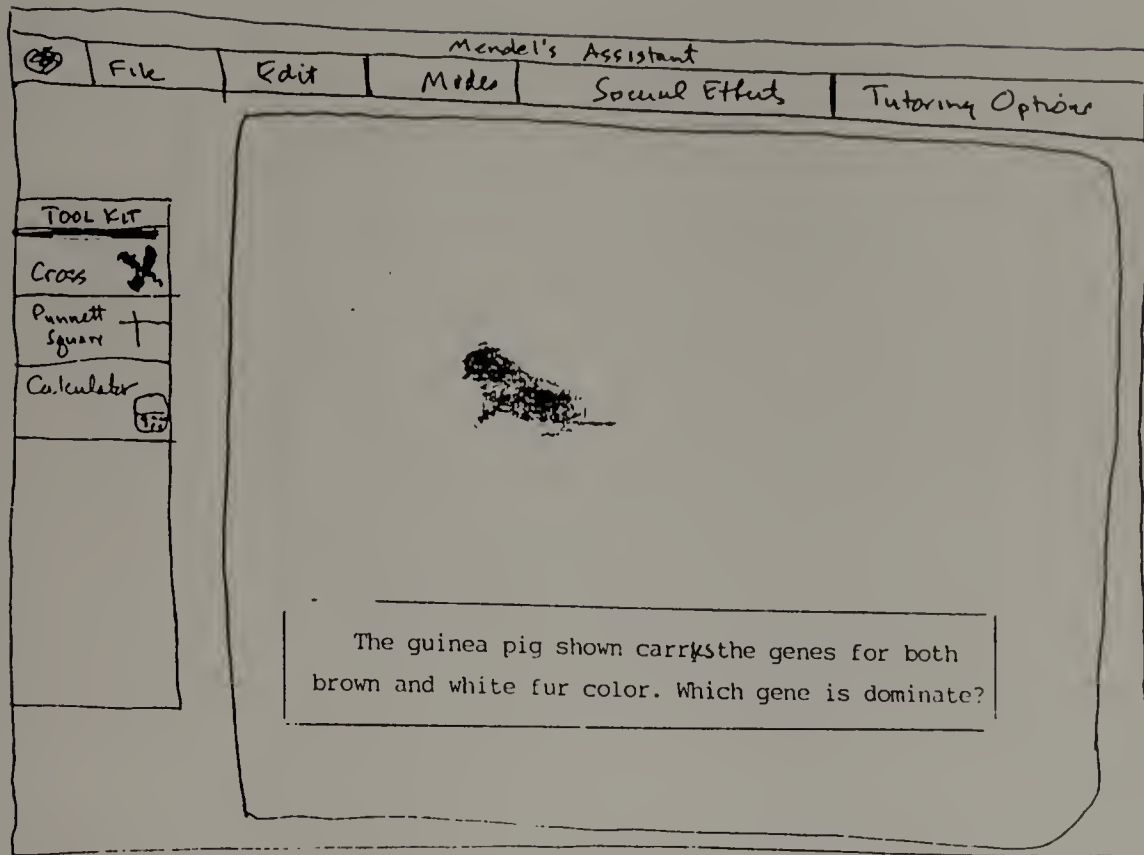


Text: The guinea pigs shown carry the genes for both brown and white fur color. Which gene is dominant?

- ?Correct
 - Continue with notation
 - Elaborate
 - Congratulate & Encourage
 - Follow up question
 - Review & Summation
- ?Wrong
 - Start over
 - Give a Hint
 - Give the correct explanation
 - Give another try
 - Probe for misconception

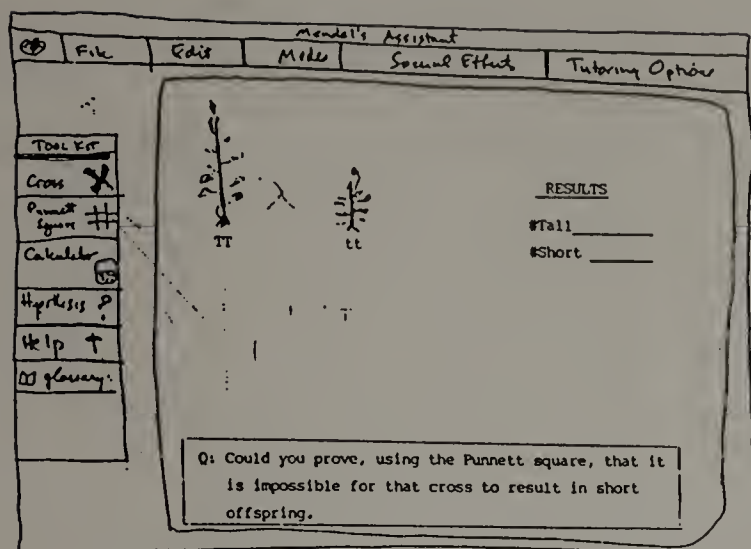
Screens for Presenting a Definition, Example, or Question

Design for a Genetics Tutor (Cont)

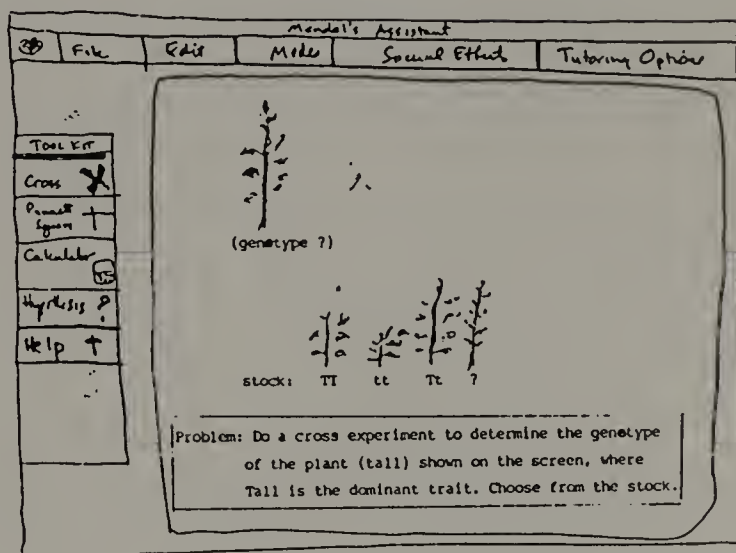


A Simple Question Presented by the Genetics Program

Design for a Genetics Tutor (Cont)

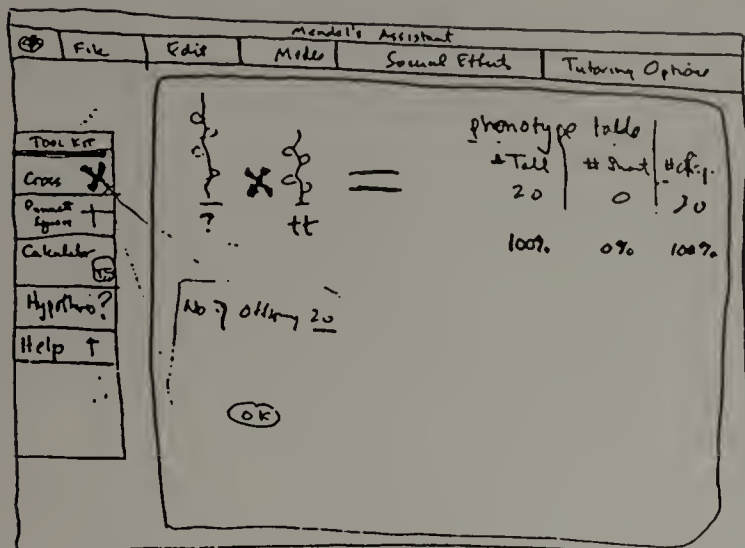


Problem Presented to Student Requesting the use of a Punnett Square

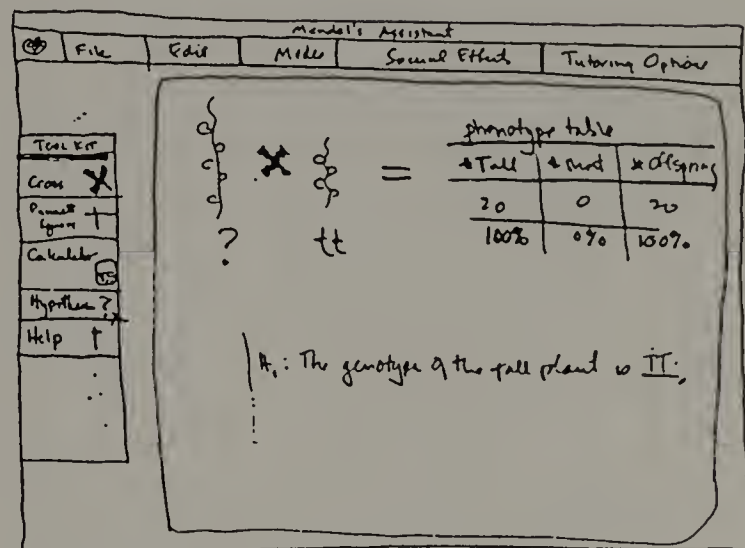


Task Given to Student Which Requires an Experiment

Design for a Genetics Tutor (Cont)



Steps in Solving the Problem
on the Previous Page



Steps in Solving the Problem
on the Previous Page

BIBLIOGRAPHY

- Ambrosio, S., & Hooper, K., *Interactive Multimedia*, Microsoft Press, Redmond, WA, 1988.
- Anderson, J. R., Boyle, C., & Yost, G., "The geometry tutor," *Proceedings of the International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, Palo Alto, CA, pp. 1-7, 1985.
- Anderson, J. R., & Reiser, B. J., "The LISP tutor," *Byte*, Vol. 10, No. 4, pp. 159-175, 1985.
- Anderson, J. R., Farrell, R. G., & Savers, R., "Learning to program in LISP," *Cognitive Science*, Vol. 8, No. 2, pp. 87-129, 1984.
- Anderson, J., "Tuning of search of the problem space for geometry proofs," *International Joint Conference on Artificial Intelligence*, British Columbia, 1981.
- Apple Corp., *Inside Macintosh*, Vol. 1, Addison-Wesley Publishing Co., 1985.
- Arbib, M., *The Metaphorical Brain*, John Wiley and Sons, Inc., 1972.
- Arons, A., "Computer-based instructional dialogs in science courses," *Science*, Vol. 224, pg. 1051, American Association for the Advancement of Science, Washington, DC, 1984.
- Atkins, T., *The Second Law*, Freedman Publishers, San Francisco, CA, Scientific American Series, 1982.

Austin, H., *A Computational View of Physical Skill*, Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1976.

Barr, A., & Feigenbaum, E., *The Handbook of Artificial Intelligence, Vol. 2*, William Kaufmann Publishers, Palo Alto, CA, 1981.

Barr, A., Beard, M., & Atkinson, R. C., "The computer as a tutorial laboratory: the Stanford BIP Project," *International Journal Of Man-Machine Studies*, 8, 1976.

Bete, C., *Education 2000+*, Channing L. Bete Co., Inc., Deerfield, MA, 1969.

Birnbaum, J., *The Domestication of Computers*, from an address delivered at Columbia Computer Science Building Convocation, Columbia University, New York, NY, Oct. 10, 1983.

Bitzer, Braunfield, D., & Lichtenberger, W., "PLATO: An Automatic Teaching Device," *IRE Trans. Education*, pp. 157-161, Dec. 1961.

Blau, L., Woolf, B., & Slovin, T., "Joining with the client in a consultant tutor," to be published as a COINS Technical Report, Computer and Information Science Department, University of Massachusetts, Amherst, MA.

Bloom, B., "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring," *Educational Researcher*, June/July, 1984.

Bobrow, D., Mittal, S., & Stefik, M., "Expert systems: Perils and promise," *Comm. ACM*, Sept. 1986.

Bonar, J., *Understanding the Bugs of Novice Programmers*, Ph.D. Thesis, Computer and Information Science Department, University of Massachusetts, Amherst, MA, 1985.

-----, "Collecting and analyzing online protocols from novice programmers," *Behavioral Research Methods and Instrumentation*, 1982.

Bonar, J., Cunningham, R., & Schultz, J., "An object-oriented architecture for intelligent tutoring," *Proceedings of the ACM Conference on Object-Oriented Programming Systems, Language and Applications*, Association for Computing Machinery, New York, NY, 1986.

Bonar, J., & Soloway, E., "Pre-programming knowledge: A major source of misconceptions in novice programmers," *Proceedings of the Conference on Human Computer Interactions*, Vol. 1, No. 2, pp. 133-161, 1985.

Bork, A., "The potential for interactive technology," *BYTE*, pp. 201-204, Feb. 1987.

-----, *Personal Computers for Education*, Harper & Row, Publishers, New York, NY, 1985.

-----, "Physics in the Irvine Educational Technology Center," *Computers in Education*, 4, pp. 37-57, 1980.

Bower, G. H., Black, J., & Turner, T., "Scripts in text comprehension and memory," *Cognitive Psychology*, Vol. 1, pp. 177-220, 1979.

Brand, S., *The Media Lab*, Viking Penguin, New York, NY, 1987.

Brewer, B., "Great Expectations," in *Hypermedia: The Guide to Interactive Media Production*, special publication of *Mix-The Recording Industry Magazine*, Mix Publications, Emeryville, CA, 1988.

Brown, D. E., & Clement, J., "Overcoming misconceptions in mechanics: a comparison of two example-based teaching strategies," in *Proceedings of the Second Seminar on Misconceptions and Education Strategy in Science and Mathematics*, Vol. III, pp. 39-53, Cornell University, Ithaca, NY, 1988.

Brown, D., & Chandrasekaran, B., "Knowledge and control for a mechanical design expert system," *IEEE Computer*, Vol 19, #7, Los Alamitos, CA, July, 1986.

Brown, J. S., Burton, R. R., & deKleer, J., "Pedagogical natural language and knowledge engineering techniques in SOPHIE I, II and III," in D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London, pp. 227-282, 1982.

Bruno, G., Elia, A., & Laface, P., "A rule-based system to schedule production," *IEEE Computer*, Vol. 19, #7, Los Alamitos, CA, 1986.

Burton, R., "The environment module of intelligent tutoring systems," in M. Polson & J. J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems*, Lawrence Erlbaum Associates Publishers, Hillsdale, NJ.

-----, "Diagnosing bugs in a simple procedural skill," in D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London, England, 1982.

Burton, R. & Brown, J. S., "An investigation of computer coaching for informal learning activities," in D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London, 1982.

Callahan, J., Cox, D., Hoffman, K., O'Shea, D., Senechal, L., & Wattenberg, F., "Calculus in context," a National Science Foundation Proposal, 1988.

Carbrera, B., "Physics simulations: Teaching aids for elementary physics instruction," *Academic Computing*, Spring, 1987.

Carr, B., & Goldstein, I. P., "Overlays: A theory of modeling for computer-aided instruction," *AI Lab Memo*, Massachusetts Institute of Technology, Cambridge, MA, 1977.

CasaBianca, L., "Publishers' Notes," in *Hypermedia: The Guide to Interactive Media Production*, special publication of *Mix-The Recording Industry Magazine*, Mix Publications, Emeryville, CA, 1988a.

-----, "Ted Nelson Hypermedia Magician," in *Hypermedia: The Guide to Interactive Media Production*, special publication of *Mix-The Recording Industry Magazine*, Mix Publications, Emeryville, CA, 1988b.

Castro, J., "Staying home is paying off," *Time*, Oct. 26, 1987.

Chase, W., & Simon, H., "Perception in chess," *Cognitive Psychology*, Vol. 4, pp. 55-81, 1981.

Chase, W., & Simon, H., "The mind's eye in chess," in W. G. Chase (Ed.), *Visual Information Processing*, Academic Press, New York, NY, 1973.

Chomsky, N., *Language and Mind*, Harcourt, Brace, Jovanowich, New York, NY, 1968.

Clancey, W., "Assessment of designs in progress," from a talk delivered to the ESE consortium Meeting on Knowledge Engineering and System Architecture, Hawaii, November, 1987.

-----, *Knowledge-based Tutoring: The GUIDON Program*, MIT Press, Cambridge, MA, 1987.

-----, "Qualitative student models," in J. F. Traub (Ed.), *Annual Reviews of Computer Science*, Vol. 1, pp. 381-450, Annual Reviews, Inc., Palo Alto, CA, 1986a.

-----, "From Guidon to Neomycin and Heracles in 20 short lessons: ONR final report, 1979-1985," *The AI Magazine*, Vol. 7, No. 3, pp. 40-60, August, 1986b.

-----, "Classification problem solving," *Proceedings of the National Conference on Artificial Intelligence*, Austin, TX, pp. 49-55, 1984.

-----, "Tutoring rules for guiding a case method dialogue," in *International Journal of Man-Machine Studies*, Vol. 11, pp. 25-49. Reprinted in D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London, England, 1982.

-----, *Transfer of Rule-Based Expertise Through Tutorial Dialogue*, Ph.D. Dissertation, Department of Computer Science, Stanford University, Stanford, CA, 1979a.

-----, "Dialogue management for rule-based tutorials," *Proceedings of the International Joint Conference on Artificial Intelligence*, 1979b.

Clancey, W., & Letsinger, R., "Neomycin: Reconfiguring a rule-based expert system for application to teaching," *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, Canada, pp. 829-835, 1981.

Clarendon, A., *New Technology at Work*, Oxford University Press, New York, NY, 1986.

Clark, A., *Profiles of the Future*, Holt, Rinehart and Winston, New York, NY, 1962.

Claxton, G., "Teaching and acquiring scientific knowledge," in Pope & Keen (Eds.), *Kelly in the Classroom: Educational Applications of Personal Construct Theory*, Cybersystems, Inc., 1985.

Clement, J., "A conceptual model discussed by Galileo and used intuitively by physics students," in D. Genter & A. L. Stevens (Eds.), *Mental Models*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

Clement, J., & Brown, D., "Using analogical reasoning to deal with deep misconceptions in physics," Cognitive Processes Research Group, Physics Department, University of Massachusetts, May, 1984.

Clement, J., "Student's preconceptions in introductory mechanics," *American Journal of Physics*, 50 (1), January 1982.

Collins, A., Warnock, E., & Passafiume, J., "Analysis and synthesis of tutorial dialogues," in G. Bower (Ed.), *The Psychology of Learning and Motivation*, Vol. 9, Academic Press, Inc., 1975.

Collins, A., "Teaching reasoning skills," in S. F. Chipman, J. W. Segal, & R. Glaser (Eds.), *Thinking and Learning Skills: Research and Open Questions*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

Copeland, W. D., "Creating a historian's micro-world," *Classrooms Computer Learning*, pp. 49-53, Oct. 1984.

Cunningham, P., *Caleb: A Silent Second Language Tutor: The Knowledge Acquisition Phase*, Master's thesis, Rensselaer Polytechnic Institute, Hartford Graduate Center, Troy, NY, 1986.

Cunningham, P., Iberall, T., & Woolf, B., "Caleb: An intelligent second language tutor," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Atlanta, GA., pp. 1210-1215, 1986.

Dede, C., "Probable evolution of artificial intelligence based education devices," *Technological Forecasting and Social Change*, 34, pp. 111-133, 1988.

diSessa, A. A., "Knowledge in pieces," Address to the fifteenth Annual Symposium of the Jean Piaget Society, "Constructivism in the Computer Age," Philadelphia, PA, June, 1985.

Duckworth, E., Kelly, J., & Wilson, S., "AI goes to school," *Academic Computing*, pp. 6-10, pp. 38-43, pp. 62-63, Nov., 1987.

Dym, C., "Expert systems: New approaches to computer-aided engineering," *Xerox PARC Intelligent Systems Laboratory Series ISL-84-2*, April 1984.

Evans, R., *Jean Piaget: The man and his Ideas*, E.P. Dutton and Co., New York, NY, 1973.

Eylon, B. S., & Reif, F., "Effect of knowledge organization on task performance," *Cognition and Instruction*, 1, 1984.

Feigenbaum, E., address at the "Applications of Artificial Intelligence Conference," Stanford University, March 29, 1989.

Fischer, G., "A Critic for LISP," *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milan, Italy, Morgan Kaufmann Publishers, Palo Alto, CA. pp 177-184, 1987.

Finin, T., "Providing help and advice in task-oriented systems," *Proceedings of the International Joint Conference on Artificial Intelligence*, Karlsruhe, W. Germany, 1983.

Fiske, B., "Lessons," in *The New York Times*, Education Section, pg. B10, August 16, 1989.

Forbus, K., & Stevens, A., "Using qualitative simulation to generate explanations," Report #4490, Bolt, Beranek and Newman, Inc., 1981; also in *Proceedings of Third Annual Conference of the Cognitive Science Society*, August, 1981.

Freire, P., *Pedagogy of the Oppressed*, Translated by Myra Ramos, Continuum Publishing Corp., New York, NY, 1982.

Fuller, R. B., *Education Automation: Freeing the Scholar to Return to his Studies*, Southern Illinois University Press, Carbondale, Ill, 1962.

Gardner, J., *No Easy Victories*, in Rowan, H. (Ed.), Harper & Row, New York, NY, 1968.

Gardner, H., *Frames of Mind: The Theory of Multiple Intelligences*, Basic Books, Inc., New York, NY, 1983.

Gerhart, S., "Knowledge about programs: A model and case study," *Proceedings of the Conference on Reliable Software*, SIGPLAN Notes, Vol. 10, No. 6, 1975.

Glasser, W., *Schools Without Failure*, Harper and Row, New York, NY, 1969.

Goldstein, I. P., "The genetic graph: A representation for the evolution of procedural knowledge," in D. H. Sleeman, & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London, England, 1983.

Goldstein, I. P., "The computer as coach: An athletic paradigm for intellectual education," AI Memo #389, Massachusetts of Technology, Cambridge, MA, 1977.

Gonzales, A. & Zimbardo, P., "Time in perspective," *Psychology Today*, pp. 21-26, March, 1985.

Goodman, P., *Summerhill For and Against*, Hart Publishing, New York, NY, 1970.

Gulis, P., "Rochester asks teachers for 'extra mile'," *The New York Times*, Feb. 18, 1988.

Half, H., "Curriculum and instruction in automated tutors," in M. Polson & J. Richardson, (Eds.), *Foundations of Intelligent Tutoring Systems*, Lawrence Erlbaum and Assoc., 1988.

Harmon, P., *Expert Systems Strategies*, Cutter Information Corp., Arlington, MA, Vol. 3, No. 6, 1987.

Haugeland, J. (Ed.), *Mind Design*, M.I.T. Press, Cambridge, MA, 1981.

Heller, J. I., & Reif, F., "Prescribing effective human problem-solving processes: problem description in physics," *Cognition and Instincts*, 1, 1984.

Hofstadter, D., *Godel, Esher, Bach: An Eternal Golden Braid*, Basic Books, New York, NY, 1978.

Hollan, J., Hutchins, E., & Weitzman, L., "STEAMER: An interactive inspectable simulation-based training system," *The AI Magazine*, Vol. 5, No. 2, pp. 15-27, Summer, 1984.

Holt, J., *How Children Fail*, Pittman Publishers, New York, NY, 1964.

Illich, I., *Deschooling Society*, Harper & Row, New York, NY, 1971.

Inholder, B., "Memory and intelligence in the child," in Etkind & Flavel (Eds.), *Studies in Cognitive Development*, Oxford Press, New York, NY, 1969.

Johnson, L., & Soloway, E. M., "PROUST: An automatic debugger for Pascal programs," *BYTE*, Vol. 10, No. 4, pp. 170-190, 1985.

Johnson, L., & Soloway, E. M., "PROUST: Knowledge-based program debugging," *Proceedings of the International Software Engineering Conference*, Orlando, FL, pp. 369-380, 1984a.

Johnson, L., & Soloway, E. M., "Intention-based diagnosis of programming errors," *Proceedings of the National Conference on Artificial Intelligence*, Austin, TX, pp. 162-168, 1984b.

Kass, R., & Finin, T., "Rules for the Implicit Acquisition of Knowledge about the User," *Proceedings of the National Association of Artificial Intelligence, AAAI-87*, Seattle, WA, 1987.

Kearsley, G. (Ed.), *Artificial Intelligence & Instruction*, Addison-Wesley Publishing Co., Reading, MA, 1987.

Kearsley, G., "Intelligent CAI: Today and tomorrow," *ACM 23rd Annual Technical Symposium*, 1984.

Kozol, J., *Death at an early age*, Bantam books, Inc., New York, NY, 1967.

-----, *The Night is Dark and We are A Long Way From Home*, Houghton Mifflin Co., Boston, MA 1975.

-----, *Illiterate America*, Anchor/Double Day, 1985.

Krendl, K. A., & Fredin, E. S., "The effects of instructional design characteristics: An examination of two communication systems," *Journal of Educational Technology Systems*, 14 (1), pp. 75-86, 1985.

Kuhn, T. S., *The Structure of Scientific Revolutions*, International Encyclopedia of Unified Science, University of Chicago Press, 1970.

Kurland, D., & Kurland, C., "Computer applications in education: A historical overview," *Annual Review of Computer Science*, pp. 317-358, 1987.

Larkin, J., McDermot, J., Simon, D. P., & Simon, H., "Expert and novice performance in solving physics problems," *Science*, Vol. 208, pp. 1335-1342, 1980.

Larkin, J., "Enriching formal knowledge: A model for learning to solve textbook physics problems," in J. R. Anderson (Ed.), *Cognitive Skills and Their Acquisition*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1982.

Lave, J., *Cognition in Practice*, Cambridge University Press, Cambridge, G.B., 1988.

Lawler, R., & Yazdani, M. (Eds.), *Artificial Intelligence and Education, Volume One*, Ablex Publishing, Norwood, NJ, 1987.

Lesgold, A., & Reif, F., *Computers in Education: Realizing the Potential*, Report of a Research Conference, U.S. Department of Education, 1983.

Lindsay, R. K., Buchanan, B., Feigenbaum, E., & Lederberg, J., *Applications of Artificial Intelligence for Organic Chemistry*, Kaufmann Publishers, Palo Alto CA, 1985

Lochhead, J., & Clement, J. (Eds.), *Cognitive Process Instruction*, Franklin Institute Press, Philadelphia, PA 1978.

McDermott, L., "Misconceptions in classical mechanics," *Physics Today*, July, 1984.

McDermott, J., "A rule-based configurer of computer systems," *Artificial Intelligence*, 19,1, Jan. 1982.

McLuhan, M., *Understanding media: the extensions of man*, McGraw-Hill, New York, NY, 1965.

Miller, M., "A structured planning and debugging environment for elementary programming," *International Journal of Man-Machine Studies*, 11, 1979, also in D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, Cambridge, MA, 1982.

Minsky, M., "Why people think machines don't," *Technology Review*, Massachusetts Institute of Technology, Cambridge, MA, 1983.

Mitre Corporation, "Computers in the instructional process: Report on an international school," *Extend Publications*, Ann Arbor, MI, 1972.

Mittal, S., Dym, C., & Morjaria, M., "Pride: An expert system for the design of paper handling systems," *IEEE Computer*, Vol. 19, No. 7, July, 1986.

Mittal, S., & Dym, C., "Knowledge acquisition from multiple experts," *AI Magazine*, 6(2), Summer 1985.

Murray, T., Schultz, K., Brown, D., & Clement, J., "An analogy-based computer tutor for remediating physics misconceptions," *Interactive Learning Environments*, Ablex Publishing Corporation, Norwood, NJ, in press.

Naisbitt, J., *Megatrends: Ten new directions transforming our lives*, Warner Books, New York, NY, 1984.

Naisbitt, J., & Aburdene, P., *Re-inventing the Corporation*, Warner Books, Inc., New York, NY, 1985.

The New York Times, Advertisement in the Science Times, Feb. 10, 1988.

Novak, G. S., "Representation of knowledge in a program for solving physics problems," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1977.

Papert, S., *Mindstorms: Children, Computers, and Powerful Ideas*, Harper Colophon Basic Books, 1980.

Parkes, A., & Self, J., "Video-based intelligent tutoring of procedural skills," in *Proceedings of the International Conference on Intelligent Tutoring Systems (ITS-88)*, Montreal, Canada, 1988.

Peelle, H., & Riseman, E., "The four faces of Hal: A framework for using Artificial Intelligence techniques in computer-assisted instruction," *IEEE Transactions on Systems, Man, and Cybernetics*, May, 1975.

Perelman, L., *Technology and Transformation of Schools*, National School Board Association, Alexandria, VA, 1987.

Piaget, J., *Genetic Epistemology*, W. W. Norton Press, 1971.

Piaget, J., & Inhelder, B., *Origins of Intelligence in Children*, Translated by Cook, International University Press, Norton, 1963.

Plato, *Laches, Protagoras, Meno, and Euthydemus*, W. R. Lamb, translator, Harvard University Press, Cambridge, MA, 1922.

Pollack, A., "Computer programs as university teachers," *The New York Times*, Dec. 7, 1987.

Polson, M., & Richardson, J., *Foundations of Intelligent Tutoring Systems*, Lawrence Erlbaum Assoc., Publishers, NJ, 1988.

Polya, G., *How to Solve It*, Doubleday Anchor Books, New York, NY, 1957.

Pope, C., "Education and emotionally disturbed children: A critique of behaviorist trends and an examination of a constructivist alternative," EdD. Dissertation, School of Education, University of Massachusetts, Amherst, MA, 1982.

Postman, N., & Weingartner, C., *Teaching as a Subversive Activity*, Delacorte Press, New York, NY, 1969.

Psofka, J. (Ed.), *Intelligent Tutoring Systems: Lessons Learned*, Lawrence Erlbaum and Assoc., NJ, 1988.

Quinn Patton, M., "Instructional information systems: dream or nightmare?" In A. Bank & R. C. Williams (Eds.), *Information Systems and School Improvement: Inventing the Future*, Teachers College Press, New York, NY, 1987.

Rappleyea, A., "Progress report on the CRANE BOOM module," unpublished document, Department of Physics, City College of San Francisco, San Francisco, CA, 1987.

Rissland, E., "Understanding understanding mathematics," *Cognitive Science*, Vol. 2, No. 4, 1978.

Rissland, E., & Schultz, K., "Knowledge engineering for building an intelligent tutor," prepared for the 4th ESE Consortium Meeting, Coronata Hotel, San Diego, CA. Unpublished document, Computer and Information Science Department, University of Massachusetts, 1987.

Rissland, E. & Soloway, E., "Constrained example generation: A testbed for studying learning," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, 1981.

Sauers R., & Farrell, R., GRAPES User's Manual., Technical Report, Office of Naval Research 82-3, Pittsburgh, Carnegie-Mellon University, Pittsburgh, PA, 1982.

Schultz, K., "Description and evaluation of an inservice model for implementation of a learning cycle approach in the secondary science classroom," *Science Education*, Vol. 69, No. 4, 491-500, 1985.

Schultz, K., "Recruiting talent into public school teaching: An analysis from one successful pilot program," *Journal of Teacher Education*, Vol. XXXV, No. 6, 2-4, 1984.

Self, J. (Ed.), *Artificial Intelligence and Human Learning*, Chapman & Hall Computing, London, 1988.

Servan-Schreiber, D., "From intelligent tutoring to computerized psychotherapy," *Proceedings of the Sixth National Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, Palo Alto, CA, 1986.

Shank, R., & Childers, P., *The Cognitive Computer: On Language, Learning and Artificial Intelligence*. Addison-Wesley Publishing Co., Inc., Reading MA, 1984.

Shortliffe, E. H., *Computer-Based Medical Consultations: MYCIN*, American Elsevier Publishers, New York, NY, 1976.

Shrobe, H., & The American Association for Artificial Intelligence (Eds.), *Exploring Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., Palo Alto, CA, 1988.

Shute, V., & Bonar, J., "Intelligent tutoring systems for scientific inquiry," *Proceedings of the Cognitive Science Society*, Lawrence Erlbaum Associates, Publishers, NJ, 1986.

Sleeman, D., & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, Cambridge, MA, 1982.

Slovin, T., & Woolf, B., "A consultant tutor for personal development," *Proceedings of the International Conference on Intelligent Tutoring Systems (ITS 88)*, University of Montreal, Quebec, Canada, 1988.

Soloway, E., Bonar, J., Woolf, B., Barth, P., Rubin, E., & Erlich, K., "Cognition and programming: Why your students write those crazy programs," *Proceedings of the National Educational Computing Conference, NECC*, No. Denton, TX, 1981.

Soloway, E., Bonar, J., & Erlich, K., "Cognitive strategies and looping constructs: An empirical study," *Communications of the Association of Computing Machinery*, Vol. 26, No. 11, pp. 853-860, 1983.

Soloway, E., Woolf, B., Barth, P., & Rubin, E., "MENO-II: An intelligent tutoring system for novice programmers," *The Seventh International Joint Conference on Artificial Intelligence*, Vancouver, Canada, 1981.

Southworth, J., "Hawaii distance learning- -Technology project" *Computer Education and Children* published by the Association for the Development of Computer-Based Instructional Systems (ADCIS) and the Soviet Academy of Sciences, 1988.

Stansfield, J. C., Carr, B., & Goldstein, I. P., "Wumpus Advisor I, a first implementation of a program that tutors logical and probabilistic reasoning skills," AI Lab Memo 381, Massachusetts Institute of Technology, Cambridge, MA, 1976.

Steele, G., *Common Lisp*, Digital Press, Burlington, MA, 1984.

Stevens, A., Collins, A., & Goldin, S., "Misconceptions in student's understanding," *International Journal of Man-Machines Studies*, 11, pp. 145-156, 1978 and in D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, Cambridge, MA, 1982.

Streibel, M., Stewart, J., Koedinger, K., Collins, A., & Jungck, J., "Mendel: An intelligent computer tutoring systems for solving genetics problem solving, conjecturing, and understanding," *Machine-Mediated Learning*, Vol. 2, No. 1 and 2, pp. 129-159, 1987.

Sussman, G., "A computational model of skill acquisition," AI Lab Memo #297, Massachusetts Institute of Technology, Cambridge, MA, 1976.

- Suthers, D., "Perspectives in explanation," COINS Technical Report #89-24, Computer and Information Science, University of Massachusetts, Amherst, MA, 1989.
- Suthers, D., & Rissland, E., "Constraint manipulation for example generation," COINS Technical Report #88-71, University of Massachusetts, MA, 1988.
- Talukdar, S., Cardozo, E. & Leao, L., "Toast: The power system operator's assistant," *IEEE Computer*, Vol 19, #7, Los Alamitos, CA, July, 1986.
- Teltsch, K., "Business sees aid to schools as a net gain," *The New York Times*, pp. 1 & 50, Dec. 4, 1988.
- Tinker, R., "Network science arrives," *Hands ON!*, a newsletter produced by Technical Education Research Centers (TERC), Vol. 10, No. 1, Winter, 1987.
- Tobias S., "Peer perspectives on the teaching of science," *Change*, March/April, 1986.
- Toffler, A., *The Third Wave*, Morrow Publishers, New York, NY, 1980.
- Turkle, S., *The Second Self: Computers and the Human Spirit*, Simon & Shuster, New York, NY, 1984.
- van Lehn, K., "Felicity conditions for human skill acquisition: validating an AI-based theory," Tech Report #CIS-21, Xerox Palo Alto Research Center, Palo Alto, CA, 1983.
- , "Student modelling," in M. Polson & J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems*, Lawrence Erlbaum and Assoc., NJ, 1988.
- Wakefield, R., "Home computers and families: The empowerment revolution," *The Futurist*, 20:5, pp. 18-22, Sept.-Oct., 1986.
- Waterman, D., *Innovation: The Attacker's Advantage*, Simon & Schuster, New York, NY, 1986.

- Waters, R., "The Programmer's Apprentice: Knowledge-Based Program Editing," *IEEE Transaction on Software Engineering*, Vol. SE-8, No. 1, 1982.
- Weiner, N., *The Human Use of Human Beings: Cybernetics and Society*, Da Capo Press, New York, NY, 1985.
- Weizenbaum, J., *Computer Power and Human Reason*, W. H. Freeman, San Francisco, CA, 1976.
- Wenger, E., *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers, Palo Alto, CA, 1987.
- Wescourt, K., Beard, M., & Gould, L., "Knowledge-based adaptive sequencing for CAI: Application of a network representation," *Proceedings of the National ACM Conference*, Seattle, WA, pp. 234-240, Association for Computing Machinery, New York, NY, 1977.
- Wilensky, R., "Talking to UNIX in English: an overview of UC," *Proceedings of National Conference of Artificial Intelligence*, Morgan Kaufmann Publishers, Palo Alto, CA, 1982.
- Winograd, T., & Flores, F., *Understanding Computers and Cognition*, Ablex Publishing Corporation, Norwood, NJ, 1986.
- Winston, P., *Artificial Intelligence*, Second Edition, Addison-Wesley, Reading, MA, 1984.
- Wolfgram, D., Dear, T., & Galbraith, C., *Expert Systems for the Technical Professional*, John Wiley & Sons, Inc., New York, NY, 1987.
- Woods, W., "Transition network grammar for natural language analysis," *Communication of the ACM*, Vol. 13:10, pp. 591-606, 1970.
- Woolf, B., "Intelligent tutoring systems: A survey," in Howard Shrobe and the American Association for Artificial Intelligence (Eds.), *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, Morgan Kaufmann Publishers, Palo Alto, CA, 1988a.
- , "20 years in the trenches: What have we learned?," in Frason & Gauthier (Eds.), *Intelligent Tutoring Systems*, Ablex Publishing Corp., Norwood, NJ, in press.

- Woolf, B., "Representing complex knowledge in an intelligent machine tutor," *Artificial Intelligence and Human Learning*, John Self (Ed.), Chapman and Hall Inc., London, England, 1988b.
- , "Theoretical issues in building an intelligent tutor," in G. Kearsley (Ed.), *Artificial Intelligence and Instruction: Applications and Methods*, Addison-Wesley, Reading, MA, 1987.
- , "Multiple knowledge sources in intelligent tutoring systems," *IEEE Expert*, Los Alamitos, CA, Summer, 1987.
- , "Design issues in building a computer tutor," *IEEE Computer*, special issue on Artificial Intelligence for Human-Machine Interaction, Los Alamitos, CA, September, 1984b.
- , "Representing complex knowledge in an intelligent machine tutor," *Artificial Intelligence and Human Learning*, John Self (Ed.), Chapman and Hall Inc., London, England, 1988b.
- , "20 years in the trenches: What have we learned?," in Frason & Gauthier (Eds.), *Intelligent Tutoring Systems*, Ablex Publishing Corp., Norwood, NJ, in press.
- , "Theoretical issues in building an intelligent tutor," in G. Kearsley (Ed.), *Artificial Intelligence and Instruction: Applications and Methods*, Addison-Wesley, Reading, MA, 1987.
- , *Context-dependent Planning in a Machine Tutor*, Ph.D. Dissertation, Computer and Information Sciences Department, University of Massachusetts, Amherst, MA, 1984.
- , *Context-dependent Planning in a Machine Tutor*, Ph.D. Dissertation, Computer and Information Sciences Department, University of Massachusetts, Amherst, MA, 1984.
- Woolf, B., Schultz, K., & Murray, T., "Working with expert teachers to distinguish knowledge from theory," to appear as a COINS Technical Report, University of Massachusetts, Amherst, MA.

- Woolf, B., Suthers, D., & Murray, T., "Discourse control for tutoring: Case studies in example generation," to appear as a COINS Technical Report, University of Massachusetts, Amherst, MA.
- Woolf, B., Murray, T., Suthers, D., & Schultz, K., "Primitive knowledge units for tutoring systems," *Proceedings of the International Conference on Intelligent Tutoring Systems (ITS88)*, University of Montreal, Montreal, Canada, 1988.
- Woolf, B., Blegen, D., Verloop, A., & Jensen, J., "Tutoring a complex industrial process," in Lawler, R., & M. Yazdani, (Eds.) *Artificial Intelligence and Education: Learning Environments and Tutoring Systems*, Ablex Publishing Co., NJ, 1987.
- Woolf, B., & Cunningham, P., "Building a community memory for intelligent tutoring systems," *Proceedings of the National Association of Artificial Intelligence (AAAI-87)*, Morgan Kaufmann Publishers, Palo Alto, CA, 1987.
- Woolf, B., & Murray, T., "A framework for representing tutorial discourse," *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI-87)*, Morgan Kaufmann Publishers, Palo Alto, CA, 1987.
- Woolf, B., & McDonald, D., "Representing discourse conventions in tutoring," *Expert Systems Symposium*, IEEE and MITRE Corp., McLean, VA, 1984.
- Yankelovich, N., Meyrowitz, M., & van Dam, A., "Reading and writing the electronic book," *IEEE Computer*, October 1985.
- Zuboff, S., *In the Age of the Smart Machine*, Basic Books, New York, NY, 1988.
- Brattle Research Corp., *Artificial Intelligence Computers and Software: Technology and Market Trends*, Brattle Research Corp., Cambridge, MA, 1984.
- Carnegie Council of Policy Studies in Higher Education, 1979.
- Computers in Education: Realizing the Potential*, US Department of Education Conference, Nov. 1982.
- National Commission on Excellence in Education, *A Nation at Risk*, 1983.

National Task Force on Educational Technology, *Transforming American Education: Reducing the Risk to the Nation*, Report to the U.S. Secretary of Education, 1984.

Power On! New Tools for Teaching and Learning, U.S. Congress Office of Technology Assessment, 1988.

Proceedings of the Office of Education Research and Improvement, US Department of Education, 1983.

National Science Foundation, *Educating America for the 21st Century*, 1983.

United States Department of Education and the National Science Foundation Report, 1981.

Walberg, H., A Series of Reports (1982-3) Concerning Computational Studies of Mathematics Skills Scores between US and Japanese Students, 1982-1983.

