

LEARNING BY TEACHING AGENTS

By

Thomas Katzlberger

Dissertation

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

DOCTOR OF PHILOSOPHY

in

Computer Science

December, 2005

Nashville, Tennessee

Approved:

Professor Gautam Biswas

Professor John Bransford

Professor David Noelle

Professor Doug Fisher

Professor Steven Schach

Professor Bethany Rittle-Johnson

ACKNOWLEDGEMENTS

I would like to thank Dr. Gautam Biswas for accepting me as research assistant and supporting me after I finished my undergraduate studies at Vanderbilt, Dr. Dough Fisher, and Dr. Stephen Schach for their influential courses during my early time at Vanderbilt and their support later on, and Dr. Patrick Fischer, my first adviser at Vanderbilt, who selected challenging courses for me.

I would like to specially thank Dr. David Dilts. He taught me in his excellent course "Research Methods" all I needed to know to complete this thesis and never hesitated to answer any questions later when I needed further help. I would like to thank Dr. David Noelle, Dr. Bethany Rittle-Johnson, for their committee duties and their suggestions and ideas that contributed to my work. Special thanks also to my mentors in Educational Psychology Dr. Nancy Vye, Dr. John Bransford, and Dr. Daniel Schwarz.

I also would like to thank all the helpers I had during my Experiments: Kadira Belyne, and all the other members of the Teachable Agents Group. Additionally, I would like to thank Joan Davis for helping to obtain the signatures for my dissertation at University of Washington. I am grateful for the help and warm care of Sandy Winters during my stay at Vanderbilt. She always found time to listen and had some advice when I needed it. I would like to thank Dr. John Veillette for help in difficult situations.

Finally, I would like to thank my mother for her support when I needed it and my wife for her patience during the very difficult time before my thesis defense. I am also grateful for the financial support from the National Science Foundation (Grant Numbers NSF KDI 9873520 and NFS ROLE 0231771), and I would like to thank the authors of free software applications that I used to write and print this thesis.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	ii
TABLE OF CONTENTS	iii
LIST OF TABLES	vi
LIST OF FIGURES	viii
Chapter	
I. INTRODUCTION	1
Learning by Teaching and its Implementation.....	4
Expected Results	6
Thesis Organization	8
II. EDUCATIONAL PERSPECTIVES	10
From Thinking to Learning.....	11
Social Learning and Development	12
Situated Learning	13
Anchored Instruction.....	14
Constructivism	15
Inert Knowledge and Knowledge Contexts	17
Repair Theory	19
Transfer	21
Metacognition and Metacognitive Regulation.....	23
Motivation.....	25
Intrinsic Motivation	26
Flow	28
Summary.....	29
III. COMPUTER AIDED INSTRUCTION	32
Pedagogies and Instructional Strategies.....	33
The Cognitive Tutoring Approach.....	33
Self-Explanation	35
Learning by Doing, Active Learning, Inquiry Learning and Related Pedagogies.....	36
Discovery Learning.....	38
Flexibly Adaptive Instructional Design Theory.....	39
Peer Tutoring, Cross-age Tutoring, and Reciprocal Tutoring.....	40
Learning by Teaching.....	42
Other Pedagogical Approaches	47
Summary.....	48

Existing System Designs.....	49
Intelligent Tutoring Systems.....	50
Computer Based Microworlds.....	54
Interactive and Intelligent Learning Environments.....	56
STAR Legacy Learning Shell.....	60
Related Teachable Agents Research.....	62
Agents.....	64
Summary.....	66
IV. SYSTEM DESIGN.....	70
The Cycle Architecture.....	72
General Introduction (Anchoring Context).....	73
Problem Description.....	75
Teach Phase.....	76
Teaching the Teachable Agent – Two Examples.....	79
Quiz Phase.....	81
Test Phase.....	85
Software Architecture Overview.....	90
Interactive Representations.....	91
Simulation.....	92
Simulation User Interfaces.....	93
Simulation Framework.....	95
Smart Tools.....	98
User Interface.....	99
Smart Tools Framework.....	102
Teachable Agent Architecture.....	104
User Interaction.....	104
Dialogue States.....	105
Sensing Rules (Agent Triggers).....	106
Teachable Agents Learn Believably.....	108
Agent Framework.....	109
Agent Knowledge Structures.....	111
Agent Scripting and Communication.....	113
Dialogues and Curriculum.....	116
Summary.....	118
V. EXPERIMENTAL DESIGN.....	121
Hypotheses.....	121
Detailed Experimental Design.....	124
Method of Analysis.....	128
Activities of Students in the Alternate Condition.....	130
Population, Sample and Randomization.....	134
Measures.....	135
Motivated Strategies for Learning Questionnaire.....	135
Experiences that Energize.....	137
Knowledge Test.....	139
Online Testing Interfaces.....	141

Near Transfer Task.....	142
Survey	145
User Action Traces.....	146
Procedures.....	146
Summary	148
Knowledge Test Results	150
Time on Task	151
Time to Complete all Cycles.....	151
Class Differences	153
Group Equivalence - Teacher Rated Problem Solving Speed	154
Correlation of Knowledge Scores versus Time	155
Quiz Behavior	156
Knowledge Test Analysis	158
Individual Extraordinary Questions of the Posttest	161
Transfer Test Results	163
Motivation Test Results	166
Motivated Strategies for Learning Questionnaire (MSLQ)	167
Experiences That Energize (ETE).....	169
Survey Results.....	172
Summary	174
VII. CONCLUSIONS	179
Appendix	
A. MOTIVATED STRATEGIES FOR LEARNING QUESTIONNAIRE.....	183
B. EXPERIENCES THAT ENERGIZE	189
C. KNOWLEDGE TESTS.....	190
D. TRANSFER TEST WORKSHEET	200
E. NORMAL LINEAR MODEL ADDENDA.....	201
Knowledge Test (Analysis of H_{KNGAIN})	201
Knowledge Test (Analysis of H_{KNDIFF})	203
Transfer Test	204
MSLQ	207
BIBLIOGRAPHY	210

LIST OF TABLES

Table	Page
1. Table of Hypotheses (O notation refers to Figure 33; \bar{O} is the mean of a measure).....	123
2. Table of Differences between Experimental and Alternate Condition	133
3. Motivated Strategies for Learning Questionnaire Example Questions.....	136
4. Experiences That Energize Measured while Students Use the System.	138
5. Near Transfer Task Grading Scheme	144
6. Survey Questions	145
7. Pearson Correlation of Knowledge Score versus Time	156
8. Descriptive Statistics of Knowledge Pretest and Posttest.....	158
9. Normality Tests for H_{KNDIFF}	160
10. Levene’s Test of Equality of Error Variances.....	160
11. Number of Correct Answers by Group on the Knowledge Tests	161
12. Descriptive Statistics of the Transfer Task	166
13. Executive Summary of Study Results.....	176
14. Modified Motivated Strategies for Learning Questionnaire (MSLQ) Pretest.....	183
15. Modified Motivated Strategies for Learning Questionnaire (MSLQ) Posttest.....	186
16. Experiences that Energize.....	189
17. Knowledge Pretest	190
18. Knowledge Posttest.....	194
19. Box’s Test of Equality of Covariance Matrices for Knowledge Tests	201
20. Normality Tests for H_{KNGAIN}	204

21. Normality Tests for the Transfer Test Analysis	205
22. Box's Test of Equality of Covariance Matrices for Transfer Test	206
23. Levene's Test of Equality of Error Variances for Transfer Test.....	206
24. Normality Tests for the MSLQ Analysis	209
25. Levene's Test of Equality of Error Variances.....	209

LIST OF FIGURES

Figure	Page
1. Teachable Agents: Applying Learning Theory in Modern Education	6
2. Intelligent Tutoring System Architecture adapted from (Wenger 1987)	51
3. Reengineered Version of Rescue at Boone’s Meadow with Planner and Timeline	59
4. STAR Legacy learning shell cycle structure (IRIS 1999).....	61
5. Teachable Agent Betty’s Brain models an ecosystem domain.....	63
6. Teachable Agent Learning Cycle.....	73
7. Problem Description of Cycle 3.....	76
8. User Interface in the Teach Phase: Simulation (top), Graph (left bottom), and Agent Interface	77
9. Teaching the Agent to Find Minutes when Miles are given.....	80
10. Quiz Phase: The Student selects a Problem and the Agent solves it.....	82
11. Quiz Phase: The Teachable Agent demonstrates and explains its Solution	84
12. Quiz Phase: The Teacher Agent corrects (left) and gives the Solution (right).....	85
13. Test Phase: The New Simulation	86
14. Test Phase: The Student "Creates" the Test by Finding Answers.....	87
15. Test Phase: The Student Reviews and Grades the Agent.....	88
16. Test Phase: The Teacher Provides the Correct Solution.....	89
17. Test Phase: The Student gets Summative Feedback	90
18. UML Structure: Package Architecture of the Learning Environment.....	91
19. Simulation: Item Inspector Inspecting an Airplane with its Pilot during a Flight	95
20. UML Structure: Simulation creates Environment that executes PlanSteps to change Actors.....	96
21. UML Structure: Actor Class Hierarchy for Velocity, Fuel-Rate and Payload Simulations	97
22. A Student Reads a Total Time from a Trip-Graph	100
23. UML Structure: The Controller instantiates SmartTools; Graph and Table share a GraphModel.....	102
24. Calculator Tool and actual Table Tool Created by a Student during our Experiment	104
25. Agent Interfaces: The Student chooses what to do next (left), Ms. Mathie instructs Slope (right).....	106

26. Interaction Schematic: Agents Sense and Act in the Environment	108
27. UML Structure: Agent Architecture	110
28. Four Step Procedural Knowledge to find the Slope of a Line	112
29. UML Structure: AgentCommunication.....	114
30. Partial Agent Script: Teacher Checks if the Student has used the Graph's Reader Lines Correctly.....	115
31. Quiz Script: Full Script for the Quiz Question to Find the Slope of the King Air Plane.	115
32. Standard Dialogue Structure on the Example of Cycle 1.....	117
33. Full Experimental Design in the Notation of Cook and Campbell (Cook, Campbell, and Cook 1979).....	125
34. Repeated Measure (Within Subjects) Design to analyze Knowledge Gain	126
35. Between Subjects Design to Analyze how the Treatment Influences Knowledge (O_1 is a Covariate).....	126
36. Between Subjects Design to Analyze how the Treatment Influences Transfer.....	127
37. Between Subjects Design to Analyze how the Treatment Influences Motivation (O_2 is a Covariate)	127
38. Quiz Phase: User Interface of the Alternate Experimental Condition.....	132
39. Knowledge Pretest Example: Which Line is a faster Speed?	140
40. Electronic Likert Testing Interface for MSLQ, ETE (this picture), and Knowledge Tests.....	141
41. Photo of Equipment Supplied to Students in the Transfer Test (without Stopwatch).....	143
42. Time to Complete each Cycle and Average Time per Cycle	152
43. Time to Complete each Cycle depends on Classroom.....	154
44. Scatter-plot of Knowledge Score versus Time.....	155
45. Quiz Questions Requested by Students in each Cycle	157
46. Histogram of the Knowledge Pretest (O_1)	159
47. Histogram of the Knowledge Posttest (O_4).....	159
48. Illustrated Multiple Choices for a Posttest Question.....	163
49. Transfer Test Scores by Task	165
50. Motivated Strategies for Learning Questionnaire Posttest Comparison of Conditions.....	168
51. ETE Time Series for cycle 1, 2 and 3 (Standard Error Mean Bars)	171
52. Survey Results	173
53. Residual Plot Knowledge Pretest.....	202
54. Residual Plot Knowledge Posttest	202

55. Histogram of the Knowledge Posttest of the Experimental Condition.....	203
56. Histogram of the Knowledge Posttest of the Alternate Condition	203
57. Histogram of the Transfer Test of the Experimental Condition.....	204
58. Histogram of the Transfer Test of the Alternate Condition	205
59. Residual Plot for Transfer Test (Graph Labels).....	207
60. Residual Plot for Transfer Test.....	207
61. Histogram of the MSLQ of the Experimental Condition	208
62. Histogram of the MSLQ of the Alternate Condition.....	208
63. Residual Plot Motivation Posttet	209

CHAPTER I

INTRODUCTION

This research aims to extend the current state of the art in intelligent learning environment design by implementing the learning by teaching paradigm using teachable software agents. Students, who are domain-novices, teach an agent and learn about the domain in this process. Our work draws from psychological theories of constructivism (Piaget 1953), learning (Dewey 1933, 1938), transfer (Haskell 2001; Bransford, Brown, and Cocking 2000), and motivation (Brophy 1998; Csikszentmihalyi 1990; Lepper et al. 1993). In addition, this work is informed by previous research on learning environments for education that was conducted in the Learning Technology Center, Peabody School of Education, and the Department of Electrical Engineering and Computer Science at Vanderbilt University (Biswas, Katzlberger et al. 2001; Leelawong et al. 2002; Leelawong et al. 2003; Bransford, Brown, and Cocking 2000; Crews et al. 1997; Owens et al. 1995; Cognition and Technology Group at Vanderbilt 1997; Bransford 1990). However, the focus of this research is on evaluating the influences of learning by teaching agents on student's learning and problem solving in middle-school mathematics. Our teachable agents do not incorporate inductive mechanisms to learn; rather, they are computer-based social agents that require explicit instruction to perform well on a given task. Our goal is to design agents that improve the student's learning, and give them a deeper understanding of the domain while making the learning task interesting. To study the effectiveness of our approach, we have conducted experiments to analyze its effects and influence on students. Specifically, we contrast performance and motivation of students who learn for themselves with students who learn by teaching agents.

We implemented our system in the domain of mathematics, as there is public interest in improving mathematical education in the US. The report "Trends in Mathematics and Science Study" (Mullis et al. 1999) documents that the United States is trailing a number of developed nations in student performance in science and mathematics. There is also evidence through the experience of many teachers that students, who come to college with A and B averages, fail to answer the same question in different forms on an exam because they cannot transfer their knowledge to new situations (Haskell 2001). As a result, there is continued effort by federal, state, and local agencies to improve education. A primary thrust in recent times has been to use technology to build learning environments that promote deeper understanding and better retention of learnt knowledge. The collective goal of education is to make knowledge applicable and useful in different problem-solving situations later in life (by improving *transfer of learning*). However, our educational system does not currently live up to its promise (Haskell 2001).

Learning by teaching is an educational technique that has its roots in research on peer tutoring with human tutors, and carries the promise of promoting deeper understanding of concepts in the knowledge domain than traditional learning techniques. Learning benefits of moderately experienced tutors who taught tutees were observed in preliminary studies, informal work, and some full studies (Cohen 1986; Palincsar and Brown 1991; Chan and Chou 1997; Gaustard 1993; Michie, Paterson, and Hayes-Michie 1989; Nichols 1994). The learning by teaching paradigm can be linked to learning gains demonstrated in self-explanation studies (e.g., Chi 1997) because explaining to a tutoring system¹ (e.g., Aleven and Koedinger 2002) and explaining to a tutee are inherently similar tasks that involve metacognitive processes like reflection. However, learning

¹ Current research on tutoring systems operationalizes self-explanation as students explaining to a tutoring system (Weerasinghe and Mitrovic 2002). See section about self-explanation for more details (page 35).

by teaching may have additional advantages, especially in fostering social interaction and motivation. We discuss learning by teaching in more detail in the next chapter (see page 35).

In contrast to classic approaches, where tutors already have experience in the domain, we are interested in novice tutors who teach computer-based agents. In this research, we explore this new instructional approach: Domain novices (6th grade middle-school students) teach social agents to solve distance-rate-time problems by using and creating graphs. While doing that, they learn about the domain.

If our system is successful, we expect to see differences in motivation, learning, and transfer between students who teach agents to learn, and students who learn but do not teach. Before discussing this in more detail, we establish why we investigated learning by teaching with software agents as tutees, and introduce our vision of the teaching process.

We let students teach software agents and not humans, because tutoring requires training and communication skills (Cohen 1986), which our novice tutors may not have. This can lead to incorrectly taught knowledge, and require continuous learning and relearning, which does not harm agents, but this process may be frustrating and harmful to human tutees. To avoid harming human tutees, we let novice tutors teach software agents. Teaching others typically requires a shared representation of the knowledge that organizes the materials so that they are easy to understand. This may be hard to achieve between novice human tutors and tutees. However, if we use a computer-based teachable agent, we can select representational structures that generally work well in advance.

Additionally, each tutor-tutee interaction is confounded by the behavior of the tutee. Even well trained humans acting as tutees could introduce substantial variability into an experimental setting. By substituting the tutee with a software agent with well-defined characteristics, we can

remove this variability. We also avoid status problems in peer tutoring, which may make tutees feel inferior, and cause friction between tutor and tutee (Gaustard 1993).

Learning by Teaching and its Implementation

We review the concept of learning by teaching. A teacher planning to teach in a new domain will generally first *prepare* to teach, then *teach*, and *reflect* during and after the teaching process.

When preparing, he or she will scout resources, choose, access, and organize the material in a meaningful way, and learn during this process. Only then, a teacher can start teaching. Throughout the teaching process, the teacher explains knowledge, then interacts with students, and reflects on questions, comments, and exam results. Unexpected or wrong solutions by tutees can cause reflection on one's own knowledge or the teaching process.

Related research mainly focused on teacher-student interactions (teaching and reflecting) as causes for the tutor's learning. We additionally include differences in preparation (scouting and organizing), because we think that students preparing to teach take a different approach to studying resources than students who prepare for an exam, especially if they are domain novices. However, our system does not enforce a strict sequence and students most likely scout or read resources only if the need arises during the teaching process (e.g., the agent asks to be taught something). We do not yet utilize the additional strategies of learning by creating tests and quizzes, although those tasks could enhance learning by teaching further.

Because learning by teaching requires an environment where knowledge can be taught, retrieved, and verified, we have implemented a powerful and effective model-learning environment based on previous research (Figure 1), which we augment with teachable agents. This environment draws from the psychological learning theory of constructivism (e.g., Piaget 1953) and uses

anchored instruction (Cognition and Technology Group at Vanderbilt 1993; Crews et al. 1997 & Bransford, 1997). We integrated and extended ideas from two separate intelligent learning environment projects into one system: Adventure Player (Crews et al. 1997), and Smart Tools (Owens et al. 1995). Adventure Player used a simulation environment that allowed students to verify their solutions, while the Smart Tools project allowed students to construct representations that helped them to solve problems. In this work we introduce the idea that these representations could be taught to teachable agents. Last, in conjunction with the teachable agent paradigm, we superimposed a learning cycle structure similar to the STAR Legacy learning shell (Schwartz, Lin et al. 1999), which organizes students' learning and problem solving tasks in a sequence that helps them understand and organize their domain knowledge easily. We have developed a modular software framework that allows us to study arbitrary variations and extensions of this baseline environment, one of which is learning by teaching software agents.

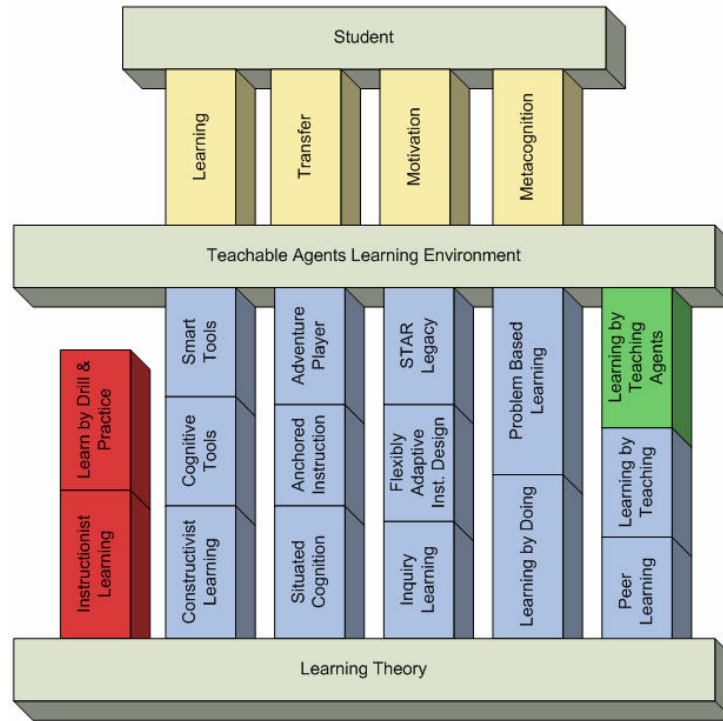


Figure 1. Teachable Agents: Applying Learning Theory in Modern Education

As we will see in our discussion of educational theory in chapter II, the central design principle of our baseline learning environment is that it situates students in a realistic context through anchored instruction, which has been shown to enhance learning and transfer of learning (Cognition and Technology Group at Vanderbilt 1993; Crews et al. 1997; Crews 1995; Bransford 1990; Leelawong et al. 2002). We also give learners considerable freedom to explore, without correcting mistakes too soon, which teaches students how to monitor their own learning process, without relying too much on external help.

Expected Results

One of the primary contributions of this work is the development of an agent architecture to facilitate the design and implementation of learning by teaching systems that include a teaching

framework, assessments, and domain resources to facilitate the learning process. We explore the learning by teaching agents paradigm within this framework, because we think that student teachers behave differently than students who do not teach. This changed behavior influences how students learn and produces differences in learning and motivation.

We think students learn better, because students look differently at materials when they prepare for teaching or grade and review answers of their tutees, than when they learn and practice for themselves and review their own answers. Additionally, teachable agents actively engage students in explaining knowledge to them and force students to reflect on what they have taught.

It is also possible that learning by teaching influences motivation. The interaction with social agents, especially the act of helping an agent by teaching it, may make students feel better about their task. Learning may become the means to accomplish the goal of teaching an agent. Thus, we try to identify the motivational influences of learning by teaching on students. Based on these expectations, we explore the following questions by comparing students who teach agents against students who do not teach:

Do students demonstrate better learning ability, and do they learn more when they teach a computer agent as opposed to students who use a learning environment but do not teach an agent?

Can students transfer learnt knowledge to new domains better when teaching agents?

Does teaching our agents motivate students during their teaching and learning tasks?

Is learning by teaching a viable approach for novice students?

For the remainder of this work we will isolate the effects of learning by teaching by contrasting students who learn from a teacher agent and teach a tutee agent with students who learn from

a teacher agent and study for themselves. Both of these conditions are embedded in our learning environment, which is a fully featured mathematics learning environment with simulation capabilities and an interactive graph representation that can be used to solve rate problems.

Thesis Organization

In chapter II of this thesis, we will discuss cognitive theories that influenced the design of our learning environment. We explore educational theories that explain how people learn. We describe Social Learning and Situated Learning that led to the development of Anchored Instruction, which forms the basis for our system design. Then we make an excursion through the theory of constructivism, which describes how we learn. We explore the possibility that learnt knowledge may stay inert¹, or become associated with a narrow context, effectively making it irretrievable in many real problem-solving situations. Later in chapter II, we describe transfer of learning, metacognition, and motivation, which are primary factors that support deep learning with understanding and transfer.

Chapter III relates the theories of the previous chapter to traditional and current computer-based learning environments. First, we introduce pedagogies and strategies employed by previous learning environments that employ cognitive tutors, discovery learning, active learning, learning by teaching, and other miscellaneous approaches. Then we survey existing system designs and their implementations: Intelligent tutoring systems, microworlds, and interactive learning environments. A discussion of agent-based approaches concludes the chapter.

¹ Inert knowledge (Whitehead 1929) is learnt knowledge that a person cannot apply, for example, in a different practical context, because he or she does not associate it with the current context.

Chapter IV presents our ideas for an agent-based approach of a new learning environment design. We will start by describing the pedagogical architecture of our system, which combines anchored instruction with a STAR-Legacy-based approach to develop the curriculum units for teaching rate problems in mathematics. Then we will introduce the design of software modules by describing their functionality and implementation. The primary functional components are simulations, smart tools, and the agent architecture that includes the teachable agent and the mentor agent.

In chapter V, we introduce our hypotheses, experimental design, and the analysis procedures. We also describe our sample, procedures, and measures, which are roughly split into motivation, learning, transfer measures, and a survey.

Chapter VI presents our results, starting with the discussion of our knowledge test results. Following this, we summarize our results of the transfer test and motivational measures, and finally present our conclusions and directions for future work in chapter VII.

CHAPTER II

EDUCATIONAL PERSPECTIVES

Most present day instructional practices in classrooms tend to produce superficial learning among students (Weigel 2002). Hence, Weigel suggests moving away from techniques that promote surface learning towards techniques that emphasize deep learning in classroom environments. He cites Entwistle, who stated that we engage in surface learning if "learners treat a course as unrelated bits of knowledge, memorize facts, carry out procedures routinely, see little value or meaning in carried out tasks, study without reflection on purpose or strategy and feel undue pressure and stress." On the contrary, deep learning involves "relating ideas to previous knowledge and experience, looking for patterns and underlying principles, checking evidence and relating it to conclusions, examining logic and arguments cautiously and critically, being aware of understanding that develops while learning and being actively interested in content" (Entwistle cited in Weigel 2002). Deep learning should help students learn skills and knowledge for life (Weigel 2002). Unfortunately, the current focus on improving standardized test scores trains students to be expert test takers, who memorize facts and procedures but find it hard to apply them in real-world problem solving situations.

An essential foundation for designing a successful learning system that promotes deep learning must be based on how students learn, a topic, which we explore in this chapter. We also discuss cognitive and educational findings, in the areas of constructivist learning, transfer, motivation and metacognition, which provide the core for a growing number of modern instructional approaches to support effective teaching and learning.

In the sections that follow, we review the educational theory of learning. Two strands of research are very relevant to designing learning environments: learning within a (social) context, and the constructivist learning theory. After that, we discuss the concept of transfer, which describes the circumstances under which people can apply knowledge learnt in one context to new situations. Transfer also serves as an advanced metric for learning. Another important issue that has been shown to improve learning is the adoption of relevant metacognitive strategies that help students set goals, plan, organize, self-asses, and seek help during learning and problem solving. Last, motivation may help learners to enjoy their problem solving tasks and improve learning gains through positive learning experiences.

From Thinking to Learning

Thinking is a continuous active flow of words and pictures in our mind that we can direct, but cannot stop unless we fall asleep. Our mind seamlessly integrates input from our senses with our thoughts to form a mental stream of consciousness which expresses our beliefs (not facts) about how the world works (Dewey 1933). To some of these beliefs we will pay attention and find them prominent or important enough to remember for later. Thus, we learn. Some of these facts simply fall into place and can be remembered without specific effort¹, while others require continued repetition and application before we can apply them effectively. Other times, we may find ourselves deliberating about our process of thinking and learning, or simply observe ourselves through reflection. In this manner, we engage in metacognition, which is a skill that can help us to improve our learning.

¹ Advertising tries to exploit this kind of passive learning, but not all learning is passive.

Amazingly, learning results cause not only electro-chemical, but also structural changes in our brain. Through technologies like positron emission tomography (PET) and functional magnetic resonance imaging (fMRI), research has found how learning changes the physical structure of the brain. Simply by absorbing knowledge we physically change the strength of our synapses and change neuronal pathways. Thus, "One of the simplest rules is that practice increases learning; in the brain, there is a similar relationship between the amount of experience in a complex environment and the amount of structural change" (Bransford, Brown, and Cocking 2000).

Knowing this, we could simply increase the amount of practice by introducing drill and practice exercises to aid learning. Still, experience has shown that other systems promote deeper learning and better transfer (Weigel 2002; Haskell 2001), as we discuss in examples presented in chapter III. Students will perform relatively well in the short run but may fail to transfer (apply) knowledge in new situations. A number of theories summarized in this chapter look at means to achieve better learning, the ability to apply what is learnt, and to achieve life-long learning.

Social Learning and Development

If we go beyond physical changes in the brain, and theories of classical conditioning and reinforcement (Pavlov, Thorndike, and Skinner), we will find ideas on how people learn in *social learning* or *observational learning* (Bandura 1977). Implications of social learning are especially interesting to us, as social agents play an important role in our learning environment.

In everyday life, we continuously learn new knowledge, for example, through the activities of conversation, play, discovery, and problem solving. This implies that we often learn in a social context. Even when we acquire knowledge on our own, we may reshape it later to fit socially ac-

cepted norms (Tripathi 1979). The simplest form of social learning is learning through observation of behaviors.

Observational learning requires attention, retention, motivation, and potential reproduction of the behavior (Bandura 1977). In his classic experiment, Bandura showed movies to children that showed an adult being rewarded or punished for punching a doll. The children learned the behavior and repeated it more frequently when the adult was rewarded. In our context, children working with social agents may learn from the agent's behaviors by observation, which we had to consider in designing our environment. The findings of observational learning soon led researchers to analyze learning in social contexts like interactions among family and friends in informal settings, and collaboration among colleagues at work, which gave rise to situated learning, which we have directly applied in our learning environment.

Situated Learning

Learning and cognition develops in a social context. Vygotsky linked social interaction with the development of cognition (Vygotsky 1978). His work on social constructivism gave rise to situated learning and other theories (Brown, Collins, and Duguid 1989; Lave and Wenger 1990), which puts learning into the context of an activity by focusing on learning within a social framework. It contrasts with traditional lecture-based learning, which involves knowledge that is often abstract and taught out of context in school.

Lave and Wenger stated that situated learning relies on the principles that (1) knowledge needs to be presented in an authentic context, for example, settings and applications that would normally involve that knowledge, and (2) learning requires social interaction and collaboration

(Lave and Wenger 1990; Kearsley 2002). Situated learning is closely related to anchored instruction (Cognition and Technology Group at Vanderbilt 1993), which has guided our system design.

Anchored Instruction

Anchored instruction situates learning in the context of meaningful problem solving tasks in a realistic environment. The environment is called the *anchoring context* or *macrocontext* (Cognition and Technology Group at Vanderbilt 1993; Crews et al. 1997). Anchored instruction distinguishes macrocontexts from microcontexts. The latter are traditional word problems found in textbooks for mathematics instruction. Each macrocontext defines an elaborate problem setting from which teachers and students can derive many sub-problems. The macrocontext allows elaborate exploration, and students may revisit it from many perspectives over several weeks. Thus, anchoring contexts support effective problem solving in multiple domains, and help one learn about specific concepts and principles in context. This allows students to think effectively about particular domains, and it avoids the problem of *inert knowledge*¹ (Bransford 1990).

Kearsley (2002) summarizes the principles of anchored instruction: (1) Learning and teaching activities should be designed around an *anchor*, which should be a case study or problem situation; (2) Curriculum materials should allow exploration by the learner. We used this approach directly in our learning environment. Anchored instruction provides one of the theoretical foundations of our research. A second direction of research, which complements situated cognition and anchored instruction, is constructivist learning theory. This is described in the next section.

¹ We will discuss inert knowledge a few sections later in more detail.

Constructivism

Tell me and I forget.

Show me and I remember.

Involve me and I understand. - Chinese proverb.

In this section, we focus on constructivist theories of how we learn. We begin the discussion by addressing the constructive nature of memory. This leads to the theory of constructivism, which, among other issues, addresses the question of how people integrate new knowledge into their existing knowledge structures. We decided to focus on constructivism because it is a compelling theory among others that explains how students learn and then recall what they have learned.

People can memorize information that is meaningful and related to previous experience much easier than meaningless unrelated information. People merge new knowledge with what they already know (Tripathi 1979). This means that our memory does not simply recall facts, but can modify or even introduce new information. Thus, our memory is constructive. Tripathi demonstrated in experiments where Indian schoolchildren were asked to memorize old folk tales that recalled stories had additions, omissions, and modifications. Many of the modifications transposed the stories into a modern world. This constructive nature of memory is the basis for constructivism, which builds the foundation for many innovative approaches to teaching.

The first documented roots of constructivism date back to Democritus, Plato, and Giambattista Vico who commented in 1710, "One only knows something if one can explain it." Immanuel Kant elaborated on this idea and hypothesized that "humans are not passive recipients of knowledge but integrate it by constructing their own representation" (Kant 1781). In this sense, the constructivist paradigm says that every person constructs knowledge from experiences and out-

side influences to form his or her own *interpretation* of what they know. For students, this not only includes material presented by teachers in classrooms but also discussions with peers, parents, observations of the surrounding world, and information gained from other sources. From all this information, learners build their own knowledge compilation, which includes correct and incorrect information. This is based on the constructivist theory of Bruner (Bruner 1966), who was heavily influenced by Piaget's Genetic Epistemology (Piaget 1953).

Piaget studied the cognitive development of children and theorized about it. He studied spontaneous learning of children and found that children integrate new knowledge with what they already know. An observation that contradicts current knowledge creates *disequilibrium* or in other words an imbalance between what one already knows and what is newly encountered. Students try to correct (*equilibrate*) this disequilibrium by *accommodation*. Accommodation means that existing knowledge structures change to fit the new information without contradictions. Students can incorporate new events into preexisting cognitive structures by *assimilation*. Existing knowledge schemes are called preconceptions and stem from the fact that no learner walks into the classroom as *tabula rasa*, but rather with their own powerful ideas of how the world works. If a student's previously constructed incorrect understanding is not challenged, they may retain misconceptions, and fail to understand new concepts (Bransford, Brown, and Cocking 2000).

Piaget conjectures that the processes of equilibration may cause conflicts if old and new knowledge do not match, and successful learning requires effective ways to resolve such conflicts. Sometimes people keep contradictory knowledge in different "compartments" of their mind that are activated by the context in which knowledge is applied, but often one has to be abandoned in favor of the other. Other times, learners are not aware of conflict and merge knowl-

edge inappropriately. An example, which illustrates assimilation, is that one might tell children whose misconception is that the earth is flat that the earth is round. Children might integrate the knowledge in their own way by constructing a new picture in their mind with a pancake shaped flat and round earth (Vosniadou and Brewer 1989).

The constructive nature of learning has powerful implications for the design of learning environments, especially when computers keep track of what a learner knows in a *student model*¹. We cannot assume that knowledge presented to the student is learnt in the way it was intended, but that the student constructs an approximation of the taught material with omissions, additions, and changes. Newly presented material may change existing knowledge structures of learners. This creates challenges in designing a learning environment that avoids, catches, and corrects misconceptions of learners whenever they evolve. We discuss pedagogies that were developed to alleviate these problems in the next chapter.

Inert Knowledge and Knowledge Contexts

Knowledge learned by a student may be associated with a specific context and stay inert in other problem settings. Whitehead identified *inert knowledge* and concluded that under certain circumstances a problem that should have been solvable by a subject could not be solved in a different context (Whitehead 1929). Inert knowledge implies that a student is unable to transfer knowledge to a new domain (lacking *transfer of learning*²). Others have observed that experts have highly conditionalized knowledge, which includes for every fact a specification of contexts in which it is useful (Glaser 1992). Thus, experts can readily apply knowledge correctly in new situations,

¹ Student models are used in intelligent tutoring systems, which we will discuss in the next chapter.

² This concept is described in a later section of this chapter.

while novices, who have a very small set of contexts where they can successfully apply their knowledge, may use incorrect procedures, or fail to solve problems at all. Students studying for a test may revert to their preconceptions outside of the classroom, because they cannot transfer their learning like experts (Bransford, Brown, and Cocking 2000).

An example for contextualization, preconceptions, and conflict resolution that leads to inert knowledge is beliefs about motion and inertia (Papert 1980). When pushing a table in a room we need force to keep the table moving (Aristotle's view). This experience contradicts the Newtonian laws that a moving body only needs force acting on it to start and stop moving. Thus, a student facing such a contradiction will resolve it by creating a context where Newtonian laws apply, like, for example, "in space" or "a car skidding on an icy road" (Bransford 1979; Bransford, Brown, and Cocking 2000). This in itself is not incorrect, but it also may lead to the belief of the student that Newtonian laws do not apply in general. In the worst case, students may associate knowledge with contexts that are meaningless in real life, like "physics in school" or "physics book - chapter 5."

Traditional classroom teaching may create inert or poorly conditionalized knowledge, and properly designed learning environments should avoid this by establishing relevant contexts, which aid students in learning to apply and then generalize learnt knowledge. Modern classroom teaching approaches can alleviate this problem by overlapping problem contexts between subjects, like, working on the same problem in a mathematics and a science class (Bransford, Brown, and Cocking 2000). However, this practice is often not implemented, as it requires coordination of efforts between teachers. We use the previously introduced teaching strategy of anchored instruction to reduce the likelihood of creating inert knowledge, by introducing realistic macrocontexts in which students perform their problem-solving tasks.

Constructivism has inspired a whole battery of modern, applied instructional strategies, such as learning by doing, active learning, and inquiry based learning, which we discuss in the next chapter. Before that, we briefly discuss repair theory, a proposed cognitive model of how students handle contradictions and incomplete knowledge during learning and problem solving.

Repair Theory

Students often form misconceptions (incorrect knowledge) while learning, and learning environments should employ strategies to correct these misconceptions. A systematic approach proposed for dealing with misconceptions is VanLehn's Repair Theory (Brown and VanLehn 1980). This theory calls misconceptions *mind-bugs*¹ (Brown and Burton 1978) that need to be identified and corrected to improve a student's understanding of the domain. When learning procedural skills, like performing subtraction or reading graphs, students tend to make systematic errors. These errors may appear individually or in a variety of combinations. Typically, these bugs are not stable, and students tend to move between different patterns (bug migration) by applying different repair strategies at different times (VanLehn 1990). An observing agent could theoretically detect such bug patterns, and use this information to generate feedback in a way that aids learning. Unfortunately, not all misconceptions are systematic. Cohen has identified that students can make inconsistent mistakes, which he called slips (Cohen 1990). Remediation after slips may not be effective, but tedious, if the student already knows the correct procedure, but inadvertently made a mistake.

Recently, Elby has criticized misconception repair as an oversimplification. He separates constructivist research into two camps: (1) Misconception constructivists, who model students

¹ This expression is similar to the term 'bug' in programmers-jargon, referring to faults in computer programs.

entering into learning tasks having bugs, alternate conceptions and misconceptions, as we have seen in the previous paragraph; and (2) Fine-grained constructivists, who believe that students have loosely connected, context activated minigeneralizations and knowledge elements (Elby 2000). For example, a student working with a speed vs. time graph that shows a horizontal line with a hump might interpret this as a car driving over a hill. The student's interpretation of the graph is literally what you see is what you get.

Although basic repair theory did not live up to its promise, and is often criticized, we could imagine that learning systems or computer agents that are able to identify misconceptions can provide useful feedback to learners to help them reflect on their misconceptions. A system based on bug identification and correction alone may only be partially successful in addressing learning problems. The main problem with mind-bugs is that one has to generate an exhaustive bug library. However, if we know a small set of common misconceptions that students often have in a domain, we can easily program an agent to react appropriately, and still provide benefit to the learning process in conjunction with other pedagogical approaches. Using this approach, an environment could point out counterexamples based on misconceptions, or initiate dialogues that lead the student to insights that his or her knowledge is incorrect. In a learning by teaching situation an agent may ask questions upon detecting such misconceptions, which leads a student to reflect on their understanding, and develop new insights. According to constructivist theories, this causes a cognitive disequilibrium, which forms the basis for students to correct their erroneous understanding of concepts and procedures.

Next, we discuss auxiliary concepts that are desired outcomes of learning, or intrinsically aid learning. First, we start with discussing the concept of *transfer of learning*, which is highly de-

sired, but elusive in education. Additionally, it is a measure of learning quality. Following that, we discuss metacognition and motivation.

Transfer

Transfer is the ability to generalize from the familiar to the less familiar, for example, it is the ability to use problem-solving knowledge learnt in the context or domain of one task in another context or different task (Bransford, Brown, and Cocking 2000; Haskell 2001). Most authors in current learning research call transfer a true (but elusive) measure of learning because it correlates to the potential utility of learnt knowledge in life, as we elaborate a few paragraphs later.

The importance of achieving transfer is enormous, because society expects that our educational system prepares students for life long learning and problem solving, and not just produces good grades in school. Despite reasonable efforts, most of the knowledge learnt in school stays in school, and research efforts to improve transfer have not been very successful yet. Haskell illustrates this calling it the paradox of transfer:

"In essence, we would like students to be able to apply what they have learnt; yet, despite governments pouring increasing amounts of money into improving education, we are unable to achieve transfer in schools." (Haskell 2001)

Psychological texts distinguish, at a basic level, *negative transfer*, which hinders solving a future task, and *positive transfer*, which improves later problem solving (Sternberg 1995). Negative transfer can occur when a person, through previous experience, uses a specific solution strategy not suitable for the current problem. Positive transfer may be initiated through analogies by either willfully introducing them in an instructional setting, or letting learners actively search for

analogies. Sometimes the student assumes incorrect analogies, because the context in which the situation is embedded appears to be the same. This is called *transparency*. If a student has to answer whether algae and fish use or produce oxygen, she might conclude, knowing that fish breathe oxygen that algae use oxygen too, because both live in water.

Learning research focuses only on the presence or absence of positive transfer. Haskell defines a taxonomy of transfer in six levels: (1) nonspecific transfer, (2) application transfer, which describes the ability to apply what one has learned, (3) context transfer, or the ability to apply knowledge in a different outside context (e.g., school vs. home), (4) near transfer to closely similar situations, (5) far transfer to dissimilar situations through analogical reasoning, and (6) displacement or creative transfer, which relates to discovering new insights (Haskell 2001). The differentiation and ordering among some entities in this taxonomy is not obvious; hence, many researchers describe their findings in simpler terms of only near and far transfer. Another theory of transfer (Salomon and Perkins 1988) distinguishes low-road transfer and high-road transfer. Low-road transfer occurs when stimulus conditions in the transfer context are similar to a prior context of learning to trigger semi-automatic responses. High-road transfer requires abstraction from the context of learning and a deliberate search for connections.

Transfer is today's benchmark for educational innovation, because traditional school systems often achieve only nonspecific and application transfer. For example, it is unclear whether students can apply their learnt knowledge in their daily lives or at work through context or far transfer. An important challenge in learning research is to promote far transfer by designing appropriate learning situations and environments.

Transfer is a true measure of learning as it correlates to the utility of a learner's knowledge. Simple drill and practice may achieve high scores on multiple-choice tests and good grades in

school, but this does not guarantee that students are able to use this knowledge in problem solving situations. Hence, many teaching strategies will look equally efficient if they are only evaluated based on facts that have been presented in the classroom. However, if we evaluate different approaches of teaching against transfer, for example, between sets of concepts, school subjects or school years, results will be more distinguishing (Bransford, Brown, and Cocking 2000). Bransford also states that to achieve transfer we need three things: (1) a sufficient initial threshold of learning, (2) learning with understanding, and (3) knowledge taught in a variety of contexts.

Systematically observing and measuring transfer is a difficult task. Detterman criticizes most studies on transfer for being flawed in one way or another. When subjects are told or can deduce that previous material may be useful then they are informed, but far transfer assumes that subjects are confronted with a new situation that is not linked in any artificial way to the learning situation (Detterman 1993). Detterman illustrates this on the example of Judd's experiments (Judd 1908), where children have been specifically instructed to use their knowledge about refraction to throw darts on targets under water. In addition, every experimental setting creates an artificial context that makes it hard to show context transfer. Yet, despite this critique, most studies measure transfer within the context of an experiment with a transfer task by observing differences in performance of solving this task. We follow this practice in evaluating our research.

Metacognition and Metacognitive Regulation

Metacognition, the art of thinking about thinking, reached widespread popularity through Flavell's work (Flavell 1979; Flavell, Miller, and Miller 1993). A metacognitive approach to instruction can help students in taking control of their own learning by defining learning goals, and in letting them monitor their own progress in achieving these goals (Bransford, Brown, and Cock-

ing 2000). The common assumption is that learning metacognitive skills (meta-learning) helps students improve in several subjects, not only one (e.g., White, Shimoda, and Frederiksen 1999).

Metacognition distinguishes the sub-categories of metamemory, metacomprehension and metacommunication (Wilson and Keil 2001). Metamemory includes knowledge about utilizing mnemonic strategies to improve memorization of items. Metacomprehension describes the intuition of how well one knows what one should know. It also allows the student to decide what to study next and which actions will lead to more success. Metacommunication is the skill to reflect upon one's own communication and to assess whether one has understood communicated information.

Zimmerman and Schunk (2001) distinguish metacognitive knowledge from metacognitive regulation. Metacognitive regulation (also self-regulation) states: "Students are self-regulated to the degree that they are metacognitively, motivationally, and behaviorally active participants in their own learning process" (Zimmerman and Schunk 2001). The key research questions addressed by self-regulation theories are:

- What motivates students to be self-regulating during learning tasks?
- Through what process or procedure do students become self-reactive or self-aware?
- How do self-regulated students attain their academic goals?
- How does the social and physical environment affect student's self-regulated learning?
- How does a learner acquire the capacity to self-regulate when learning?

Computer-aided instruction systems can adopt strategies to teach metacognitive skills, like metacomprehension, reflection, and self-regulation. If students learn these strategies, they will not only improve scholastic performance in the subject currently taught, but also be able to re-

flect better on their own knowledge in other subjects, and therefore show transfer of metacognitive skills. White et al. has created a system based on this strategy (White, Shimoda, and Frederiksen 1999), that we discuss in detail the section "Learning by Doing, Active Learning, Inquiry Learning and Related Pedagogies" in the next chapter.

Motivation

Educational literature notes links between learning, motivation, and attitude. Motivated students tend to spend more time on learning and pay more attention to the presented material. Motivated students learn more and learn better (Brophy 1998; Haskell 2001; Lepper et al. 1993; Stipek 1988).

We believe that learning by teaching agents increases a student's motivation as it shows similarities to approaches discussed later in this section: Students who teach to learn are in a position of power and control, which includes their responsibility for the tutee's success. Anchored instruction, which situates the student in a realistic context, stimulates fantasies of students. Additionally, the student is co-operating with our social agent, which adds an interpersonal motivation component. Our learning environment provides tools for exploratory learning (a simulation) which stimulate cognitive curiosity.

Cognitive theories about motivation distinguish extrinsic motivation and intrinsic motivation. Intrinsic motivators make us do things because we enjoy doing them, for example, we may like astronomy because we have a telescope and enjoy stargazing. Although a learning environment only can foster intrinsic motivation, extrinsic motivation is omnipresent in a scholastic environment and may interfere. Extrinsic motivators come in the form of rewards and punishments from the world around us. We thrive to get a good grade in school to obtain rewards from family

members (Sternberg 1995). Remarkably, there is an inverse relationship between intrinsic and extrinsic motivation. If a person, who is highly motivated intrinsically on a task, receives additional extrinsic motivators then their intrinsic motivation may diminish (Sternberg 1995).

Intrinsic Motivation

Malone and Lepper studied this kind of motivation in children, and found that some tasks are motivating by themselves, because the urge to solve them comes only from the problem and not from outside, as it would when studying to pass an exam (Lepper and Malone 1987; Malone and Lepper 1987). For example, many children spend several hours each day playing intrinsically motivating computer games, time that is essentially lost, or may even displace education. If we could prepare educational materials to make learning more intrinsically rewarding, we could improve learning environments. This may even include combining games and education. Thus, Malone and Lepper formulated their taxonomy of intrinsic motivation to analyze factors that make learning fun. They distinguish two groups of factors, individual and interpersonal motivation. We start by discussing individual motivators, which are challenge, curiosity, control, and fantasy, while competition, cooperation, and recognition are interpersonal motivators.

It seems to be commonly agreed upon that an optimal level of challenge that may vary between individuals, results in the highest motivation. Attractive tasks give the learner an explicit, fixed goal, while more open-ended learning environments, like Logo, may let the user choose emergent goals (Csikszentmihalyi 1978). Bandura and Schunk also stated that proximal (near) goals are always superior to distal (far) goals (Bandura and Schunk 1981). Another factor contributing to challenge is uncertain outcome, which seems to be optimal when the initial probability for succeeding in a task is 0.5 (McClelland et al. 1953). Thus, to create a motivating learning

environment, we need randomized tasks of variable difficulty that allow a success rate of 0.5, and give multiple levels of explicit goals. Performance feedback should desirably be frequent, clear, constructive, and encouraging. To enhance self-esteem a system should make performance goals personally meaningful and relevant (Malone and Lepper 1987).

A second component of individual motivation is sensory and cognitive curiosity (Malone and Lepper 1987). Sensory curiosity is stimulated by physical and psychological means (light, sound, zoom, highlight, and so forth) to increase attention, while cognitive curiosity uses lack of completeness, consistency and parsimony to evoke interest. Cognitive curiosity is satisfied by exploration of undiscovered terrain or facts and seems to be a very powerful motivator that occurs in many best selling games.

According to Malone and Lepper, power and control are the most cited explanations for the attractiveness of computer-games (Lepper and Malone 1987; Lepper et al. 1993; Malone and Lepper 1987). The perceived level of control makes a player feel competent and empowered. It is important that outcomes are contingent to the user's responses. Zuckerman et al., for example, showed that explicit choices by students enhance intrinsic motivation (Zuckerman et al. 1978). Usually, the illusion of choice is motivating enough to be noticeable (Langer 1975). Finally, the student's actions should have powerful effects. Large and spectacular effects due to minor alterations in the environment (graphics and sound) will lead to subsequent motivation (Lawler 1982; Papert 1980).

Fantasy plays a major role in being motivated, for example, when reading a book. It may help to satisfy unconscious emotional needs and lets us master situations that are not available to us in reality. A user would identify with another character if there were perceived similarity, ad-

miration, and salience of that character's perspective. Lepper and Malone suggested that students should be able to name actors in the environment.

Interpersonal motivations are competition, cooperation, and recognition. These factors are prevalent in multi-user Internet games. Competition may work for or against individual motivation (e.g., self-esteem), and it may have negative influences on social relationships. Cooperation can be used to master difficult situations. Recognition can be shown through the process of public performance, the product (e.g., painting of an artist), or the result (e.g., high-scores, certificate).

Flow

A different kind of motivation is that learning is often associated with the state of the mind that finds balance between challenge and skill, and people find themselves in *flow* (Csikszentmihalyi 1990, 1978). Although *flow* is a somehow elusive concept, we include it at this point to make the conceptually related measure Experiences that Energize (ETE) (Brophy 1998, 2003) better understandable. We describe ETE later in this section and revisit them in our discussion of experimental measures.

Csikszentmihalyi has found that people are generally unhappy when they are doing nothing and generally happy when they are doing things. Sometimes, individuals reached a state of the mind that made them very productive in the task that they perform. Csikszentmihalyi called this flow. Persons in flow feel completely involved in, and focused on their task because of their curiosity or training. They feel great inner clarity, know that the activity is doable, do not notice time passing, and feel intrinsically motivated. Flow is a highly activated state of the mind that puts the person performing a task in control. People in flow devote 100 percent of their attention

to the task they are performing. Putting a learner in flow some of the time would be a highly appreciable goal for learning environments. Unfortunately, there is no known way of inducing flow artificially as it has to come from within.

However, flow may be useful as evaluation metric of a system. Brophy has developed a measure of intellectual flow that he has called Experiences that Energize (ETE) (Brophy 1998, 2003). Subjects repeatedly report their energy level associated with an activity that they are currently involved in or have just completed, with a single seven point Likert question (see page 137 for a detailed discussion of this measure). The answers provide a rating of self-reported intellectual enjoyment that relates to the idea of flow.

Summary

This chapter discussed theories that build the basis for understanding of how people learn that influenced the design of our system in many ways. A peculiarity of the learning process is that learners integrate new knowledge with knowledge they already possess to construct their own understanding of the world. When asked to reproduce what they have learnt, learners show that they have transposed learnt stories into the modern world (Tripathi 1979), or that they have merged what they knew with what they learnt anew. This is the core idea behind constructivism, which describes how people accommodate new knowledge and handle contradictions to arrive at something they feel comfortable. In recent work, these ideas of constructivism seem to have an increasing influence on newly developed educational methodologies and computer aided instruction systems.

One issue illustrated in this chapter was that students may associate weak contexts with knowledge, or that knowledge simply stays inert (Whitehead 1929). Thus, learning requires rich

contexts. We presented work that suggests that situating learning in realistic problem-solving macrocontexts helps learning. An applied strategy to implement this theory in schools is anchored instruction, which helps transfer and learning. Transfer of learning is the ability to generalize learnt knowledge and apply it in new situations. Therefore, transfer is also a good benchmark for assessing how good a learning strategy or system is in helping learners to apply their knowledge in real-world situations.

Students often categorize school as boring, thus, increased motivation might help to stimulate interest and could indirectly improve learning. Lepper and Malone's Taxonomy of Intrinsic Motivation for Learning (Malone and Lepper 1987) helped us to identify criteria that make learning by teaching agents motivating. In addition, metacognition and metacognitive regulation describe the ability of students to control their learning through reflection and self-regulation. This plays a role in our context, because agents make students reflect on what they taught to the agent, and we are interested how this affects students.

This lets us arrive at the following conclusions:

- Foster transfer and deep learning by providing rich environments that are anchored in realistic macrocontexts already in the learning stage.
- Recognize that students merge new knowledge with what they already know.
- Intrinsically motivate students through learning by teaching. Students are in a position of power and control with the responsibility to teach, and realistic fantasy contexts stimulate cognitive curiosity. Additionally, co-operation with a social agent adds interpersonal motivation component.

This concludes the theoretical framework for learning for our system design. In the next chapter, we will discuss related work in computer aided instruction and pedagogies that have been derived from practical research.

CHAPTER III

COMPUTER AIDED INSTRUCTION

This chapter surveys past work in computer-based learning, and instructional systems. We will adopt the term *computer aided instruction* in its all encompassing meaning, but we do not intend to limit this term to intelligent computer aided instruction systems, which has been used as synonym for intelligent tutoring systems (Wenger 1987). We describe a number of representative strategies and systems and relate them to the theories that were described in the previous section. This chapter is not an exhaustive overview of systems that have been developed in the domain, but it focuses more on mainstream research and technologies that may contribute to designing a learning system based on the principles of constructivism, transfer, metacognition, self-regulation, and motivation.

The first section of this chapter discusses specific learning strategies that have been employed in classrooms and in computer-aided instruction. Among others we discuss: learning by being taught (e.g., Anderson 1995; Wenger 1987), and approaches influenced by constructivism such as active learning, learning by doing, and guided discovery methods. Research that is more recent integrates peer tutoring and self-explanation with traditional systems. In general, the trend goes towards hybrid systems that employ multiple strategies at once. In the final paragraphs, we focus on learning by teaching.

The second section of this chapter introduces designs and implementations of representative systems that implement these strategies in intelligent computer aided instruction systems. Intelligent tutoring systems traditionally use the tutoring approach, but also implement a wide range of

strategies. Constructivists initially devised the not very successful idea of microworlds (Papert 1980) that supported pure discovery learning. Later mutual interactive and intelligent learning environments combined multimedia, anchored instruction, situated cognition, and other approaches with coaching into various levels of guided discovery.

The last section of this chapter gives a brief overview of how computer agents may help in achieving educational goals. Of special interest are design, social, and educational aspects of the agents.

Pedagogies and Instructional Strategies

This section gives an overview of various teaching and learning strategies from the theoretical perspective that the scientific community has developed and incorporated into computer-aided instruction systems. The topics covered include cognitive tutoring, discovery learning, active learning, inquiry-based learning, peer and reciprocal tutoring, learning by teaching, and learning of metacognitive strategies. All these strategies have been shown to be useful in designing learning environments and have enhanced the effectiveness of teaching tools. We also discuss the strengths and weaknesses of each approach. It should also be made clear that these approaches are not mutually exclusive, and students could engage in activities that incorporate multiple strategies within a learning session. For this reason, we defer the discussion of practical implementations and system designs of these strategies to the next section.

The Cognitive Tutoring Approach

Original cognitive tutors adopted the principle of drill and practice to educate the user in domains, such as algebra or geometry. The most cited and well-known tutor designs are based on

Anderson's ACT* theory (Anderson 1983), which fuelled the development of cognitive tutors and a number of intelligent tutoring systems that are discussed later. The first application of this theory was a tutor that taught students to program in LISP. Later, this system was adapted to implement a geometry tutor (Anderson 1995). These systems have now evolved to a generic mathematics tutoring system that is sold by Carnegie Learning® (CarnegieLearning 2004). After learners study a paper curriculum in mathematics class, students practice on the tutoring system. As a general principle, these systems implement production rules that follow the students' problem solving trace step by step. The system does not intervene as long as the student stays on a reasonable solution path. Once a student's solution deviates from a pre-defined path, the system intervenes immediately, flags the incorrect answer, and provides feedback that leads the student in the right direction. This may prevent the student from making further wrong moves¹. Anderson further explains that students also can request help prior to answering. These messages should be short and to the point. However, some students tend to overuse help, and the system has to moderate requests (Anderson 1995). Nevertheless, students cannot go on until they have solved a problem without help at least once.

The advantage of this approach is that it keeps the student's activities focused on the correct problem solving trace, which makes the problem solving process efficient. By observation and practice, the student may quickly pick up the correct problem solving procedures. Anderson states that keeping students on track reduces their chances of developing misconceptions. Additionally, he notes that students' attitudes are quite positive in classrooms using cognitive tutoring systems, and that cognitive tutors encourage peer help in how to use the systems.

¹ This is the original cognitive tutoring strategy. Today there are numerous improved variants of this approach that let students make some mistakes or use alternate strategies to provide feedback.

Critics of this approach point out that it prevents students from exploration, and from learning how to recover from incorrect solution paths, or to identify wrong solutions, and leaves very little room for creativity, exploration and inventiveness (e.g., Self 1990; Rickel et al. 2000). More recent work (Alevan and Koedinger 2002) extends cognitive tutors with metacognitive strategies, as we see next.

Self-Explanation

Self-explanation is the process of spontaneously explaining to oneself available instructional material in terms of the underlying domain knowledge (Chi et al. 1989). Researchers operationalize self-explanation by letting students explain the reasons for choosing an action, or the whole solution path of a problem after providing a correct answer (e.g., Weerasinghe and Mitrovic 2002). Thus, the term self-explanation is also used if the explanation is enforced by, or given to a tutoring system.

Alevan and Koedinger combined cognitive tutoring with self-explanations (Alevan and Koedinger 2002). Students were required to justify their answers with correct explanations; otherwise, the cognitive tutor did not let them move on. They found that students who explained their steps to the cognitive tutor performed better than students who did not explain their problem solving steps.

A learning by teaching system uses similar mechanisms as self-explanation, because students elaborate their understanding during teaching. We discuss these similarities in the section about learning by teaching. This concludes our discussion of tutoring strategies, and we will focus on strategies that involve the learner more actively in the learning process.

Learning by Doing, Active Learning, Inquiry Learning and Related Pedagogies

This section discusses teaching strategies that have been devised to supplement classroom instruction. The same ideas can also be applied to designing learning environments. The roots of these pedagogies date back to the educational theories of John Dewey (1859-1952).

Two pedagogical approaches that comply with the principles of constructivism are learning by doing and active learning. These strategies are based upon the idea that people learn best by doing things, not by being passive recipients of knowledge (Lander et al. 1995; Modell and Michael 1993). Some studies have found that higher order thinking skills are not acquired through didactic approaches, but rather through learner's active involvement with information (Collins, Brown, and Newman 1989). Discussion, reading, writing, evaluation, analysis, synthesis, and teaching are tools of the trade to support active learning.

From this general principle, research derived three closely related pedagogical approaches: project-based learning (Katz and Chard 2000), problem-based learning (Boud 1985; Boud and Feletti 1991; Torp and Sage 2002), and inquiry-based learning (Dewey 1938). Project-based learning focuses on developing a product or creation, and may be combined with any other strategy. Problem-based learning is that content is introduced in the context of complex real-world problems (the problem comes first). The approach stems from medical instructional research and appears to be conceptually similar to the idea of macrocontexts/anchors in Anchored Instruction.

Inquiry-based learning approaches are derived from the educational theories of John Dewey (Dewey 1933, 1938). In a generic inquiry-based learning environment, students work in groups and cycle through five tasks: ask, investigate, create, discuss, and reflect¹ (compare with the

¹ The phases of this strategy are similar to Flexibly Adaptive Instructional Design and the STAR Legacy learning shell cycles. We discuss both of them later in this chapter.

STAR Legacy cycle in Figure 4 on page 61). First, students ask meaningful questions (generate a hypothesis) to create genuine curiosity about real life experiences. This curiosity provides the motivation for students to gather information, research resources, and conduct experiments. The learner can then recast the question that she has asked or redefine the investigation on the fly. After enough information is gathered, the learners begin to integrate their knowledge to create new thoughts and ideas. Then, inquiry learners share their ideas with others and engage in discussions. The students conclude the cycle with a reflection phase where they evaluate if a solution has been found, or new questions arise, and the cycle is repeated.

One computer-based learning environment that implements inquiry-based learning combined with meta-learning (learning of metacognitive skills) is the SCIWISE project (White, Shimoda, and Frederiksen 1999). The system includes a number of task adviser agents, such as Ingrid Inventor and Harry Hypothesizer, who help the student in specific situations with strategic advice, for example, the agents suggest how to create hypotheses, plans, and ideas. White et al. believe that this makes students learn cognitive processes, which facilitates inquiry learning, collaborative work, and peer tutoring. These task advisers help the students to develop metacognitive skills while solving problems. The authors state this as follows:

"Our claim is that an agent's meta-level expertise can be internalized by students and then consciously invoked if, through a process of reflected abstraction (Piaget, 1976), it has been identified, explicitly labeled, and interacted with as a functional unit. By internalizing expertise as a system of such functional units in the form of advisors, they become accessible to reflected abstraction and conscious control, enabling students to 'put on different hats' and 'invoke different voices' when needed as they solve problems or engage in inquiry learning."
(p. 176, White, Shimoda, and Frederiksen 1999)

A major risk in implementing the pedagogies, which we have discussed in this section, is that engagement of students can be mistaken for learning (Schwartz, Brophy et al. 1999). Students may be enthusiastic about the task and work hard, yet assessments of their understanding of domain knowledge may produce disappointing results (Barron et al. 1998). In addition, implementation of these approaches in regular instruction requires more time than traditional lecture based instruction. Nevertheless, active learning and learning by doing influenced the development of discovery learning. Problem based learning and inquiry learning influenced the Flexibly Adaptive Instructional Design Theory. We discuss both pedagogies in the next sub-sections.

Discovery Learning

According to Schank and Edelson, pure discovery learning (Bruner 1961) works as follows: People reason about situations they encounter by referring to similar situations they have encountered earlier, and if people see a new situation and all their expectations are met, then no learning takes place. However, if their expectations are violated learners will start to question their experiences and begin to integrate new knowledge into their existing knowledge structures (Schank and Edelson 1989). A pure discovery system will allow a student to learn actively by solving problems. Learners typically choose strategies freely without intervention or specific help.

The main weakness of the pure discovery process is that it may be quite time consuming to accomplish learning. In some cases where students have full autonomy, but very little comprehension of the domain, learning may not occur because students cannot interpret the situations they see (Brown and Campione 1996). The student may use trial-and-error to accomplish the learning task or solve problems, and explore the environment in an inefficient way. Sometimes, students may feel unmotivated and lose interest in the learning task, because they do not make

progress (e.g., Malone and Lepper 1987). Other researchers demonstrated that pure discovery does not work very well (e.g., Klahr and Nigam 2004; Mayer 2004).

To alleviate these shortcomings researchers developed guided discovery and expository methods. To provide guidance, a teacher or system provides hints, direction, feedback, and coaching. Expository methods provide the correct solution before learners start the discovery process. Mayer concludes that students in a pure discovery condition performed the worst, and learners in a guided discovery condition performed the best on tests of immediate retention, delayed retention, and transfer in solving new problems, although guided discovery requires the most learning time (Mayer 2004). He also states, "Students need enough freedom to become cognitively active in the process of sense making, and students need enough guidance so that their cognitive activity results in the construction of useful knowledge."

A discovery learning system conforms to the principles of constructivism because it allows students to construct their own understanding of the domain, to learn how to develop solution strategies, to derive and improve hypotheses, to learn from their own mistakes, and to deal with incomplete knowledge. Implementations of pure discovery systems are Microworlds (Papert 1980). An example of a guided discovery system is the Adventure Player learning environment (Crews 1995). We introduce both systems in the section "Existing System Designs" (page 49).

Flexibly Adaptive Instructional Design Theory

Flexibly Adaptive Instructional Design (FAID) has been introduced to help foster deep understanding while simultaneously promoting the skills for problem solving, collaboration and communication through the use of problem based learning followed by more open ended project based learning (Schwartz, Brophy et al. 1999; Schwartz, Lin et al. 1999). According to FAID,

instruction should integrate (1) learner-centered environments, which integrate knowledge, skills, and attitudes of students, (2) knowledge-centered environments, which are organized around big ideas that support learning, (3) assessment-centered environments, which help students to create a representation of their knowledge that makes their thinking visible to them and their teachers and thus revisable, and (4) community-centered environments that support collaboration among students. FAID combines a problem-based/anchored instruction approach with a modified inquiry learning cycle (initial challenge, generate ideas, multiple perspectives, research and revise, test your mettle, and go public - see Figure 4 on page 61). This pedagogical design was applied in the STAR Legacy learning shell, which we will discuss on page 60. FAID and STAR Legacy are pedagogical foundations for our system design.

Peer Tutoring, Cross-age Tutoring, and Reciprocal Tutoring

Peer tutoring implies that tutor and tutee are of the same age, while in cross age tutoring the tutor comes from an advanced class and is older. Publications suggest that peers understand each other better because they are cognitively closer. Allen and Feldman found that third and sixth graders were more accurate than experienced teachers in determining from nonverbal behavior whether age-mates understood lessons (Allen 1976). Peer tutors may also have advantages in explaining materials due to their cognitive similarity (Cohen 1986). A related strategy is reciprocal tutoring (Palincsar and Brown 1991) when students alternatively take on the role of tutor and tutee. Due to the close relationship of peer tutoring and learning by teaching, we defer discussing cognitive benefits of tutors to the next section.

Peer and cross-age tutoring are not easy to adapt, as simply assigning tutor and tutee may lead to problems. Tutoring requires training and communication skills. Cohen suggests to assess

a potential tutor's comprehension before assigning them to tutoring tasks (Cohen 1986). Lippitt states that especially in cross-age tutoring also lower performing students can be effective tutors (Lippitt 1976). In addition, peer tutoring can give rise to status problems by making tutees feel inferior, and cause friction between tutor and tutee (Gaustard 1993). As we have discussed in chapter one, computer-based systems can avoid these dangers by removing one or the other human factor by substituting a system or an agent. In the following paragraphs, we discuss a representative computer aided instruction system based on reciprocal tutoring.

Scott and Reif have implemented a reciprocal tutoring system to teach Newtonian physics (Reif and Scott 1999; Scott 1991). A computer and a student take turns in coaching each other. They have demonstrated that their system performs almost as well as individual human tutors. The system is implemented in AuthorwareTM¹. In the following paragraphs, we give an overview of this system.

As first step, the computer coaches the student. The system detects any errors in the student's implementations, and helps them to diagnose the reasons for their incorrect answers. Then the computer guides students to the correct solution. Each tutorial conducted by the computer coach has one problem that can be solved by applying several aspects of Newton's law that students can select from a menu. Coaching the student commences in three tasks: (1) The student decides to use a concept from a list, which the computer assesses and only draws in a diagram if the concept is applied correctly, otherwise the computer coaches the student with hints and other feedback; (2) To let the computer generate equations based on Newton's laws, the student has to click on appropriate parts of the system's diagram; and (3) The computer asks about qualitative relationships in the diagrams.

¹ AuthorwareTM is a visual authoring tool that allows creation of interactive (e.g., educational) content with little programming skills. It is similar to another product, which is called HyperCardTM.

In the reciprocal step, the student coaches the computer. The student chooses actions to perform, and assesses the implementations generated by the computer, which may make mistakes: (1) The computer randomizes the solution path, and the student has to select the proper steps in the right order; and (2) The computer implements this solution path step by step, and the student has to approve every step. Should the student approve a wrong step, the system asks the student to check more carefully and gives hints.

We believe that this system combines reciprocal tutoring with traditional cognitive tutoring by implementing learning and correcting false moves of the student. However, our goal for a learning by teaching system is to give the student more freedom to explore. We accomplish this by moving towards a guided discovery approach in the design of our learning environment.

Learning by Teaching

One only knows something if one can explain it. Giambattista Vico, 1710

The idea of learning by teaching is closely related to studies in peer tutoring, which has provided some results that illustrate benefits for peer tutors. Cohen found that preparing to teach "facilitates long-term retention, as well as aiding in the formation of a more comprehensive and integrated understanding" (Cohen 1986). Gaustard observed: "Strikingly, student tutors often benefit as much or more than their tutees" (Gaustard 1993). In addition, learning by teaching seems to provide motivational and cognitive benefits for the tutor. In preparing to teach, students consider the larger context of the knowledge, spontaneously discover flaws, question the purpose or mention alternatives significantly more often than students preparing for a test, and "the challenge of teaching others appears to create the sense of responsibility that is highly motivating to individuals of all ages" (Biswas, Schwartz et al. 2001).

Learning benefits of tutors have been observed in the following publications (Cohen 1986; Michie, Paterson, and Hayes-Michie 1989; Palincsar and Brown 1991; Gaustard 1993; Nichols 1994; Chan and Chou 1997; Obajashi, Shimoda, and Yoshikawa 2000). In the following paragraphs, we discuss systems that combine learning by teaching with computer-aided learning.

One of the first approaches implementing learning by teaching a computer is the pilot study of Michie et al. (Michie, Paterson, and Hayes-Michie 1989). The knowledge domain consisted of basic algebra concepts (solving of linear equations). Students taught a computer by providing example solutions to problems. From these examples the system learned one general rule to solve equations by using a machine learning algorithm (ID3 decision tree induction algorithm, Quinlan 1986). The student used the commands: (1) show to provide an example to the system, (2) look to display the induced rule, (3) test to try to solve a problem, and (4) ask to go through the induced rule step by step to debug the rule. Their hypothesis was that having taught the machine, a pupil must have mastered the given skill and that this use of a rule-induction algorithm has significantly better advantages for the learner than learning through drill and practice. Michie et al. compared three conditions: drill and practice, and two versions of their learning by teaching system with 30 subjects. The researchers concluded: "Although, motivational gains were not observed, students in the experimental condition showed enough learning gain to warrant a full-scale study."

Palpetu et al. conducted experiments to determine whether explaining concepts to others has advantages over plain studying. The idea was that dialogues with others would expose possible argumentative flaws through previously unconsidered justifications. The group proposed a system where the student teaches the computer in a fact-based science domain (Palthepeu, Greer, and McCalla 1991). In this system, users want to perfect their own domain knowledge, while the

computer starts as *tabula rasa*. This system does not need an expert or domain model (see page 50), but only makes inferences from what it has learnt. Input from students improves the domain knowledge base and the learners evaluate the system's inferred responses. The researchers concluded their system performed best in situations when the student "almost knows" the domain. With their approach dialogue history provided information to keep the dialogue active, focused and pedagogical. The authors raised the question if a learning by teaching system needed a student model, and if it would be helpful to have a domain model to guide the student towards unexplored areas.

Chan and Chou tried to answer the question whether the computer should teach the student or vice versa. They explored various combinations of reciprocal tutoring (Palincsar and Brown 1991) where human and virtual agents worked together in dyads and triads, in various combinations as tutor, tutee or companion to solve recursion problems in LISP (Chan and Chou 1997). In this approach, agents (including humans) took turns in filling the role of tutor or tutee after each problem. Unfortunately, we do not know that a full study followed the preliminary experimental trials with only five subjects per condition. The results suggested that distributed reciprocal tutoring (virtual tutor + real tutor + real learner on different computers), outperformed centralized reciprocal tutoring (virtual tutor + real learner) and intelligent tutoring (virtual tutor + real learner, not switching roles). Students in the "learning by tutoring"¹ condition (virtual tutor + real tutor + virtual learner) performed worse than in other conditions. The authors had expected the low performance of learning by (observing) tutoring group, because those students essentially only watched, and occasionally enhanced the virtual tutor's answer to the virtual tutee. However, students preferred this condition to other activities that involved more work. The view of learning

¹ The authors use alternatively learning by teaching and learning by tutoring to explain their abbreviation LBT. Later, they restrict this term to the meaning of learning by observing tutoring between virtual tutor and virtual student with the option of enhancing the tutor's instructions.

by tutoring adopted in this paper does not conform to our view of learning by teaching, which requires the student to take on a more active role in the teaching and learning processes.

Nichols pointed out that learning by teaching systems need to focus on defining a mutual communication language to limit the content of the dialogue to the essential facts in the domain (Nichols 1994; Nichols 1994). His system DENISE started as *tabula rasa* (empty knowledge model) and built a *learnt model* from the dialogue interactions with the student in the domain of qualitative economics. Each resulting learnt model contained 65 percent terms not used by other tutors. The models had many unrelated and general terms (irrelevant knowledge like religion or civil war) that often could not be found in the economics literature. Nichols found that students created long causal chains of little value with general terms. He also pointed out that learning by teaching systems were less reliant on a domain model than intelligent tutoring systems. We believe that the weakest point in this design was that students lack an intuitive way to review and modify the learnt model efficiently. For example, students could not look at a visual representation of the entire learnt model.

The newest learning by teaching prototype to date was developed by Japanese graduate students at Kyoto University (Obajashi, Shimoda, and Yoshikawa 2000). The system enhances a web-based computer aided instruction system with personalized agents that played the role of virtual students. Students first studied materials on their own by reading the basic lecture in their web-browser. Then, the learner has to solve given exercises, which allowed the system to modify the behavior of the learner's virtual student. In the next step, the learner taught the virtual agent by typing text, which was then evaluated by the agent. The paper did not describe how this evaluation took place, only that the agent was able to recognize if the student copied and pasted text from the lecture materials. After teaching, the learner asked the virtual student agent ques-

tions, and corrected its answers. In the fourth step, virtual students of different tutors meet in a virtual classroom, where they present their knowledge and a virtual teacher provided them with correct answers, while the human tutor observed the dialogue. Additionally, this system provided an online, anonymous question and discussion room, where the learners could communicate about the material. The virtual students' questions came from a question database that depended on the human students' answers to exercises during teaching. The virtual student learned the answers to these questions. The system was evaluated against a control group, which studied only from text pages ($n = 20$ for each group) and the authors claim that their learning by teaching system deepened understanding, that learners could easier monitor their own understanding and gained diversity of understanding by observing other learners' thought processes.

Remarkably, most learning by teaching approaches use machine learning strategies to learn knowledge from the tutor. However, there is currently no evidence that it is necessary or beneficial to use a learning algorithm to maximize the learning benefits *of students*. One flaw in current systems that use machine-learning algorithms is that they create esoteric internal representations that are not comprehensible or viewable by the user, and eventually are hard to modify. Moreover, students have difficulty to understand why or how a computer has learnt certain concepts from what was taught.

We think that learning by teaching benefits from letting students teach the computer by constructing shared representations that are visible to the tutor and the agent during teaching and testing. These shared representations replace (to various degrees) hidden student models. Thus, learners can trace incorrect responses of the computer or agent transparently, and in this process view their own progress. Students can develop a feeling for what is correct by reflecting on their taught representations without requiring refined troubleshooting or debugging skills for complex

internal knowledge representations. In addition, this strategy conforms to a constructivist approach, because students effectively build their own knowledge structures, and then study how well their taught agent can solve problems or answer questions with these structures. If their agent makes errors, students have to reflect on these errors, and find out how to teach their agent better. In reality, students seem to be learning from their own mistakes, and from the feedback they get about these mistakes.

In addition to peer tutoring and the systems mentioned here, we see similarities between learning by teaching and self-explanation approaches. As we have discussed before (page 35), current research operationalizes self-explanation as explaining to a cognitive tutor (e.g., Miller, Lehman, and Koedinger 1999). A learning by teaching system that lets students explain solutions to an agent involves inherent self-explanation and reflection, because students have to focus on their agent's errors, and figure out why these errors occurred, and how to correct them. For example, Chi states:

"Thus, since the construction of self-explanations, directed by oneself without any guidance from another, is an effective means of learning, then it seems that the construction of explanations, elicited by others, may have the same beneficial effect and may account in part for the effectiveness of tutoring." (Chi 1997)

Other Pedagogical Approaches

Kafai has introduced learning through design and learning by programming. In her research, she has verified the assumption that "making something is a powerful way of learning" (Kafai 1995, p. 286). Kafai led a project where children had the task to design games in Logo to teach other children about elementary school mathematical concepts, like fractions. Children had to fill de-

sign sheets where they could jot down ideas of how to structure their game to teach another child and then implement it. These design sheets were collected and evaluated as case studies. Kafai concluded that making games for learning combined the creative tasks of designing and playing, and this helped students to develop their individual styles, and build a learning culture in the classroom. However, only a few students succeeded in integrating fractions into their game project, but learners developed good understanding of complex programming concepts. Similar to Nichols' system and discovery learning, this strategy provided too much freedom and allowed students to deviate from the path of learning target concepts. Otherwise, this approach appears to be very similar to the concepts of active learning and learning by doing.

Mengelle and Frasson introduced the learning by disturbing strategy. In their intelligent tutoring system the authors used actors, who play different pedagogical roles (Aïmeur et al. 1997; Mengelle 1996). One of them is a troublemaker agent, who is unreliable in its cooperation and may mislead the learner in an attempt to improve the learner's self-confidence. We will discuss actors in more depth in the section on Agents (page 64), where this system is revisited from a different perspective. The paper did not present evidence for or against learning by disturbing as it has evaluated a whole system where this strategy played a minor part. We have included this strategy here as an idea that could be evaluated in the future as part of a learning by teaching system, as it has similarities to a teachable agent that learns unreliably.

Summary

In this section, we discussed various learning strategies that were developed to implement learning theories or aid students in conjunction with computer aided instruction. As a basic strategy, we discussed discovery learning, which lets the student explore a domain without guidance, but

might be time consuming, inefficient and sometimes unmotivating for students, especially when they do not make progress in learning and understanding. We can supplement this discovery approach by adding scaffolding, feedback, and coaching to create guided discovery learning environments.

The constructivist philosophy of learning is supported by learning by doing and active learning, which help students to gain hands-on experience in selected topics. These pedagogies find their application in anchored instruction, project-based learning, problem-based learning, and inquiry-based learning.

A different approach is learning by teaching, which in essence also adopts a constructivist approach in which students have to explore, study, and understand the domain to be able to teach it to others. We can summarize that learning by teaching systems (1) Have motivational and cognitive benefits for the tutor (Biswas, Schwartz et al. 2001; Gaustard 1993); (2) Require a limited communication language to prevent students from wasting time by teaching irrelevant knowledge (Nichols 1994; Nichols 1994); and (3) Potentially improve on learning gains seen in studies with self-explanation.

Existing System Designs

In this section, we discuss existing formal and informal designs of intelligent tutoring systems, cognitive tutors, microworlds and intelligent learning environments, each supporting different theories, or pedagogies that we have discussed previously. We start with the theory and implementations of intelligent tutoring systems, and continue by introducing microworlds, which offer a more constructivist perspective on computer-aided learning. Last, we will discuss direct predecessors of our work.

Intelligent Tutoring Systems

Intelligent tutoring systems (ITS) use artificial intelligence techniques to provide one on one tutoring (Wenger 1987). A number of ITS systems implement the cognitive tutoring strategy (Koedinger 2001; Anderson 1995). Wenger describes intelligent tutoring systems as knowledge communication systems that have intelligent mechanisms to adapt their teaching and feedback to the learner's needs (Wenger 1987). Most intelligent tutoring systems use a standard architecture, containing four modules: (1) the expert module containing the domain knowledge, (2) the student module, which accumulates information about the student's knowledge, misconceptions and behavior, (3) the curriculum module, which includes the pedagogical expertise, and (4) the interface module (Figure 2).

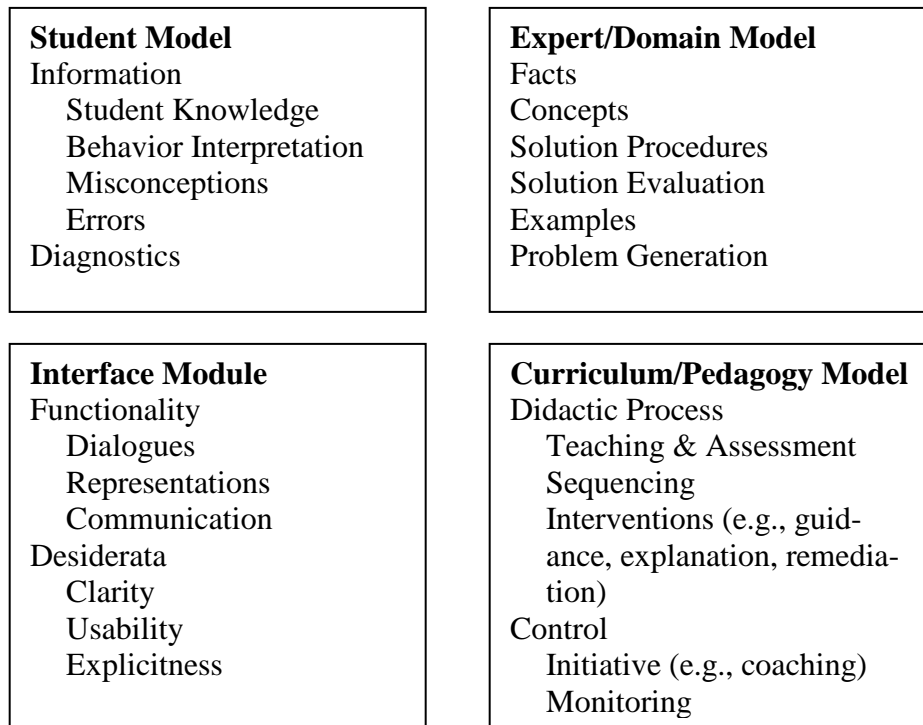


Figure 2. Intelligent Tutoring System Architecture adapted from (Wenger 1987)

The *expert module* (domain model) is a dynamic representation of the knowledge domain. It contains domain knowledge, such as facts, concepts, processes, productions, and procedures required to solve problems. The model allows evaluation of the student's solutions, and can provide examples of correct problem solutions. The *curriculum module* (pedagogy model) sequences the curriculum by comparing the expert and the student model, has testing procedures that indicate the extent of the student's knowledge, contains strategies that focus on how to teach best, and controls feedback. If the system is sensitive to misconceptions, this module contains remediation procedures. The *interface module* provides a uniform view of the environment to the user. It allows the user to interact with the system (as the curriculum module prescribes) by accessing the tutorial discourse, problems, examples, scores, progress summaries, representations, and re-

sources (e.g., examples, diagrams, lectures). The probably best investigated module of intelligent tutoring systems is the *student model*, which we discuss in more depth in the following paragraphs.

The student model captures the knowledge status of the learner at any point in the teaching process. Simple student models only keep track of topics that the student has mastered, and which topics have not yet been covered (Knowledge Spaces, Doignon and Falmagne 1999). More advanced student models record misconceptions, build bug libraries, and implementing VanLehn's repair theory (VanLehn 1990; Brown and VanLehn 1980), or record which teaching strategies work best for a specific student. Unlike other modules, the student model deals with information that is specific to individual learners.

Sison states in his student model survey that every student modeling system is obviously limited to observable responses of the student to a stimulus in a domain (Sison 1998). He calls this student behavior, which can be plain input, an action, a result, or intermediate scratch work. Each of these options entails a different behavioral complexity and, thus, requires different strategies to extract useful modeling information. Sison categorizes systems by how many atomic behaviors they need to gain some information about the student. He states that systems, such as Anderson's tutors (Anderson 1995) build one extreme of the spectrum, as they verify each "key-stroke" of the user and, thus, use single behaviors to build the user model. Other modelers derive higher-level structures through rule or decision tree induction by using multiple behaviors as input (Sison 1998).

Self (1990) states that the philosophy behind intelligent tutoring systems is only concerned with "knowledge communication" (Wenger 1987) and that the goal of teaching is to "transmit a particular subject matter to the student" (Ohlsson 1986). Self states that this view neglects the

general constructivist educational philosophy, and that for effective learning, knowledge cannot be communicated or transmitted, but has to be actively constructed by students themselves.

However, the research community developing intelligent tutoring systems proved to be resistant to the adaptation of ideas from educational psychology for a long time (Self 1990).

Combining the architecture of an intelligent tutoring system with constructivist approaches may be problematic. The process of accommodation may change what a student already knows and make it incorrect when new knowledge is learnt after a learning system has verified it. In addition, misconceptions of students may not be easily identified in an assessment. For example, students may answer that the world is round, but think of a flat and round pancake shaped world (Vosniadou and Brewer 1989). Additionally, for building an accurate student model, it is essential to know the intention of the student. However, when interacting with computer systems students, especially young students, like to use generate and test for finding solutions. This buries conclusive interaction data in noise and often makes it impossible to build a meaningful student model. To help students learn critically monitoring their solution and recognizing inconsistencies, constructivist approaches allow learners to descend incorrect solution paths. Traditional tutoring systems, especially those that used the cognitive tutoring strategy, often did not pay attention to this issue, and did not support learning of reflective skills.

A representative system of one extreme position (drill and practice) in intelligent tutoring systems in the mathematics domain is Assessment and LEarning in Knowledge Spaces (ALEKS) (Doignon and Falgagne 2003). It is based on the theory of knowledge spaces (Doignon and Falgagne 1999; Doignon and Falgagne 1985), which models knowledge as units to be learnt. Basic units must be learnt before advanced units; hence, each unit is a precondition to another unit. Students can choose to study any unit for which they have learnt all precondition-knowledge.

Knowledge is taught through examples and drill-exercises that are accessed as web pages. When learners solve a mathematics problem five times (with different numbers) correctly, ALEKS considers it learnt, remembers this in the student model, and opens new choices for the student. At the beginning of the curriculum, learners have to solve placement exercises that determine which units they know. Then, students select a module with satisfied prerequisites, read the instructions, and solve the related exercise. In this system, the student model is the set of all problems solved correctly for five times, which is a subset of the knowledge space. Repeating learnt material is the choice of the student, as the system is not sensitive to misconceptions, bugs or other higher-level inferable constructs. The system is available online with a full math curriculum that matches standardized tests. To my knowledge, the knowledge space theory was not evaluated in studies with subjects, but mathematically proven to be correct.

Computer Based Microworlds

In 1980, Papert introduced a new vision for teaching children in a constructivist way through pure discovery learning, and published it in the influential book *Mindstorms* (Papert 1980). This vision sees students working in computer environments, called microworlds that allow them to explore a concept in many aspects without having to worry about the full complexity of the real world. Papert defines a microworld as a self-contained world in which certain questions are relevant and others are not. Each microworld has its own set of assumptions and constraints, and students use their natural habits of exploring, playing, and creating to accomplish the learning tasks in the microworld. In this sense, every microworld is an abstraction of the real world, which focuses on relevant aspects of the problem-solving task, and students learn this abstracted knowledge through discovery.

Pure microworlds are single domain, pre-defined simulations of real-world phenomena that require students to acquire skills that they do not possess. Learners often struggle in setting their own goals when tasks are too open-ended, and when they lack self-regulation and other metacognitive strategies. Papert implemented his idea in the programming language called Logo to let students acquire "powerful ideas" about the geometry domain (Papert 1980). In this microworld, students manipulated the movement of a graphical cursor (the turtle) with programming instructions to create geometrical shapes.

As an implementation of pure discovery learning, microworlds share the shortcomings of the discovery approach, and may fail to teach target knowledge if the learning context is not well engineered, and if gaming aspects overpower educational goals (e.g., Miller, Lehman, and Koedinger 1999). Evidence against pure discovery learning and microworlds is documented in many articles (e.g., Pea and Kurland 1984; Klahr and Nigam 2004; Mayer 2004). However, if the idea of a microworld is incorporated into a larger learning environment, and is combined with feedback to implement guided discovery learning, we can draw from its constructivist benefits. Mayer (2004) summarizes the literature as follows: "Children seem to learn better when they are active and when a teacher helps guide their activity in productive directions."

Miller et al. recently explored learning effects when giving students different goals in their electronic field hockey microworld (Miller, Lehman, and Koedinger 1999). This microworld was intended to help students develop a qualitative understanding of electrical charges. Students could place several fixed and one moveable charge on the computer screen and start a simulation to shoot a "puck"-charge into a goal. The work assigned 30 college students to three conditions:

no-[hockey]goal¹ (experiment with the environment without obstacles and net to prepare solving a problem with obstacles and net), a specific-path (six prepared levels plus trajectories to hit the puck into the net), or a standard-goal (six prepared levels no trajectories). The no-[hockey]goal and specific-path conditions showed the highest learning gains in this experiment. The authors concluded that it had to be made clear to the learners what learning outcomes were expected from them, and how they should use the microworld. Thus, the authors conclude that, "goal-based problem solving will transfer to pedagogically relevant material exactly when the goal-dependent relationships coincide with pedagogically relevant relationships." In addition, the gaming aspects of the microworld often distracted students without letting them learn qualitative relationships of charges. Augmented microworlds with guided discovery have similarities to interactive learning environments, which we discuss next.

Interactive and Intelligent Learning Environments

Interactive learning environments (ILE) are built around constructivist principles of learning. The student has freedom to explore the environment, and learn from mistakes. Wilson defines interactive learning environments as:

"... environments that allow for the electronically integrated display and user control of a variety of media formats and information types, including motion video and film, still photographs, text, graphics, animation, sound, numbers and data. The resulting interactive experience for the user is a multidimensional, multisensory interweave of self-directed reading, viewing, listening, and interacting, through activities such as exploring, searching, manipulating, writing, linking, creating, juxtaposing, and editing." (p. 186, Wilson 1992)

¹ The authors used the label no-goal for this condition, but the students received the instructions to prepare for a situation with obstacles and net. The assumption is that the authors use goal to refer to the field hockey goal (net) and not the task's goal.

Obviously, we can distinguish different levels of interactivity. Schwier and Misanchuk created a three level hierarchical taxonomy: reactive interactivity (in response to a given stimuli, tutoring), proactive interactivity (user generation of unique constructions) and mutual interactivity through use of artificial intelligence (Schwier and Misanchuk 1993). Simple tutoring systems and basic cognitive tutors fall into the class of reactive systems. Due to our focus on artificial intelligence based systems, we henceforth discuss mutual interactive systems. A mutual interactive system adapts to the learner, advises, assists and modifies the environment to optimize learning, and shows intelligence and adaptive behavior.

It appears that some authors adopted the term intelligent learning environment (also ILE) for mutual interactive learning environments, but at this time, no formal definition of an intelligent learning environment seems to exist. All systems that we have reviewed, which classify themselves as intelligent learning environment, use artificial intelligence (e.g., Forbus and Whalley 1994; Nichols 1994). Other authors may also use the term to simply distinguish their systems from intelligent tutoring systems. Various authors state: "An intelligent learning environment (ILE) is primarily one, which understands the individual student well enough to be able to determine individualized actions." (Self 1991), and "We use the term intelligent learning environment to denote an intelligent artificial environment, which does not necessarily replace a human teacher." (Gust et al. 1999)

Research at Vanderbilt University's Learning Technology Center (Crews et al. 1997) has embraced the following view of interactive learning environments. These systems offer safeguards to help students back on track if they are stuck, and present multiple representations of a concept to the student. Thus, they help the learner to develop a deeper understanding of conceptual relationships in the domain. Such environments also draw from situated cognition and an-

chored instruction, by putting learning into a realistic, complicated macrocontext that provides a rich problem-solving environment. The student is often asked to solve new problems by using available resources, while the system itself observes the student, and offers help when it is needed. The environment typically does not correct the learner by providing a correct solution, but may coach the learner through various levels of hints. Artificial intelligence facilitates these goals, if necessary.

In the following paragraphs, we describe two intelligent learning environments that use these strategies: Smart Tools and Anchored Interactive Learning Environments. Both environments use Adventures of Jasper Woodbury episodes (Cognition and Technology Group at Vanderbilt 1997) as anchoring contexts and are direct predecessors to the system that is introduced in this work.

Smart Tools (Owens et al. 1995) combine anchored instruction with macrocontexts and microworlds in the domain of teaching functional relationships, like, distance-rate-time. Students create graphs and tables (their own Smart Tools) to explore and solve problems. Task activities in the environment have the purpose of leading students to generalize from specific problems to concepts of the domain. The system provides a simulation in which the student can explore different settings, a table to write down experimental results and a time-distance graph to plot results. The system works in two modes: the exploration mode in which students can perform experiments and collect data, and the challenge mode in which students solve a set of problems to verify their graphs and tables.

Crews et al. developed Adventure Player, an Anchored Interactive Learning Environment (Crews et al. 1997). This environment situates the student in the Rescue at Boone's Meadow adventure, where Jasper finds a wounded eagle in a remote meadow and the learner has to devise a rescue plan to get the eagle back to a veterinarian as fast as possible. The microworld provides

limited resources, like vehicles with payload limits, speed limits, footpaths, and so on. Students need to consider different candidate solutions to solve the problem optimally. Crews derived the following principles for such a learning environment in his work:

- Implement the principles of generative and active learning.
- Provide anchored learning
- Facilitate and allow guided discovery learning
- Explicate domain knowledge through multiple representations (e.g., tables and graphs)

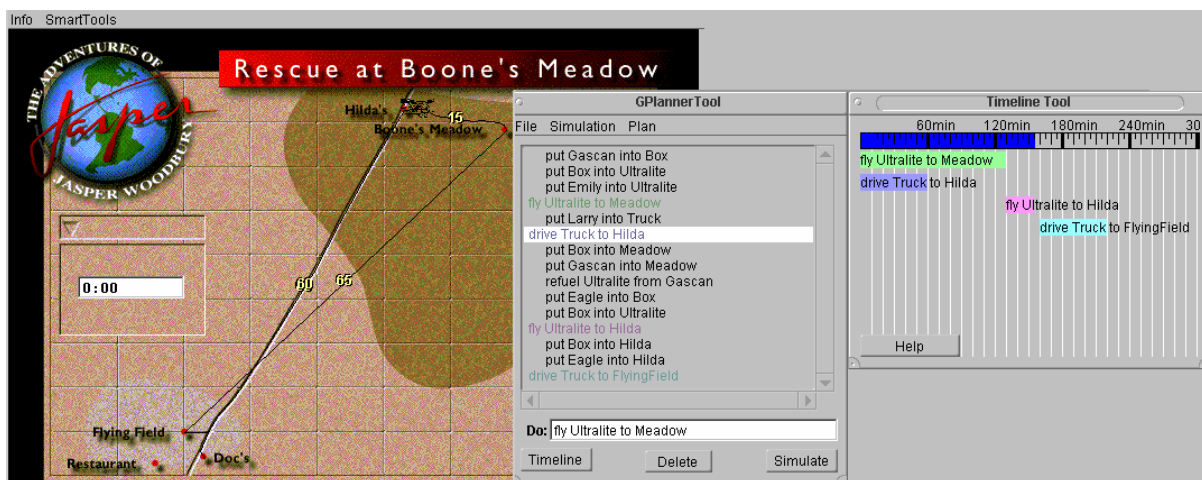


Figure 3. Reengineered Version of Rescue at Boone's Meadow with Planner and Timeline

Crew's Adventure Player asked students to create an optimal rescue plan in a planning-notebook, which contained a list of plan steps, like "fly Ultralite from City to Meadow." Plan steps could execute concurrently. As students developed their plan, they optimized it by observing how their plan steps worked in the simulation. After the student chose a planning action, the learning environment checked the validity of the planning action and responded appropriately. If

an action was invalid in the real world due to physical limitations, like loading the rescue aircraft on a truck and driving it to the destination, the system disallowed it and provided an instant explanation. Students could also derive plans that failed, for example, because the airplane ran out of fuel, and see the results in their simulation. Students learned which planning actions had which consequences and solved many distance rate time problems on the path of finding an optimal solution. Additionally, the system implemented the principle of minimal help – give students just enough help to solve the problem (Reusser 1993). Hence, the system included a coaching component that gave feedback on five different levels with increasing specificity. The results showed that students using the full system solved the challenge significantly more often than students using a core system without the simulation environment for exploratory learning.

STAR Legacy Learning Shell

Schwartz et al. implemented the Flexibly Adaptive Instructional Design (see page 38) as Software Technology for Assessment and Reflection (STAR) Legacy learning shell to help curriculum designers organize learning activities into pedagogically sound inquiry-based learning cycles (Schwartz, Brophy et al. 1999; Schwartz, Lin et al. 1999). The system, which was initially implemented in HyperCard^{TM1}, is a foundation for the cycle design of our learning by teaching agents environment. The student solved progressively more complex challenges by iterating several times through the phases of a cycle. The design philosophy behind STAR Legacy's learning cycles conforms to constructivism, situated learning, anchored instruction, and inquiry learning. A brief outline of the system states that ...

"Research has shown that effective instruction often begins with an engaging challenge or scenario that introduces the lesson and invites student inquiry. The combination of a chal-

¹ HyperCardTM is a multimedia audiovisual content authoring system similar to AuthorwareTM.

allenge, interactive activities, and multiple opportunities for sharing, assessment, and revision is called a STAR Legacy module." (IRIS 1999)

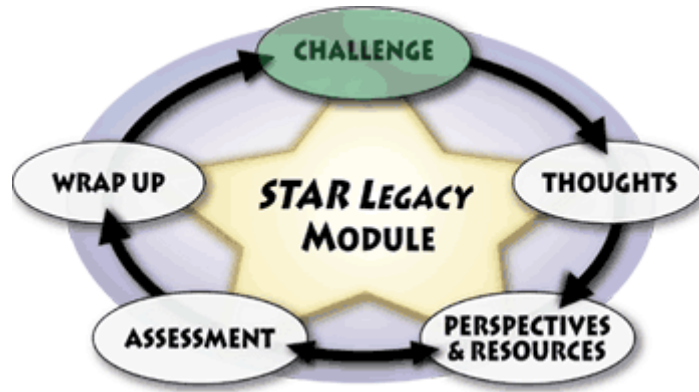


Figure 4. STAR Legacy learning shell cycle structure (IRIS 1999).

In each challenge, learners progressed through a cycle (see Figure 4) with the following steps: (1) Review a new challenge; (2) Generate ideas and thoughts; (3) Access multiple perspectives, resources and solve the problem; (4) Assess their own learnt knowledge; and (5) Write a summary, and compare with others (see Figure 4). Generating ideas captured the student's pre-conceptions, making successive viewpoints of expert perspectives and resources more relevant and interesting, because in the following step it led students to recognize, "Oh! I didn't think about this (... or like that)" (Bransford 1990). Figure 4 shows a double arrow between the steps of perspectives and assessment, which indicates that learners could revise their understanding after getting feedback following formative assessment. The term Legacy captures the notion that students preserve their reports and notes they made during the learning and problem-solving phase for future students to use as exemplars. In the next chapter, we introduce how we have adapted this strategy to learning by teaching.

Related Teachable Agents Research

Betty's Brain is based on constructivist and inquiry based learning (Davis et al. 2003; Leelawong et al. 2002; Leelawong et al. 2003; Leelawong et al. 2001). Students teach an intelligent software agent, Betty, by constructing concept maps about science topics. To do this, students add concepts to an initially blank concept map (they fill Betty's empty brain), and connect the concepts with causal relationships that express increase or decrease relationships between concepts (Figure 5). Later, students can ask questions or quiz their agent. The agent uses the concept map to derive answers using qualitative reasoning mechanisms (Leelawong et al. 2001). Betty's answers lead students to identify mistakes in their concept map, and this helps them reflect on their own understanding. Thereafter, students can access and review learning materials, revise what they know, and improve their concept maps.

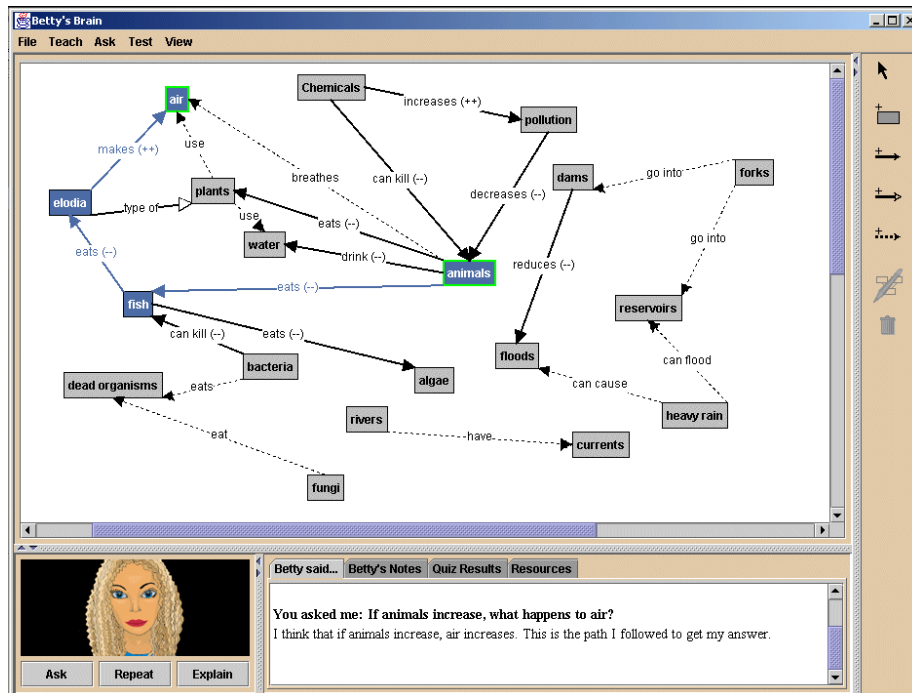


Figure 5. Teachable Agent Betty's Brain models an ecosystem domain

Research on Betty's brain established the role of formative feedback through questioning and quizzing (ask a set of questions; plain correctness feedback) the agent in the science/concept map domain (Leelawong et al. 2002; Leelawong et al. 2003). Questioning the agent implies that students can ask the agent questions like the one illustrated in Figure 5, and the agent answers by animating its reasoning on the concept map. Students quiz by asking the agent a whole set of expert prepared questions, and receive the agent's answers plus correctness feedback from the teacher. Results indicated that providing students with opportunities to quiz their agent decreased the amount of irrelevant information and increased the proportion of causal information in students' maps; whereas having opportunities to query their agent increased the interconnectedness of concepts in students' maps. These results held despite observations of students during the study, which indicated that students who quizzed might have been overly focused on "getting the quiz questions correct" rather than on understanding the domain (Leelawong et al. 2003).

Agents

At the beginning of this work, we mentioned that learning is, in some aspects, a social process, defined by social interaction in schools and in the community. It was determined that helping somebody to understand material may be rewarding for the tutor (Biswas, Schwartz et al. 2001). This motivates the use of agents to promote deeper understanding. The fundamental question for deriving the software architecture for teachable agents is how to define and what to expect from an agent embedded in our system.

An agent, in everyday sense, is one who is authorized to act for or in the place of another to accomplish a goal (Merriam-Webster 2005). Our computer agents were designed to act to accomplish the goal of improving learning through means that we have discussed in the previous sections. Remarkably, this does not require our agents to learn presented knowledge using machine-learning techniques like a human. Rather our agent's goal is to facilitate the student's learning process by participating in exploratory tasks, helping the student to formalize his knowledge using a systematic human-agent shared representation, and helping the student reflect on her problem solving by applying the derived representation to problem solving tasks.

Etzioni and Weld (Etzioni and Weld 1995), and Franklin and Graesser (Franklin and Graesser 1996) enumerated traits of agents. While discussing them, we apply them to our learning by teaching agents context. Teachable agents should have the ability to selectively *sense* the actions of the user in the learning environment and *act* upon them. Our agents should exhibit *autonomy* by directing their actions towards the goal of helping the user gain deep understanding of domain material. To a limited degree our agents need to *collaborate*, for example, a teacher-agent may need to correct the answers of the teachable agent. Teachable agents need to *learn*

from the user and *react* by expressing behavior consistent with their learnt knowledge, to make misconceptions in the user's knowledge evident. To which degree *inferential capability* of teachable agents has benefits for the learning of its user may be subject of future research. Our agents need to be *temporally continuous* and preserve their state (knowledge) over the duration of their use for each student. Another important trait of teachable agents is their *personality*. To demonstrate characteristics of a good student, they must believably learn, ask challenging questions, and be motivating for a learner. *Interactivity* between agents helps coordinate learning and lets the teacher agent point the student to the right materials. *Mobility* (that agents can move transparently to different hosts or systems) is not required for the scope of our current research.

Frasson et al. see intelligent tutoring systems evolving towards implementing multiple strategies with agents that model human behavior in learning situations (Frasson et al. 1996). They call these cognitive agents *actors*, who are reactive, adaptive, and intractable. In their system design, the authors include a tutor, supervising a learning session, a troublemaker who is a colearner agent that solves problems often incorrectly, and this may improve the learner's self-confidence, and an artificial learner that aids the dialogue and synchronizes the human learner's activities with different agents. Having a troublemaker in the environment may have similar effects as a teachable agent that learns or reproduces knowledge unreliably (e.g., it solves problems 90% of the time correctly). Assigning different roles to different agents may address a number of issues that facilitate learning. For example, an agent that is being taught cannot know whether knowledge is correct or not, however, a teacher agent can believably correct the user or another agent. Unfortunately, Frasson does not report any data or evaluations on subjects for this system.

Once computer agents learn with human students, the question arises how competence of these agents influences collaborative learning. Hietala and Niemirepo looked into this issue and

divided subjects into four groups of low achieving and high achieving crossed with introverted and extroverted students (Hietala and Niemirepo 1998). The study found that introverted high-achieving students liked strong companion agents. However, these subjects tried to solve the problems by themselves first before asking the agent. Lower achieving students preferred to collaborate with weak agents and generally asked for a suggestion, before trying to solve the problem by themselves. The implications from this study are that due to the variation across different personality types of learners, we would have to give different students different agents. However, this would introduce confounding variation in our experiments and the cost of implementation would rise. Therefore, a medium strong agent is the best compromise.

Another dimension is the embodiment of the agent. Embodiment is typically achieved through gestures, hand and body movements, animation, and synthetic speech. A study by Johnson and Rickel reported statistically significant increases in learning when pedagogical agents were animated (Johnson and Rickel 2000). A multi-modal effect with visual and auditory qualities and additional practical and theoretical advice could yield further improvements in learning. The study suggested that the benefits of animated pedagogical agents increase with the complexity of the problems. Visual animation is currently a hot topic in research, and its benefits appear to vary with applications in different domains. Our research group is working on animating teachable agent Betty in the Betty's Brain learning environment. Because animation is not part of this design, we will not discuss any further papers on this topic here.

Summary

In the last two chapters, we explored cognitive science literature on learning, computer aided instructional systems and agent technology that helped us to define a framework for an intelligent

learning environment. Our research specifically focuses on the distinguishing features of learning by teaching agents, even though most of the justifications in the previous chapters apply to the whole learning environment that we introduce in the next chapter.

The psychological knowledge of how people learn helps to define the base effectiveness of our system. Constructivist learning (Piaget 1954) is a complex process of integrating new knowledge with preconceptions while providing means for correcting misconceptions. This involves placing the student in a feedback loop that involves learning new concepts, practicing their use by solving problems, reflecting on the solutions generated, and using the feedback from the system to improve one's learning and understanding of the relevant concepts. We accomplish this by adapting strategies applied in the STAR Legacy Learning Shell, which organizes knowledge in inquiry-based learning cycles (Schwartz, Brophy et al. 1999; Schwartz, Lin et al. 1999).

As we have established, situating or anchoring a learning task in a meaningful context (Bransford 1990; Cognition and Technology Group at Vanderbilt 1993; Crews et al. 1997) will enhance the likelihood that transfer of knowledge occurs so that students can apply it in new situations. Thus, we introduce the problem-solving domain as elaborate macrocontext (as required by anchored instruction), from which we draw specific problems and proximal goals of increasing difficulty. Additionally, this may motivate the learner by demonstrating task value. While discovery learning may accomplish this in the end, if the student stays motivated, we may be able to accelerate this process by engaging the student in learning by teaching.

Research in computer-aided instruction developed in two parallel mainstreams following the exploratory approach (e.g., Papert 1980) and tutoring approach (e.g., Anderson 1995; Wenger 1987) separately for many years. Only in the last decade, researchers started to explore synergistic approaches. The tutoring approach mainly focused on devising intelligent tutoring systems,

which followed a strict modular architecture that taught knowledge to the student. Research suggests that teaching correct procedures and knowledge, while flagging the student's mistakes, is a good way to teach. Many papers in the area of intelligent tutoring systems describe how to model the student and derive misconceptions and remediation from the students' actions. However, one of the shortcomings of the tutoring approach is that it often neglects the necessity for students to learn from their own mistakes, and monitor their progress using metacognitive (reflective) skills. This shortcoming is addressed in exploratory approaches.

The early developments of the exploratory camp were a number of microworld-based systems (Papert 1980), which were pre-defined simulations of real-world phenomena that allowed the learner to explore freely, observe the consequences of making mistakes, and learn from these interactions. In these environments, learners sometimes struggled to set their own goals, unless they already had a good understanding of the domain of study, and possessed strong self-regulation characteristics. However, the biggest disadvantage was that though students often enjoyed the gaming aspects of microworlds and learned to use a microworld effectively, they still failed to learn parts of the target domain knowledge.

Recent developments in mutual interactive and intelligent learning environments combined anchored instruction, and situated cognition with coaching to provide guided discovery learning (Crews et al. 1997) and emphasized the role of representations in teaching mathematics (Owens et al. 1995). Students solved problems, while using smart tools that assisted them in acquiring knowledge and discovering misconceptions. In the mathematical domain, creating a representation, like a graph, helps the learner to acquire knowledge consistent with the constructivist idea, while a simulation aids the student in establishing the ground-truth for attempted solutions.

Observations in peer tutoring showed that tutors also learn when they teach their tutee (Cohen 1986; Gaustard 1993; Biswas, Schwartz et al. 2001). Thus, letting the students learn for themselves in the process of teaching an agent may improve the effectiveness of a learning environment even further. This may help if agents point out the contradictions in the knowledge structures that the students formulate while teaching or it may introduce new insights that the learner has never had before. Interaction with agents can also provide motivation through a social context and the desire to help one's agent, especially in young learners. As the teaching task involves dealing with other's mistakes it is possible that this aids the ability to manage own mistakes. A primary causal mechanism for this might be the acquisition of metacognitive, reflective skills, which are believed by many authors to transfer readily between domains. In this sense, we hope that the findings that we discussed here help us to successfully design and evaluate a framework for an intelligent learning environment that can serve as a model implementation.

CHAPTER IV

SYSTEM DESIGN

This chapter describes the design and implementation of our learning by teaching agents environment. We follow the design principles for such environments that were established by Biswas et al. (Biswas et al. 2005), which are outlined below.

- Build on well-known teaching interactions to organize student activities.
- Agents are taught through visible representations.
- Keep the start-up costs of teaching agents low.
- Ensure the agents have independent performances that provide feedback on how well they have been taught.

To build on well known teaching interactions, we used an anchored interactive learning environment design similar to the one in Adventure Player (Crews 1995; Crews et al. 1997). Adventure Player is an interactive learning environment linked to the Adventures of Jasper Woodbury series (Bransford 1990; Cognition and Technology Group at Vanderbilt 1997, 1993). A second component that we used to structure teaching interactions are learning cycles that were adapted to the learning by teaching paradigm from STAR Legacy (Schwartz, Brophy et al. 1999). The anchoring context situated the students' problem solving in a realistic context, for example, students were told that they were interns at a company. In this context, students are asked to solve many distance-rate-time problems quickly and correctly. This provides intrinsic motivation (challenge, control, fantasy, and cooperation) to gain deep understanding of the domain material in the context of problem solving activities. The need to be fast and accurate also motivates stu-

dents to develop their own Smart Tools¹ (Owens et al. 1995) to solve problems. To adapt STAR Legacy's inquiry cycles to learning by teaching, we merge phases two and three of the cycle (Figure 4) into the teach phase, which lets students access resources and perspectives of the teacher agent, and learn for themselves, before they teach their computer-based agent. Following this, students receive feedback through formative assessment in the quiz phase, and they can return to the teach phase to revise their understanding. Then they proceed to the test phase to conclude a cycle and receive summative feedback after taking a test.

Students teach agents by creating a visible, shared representation of their understanding (in our case a graph), and teach the remaining declarative and procedural knowledge of how to use these representations through dialogue choices and demonstration. In our domain of distance-rate-time problems, the representation structure for problem solving is a graph. Therefore, in each of our learning cycles students learn how to use, create, and modify graphs and teach this to the teachable agent, Billy. Because students also have to teach the agent procedures how to use graphs (e.g., read it), but performing actions by demonstration alone was not sufficient for unambiguous learning, we used a mixture of student-agent dialogues along with demonstrations of problem solving steps for this purpose. For example, the agent wants to solve a specific problem and asks, "How do we start?" Then the student selects an action from a set of dialogue options that the agent offers, and Billy implements the student's choice, or asks for a demonstration on how to proceed. This reduces the start-up costs of teaching our agents, as students can demonstrate solutions and choose actions among dialogue options.

We provide formative feedback of how well our agent has been taught by letting the students quiz the agent. A student selects from a set of predefined problems. Billy solves the selected

¹ In our context, Smart Tools are graphs and tables.

question and provides an answer if he was taught by the student beforehand. Otherwise, the agent says that it does not know a solution, and needs to be taught how to solve the problem. If Billy can provide a solution, the student can request an explanation (demonstration and verbal explanation), and then choose between teaching the agent better and asking the teacher for the correct solution. We discuss this in the next section in more detail.

In the following section, we give an overview of the cycle architecture. Some design issues are dictated by our experimental design, which splits students into two groups: the experimental (learning by teaching) condition and the alternate condition. Because Billy is not taught in the alternate condition, some shared resources and the cycle do not mention teaching an agent. In this chapter, we focus on the learning by teaching system and only occasionally mention experimental design issues, which are discussed in the following chapter.

The Cycle Architecture

The system organizes the student's learning by teaching in cycles. We have adapted this idea from the STAR Legacy learning shell (Schwartz, Brophy et al. 1999; Schwartz, Lin et al. 1999) to the learning by teaching paradigm (for a discussion of STAR Legacy see page 60). The learning cycles are organized in sequence of increasing difficulty and present the activities in an organized way.

We have replaced the individual phases of the STAR Legacy approach with a general introduction, a problem introduction, and the three phases of teaching, quizzing, and testing. For the

sake of our experimental design and our anchored instruction approach¹, we label the last three phases Make Smart Tool, Try Smart Tool, and (solve problems for) Real Customers (instead of Teach Billy, Quiz Billy, and Test Billy) to hide the fact that there are two treatment groups (see Figure 6). In the following sections, we describe each step of a cycle in more detail.

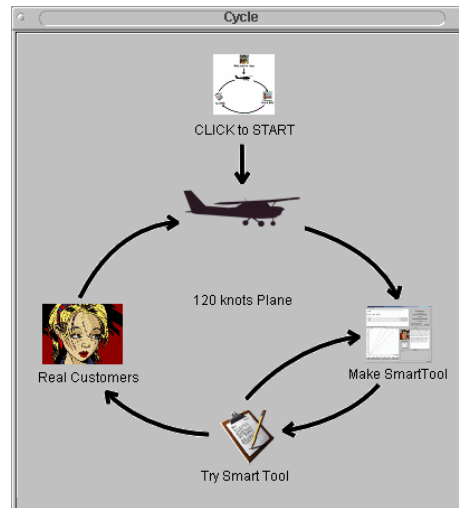


Figure 6. Teachable Agent Learning Cycle

General Introduction (Anchoring Context)

Before a student uses the system for the first time, we introduce the anchoring scenario in the general introduction. This situates the student in a realistic fictional problem-solving context that stimulates the student's fantasy for motivational purposes. In our case, the anchoring context involves a small airplane and jet transportation company, JetEx. The company is owned by the trio Jasper Woodbury the CEO, Emily the pilot, and Larry the engineer. They have hired the learner²

¹ We situate the learning task in a realistic context: The student teaches the teachable agent (an intern at a jet transportation service) to make smart tools that help compute flying durations for real customers. See page 14 for more information on anchored instruction.

² Due to our experimental design, the teachable agent Billy is not introduced in shared lectures. Later on the computer, learners of the experimental group will find out that, they work together with Billy.

as summer intern to interact with customers. The characters are linked to the Adventures of Jasper Woodbury (Cognition and Technology Group at Vanderbilt 1997), which allowed us to reuse introductory movies with professional actors to introduce the anchoring context. This story situates our students in a realistic macrocontext, from which we draw all our mathematical problems.

This anchoring context is initially presented as a twenty-minute long movie to all participants simultaneously in class. This movie flashes back to the childhood of Jasper, Emily and Larry and shows how the friends won a contest at a local traveling agency by constructing smart tools for distance-rate-time problems. This has obviously contributed to their current success in running their company. At that point, the story switches back to the current time, when the students are told that they have been hired as summer interns by just that company. Students are informed that they will learn from the company's mathematics expert, Ms. Mathie, who is the teacher agent in our environment, how to develop smart tools. These smart tools will then help students to provide quick answers to questions of customers that want to rent aircraft on transportation between different cities. Later, when students work with the learning environment, they can review a summary of this introduction as pages in a web-browser by clicking on the first icon in the cycle ("Click to Start" in Figure 6).

In the learning by teaching condition, we introduce students additionally to our teachable agent Billy, who says that he has trouble understanding graphs. Billy elaborates further that he would appreciate the student's help to learn more about the domain, so that he can successfully solve problems for the customers. Students in the alternate condition meet a different Billy, who writes a report about Jasper's company without being involved in solving mathematics problems.

Problem Description

At the beginning of each cycle, students read a web page that introduces data, problems, and goals (Figure 7). This page includes pictures of airplanes, a view of the cockpit, and technical data about the planes. The problem statement contains essential and unessential data similar to how materials were presented in the Adventures of Jasper Woodbury video resources (Cognition and Technology Group at Vanderbilt 1997). This prepares students for another complexity of real-world problem solving, which is the extraction of essential information from realistic problem contexts¹. Traditional word problems in mathematics books (microcontexts) do not develop this skill. Additionally, this wealth of information may fuel the student's natural curiosity and fantasy about airplanes, which could influence their motivation (Malone and Lepper 1987).

¹ Our current system only fosters this skill during the problem introduction, but future versions may improve on this.

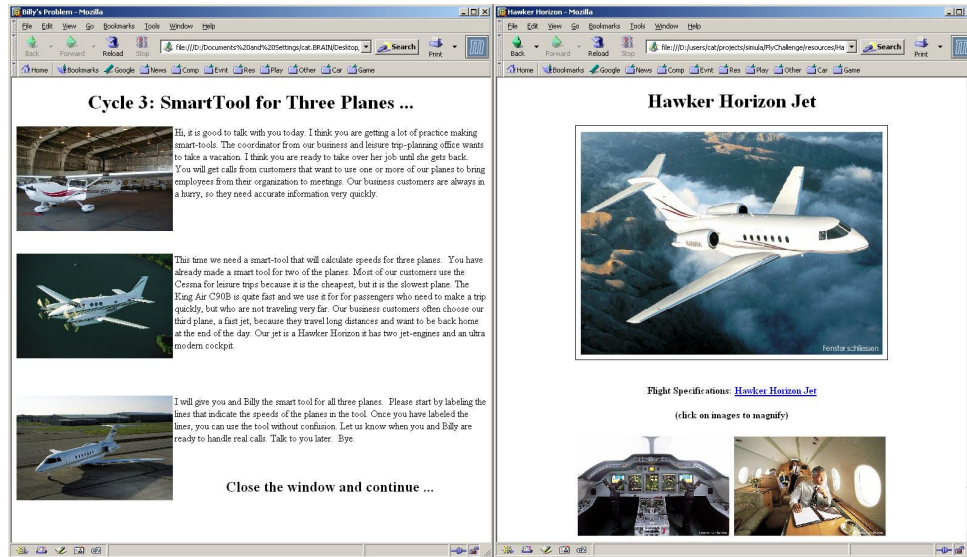


Figure 7. Problem Description of Cycle 3

Teach Phase

As we discussed earlier, the teaching task involves an initial phase where the student teacher prepares for teaching by learning, reviewing, and organizing material in a manner that it can be taught before engaging in the actual teaching task itself. Teachers scout, choose, access, and then organize material in a meaningful way, and learn by reading, planning and reflecting, before teaching. We do not expect that novice tutors will differentiate these steps, therefore the tasks of preparing to teach and teaching will occur together during the teach phase. Students may switch between asking the teacher agent and teaching at their disposition.

In the teach phase, students use three components of the system: (1) the simulation, (2) the smart tool (an interactive graph), and (3) the agent interface (Figure 8). The simulation provides feedback if a distance-rate-time problem has been solved correctly, but students cannot use it to find problem solutions directly without using the graph or calculating. Students learn by teaching the agent how to use or create a smart tool in each cycle. In this process, students communicate

with the teachable agent in dialogues to tell it what to do next, and ask the teacher agent about domain knowledge if they are stuck. Details about the design and implementation of the dialogue and smart tool components are provided later.

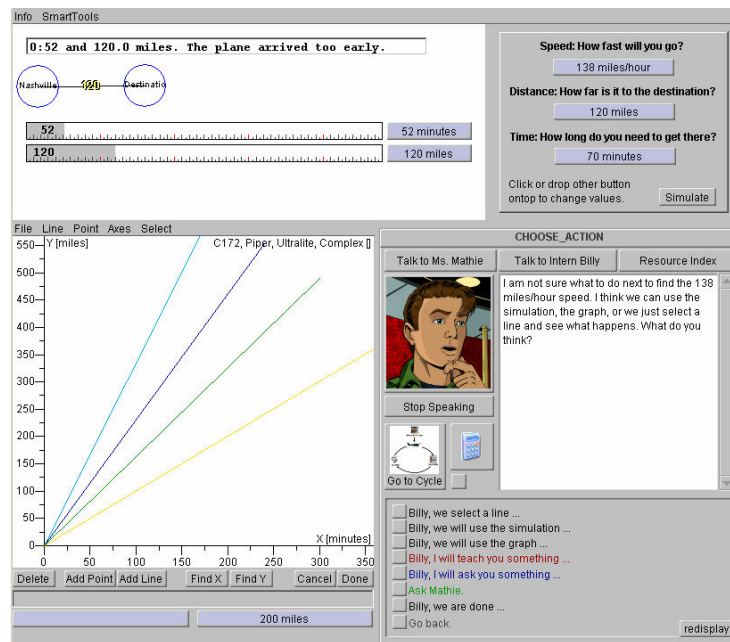


Figure 8. User Interface in the Teach Phase: Simulation (top), Graph (left bottom), and Agent Interface

In the teach phase the student has two primary goals: (1) To use, create, or modify a given graph in order to "teach the agent" (student and agent share this representation); and (2) To teach Billy directly in a dialogue by telling him what to do step by step. In each cycle, upon entering the teach phase, the teachable agent initiates the dialogue by repeating parts of the problem description¹, and then offers a few choices of how to proceed to the student. To start, the student has three primary choices to teach the agent: (1) use the simulation, (2) use the graph tool, or (3) start talking to the teacher or teachable agent. However, these choices depend on the task and, like a main menu, offer major paths in solving the problem. Two additional choices offer to teach

¹ In the control condition, the teacher agent will repeat the problem, because Billy plays a different role.

or ask the agent. Through these dialogues, the agent offers options of how to proceed to the student, which are explicit and proximal goals (Bandura & Schunk, 1981; Csikszentmihalyi, 1978). Still the student is in control and can explore freely by solving the problem on his own (run simulations, draw the graph), or asking the teacher for more information if the choices provided in the dialogue with the agent seem unattractive. Thus, students can choose the amount of interaction with the agent, and the amount of guidance from the agents.

Some students will attempt to solve the problem on their own before teaching and work on basic domain knowledge with the teacher agent; others will follow closely the options that the teachable agent provides. The teachable agent often refers the student to the teacher agent to explore appropriate topics. We think that this choice avoids problems with strong and weak agents, as they have been described by Hietala (Hietala and Niemirepo 1998).

In the mathematics domain, the student has to teach procedural and declarative knowledge to the teachable agent. First, students learn knowledge by explanations and demonstrations from the teacher and try it out in the learning environment. Then, the learner starts talking to Billy. In some dialogues, our teachable agent asks the user to demonstrate a procedure (like how to read a graph), while in others he asks for declarative knowledge. The learner teaches declarative knowledge by choosing an answer to a *what-is* question of the agent in the dialogue interface (see examples that follow), and procedural knowledge by a combination of dialogue and demonstration.

The overall communication can be looked upon as a mechanism by which the student explains to Billy how to solve individual steps of a problem. A similar approach, *self-explanation*, has been used in cognitive tutors (Alevan and Koedinger 2002) and has produced significant improvement in learning.

Teaching the Teachable Agent – Two Examples

In this section, we illustrate how the student teaches an agent on two examples. In the next three paragraphs, we discuss how students teach the agent the procedure to read a graph. Then, we describe how learners teach declarative knowledge to the agent.

The main learning goal in cycle one is to understand how to read a graph. For this, the student first requests instruction from the teacher agent, and tries to understand it. Then, the student selects, "Billy, I will teach you something..." in the agent dialogue. This will branch to a new dialogue with a choice of several procedural and declarative topics. For our example, the learner chooses, "Teach Billy how to read the graph when miles are given." Then, the agent asks, "What should I do to find how long a trip of, let's say ... 350 miles takes?" The agent interface offers the following four choices to the student: (1) Billy, press "Find X" in the graph tool; (2) Billy, press "Find Y" in the graph tool; (3) I will teach you later; and (4) Ask the teacher.

Now, the tutor has to teach the agent by suggesting the first step. If the student teaches the agent correctly, she will select choice number one. The agent will reply, "OK, let's start with finding X..." and the agent presses the "Find X" button in the graph tool. This shows two blue reader lines in the tool, which the user can manipulate by individually dragging them with the mouse. When find X is used, only the horizontal reader line displays a number that changes when it is dragged with the mouse (more details on this when we discuss the graph tool). Then, Billy asks, "Now what do we do with 350 miles? Please show me until you get the result." The agent interface displays four new choices: (1) Billy look, I have the result; (2) I will teach you later, let me try out myself for now; (3) Sorry, wrong thing. Let's start over ...; and (4) Ask the teacher.

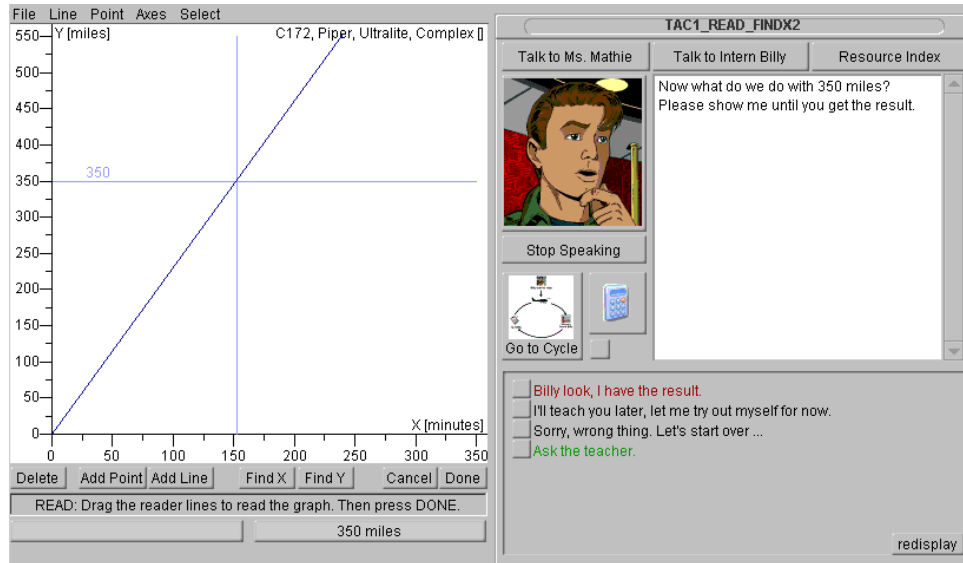


Figure 9. Teaching the Agent to Find Minutes when Miles are given.

At this time, the student has to finish reading the graph by first moving the horizontal reader line to 350 miles, and then intersecting both reader lines on the velocity line in the graph (Figure 9). To display the result, the user has to press the "Done" button in the graph tool. Then, she tells the agent, "Billy look, I have the result." If the tutor forgets to press "Done", then the agent will ask, "I cannot see any result. Did you click the 'Done' button in the graph tool?" Finally, the agent will note, "So that is the result... 152 minutes. I will try to remember that." This concludes teaching Billy how to read a graph for finding time when distance is given. Using an almost identical procedure, the student can teach the agent how to find distance when time is given.

Teaching declarative knowledge is less involved. For example, to teach the agent what an ordered pair is, the student has to answer the following question correctly, "How can I write down a point? My mathematics book says. $P = (20, 30)$ is an ordered pair. I suppose an ordered pair is ordered." The agent interface displays the following choices: (1) $P = (y, x)$ first Y and second X coordinate, or 20 minutes, 30 miles; (2) $P = (y, x)$ first Y and second X coordinate, or 30 miles, 20 minutes; (3) $P = (x, y)$ first X and second Y coordinate, or 20 minutes, 30 miles; (4)

Ask Billy, "How can you use a point?"; and (5) Ask Ms. Mathie about points. The student has to select one of the first three choices to teach the agent. Choices to teach the agent use the key-color red. After teaching, using choice four will provide the agent's answer, which is an interpretation of what the student has taught. Choices that cause the agent to react use the key-color blue. If taught right, the agent answers the following, "I always use the first number of an ordered pair on the x axis and the second on the y axis. This is the same order as the alphabet. X is before Y. That is easy to remember." The student can verify taught knowledge, other than verbally, in the quiz phase.

Quiz Phase

After the student decides that she has taught Billy enough, she will quiz the agent to check his knowledge in the quiz phase. Again, this section will not illustrate the internal workings of the environment, but the interaction of the user with the learning system. We discuss the differences in the user interface for the alternate condition to the section on experimental design.

The quiz-phase corresponds to the formative assessment step of the STAR Legacy learning shell (Schwartz, Brophy et al. 1999; Schwartz, Lin et al. 1999). Leelawong et al. have extended, adapted, and validated the quiz as formative assessment tool (Leelawong et al. 2002; Leelawong et al. 2003). In this system, the quiz questions are relevant distance-rate-time problems. Students are permitted to work on the quiz problems as long as they like, and this gives them the opportunity to reflect on their knowledge, and gain better understanding of domain concepts. Feedback from the quiz phase (teachable agent and teacher agent) helps students to correct their lack of understanding or their misunderstanding of domain knowledge.

The student enters the quiz phase by clicking on the next step in the cycle. Then, the agent dialog interface is replaced by the quiz interface, which provides up to 15 expert prepared text problems (Figure 10). This interface has three tab-panels that allow the student (1) to ask the agent to solve a problem, (2) to request an explanation of the agent how it has solved the problem, and make suggestions on how Billy may improve, and (3) to ask the teacher for the correct solution. The simulation (not shown in the figures) and the graph tool are still available to the learner, and can be used to verify the agent’s answers at any time.

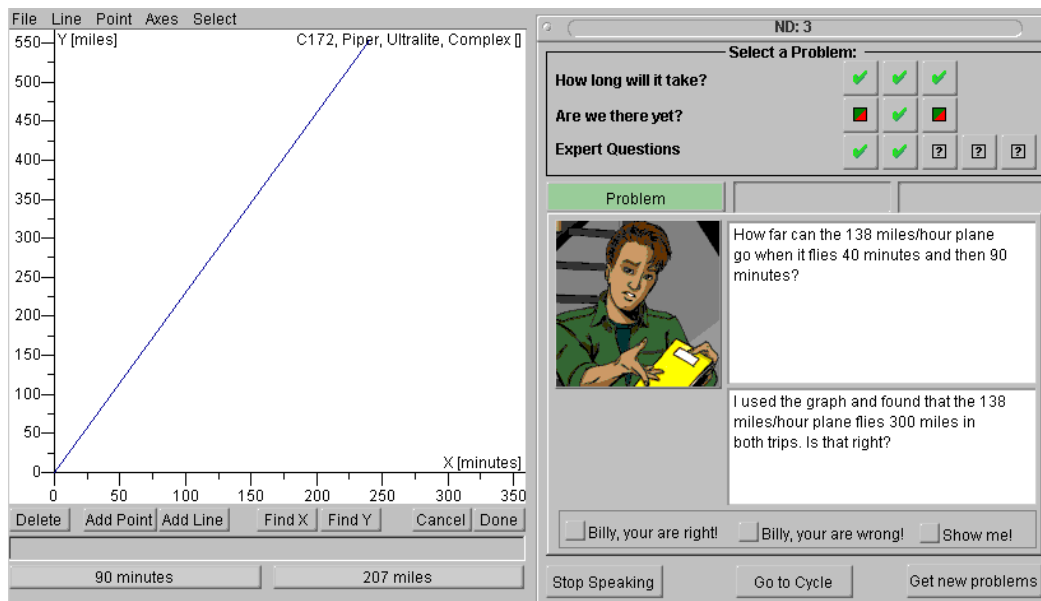


Figure 10. Quiz Phase: The Student selects a Problem and the Agent solves it

The learner starts quizzing the agent by choosing a problem category, and clicking on a question to ask the agent to provide a solution. The quiz grid (Figure 10, top right) gives a quick overview of how Billy has performed on quiz questions. If the student has not asked the agent to solve a particular problem that grid position is marked by a question mark icon. If Billy solved a question and the student asked the teacher to check the answer, the grid displays if a question was

answered correctly. If Billy answered a question correctly, that position is marked by a green checkmark. If Billy has answered wrong, the interface displays a red X. Because the correctness of a question may change if the student modifies the graph, a red/green square marks a question that needs to be revisited.

If the learner chooses a problem, the problem text is displayed in the upper text box of the quiz interface, and shortly after that, the agent provides the solution in the text box below the problem (Figure 10). Billy derives an answer, if he has been taught the procedure to solve the problem, or asks the student to teach him what he still does not know. The agent uses the graph that the student has created, but does not demonstrate how he has derived the solution unless the student specifically asks for it.

The correctness of an agent's answer depends on (1) the correctness of the representation, which the student has created during the teaching phase, and (2) that the agent was taught the correct procedure to solve the problem. If the student did not teach a topic, the agent will ask to be taught. If the representation is wrong, the agent complains if it cannot use a procedure that it has been taught (e.g., if the line is too short). Billy provides a solution as long as what he has learnt is applicable (for example, if a line has an incorrect slope). In the latter case, the student has to rely on the answers of the teacher to find an error.

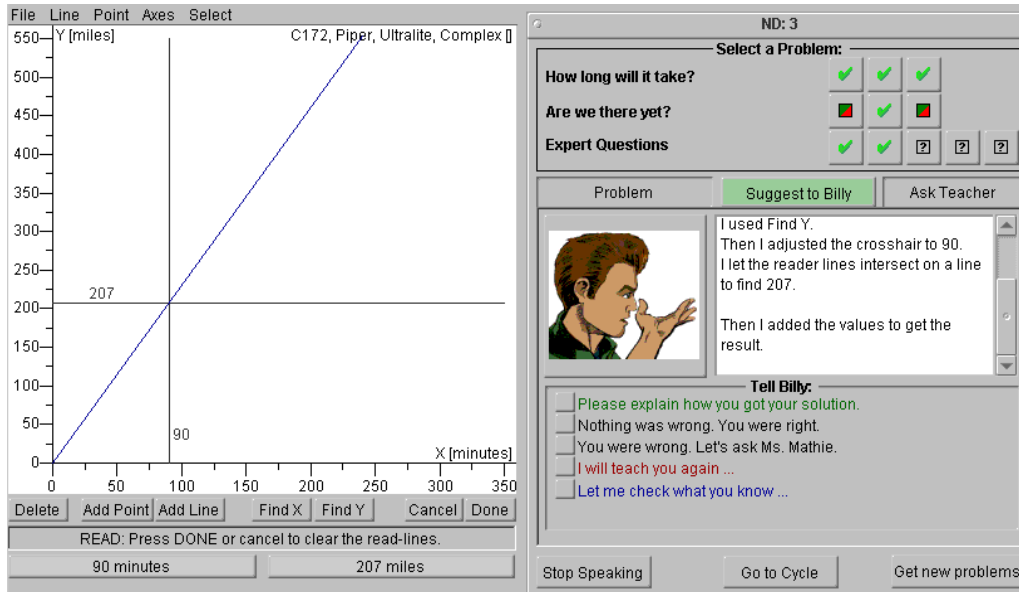


Figure 11. Quiz Phase: The Teachable Agent demonstrates and explains its Solution

Once Billy generates an answer, the student can indicate whether his answer is right or wrong. If the student is unsure, he or she might ask Billy to show how he has derived the answer. For this, the quiz interface switches to the suggestion mode (Figure 11). Here, the student can ask for an explanation, request a judgment from the teacher, or quickly go back to the teach phase again to revise Billy's knowledge.

The key feature is when the agent gives feedback by demonstrating his solution in words, and animation in the graph. Figure 11 shows how the agent describes the problem solution. Billy says, "I used Find Y. then I adjusted the reader lines to 90. I let the reader lines intersect on a line to find 207." At the same time, the agent animates dragging the reader lines in the graph to show how it obtained the solution. On problems with two legs, the agent repeats the description a second time and shows how he has added the results. If the student has taught the agent wrong, Billy makes the same mistakes that he was taught. For example, he might read the graph in the wrong

direction. To correct any combination of mistakes the student has to teach the reading procedure to Billy again.

To obtain a check mark, the student has to ask the teacher for her review of the problem by selecting the ask teacher tab. This leads to the teacher feedback interface (Figure 12). Initially the teacher only reports if the agent found the correct solution, and if the number and/or unit of the solution are correct. After reading this, the student can request a full solution from the teacher, which describes in words how to obtain the result.

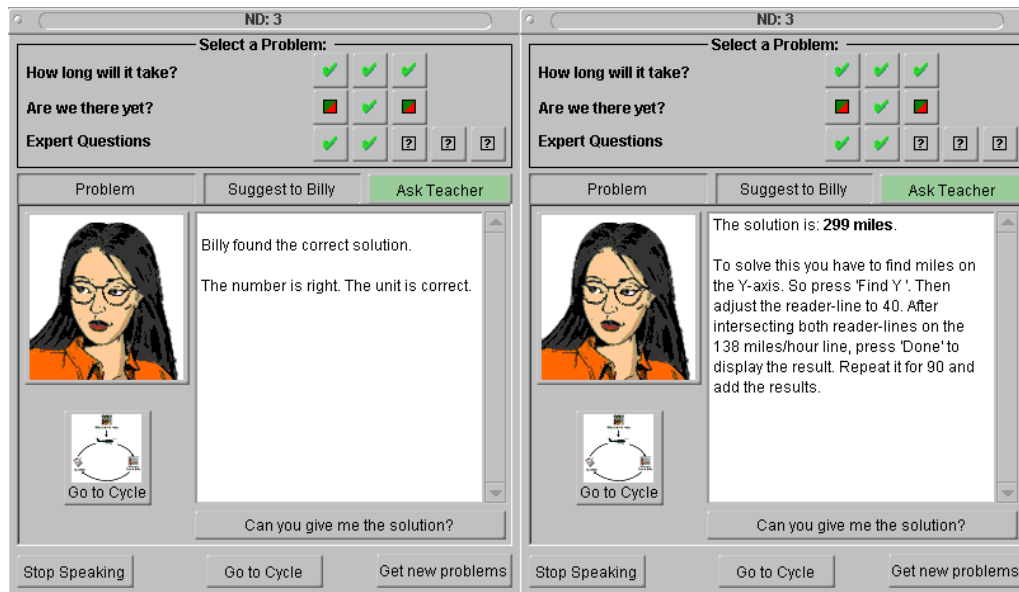


Figure 12. Quiz Phase: The Teacher Agent corrects (left) and gives the Solution (right)

Test Phase

When students think Billy learnt enough and is ready to solve all problems of this cycle they send him to take the test. The simulation is replaced by a more advanced version that shows the map of the USA, and that allows simulations of flights between cities (Figure 13). Problems are

drawn from this context, but are still very similar to those that the students solved in the quiz phase.

As special task, students are told that they have to grade Billy's answers. As teachers, they need to create a test for Billy. Although we provide the questions, students have to solve all problems on their own before they can grade the teachable agent. Right after that, the teachable agent solves the problems using the graph and succeeds or fails, depending on the correctness of the representation and taught knowledge. Students grade the agent by comparing their own prepared answers with those of the agent and telling the agent if it is correct or wrong.



Figure 13. Test Phase: The New Simulation

In the advanced simulation, the student sees a map of the Southeastern US with major cities around Nashville (Figure 13). The learner can read distances in miles from the map, and fly airplanes on designated routes between the circled cities. To run a successful simulation, the student has to enter the speed of the airplane in the simulation control panel, select an origin and a destination city, and provide a correct flight duration. Like in previous steps of the cycle, the simulation will give feedback to the student if it succeeds.

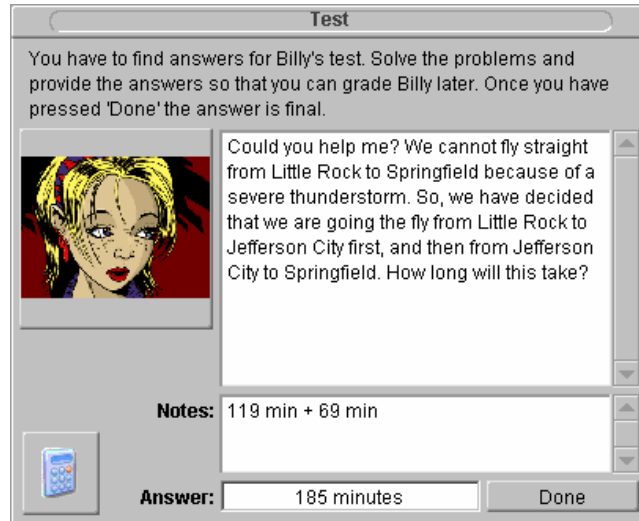


Figure 14. Test Phase: The Student "Creates" the Test by Finding Answers

When students enter the test phase, they "create" a test for Billy, by finding answers to given problems. The system tells them, "Find answers for Billy's test. Solve the problems and provide the answers so that you can grade Billy later. Once you have pressed done the answer is final." Then, the system displays the first problem in the text box beside the image of the customer (Figure 14). Students can use a calculator, which is provided by the system, and can write down notes or intermediate results in the notes box. Then students provide the answer and click the "Done" button to proceed to the next problem. This repeats until the student has solved all of the prepared problems.

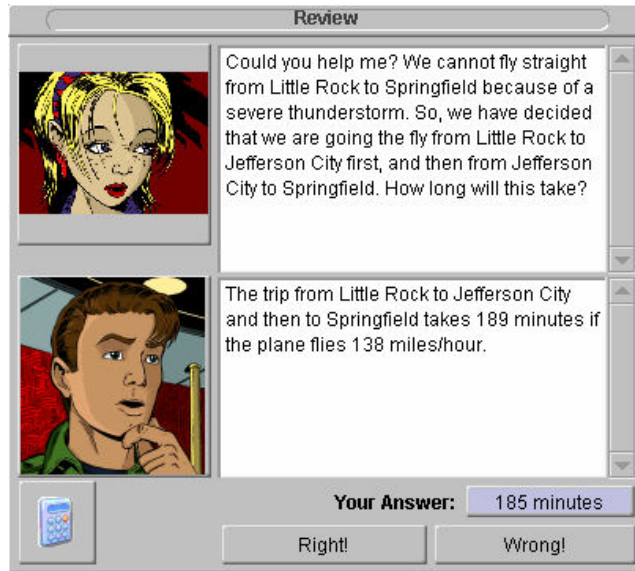


Figure 15. Test Phase: The Student Reviews and Grades the Agent

Creating the test is followed by the test review (Figure 15), where the student reviews the agent's answers and grades them. The agent solves each problem, and displays its solution. The learner's previous answer is available just below the agent's answer. Typically, the student only needs to click the right or wrong button depending on if his answer matches the agent's answer or not. However, better learning occurs if students will try to find out why there is a mismatch between the agent's answer and their own, and use the graph and the simulation to check what happened.

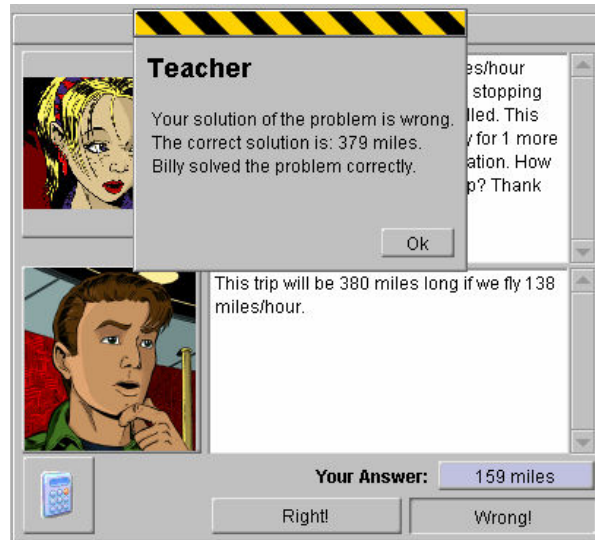


Figure 16. Test Phase: The Teacher Provides the Correct Solution

Immediately after pressing the grading button, an alert panel (Figure 16) displays the teacher agent's correct solution to the student. There are arguments for and against using this strategy: Students revise their answer, but cannot get final feedback for it, or students may not try to revise their answer because the teacher gives the correct solution too soon. A more prudent approach would be to delay the teacher's feedback more, so that students get additional time to reflect.

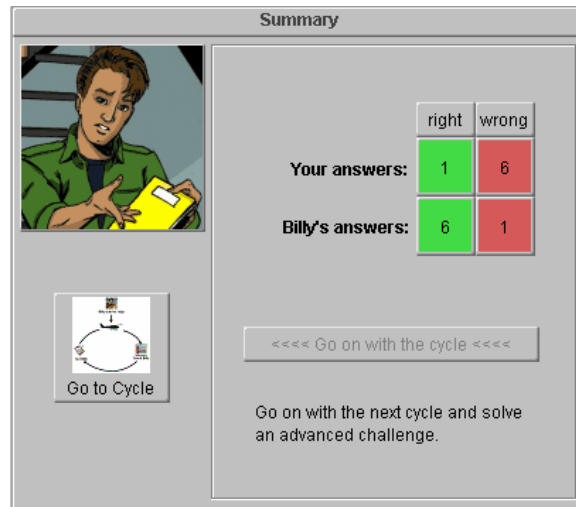


Figure 17. Test Phase: The Student gets Summative Feedback

The end of the test phase provides summative feedback (Figure 17). It tells how many answers the student has solved correctly and incorrectly, and compares this to how many answers the agent could solve. Finally, the student proceeds to the next cycle to solve another challenge.

Software Architecture Overview

The software architecture overlaps with the user interface design, but additionally separates interactive representations into two components: *smart tools* (Owens et al. 1995) and simulation¹. It also adds an agent communication structure to the system, which is transparent to the user. Therefore, the system design has five modules (Figure 18): cycle architecture (Modules package), smart tools, simulation, teachable agents, and agent communication. All five modules and even some of their components (e.g., each smart tool) provide information about their state through the agent communication channel upon request. The package Testing contains automated

¹ This separation exists mainly for historical reasons, as the simulation has been designed as the first part of the system. It would be possible to implement the simulation as smart tool without any visible changes to the user.

pretest, posttest, and Likert-test managers, and the package Utilities provides tools for persistence, logging, and text-to-speech control.

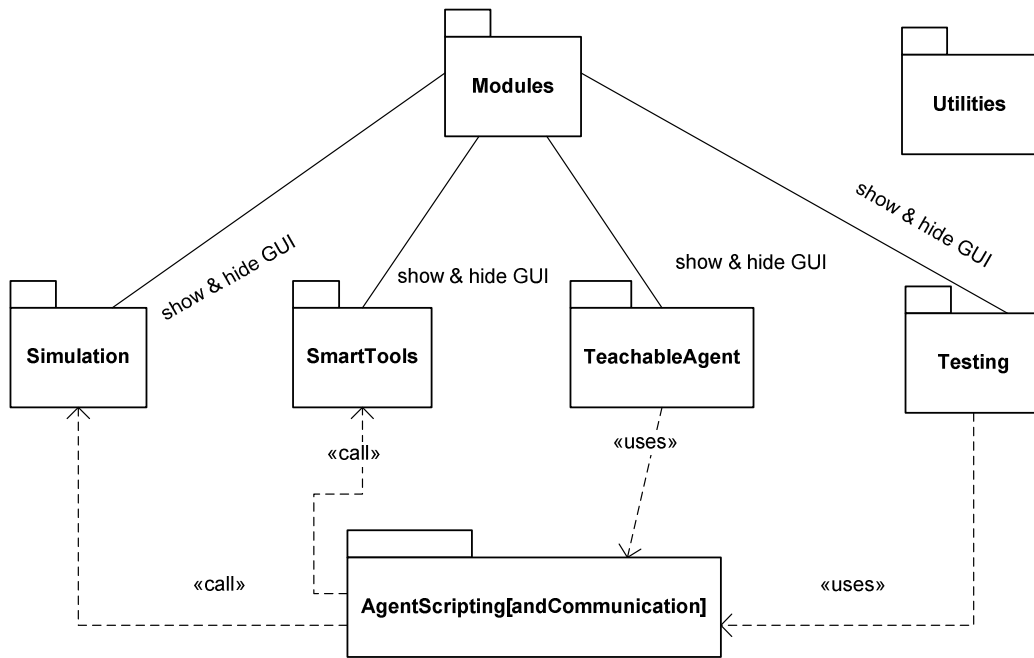


Figure 18. UML Structure: Package Architecture of the Learning Environment

Interactive Representations

In this section, we will discuss representations (smart tools), like the graph tool. Generally, we distinguish between simulations and representations, like graphs and tables. Students create representations to teach, solve problems, and learn, while simulations provide feedback for correcting representations.

Representations, like the graph tool, conceptualize the approach of Smart Tools (Vye et al. 1997; Owens et al. 1995), which integrate the constructivist approach in a multi-representational framework to help students develop a deeper understanding of variables and functional relations,

in our case, between distance, rate and time. Smart Tools help learners to build personalized representations to help them solve complex problems effectively and efficiently. We have adapted this approach to the learning by teaching paradigm and reengineered enhanced versions of these tools, to add them to our environment.

Simulation

One of the most important components of an exploratory environment is the feedback mechanism, which helps students to verify their knowledge. One of these feedback mechanisms in our learning environment is the simulation, which establishes the ground-truth, and this helps learners decide whether they have solved a problem correctly. Our simulation can model a whole microworld, but we shield the user from its full complexity by providing scaffolded user interfaces.

Simulations, like microworlds, use the idea of abstracting relevant facts of the target domain so that the user can learn without distraction. However, in contrast to microworlds we scaffold the complexity of the simulation by moving from a simple two city setting in the teach phase (Figure 8), to a more realistic map of the South East US in the test phase (Figure 13). The simplicity of the initial simulation prevents students from playing with it too much and wasting their time. In previous observations, we found that students initially like to use the simulation to solve problems by trial and error until they understand that this approach does not work. Simulations that are more complex appear to increase the time that students spend on trial and error tactics.

Generally, we exploit three uses of the simulation in our learning environment, which are verification of solutions, generation of data points, and illustration of covariation of time and distance¹ (Thompson 1994, 1994). Initially, students only use the simulation to verify data points by

¹ Students should learn that if time changes distance changes proportionally.

entering desired speed, distance, and time. This essentially over-specifies the problem, because we do not want that students use the simulation to solve problems, or encourage trial and error. The students will receive feedback when the simulation ends by, "the plane arrived too early," "the predicted time is too short," or "simulation successful." In some cycles, students have to create graphs by using the simulation to generate distance-time points for a specific speed. Students can do this by specifying a speed, the maximum distance and a long time. Then students start the simulation and stop it after any time they like to get one data point each trial, which they can transfer to their graph. We also visualize the covariation of distance and time as progress bars below the simulation, which shows students that when time varies distance varies at a different rate.

Additionally, our simulation can support planning and scheduling of parallel trips with several vehicles. This was necessary to solve the Rescue at Boon's Meadow adventure (Cognition and Technology Group at Vanderbilt 1997). However, we found that the latter is not essential in investigating learning by teaching, so this functionality is not used in the current learning by teaching environment.

Simulation User Interfaces

The simulation supports various user interfaces to accomplish different goals. We distinguish the simulation control interface, which lets the user specify how to run the simulation, an item inspector, and the actual simulation, which provides animation and results.

In its full complexity, the simulation can model a microworld with a control interface that allows the student to enter a plan, like in Jasper Woodbury's adventure Rescue at Boone's Meadow (Crews et al., 1997). In this adventure, students try to derive an optimal rescue plan that

requires vehicles to carry drivers, and transportation of extra fuel containers; vehicles have payload limits, fuel consumption and can drive concurrently (Figure 3 in chapter III). Learners can simulate various plans until they fail or succeed and eventually watch the fuel drain in the item inspector while planes fly. This functionality exists because the simulation has been reengineered from Adventure Player, which is described in the section Interactive and Intelligent Learning Environments (page 56). All this planning functionality is hidden from users in our current learning by teaching application.

The simulation control interface (e.g., Figure 13, right side) provides various abstractions for the simulation planning language, which is used to specify internally or externally how a simulation executes. In the simplest case, it translates the three values of speed, distance, and time into the internally used plan step "fly plane to Destination at 138," and rescales the distance between cities to match the problem, like it is done for the simulation of the teach phase. In the test phase, the user specifies mileage by selecting origin and destination city from a pop-up menu; otherwise, the translation is similar.

To verify or use data, users can drag and drop values from their graph to the simulation control interface or from the simulation to their graph, table, quiz and test interfaces. Alternatively, learners can enter values directly by clicking and editing the numbers on the buttons of the interface.

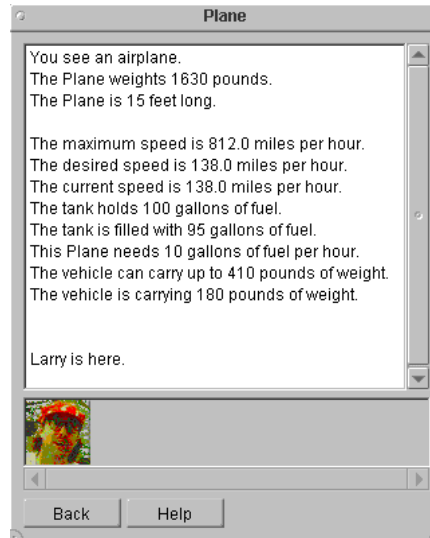


Figure 19. Simulation: Item Inspector Inspecting an Airplane with its Pilot during a Flight

The simulation inspector allows finding data about any thing and its contents in the environment. A double-click will show the interface at any time displaying basic technical data of items. Size, weight, fuel consumption, fuel content, payload, and other parameters are displayed live. If user visible plans are needed and a planner interface is present, dragging and dropping items on each other creates most plan steps.

Simulation Framework

After discussing the user interfaces of the simulation, we now discuss the internal framework of the software package. The simulation is roughly split into actor objects, which visualize and animate, and the executing components that run the simulation. We first introduce the simulation engine and then follow with the actor classes.

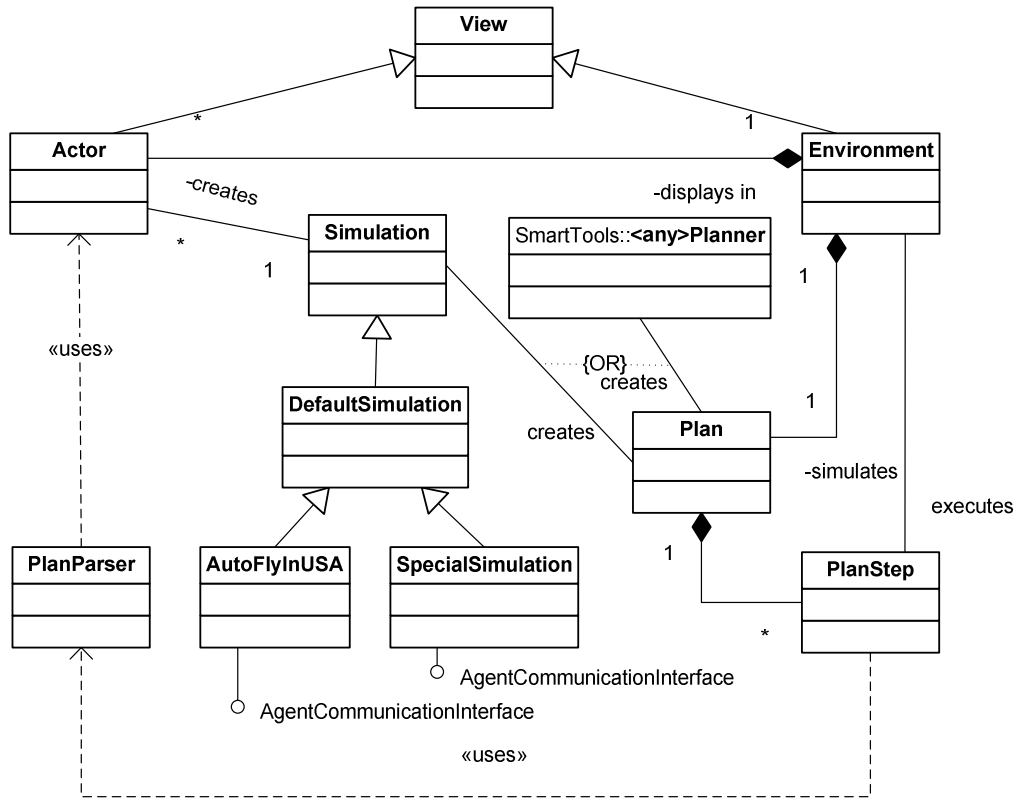


Figure 20. UML Structure: Simulation creates Environment that executes PlanSteps to change Actors

The central object in running a simulation is the Environment¹ (Figure 20), which is responsible for timing and execution of plans. The simulation is initialized by a subclass of Simulation, which knows how to create the initial state of the Environment. Thus, it either loads the state from a file or creates Actors programmatically. Then, a simulation executes a Plan, which is a vector of PlanSteps that originates from one of the interfaces, which we have discussed earlier. This Plan is passed to the Environment and the simulation starts. Upon user request, the simulation can be paused, resumed, or reset to its initial state. The simulation runs in discrete time steps, which is usually preset by the simulation designer. In our case, we use one-minute steps to let students

¹ In describing class diagrams, I use words starting with a capital letter to refer to the classes in the diagram. For example, Simulation refers to the class in the diagram, while simulation refers to the product or package.

solve problems approximately between 30 minutes and 4 hours. The timing is passed at startup to the Environment and is constant for each application.

Each PlanStep is specified by a starting time, an action, and an optional duration. If duration is given, the PlanStep throws errors to its user interface if the execution does not finish on time. It is up to the Simulation object that owns the user interface to display an appropriate message to the user. The Environment only initiates PlanSteps, distributes timing events to actors, displays the background image, and handles errors. Actors animate themselves, as we will see next. A PlanStep's action is applied globally to the Environment, which requires actors to be named distinctively.

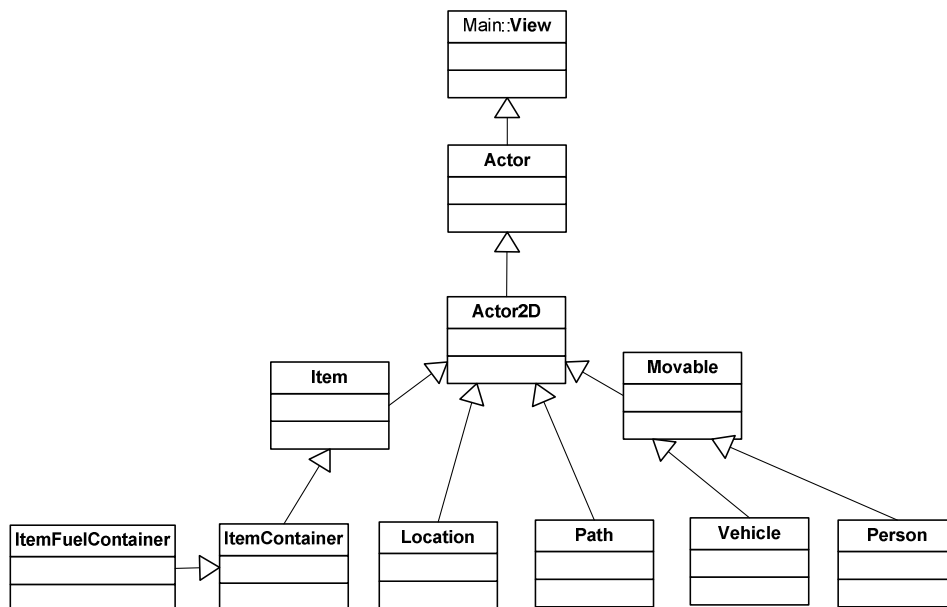


Figure 21. UML Structure: Actor Class Hierarchy for Velocity, Fuel-Rate and Payload Simulations

Next, we discuss the actor class hierarchy (Figure 21). The top-level object is View and stems from our graphical user interface library (Netscape IFC). Our own top-object is Actor and implements the public method oneStep, which is called for each time-step by the Environment while the

simulation runs. Actor mainly handles nesting of objects, which can contain each other. Actor2D implements basic drawing functions and transformations. A simulation designer will place Locations into the Environment and connect them with Paths. If the Environment has a scale, the Paths calculate their distances automatically, but if a designer desires she can set them manually, too. Moveables can follow Paths automatically with their desired speed. If a Path has a speed limit, Moveables slow down automatically. Vehicles add payload and fuel consumption management to Moveables and need a Person inside to drive or fly. Initially all Actors are placed in Locations and are assembled by an executing plan. This concludes the discussion of the simulation and we focus on representations next.

Smart Tools

Our primary goal is to help students learn rate problems. Standard curricula in middle schools often use different representations simultaneously to teach this topic, each with its own advantages and disadvantages. Students generally learn to use graphs, functions, and tables. To gain sufficient expertise in solving problems, students need to go beyond basic understanding of these representations, and learn their weaknesses and advantages in different problem solving situations. For example, a student will find that graphs are quick in providing results, but are limited in granularity. For this reason, we have integrated interchangeable representations into our environment.

Because the word "representation" is probably uninteresting to students, we use the catchphrase *smart tool* (Owens et al. 1995). Thus, we imply that using the right tool in the right way helps solving problems quickly. The introductory movie and our anchoring context, which we have adapted from Jasper's Adventure, Working Smart, does just this by implying that Jasper's

success in building his company results from his mathematical knowledge about smart tools. Thus, we also motivate students by illustrating the potential utility of knowledge.

We have built our curriculum around graphs to help students gain a visual understanding of slope as rate, before being confronted with the advanced functional representation $y = k * x + d$. A visual representation also has the advantage that our agents can ask questions about it, and can observe the student while modifying it. The primary Smart Tool that students work with in this environment is the interactive graph tool, which is augmented by a table tool and a calculator. Currently, we do not have a representational tool, which can interactively manipulate functions.

User Interface

To allow exploration, inquiry, reflection, and active learning, our representations allow the student to make mistakes and create incorrect representations. This way, students learn to reflect and critically review their results. For example, when users create a graph in our graph tool, they can draw lines that are not straight and consist of multiple segments with different slopes (Figure 22). In a learning by teaching situation, the teachable agent has to be able to read such a line correctly, if he has been taught reading graphs right.

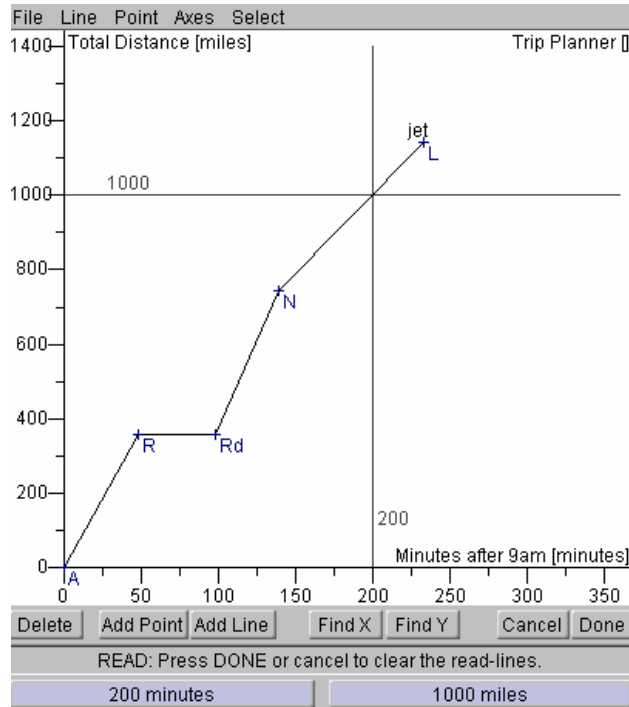


Figure 22. A Student Reads a Total Time from a Trip-Graph

In addition to the visible features of our graph tool, the user interface design also needs to make actions of the user unambiguously interpretable by agents. Thus, it provides controls that make the intention of students explicit when using a tool, that go beyond what is necessary to create the result. We illustrate this on the example of reading a graph: Originally, our user interface only provided the function "Read Graph" to display horizontal and vertical reader lines in the form of a crosshair. Students could move both reader lines simultaneously by dragging the mouse, which made the intersection point of the reader lines, follow the mouse pointer. We found that this interface design, although very efficient for the user, is at a too high level of abstraction for agents to learn if the user has understood the right procedure. The agent could not tell if the student read the result value on the X-axis, or on the Y-axis, and which value the student used to find the result. Hence, in our new interface design the user has to use a more detailed procedure.

To read the graph a learner must follow a four-step procedure that allows an observing agent to deduce the correctness of the reading procedure: (1) Decide to find an X or Y value; (2) Position a reading line on the given value; (3) Position the second reading line correctly; and (4) Obtain the result. Now, the agent can analyze each action independently and learn it correctly or incorrectly as needed. For example, students can make the mistake to use the given value on the wrong axis, and the agent can learn it.

In addition to providing a didactic approach to reading a graph, our tool allows many operations that change it. The user can label and specify minima and maxima of axes, add and delete points, add, delete, extend and label lines to create smart tools in different cycles. Additionally, operations like translating, rotating, and extending lines provide options to teach concepts, like overtake problems.

An issue similar to the one discussed in the previous paragraphs arose when we challenged students to create graphs, and the lines they created were initially too short. Students often extended their lines by adding new lines to the end of an existing line. This created internally two distinct line objects that appeared as single line in the user interface. Again, we had to modify our user interface. To teach an agent, students must express their intention of creating a single line by extending it to the left or to the right. From this, we derive the principle that *user interface actions of the learner must be unambiguously interpretable by teachable agents*. This allows agents to know the intention of their tutors and learn appropriately and believably.

Smart Tools Framework

The smart tools framework uses two design patterns: a model view controller for each Smart Tool, and a plug-in pattern that allows the Smart Tool controller to instantiate arbitrary tools, or even multiple instances of a single tool. Figure 23 shows the static structure UML diagram.

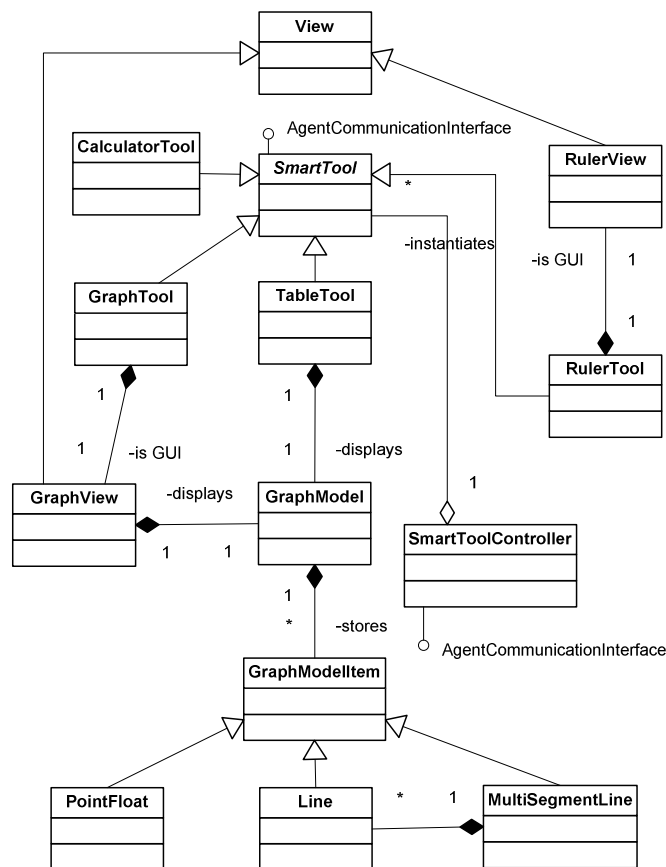


Figure 23. UML Structure: The Controller instantiates SmartTools; Graph and Table share a GraphModel

At system start-up, the Smart Tool controller registers known classes of tools in the application's main menu, and makes them accessible to the user. Additionally, the SmartToolController connects to the AgentCommunicationInterface, to enable agents to instantiate SmartTools with spe-

cific names at any time. Therefore, agents can pop up new graph tools or tables and load their content from a file to provide examples to the learner.

Each Smart Tool must implement a controller class that inherits from the abstract base class `SmartTool`, which provides methods that can display each tool in windows. This class also provides a naming and look up system for tools that is attached via the `AgentCommunicationInterface` to the `AgentCommunicationChannel` (Figure 27).

When students teach our teachable agent Billy, we instantiate a graph tool named "Billys-Graph", which is manipulated by the agent through this interface. Thus, incoming agent communication calls are forwarded from `GraphTool` to `GraphView` to `GraphModel` and eventually resolved through method calls in the objects `Line`, or `MultiSegmentLine`. An example of an agent request that travels the whole way is: `BillysGraph.slopeOfSelectedLine()`; a request that is resolved at higher level in `GraphView` is: `BillysGraph.crosshairsOnLine()`. Thus, requests of agents are always resolved at the appropriate level of our class hierarchy, without letting agents deal with addressing multiple targets.

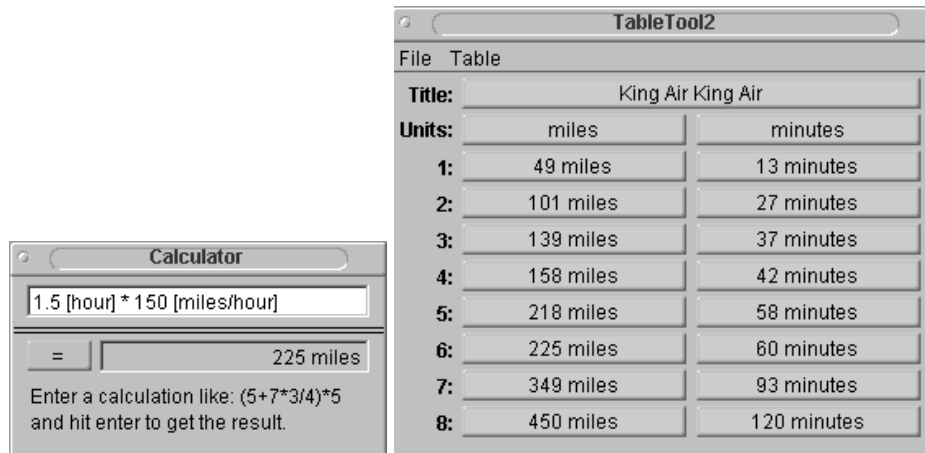


Figure 24. Calculator Tool and actual Table Tool Created by a Student during our Experiment

Simple Smart Tools, like the CalculatorTool, only load a generic interface, and have no model and view. Model free tools, like the RulerTool, have a view but no model and thus no persistent state. Advanced tools, like graphs and tables, have a persistent model and provide file operations to load and save the tool.

Teachable Agent Architecture

The agent architecture has three components: the graphical user interface that lets the user choose an agent and respond to it, and two underlying packages that build the software framework, which are the teachable agent and agent communication packages (Figure 18). We will start by discussing the interaction of the user with the teachable agent system.

User Interaction

Each agent has a dialogue interface to communicate with the user (Figure 25). Students choose which agent they want to talk to by pressing a button on the top of the interface at any time during teaching. When the dialogue advances, the agent displays what it has to say in a text panel,

and speaks the text using a text to speech engine. Displayed text can include hypertext and graphics (Figure 26). Text to speech technology prevents disadvantaging poor readers, and it provides auditory qualities of multi-modal effects that Johnson suggested as beneficial for learning (Johnson and Rickel 2000). The interface also includes a still picture of the agent, a navigation button to open the cycle, and a button to open the calculator tool (Figure 25).

Dialogue States

When displaying a specific dialogue state the agent provides multiple-choice answers to advance the dialogue, or teach the agent (Figure 25, bottom). If the user provides answers, the agent will react or learn, depending on the task or question in the current dialogue state. Additionally, Billy may learn through demonstration, if the user performs a requested demonstration. In other cases, he may perform an action, like modifying the graph, querying a representation, or simply branch the dialogue to a new state (Figure 26). Agent dialogues can be characterized as event-driven finite state machines with rule-based transitions.

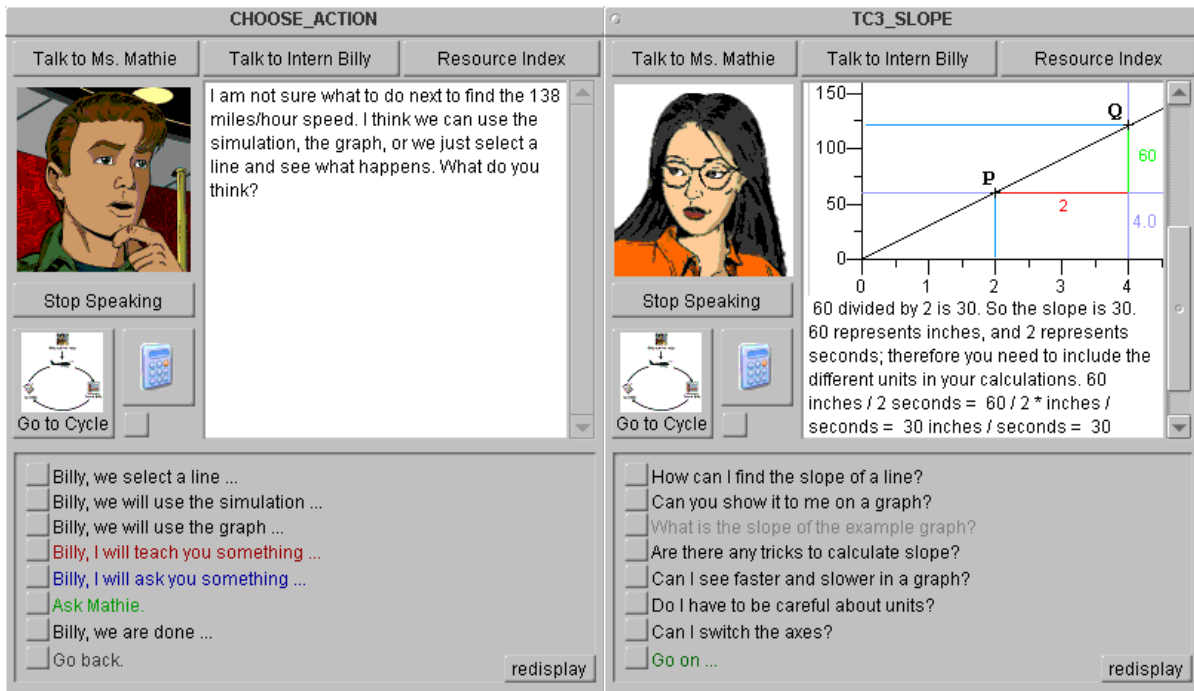


Figure 25. Agent Interfaces: The Student chooses what to do next (left), Ms. Mathie instructs Slope (right).

When the system starts, each agent loads its own dialogue structure, which is modeled like a finite state machine. The current state is what the agent says and can refer directly to specific user changeable features of representations. When invoked, a state executes an interpreted program (a script, described in a later section) that generates the agent's words, actions and installs sensing-rules in the environment (Figure 26). The state also displays answer choices for the student, which can run scripts when they are selected. Thus, the agent establishes a context that allows meaningful interpretation of user actions.

Sensing Rules (Agent Triggers)

Sensing rules (instances of AgentTrigger in Figure 27) are the core mechanism to direct interaction with the user, other than through direct dialogue choices. These rules listen to the event stream of the system and can be activated or retracted by agents at any time when a transition occurs or the

learner makes a new dialogue choice. The rules react to events that originate anywhere in the environment and may be, for example, mouse clicks of the user, a simulation that finishes successfully, or an unsuccessful attempt to read the graph. Thus, the system has a dynamically changing rule-base, which contains only rules relevant to the current dialogue context. Additionally, an agent can specify a rule's default lifetime (permanent, until next change of state, n-times activation), if it does not intend to retract it later. Sensing-rules carry a payload, which is a script that can act on behalf of the agent, for example, it can let the agent say something, modify a representation, or change the agent's state by branching to a new dialogue.

Using scripts, each state can make intelligent transitions following dialogue choices of the user. This means that a single choice of the student may branch to different dialogues at different times depending on the state of the environment. Each intelligent transition can execute a script that can do virtually anything. In some cases, this means that the agent makes a comment about the user's choice without leaving the current state, in other cases the agent modifies a representation, says something, and branches to a new dialogue a few seconds later. Sometimes, one agent will directly influence the state of another agent; for example, the teachable agent may ask the teacher agent about a specific topic. However, the specifics of these interactions are entirely up to the dialogue designer, who may want to limit the branching factor of the dialogue in sensible ways.

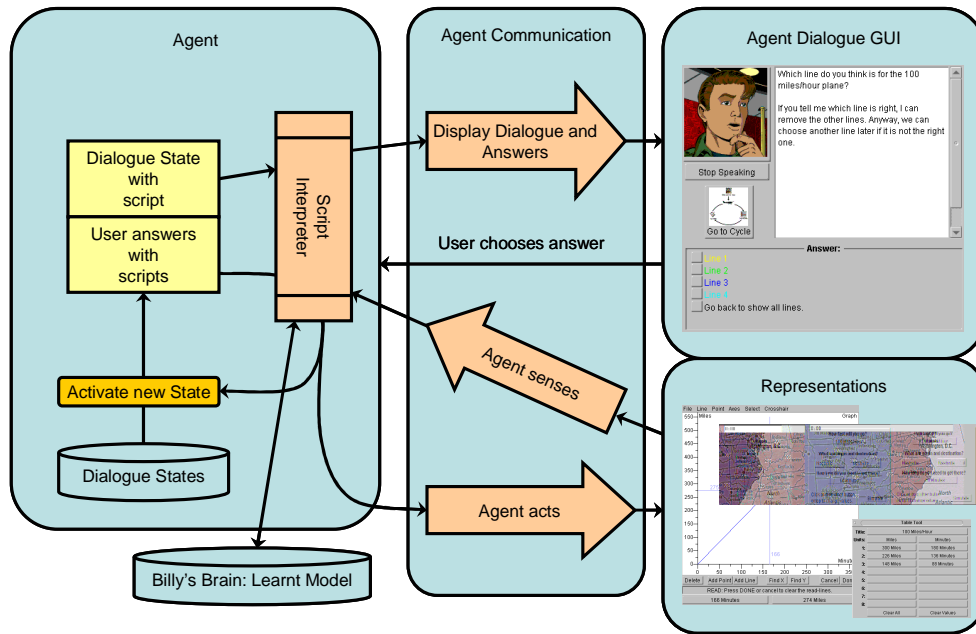


Figure 26. Interaction Schematic: Agents Sense and Act in the Environment

To recapitulate we will explain the agent's interactions by tracing Figure 26: If a new dialog state is activated (orange box, yellow box top half), an agent uses a script to display text in the graphical user interface (right top), speaks the text, and provides multiple-choice answers. After that, it installs sensing rules in the environment that can act on its behalf (agent senses arrow). If the user chooses an answer (yellow box bottom half), or triggers a sensing rule the agent executes an appropriate script, which makes the agent learn, act (agent acts arrow), or transition to a new state (orange box).

Teachable Agents Learn Believably

Agents have to learn, to be believable students. However, learning in our case does not mean inferring from actions of the user, but asking questions that promote deep understanding and initiate reflection. During the dialogue, the student may select to teach the agent. At this point, we distinguish procedural and declarative knowledge. To teach declarative knowledge the learner

simply has to answer a multiple-choice question and the agent will remember the correct or incorrect answer.

If the user teaches procedural knowledge, the agent will learn through a sequence of demonstrated and/or user suggested actions, which the agent executes. An agent learns a multi-step procedure, like reading a graph or finding slope (Figure 28), by letting the student demonstrate it, and then verifying the outcome. For an example of the process from the perspective of the user, see page 79 (Teaching the Teachable Agent – Two Examples). A detailed description of internal processes follows in the section Agent Knowledge Structures.

Another mechanism to let the agent behave believable is the ability to explain procedures through animation and in words. Especially in the quiz-phase, the student has to judge the correctness of the agent's answers, but initially the agent only provides the correct solution. Should students decide that this solution is incorrect, they may ask the agent to explain how this solution was derived. Then, the agent will replay an animation script, and explain in words what it has done to solve the problem.

Agent Framework

After discussing the user interaction in the previous section, we will now focus on the agent framework, which implements this functionality. The framework is illustrated in Figure 27 as UML static structure diagram.

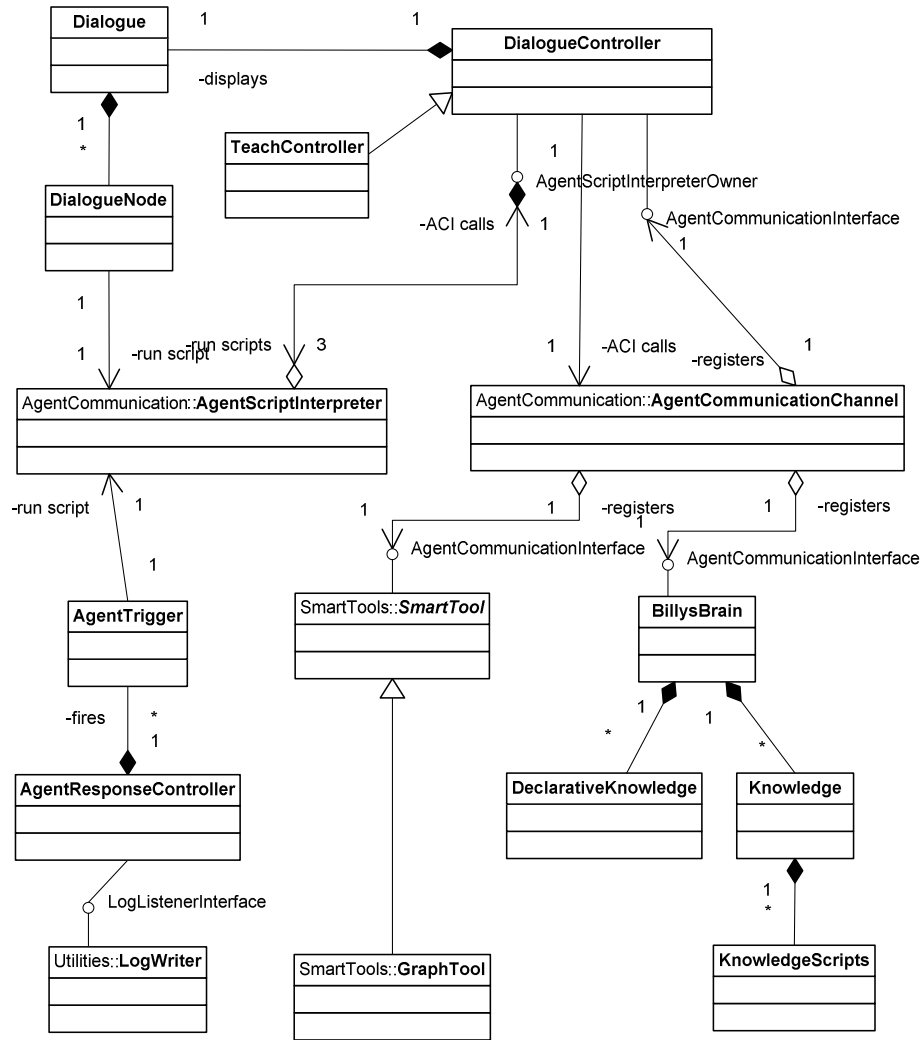


Figure 27. UML Structure: Agent Architecture

Each agent is an instance of a class inheriting from DialogueController. For example, in the teach phase this class is the TeachController. This class is mainly responsible for displaying the agent’s dialogue user interface and implementing the peculiarities of each phase. The abstract base class DialogueController loads, when its subclass is instantiated, the agent’s Dialog from a file, and displays the initial DialogueNode. The dialogue node tells the controller what the agent says and which answers to display, and then the user is back in control.

If the user selects an answer, the DialogueController notifies the current DialogueNode, which sends the corresponding answer's script to the AgentScriptInterpreter that is owned by the DialogueController. Now this script initiates a sequence of agent actions, which is funneled through the DialogueController to the AgentCommunicationChannel, where actions are distributed to their targets. This way, the DialogueController can veto any action of a DialogueNode, if this is desired. The controller handles specific commands to advance the dialog right away, while other actions modify or query smart tools, store or retrieve knowledge from the agent's memory (BillysBrain), or modify the state of other agents.

Agent sensing rules (AgentTriggers) are optionally instantiated when a dialogue node is displayed, and handed over to the AgentResponseController, which listens to the system's log file (LogWriter) via the LogListenerInterface. Any object in the system can write a line (a plain string) to the log file if it undergoes an essential change in its state, for this it uses a static method in LogWriter. Every time this happens, the agent response controller matches its triggers against this new incoming event, and fires them if appropriate. When fired, a trigger executes its script as the agent, which has instantiated it. This lets the agent respond to events.

Agent Knowledge Structures

Teachable agents learn from three agent commands: teach, ask, and act. When taught new knowledge, it first has to be stored in the agent's memory using the method BillysBrain.teach("read the graph", index, askScript, actScript, isCorrect). Then, BillysBrain.ask("read the graph") provides a verbal explanation of what the agent would do, and BillysBrain.act("read the graph") will perform the skill, for example, in a quiz or test situation. Knowledge scripts receive parameters by querying the environment or relying on shared variables with quiz scripts and test scripts. If a student

asks the agent about something that it has not been taught, the agent responds, "I do not know how to read the graph. Can you teach me?"

As mentioned before, we distinguish declarative and procedural knowledge. First, we will describe the more complex procedural knowledge (Knowledge in Figure 27). Procedural knowledge consists of multiple steps performed in sequence to solve a problem (Figure 28). We store this sequence as a vector of steps (KnowledgeScripts). Hence, each knowledge object has an array of one or more knowledge scripts that when executed in sequence produce an explanation (ask) or demonstration (act) by the teachable agent. For example, a four step procedural skill requires four teach statements with different indices. Declarative knowledge has no act script, only a single, one-step script, and its ask script uses plain text.

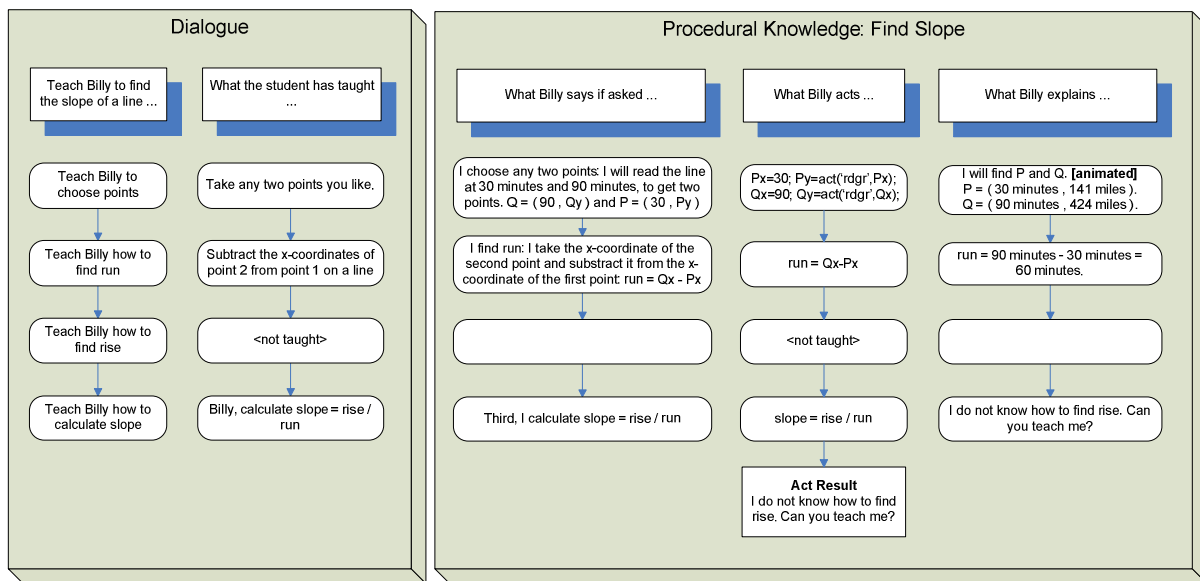


Figure 28. Four Step Procedural Knowledge to find the Slope of a Line

When an agent acts on procedural knowledge, it executes its vector of knowledge scripts in ascending order of indices, while skipping untaught steps silently. While the agent acts, it pre-

pares an explanation script that explains each step of finding a solution in words and animation (Figure 28, rightmost column), if the student requests it later. Each of these steps may be correct or incorrect and can be retaught individually by a student. Because a dialog designer knows the sequence of a skill's steps beforehand, an agent always applies skills in the correct order, regardless of how they are taught. This relieves the student of worrying about ordering problems, which in our case are quite simple to deal with. If Billy encounters a variable that he does not know, he asks the student to teach him how to find it. Then the student goes back to the teach phase and selects a dialogue that teaches the agent how to derive this variable.

Agent Scripting and Communication

The scripting language supports the agent dialogue with intelligent transitions and sensing-rules, knowledge scripts, quiz scripts, and test scripts. If an agent needs information, its script will make a request that is resolved through the agent communication channel, as we have described in the section Agent Framework. When an agent or a representation is created, it registers by name with the communication channel. Thus, its functionality becomes available to other connected agents.

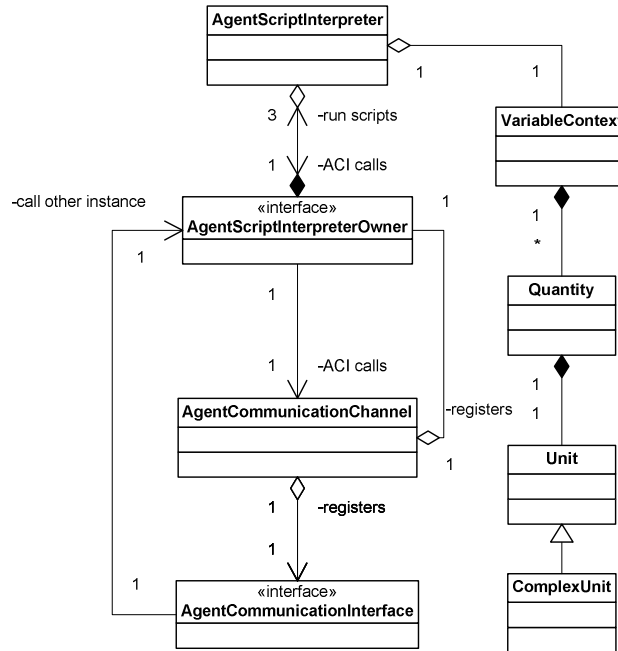


Figure 29. UML Structure: AgentCommunication

The scripting language itself is a left recursive parser specified with the Java compiler compiler (JavaCC) that executes the commands while parsing. JavaCC uses a grammar as input that is similar to the Bachus Naur Form (BNF). The interpreter is encapsulated in the AgentScriptInterpreter object, which is instantiated by agents and other sub-systems (Figure 29). Each interpreter has an optionally independent variable context, and supports a full set of mathematical and logical constructs in addition to basic flow control with statements like if, then, else, return, and so forth. An agent action is defined in object-oriented syntax: Agent.function(parameters) (Figure 30 and Figure 31). The language interpreter also supports arbitrary arithmetic calculations with unit calculation support, like $x = 20 \text{ [mi/h]}$; $y = x * 2 \text{ [h]}$, which results in $y = 40 \text{ [mi]}$.

```

if(BillysGraph.crosshairOnLine()==false)
{
    print("The blue reading lines do not intersect on any line. I will show you ...");
    BillysGraph.chooseLineByNum(0);
    BillysGraph.animateDragCrosshair(BillysGraph.yToX(150),150,-1);
    BillysGraph.sleep(2000);
    print("See. Both blue lines cross each other on top of a line. Try again.");
    end();
}

```

Figure 30. Partial Agent Script: Teacher Checks if the Student has used the Graph's Reader Lines Correctly.

Each script interpreter has an owner, which is generally an agent (usually a subclass of DialogueController). When a script executes, it generates a sequence of agent actions that the interpreter forwards to its owner, which can veto them or forward them to the AgentCommunicationChannel. Through attached AgentCommunicationInterfaces the agent command gets resolved and eventually returns a value. In case of errors, an exception is thrown back to the method that initiated the script.

```

question("Calculate the slope of the King Air in miles/minute.");
currentQuestionSolution = 4.71 [miles/minute];
correctSolution(currentQuestionSolution);
tutorSays( <lengthy explanation of the tutor> );
BillysGraph.chooseLine("King Air");
currentQuestionAgent=BillysBrain.act("find slope");
print("I used the graph and found that the solution is ",currentQuestionAgent);
agentSolution(currentQuestionAgent);

```

Figure 31. Quiz Script: Full Script for the Quiz Question to Find the Slope of the King Air Plane.

Figure 30 and Figure 31 show two examples of agent scripts. The first is a partial script that the teacher uses to verify if a student has understood how to read a line in the graph. The actions verify that, after the student says that she is done, the reader lines of the graph intersect on a line. If this is not the case, the teacher agent will tutor and demonstrate the correct procedure, which the student has to learn and then teach to the teachable agent.

In Figure 31, we see an example of a quiz question. The script displays the problem, and prepares the correct solution and the tutor's explanation. The latter is stored in the QuizController until the student requests explanations. Then the agent attempts to select the line labeled "King Air" in the graph. If this does not succeed, Billy asks the student to label or select a line in the graph. Otherwise, the agent acts by finding the slope of the selected line in the graph, and prepares simultaneously an explain script, which again is stored in the quiz controller for later use. Finally, the agent prints (and says) the solution. Most questions use randomized values and derive a solution by calculation if the problem allows it. Test questions use identical mechanisms, but the system displays them in a different interface.

Dialogues and Curriculum

Our curriculum to teach graphs and related material has three cycles. In each cycle, the agent needs help with a different class of problems. Each agent has one script per cycle for its dialogue. Additionally, there are two problem scripts one for the quiz phase, and one for the test phase. In cycle one, the learner helps the agent to select the 100 mph line from four given ones, in cycle two the task is to create a graph for a given speed, and in cycle three understanding slope and speed is desired. In the following paragraphs, we illustrate the curriculum for cycle one very briefly because we discussed most other issues already.

In cycle one, the problem is to help the agent to select the 100 miles per hour line among four given ones (Figure 8). The agent offers the student the following main choices: (1) Choose a line; (2) Use the simulation; (3) Use the graph; (4) Teach Billy; (5) Ask Billy; and (6) Ask Ms. Mathie. Each choice branches into a set of dialogues in which the agent discusses ways for solving the task, or requests procedural or declarative knowledge.

In this process, the student teaches the agent the procedure of how to read a graph, and declarative knowledge about axes, ordered pairs, lines and other basic concepts. In the quiz phase, the agent provides answers to prepared questions that the dialogue has addressed before and gives feedback as we have discussed earlier. Depending on whether the learner has selected the right or wrong line, and has taught the agent correctly, the agent succeeds or fails in solving each problem. Then, students proceed to the test phase to conclude the cycle.

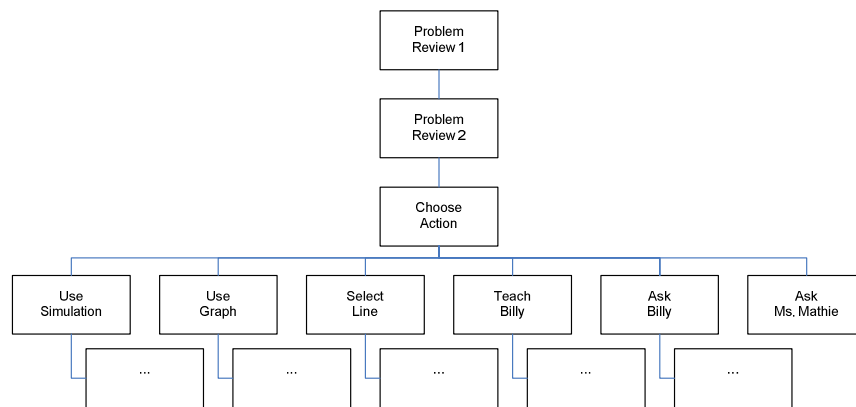


Figure 32. Standard Dialogue Structure on the Example of Cycle 1

Dialogues for all cycles share a common structure to make it easier for students to navigate the dialog tree (Figure 32). In the first two dialog steps, the agent reviews the problem very briefly again and sketches potential goals. Then, Billy asks the student what to do next, and offers about three choices to attack the problem. In addition, the student has access to two dialogs that allow her to choose among all possibilities to teach or ask Billy. The last choice (Ask Ms. Mathie) will hide Billy and transfer the student to the teacher agent. Beyond each of these top-level items, the student finds opportunities to discuss topics with the agent. Some dialogues let the student teach the agent, in others, the student can ask the agent to modify the graph or apply taught knowledge.

Summary

In this chapter, we saw how we built a constructivist learning environment that uses anchored instruction, smart tools, and a cycle design with increasingly difficult challenges, and how we could integrate it with learning by teaching agents. Before students used the system, we established the anchoring context, by situating students in a macrocontext. This placed the students as interns in a jet service and small plane transportation company, which needed a mathematical skilled helper in providing customer services. This macrocontext was motivated by an introductory movie taken from the adventures of Jasper Woodbury (Cognition and Technology Group at Vanderbilt 1997), which flashed back into the youth of the owners, and implied that the success of this company resulted from the mathematical skills of its owners. To solve their problems, students needed to separate essential from inessential data, like in the real world.

Then, we introduced how our system fosters the activities of a novice teacher who learns and teaches at the same time, and how our system helps students reflect on their teaching and knowledge by observing the agent's results in quiz and a test phase. Learners taught the system declarative and procedural knowledge by selecting answers in multimodal dialogues and demonstrating actions in representations, like the graph tool. During teaching, our teachable agent may also modify representations according to instructions of the student, which influences the correctness of that representation.

We illustrated the role of our representations, especially the simulation and the graph tool. The student gets essential correctness feedback from the simulation, which avoids providing solutions by requiring the student to over specify its parameters. The simulation can powerfully scaffold its complexity with different user interfaces from simple two location settings to full-

scale rescue missions that require simultaneously moving vehicles, planning, or even plan optimization to succeed. Learning is supported by plugging various smart tools in the environment that aid learners in achieving their goals, while not preventing the user from making mistakes. Additionally, smart tools must make user actions unambiguously transparent to agents so that they can learn and act appropriately.

Agents learn and reproduce knowledge through a teach-ask-act strategy to help the user reflect on taught knowledge. Initially students teach an agent through demonstration and multiple-choice dialogs, then they can ask the agent for a verbal interpretation of their current knowledge, and finally the agent applies its knowledge in quiz phase and test phase by acting. Acting solves a problem, provides its solution, and additionally generates an explanation in words and animation, which a learner can request to identify mistakes in the agent's knowledge. Agents represent their procedural knowledge internally as a vector of individual steps (knowledge scripts), which the student can teach and reteach individually.

Agents act and communicate through the agent and scripting and communication framework, which includes a language interpreter and a communication channel that resolves actions and requests for information among its registered peers. Representations participate in this communication framework as reactive agents, who provide essential state information and reflect other agent's changes. Our agents do not distinguish between ask and tell, like KQML (Finin, Labrou, and Mayfield 1997). If an agent requests information from another agent through the communication channel, it simply performs a function call and receives a return value for *ask* from the receiving agent, which can be a value or a string. All our communications are synchronous, because this simplifies the software architecture and avoids many engineering problems. This con-

cludes the chapter about the design of our system, and we continue by describing the experimental design of our research.

CHAPTER V

EXPERIMENTAL DESIGN

This chapter describes the design of the experiments that we employed to study the motivation and learning of students using our learning by teaching system. We outline the primary hypotheses for the study, and then introduce the measures for deriving student performance and motivation. A detailed description of treatment differences of the alternate condition is presented, and contrasted with the system used for the experimental condition. Then we present the experimental setup of the study, which included students from two sixth grade middle school classrooms.

This study compared the learning and motivational measures between two groups (Figure 33): the first group, our experimental condition (X_1), is the learning by teaching group, which engaged in all activities that we have described in the previous chapter. Students in our alternate condition (X_2) used the same baseline environment, but were taught by the teacher agent, and did not get an opportunity to teach. Therefore, learners in both groups used the same learning environment, with the same representations and the same anchoring context, but one group taught a teachable agent, while the other learnt from a pedagogical agent.

Hypotheses

Our study focused on two sets of hypotheses: knowledge hypotheses and motivation hypotheses (see Table 1). As a first step, we wanted to show that all students using our system learn. Second,

we wanted to illustrate treatment differences between the two conditions in knowledge gain, transfer, motivation, learning strategy, and experiences that energize.

We hypothesized that students in both groups would learn effectively using their respective learning environments. This should be illustrated by an increase in mean scores from knowledge pretest to posttest for all students (regardless of group). The pre- and post-tests were the only repeated measure in our experiments. Our null hypothesis for knowledge gain (H_{KGAIN0}) stated that the change in mean scores from pretest to posttest will not be statistically significant, and our alternate hypothesis (H_{KGAIN}) stated that mean scores from knowledge pretest to posttest should increase significantly for both groups (Table 1, Figure 34).

To study the effects of the teachable agent system (Figure 1, green box), we formulated three hypotheses. Our first hypothesis was that teaching social agents lets students learn better than students who do not teach. Specifically, we thought that the teaching interactions would produce significant differences between the students' post-test scores in the two groups. Thus, our knowledge gain difference null hypothesis (H_{KDIFFO}) was that students in both groups learn the same amount of knowledge on the average, while our alternate hypothesis (H_{KDIFFF}) was that students in our experimental condition outperform students of our alternate group in the post-test (Table 1, Figure 35).

Second, we believe that students in the experimental group are better in applying their learnt knowledge in a transfer task¹ (H_{T} , different transfer between conditions). Specifically, we measured if students in the experimental condition would perform better in the transfer task. We demonstrated this by comparing the mean scores in the transfer test between the two conditions. The null hypothesis (H_{T0}) for the transfer test states that students in both groups are equally successful

¹ For the design of this measure, see section "Near Transfer Task" on page 142.

in the transfer test and our alternate hypothesis (H_T) is that students in the learning by teaching condition outperform the control group (Table 1, Figure 36).

Table 1. Table of Hypotheses (O notation refers to Figure 33; \bar{O} is the mean of a measure)

Knowledge Hypotheses	
H_{KGAIN}	Alternate Hypothesis: The students in both the control and experimental condition gain knowledge over time by using our system. Knowledge Pretest Mean > Knowledge Posttest Mean ($\bar{O}_1 > \bar{O}_4$)
H_{KGAIN0}	Null Hypothesis: Knowledge Pretest Mean = Knowledge Posttest Mean ($\bar{O}_1 = \bar{O}_4$) for both groups.
H_{KDIFF}	Alternate Hypothesis: Students who learn by teaching do better than students in the control condition. Knowledge Posttest Experimental Mean > Knowledge Posttest Alternate Mean ($\bar{O}_{4X1} > \bar{O}_{4X2}$)
H_{KDIFF0}	Null Hypothesis: Knowledge Posttest Experimental Mean = Knowledge Posttest Alternate Mean ($\bar{O}_{4X1} = \bar{O}_{4X2}$)
H_T	Alternate Hypothesis: Students who learn by teaching show different transfer than baseline students. Transfer Test Experimental Mean > Transfer Test Alternate Mean ($\bar{O}_{6X1} > \bar{O}_{6X2}$)
H_{T0}	Null Hypothesis: Transfer Test Experimental Mean = Transfer Test Alternate Mean ($\bar{O}_{6X1} = \bar{O}_{6X2}$)
Motivation Hypothesis	
H_M	Alternate Hypothesis: Students who learn by teaching have different MSLQ scores than baseline students. MSLQ Posttest Experimental Mean > MSLQ Posttest Alternate Mean ($\bar{O}_{5X1} > \bar{O}_{5X2}$)
H_{M0}	Null Hypothesis: MSLQ Posttest Experimental Mean = MSLQ Posttest Alternate Mean ($\bar{O}_{5X1} = \bar{O}_{5X2}$)
H_{ETE}	Alternate Hypothesis: Students who learn by teaching have different ETE scores than baseline students. ETE Experimental Mean > ETE Alternate Mean ($\bar{O}_{3X1} > \bar{O}_{3X2}$)
H_{ETE0}	Null Hypothesis: ETE Experimental Mean = ETE Alternate Mean. ($\bar{O}_{3X1} = \bar{O}_{3X2}$)

Third, we hypothesized that the act of teaching social agents would result in differences in self-reported motivation (H_M , motivation difference between groups). We characterize motiva-

tion as composite score of intrinsic and extrinsic goal orientation towards mathematics, perceived enjoyment and value of the subject, and other such factors that are measured by the Motivated Strategies for Learning Questionnaire MSLQ (Pintrich et al. 1993, 1993). Our motivation null hypothesis (H_{M0}) was that there would be no difference in motivation scores between the two groups, while our alternate hypothesis (H_M) stated that students in our experimental condition would report higher scores (Table 1, Figure 37).

In addition, we expected to find differences in the measure Experiences That Energize (Brophy 1998) between treatment groups. Our null hypothesis (H_{ETE0}) was: there are no difference in average total score between groups, and our alternate hypothesis (H_{ETE0}) is that the learning by teaching group reports a higher average total score.

Detailed Experimental Design

We conducted a fully randomized experiment with two conditions. As we stated before, we focused on differences between students who teach a peer agent and students who do not teach. Both conditions use our baseline environment¹, which is an anchored, exploratory environment that supports active learning and discovery. Students of both conditions learn from a teacher agent. We selected two classrooms of a middle school non-randomly, and used a stratification scheme so that each class contributed 50 percent of its students to each condition (see section Population, Sample and Randomization for more details). Figure 33 gives an overview of the sequence of measures in the notation of Cook and Campbell (Cook, Campbell, and Cook 1979). In this notation, O stands for observation and X stands for treatment.

¹ Simulation, representations (graph), and teacher agent.

O_1	O_2	X_1	O_4	O_5	O_6
O_1	O_2	X_2	O_4	O_5	O_6

Figure 33. Full Experimental Design in the Notation of Cook and Campbell (Cook, Campbell, and Cook 1979)

We refer to the learning by teaching group as the experimental condition (X_1), and the group that was taught by a mentor agent (X_2) as the alternate condition (Figure 33). For these two conditions, we computed scores for the following variables: domain knowledge (O_1 , O_4), Motivated Strategies for Learning (O_2 , O_5), and transfer (O_6). Measures during the treatment (O_3 ; not shown in Figure 33) were Experiences That Energize and miscellaneous usage metrics that we could extract from user action traces in log-files. A small set of survey questions preceded the MSLQ (O_5). In the following paragraphs, we describe how we have used the computed measures in individual analyses.

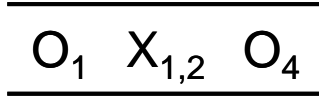


Figure 34. Repeated Measure (Within Subjects) Design to analyze Knowledge Gain

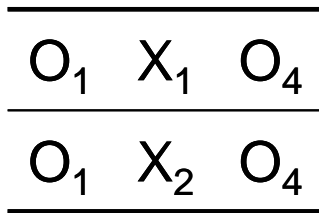


Figure 35. Between Subjects Design to Analyze how the Treatment Influences Knowledge (O_1 is a Covariate)

Our data analyses dealt with subsets of collected data. Specifically, we analyzed domain knowledge (O_1, O_4) in two different ways. The knowledge gain hypothesis required a pretest-posttest design that compared individual knowledge gain over the duration of our experiment regardless of treatment group. Thus, we compared pretest scores with posttest scores with a within subjects design (Figure 34). To analyze how our treatments influenced student learning, we used the same raw data in a posttest only comparison between treatment groups. In this case, the pretest data was a covariate for the posttest data (Figure 35). The experimental design for our transfer analysis was a posttest only, between subjects design (Figure 36). We looked at how our treatment influenced the scores of the transfer test.

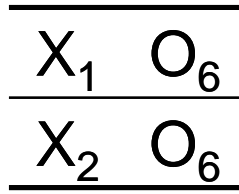


Figure 36. Between Subjects Design to Analyze how the Treatment Influences Transfer

Our motivation analysis also was a between subjects designs. In the case of the MSLQ, we compared the posttest results between groups and used the pretests as covariates (Figure 37). Our analysis of Experiences that Energize followed a similar design with the only exception that the covariate data were the averages of the four pre-treatment questions (Table 16 in Appendix B). This accounted for individual differences, for example that some people might have reported that they were energized by activities like copying grammar lessons from the board.

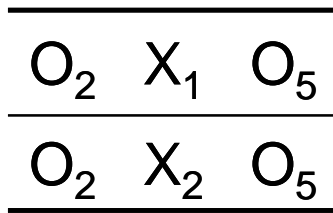


Figure 37. Between Subjects Design to Analyze how the Treatment Influences Motivation (O_2 is a Covariate)

The controlled variables in this study were the working-environment, computer usage, the exploratory, anchored learning environment, the presence and speech of agents, and external resources. Additionally, we decided to control for exposure to instructional materials and the opportunity to learn them. Because our system design was cycle-based, we gave students time to complete all cycles, before advancing to the posttests. Because we could only control for exposure or time but not both, this decision prevented us from controlling for time. The dependent variable was learning by teaching the teachable agent.

Because we had a heterogeneous set of laptop models, technology savvy students were excited about using the latest, fastest, or smallest computer in our set. Because assigning one computer over the duration of the whole study would have evidently influenced motivation, we controlled this variable by rotating students to a different computer in each session. To control for influences through the working-environment, we used a single room that the middle school provided during the whole experiment for all groups.

We could only partially control for the confounding effect of resentful demoralization. Initially we discouraged students to talk to each other about the environment to avoid that learners from the alternate condition would become discouraged because they could not teach Billy. However, it would be too idealistic to think that all students complied with our request. Therefore, we only relied on the rich compensatory treatment that included the simulation, representations, the teacher agent, and a non-teachable Billy-agent.

Because our experiment spanned multiple days for each group, we assessed maturation in our survey. However, because we used a fully randomized design, the most likely cause for any observed differences in maturation would be a result of the treatment.

Method of Analysis

We expected that our collected data would be the normally distributed in most cases. Therefore, we used the parametric normal linear model¹ (LNM) on all tests, unless tests showed that the data were not normal. This model includes analysis of variance (ANOVA), and analysis of co-

¹ In literature, this model is also called general linear model (GLM), normal linear model, or linear normal model (LNM). The different names originated probably in attempts to avoid confusion with the *generalized* linear model (GLZ or GLM). The GLZ is a further generalization of the LNM. It removes the assumption of normality, and can be parameterized with different distributions. However, despite its similar name it shall not be confused with the model we use.

variance (ANCOVA). We used the software package SPSS™ version 11.5 to perform the analyses.

The normal linear model makes the following assumptions: (1) The response variable can be expressed as a linear combination of functions of the covariates, plus a random error term; (2) The residuals are distributed normally; (3) The error variances of residuals are normal (*homoscedasticity, homogeneity of variances*); (4) Fixed independent variables are measured without error; and (5) There are no influential outliers.

We verified that our data did not violate normality by plotting histograms, and performing the Shapiro-Wilk normality tests. The Kolmogorov-Smirnov test was not used because of the problems it has with small sample sizes. Instead, we rely on the Shapiro-Wilkes test, which is more reliable in comparison studies with other goodness of fit tests (NIST/SEMATECH n. d.). Still, it is possible that the Shapiro-Wilk test fails on normal data if outliers have to be included. However, the normal linear model is especially robust against assumption violations if sample sizes are almost equal, like in our case. If we had sufficient reason to believe that normality was violated, or that our data did not follow the normal distribution we performed non-parametric Mann-Whitney tests.

In addition, we tested the null hypothesis that the error variance of the dependent variable was equal across groups with Levene's tests, and the null hypothesis that the observed covariance matrices of the dependent variables were equal across groups with Box's M tests, where applicable. Results of these tests are discussed in the next chapter.

Activities of Students in the Alternate Condition

The only difference between our experimental and alternate condition was that the students in the alternate condition did not teach an agent. Unfortunately, this resulted in other small changes to the cycle structure. In this section, we summarize treatment differences that were described in Chapter IV by paying special attention to the experimental design. We conclude this section with a high-level overview of our experimental design in Table 2.

At the beginning of the study, we introduced students to the system and our agent Billy in a classroom lecture that all participants attended at the same time. When students started using the learning system, the system introduced the teachable agent, Billy, as a co-learner who had low self-confidence in mathematics in the experimental condition, and Billy as an intern writing a paper about the company, who would not participate in learning and problem solving tasks in the control condition.

The differences in the activities of the two groups were most pronounced in the teach phase (Make Smart Tool in Figure 6). In the experimental group, Billy reviewed the problem first. In the alternate condition, students started by reviewing the problem with the teacher agent instead. However, the dialogues and answer options were the same for both groups.

To control for the presence of the teachable agent, learners of the alternate group could also talk to Billy, but all of the dialogues just involved small talk and had no learning content. The remainder of the teacher's dialogue was identical in both conditions. Overall, students in the alternate condition were likely to spend less time in the teach phase than the experimental condition. This was because students in the experimental condition had to spend time in teaching their agent using the dialog mechanisms provided by the system.

In the alternate condition, Billy was not involved in the quiz phase. Instead, students generated the answers to the problems on their own. Like in the experimental condition, the student could ask the teacher to check their answers, and to provide correct solutions. The feedback from the teacher was identical in both conditions, but the user interface to access this information was different (Figure 38). To obtain feedback from the teacher, the student had to enter a solution to the problem. Ms. Mathie, the teacher, checked the answer (i.e., the students had to click on the check answer button), and then told the students whether their solution was right or wrong. If the student asked her to provide a solution path (by clicking on the give solution button), Ms. Mathie then explained how she solved the problem. This process of obtaining feedback was identical to the process of getting feedback from the teacher agent in the experimental condition (Figure 12). Students in the control condition did not interact with the teachable agent. Therefore, the user interface for the control condition did not have two tab panels (Figure 10, Figure 11) to communicate with Billy. Overall, the students in the experimental condition had to go through many more steps, in first teaching Billy, then getting him to answer quiz questions, and then observing the feedback that the teacher Ms. Mathie provided. This resulted in the experimental group having to spend much more time in going through the same amount of material. Often their interactions with the teacher and Billy were quite repetitive. This could be a reason why we did not see the differences in knowledge test scores that we had expected between the experimental and alternate group.

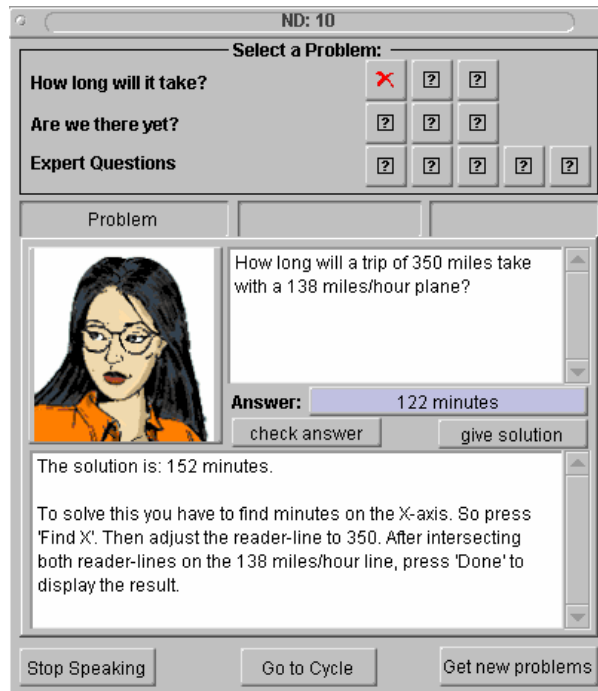


Figure 38. Quiz Phase: User Interface of the Alternate Experimental Condition

Since we decided to randomize quiz-questions after the first time only on request by students (get new problems button in Figure 38), learners in both conditions had the same kind of opportunities to solve quiz problems by trial and error¹. Randomizing quiz questions only on demand avoids that questions flip-flop between right and wrong each time they are asked. Students could retry the same question with the same numbers again, and would not be confused that a question suddenly reverted to unsolved again because the new random numbers were out of range of their line on the graph. In the alternate condition, learners could enter an intentionally incorrect answer, let the teacher check it, and provide a correct solution, which they memorized to answer the same question later. In the experimental condition, students could skip checking the agent's answer by just stating that Billy's answer was correct, then asking the teacher for a correct answer.

¹ A future version of the system should be modified to avoid this better.

Then they could return to the teach phase and use another dialogue option relating to this question to teach the agent differently and return to the quiz.

Table 2. Table of Differences between Experimental and Alternate Condition

	Experimental Condition	Alternate Condition
General Introduction	Classroom lecture. No differences between groups.	Classroom lecture. No differences between groups.
Problem Introduction	Billy is a co-learner with low self-confidence in mathematics.	Billy is an intern who is writing a paper about the company and does not participate in learning and problem solving tasks
Teach Phase	Billy reviews the problem introduction. Student and Billy interact through dialogues; Billy is taught and questioned. Students learn from the teacher agent.	The teacher reviews the problem introduction instead of Billy. Student and Billy can engage in 2-3 small-talk dialogues without domain or learning content. Students learn from the teacher agent.
Quiz Phase	Three tab interface: (1) The student displays the problem & Billy's solution; (2) The student makes suggestions to Billy; and (3) The student requests the teacher to check, requests teacher's solution.	One tab interface: The student displays the problem, provides an answer, requests the teacher to check, and requests the teacher's solution (Figure 38). All in one user interface.
Test Phase	(1) Students solve a test to grade Billy later; (2) Students review own and Billy's answers; and (3) Students compare the number of their and Billy's correct answers.	(1) Students solve a test; (2) Students review their answers; and (3) Students view the number of their own correct answers

The test phase helped differentiate between the two groups in a more substantial way. To recapitulate, students in the test phase first solved a set of problems, and then reviewed their solutions to obtain summative feedback. In the alternate condition, students took an exam. Except for

this arrangement, the user interface, problems, and interaction with the system were the same for both conditions. During the review, the agent's solution was replaced by the teacher's solution, which was displayed in an alert panel in the experimental condition (Figure 16), and the student clicked "OK" instead of "Right" and "Wrong" to advance to the next question. In the summary, the lower two feedback boxes relating to the agent's right and wrong answers were removed.

Population, Sample and Randomization

Our study was conducted in two 6th grade mathematics classrooms in a public school in the southeastern United States. The two classrooms were picked non-randomly, as we had to rely on voluntary participation of the classroom teachers. We invited all 49 students from the two classrooms to participate in our experiment. Class P contributed 25 students and class Q 24 students.

The demographics of students in the county of our school were White 41.1%, African American 46.1%, Hispanic 9.2%, Asian 3.4%, but our sample in the particular school apparently had fewer than average African Americans. The students in this school were high achieving, and selected based on test scores and a lottery system. To address integration problems, the school makes a strong attempt to maintain a 33% minority student ratio in the school.

We used two strata to assign our students to experimental and control condition, to control for a teacher or classroom effect. We ensured that each class was equally divided into our two treatment conditions.

Three students of class Q decided not to participate. Additionally, one student decided to drop out during the treatment. This reduced our final count of participants to 45 students. The

experimental condition had 23 participants (12 from class P and 11 from class Q) and our alternate condition had 22 students (13 from class P and 9 from class Q).

Measures

To verify our hypotheses, we selected a set of motivational and learning measures. For the first set of hypotheses, we used the Motivated Strategies for Learning Questionnaire and Experiences That Energize (ETEs), and to measure changes in learning and transfer we used knowledge tests that we created particularly for this study. We describe all measures in the following sections.

Motivated Strategies for Learning Questionnaire

We used a subset of the Motivated Strategies for Learning Questionnaire, MSLQ, (Pintrich et al. 1993, 1993) that we modified for the purposes of our study to measure the motivation and the learning strategies employed by the participating students. This questionnaire was administered along with the knowledge pretest (O_2 ; Figure 33) and the knowledge posttest (O_5). The MSLQ scale is a seven point Likert self-report scale, which was created and normed on 400 college students from 37 classrooms, spanning 14 subject domains and five disciplines. Correlation with final grades was reported as significant, albeit moderate.

Table 3 lists all the variables used in the MSLQ with their internal consistency reliability coefficients (Cronbach's alphas) in parentheses. Due to time constraints, we removed a few variables, which we thought were more relevant to course-based instruction. Table 3 also gives example questions for each variable. The full test is included in Appendix A. We used all three *motivational value* components of this scale, which are intrinsic and extrinsic goal orientation and task value ($\alpha = 0.68, 0.62, 0.90$). *Expectancy components* that we measured are self-efficacy (for

learning and performance), and control beliefs ($\alpha = 0.93, 0.68$). We did not measure test anxiety, as it was unrelated to our study¹. For measuring *cognitive and metacognitive strategies* we selected critical thinking, and metacognitive self-regulation ($\alpha = 0.80, 0.79$). We skipped rehearsal, elaboration, and organization because those strategies were not relevant to our learning environment, and we did not expect to see an effect during our study. From *resource management strategies* we used effort regulation, peer learning, and help seeking ($\alpha = 0.69, 0.76, 0.52$). We did not measure the variable time and study environment as it is a fixed factor in our setting.

Table 3. Motivated Strategies for Learning Questionnaire Example Questions

Control Belief	It is my own fault if I do not learn the ideas in mathematics class.
	If I try hard enough, I will understand the ideas in this mathematics class.
Critical Thinking	I often find myself questioning things I hear or read in mathematics class in order to decide if I believe them.
	I like to play around with ideas of my own that are related to what I am learning in mathematics class.
Extrinsic Goal Motivation	I want to do well in mathematics class because it is important to show my ability to my family, friends, teachers, and others.
Effort Regulation	Even when the mathematics book is dull and uninteresting, I manage to keep reading until I finish the assignment.
Help Seeking	When I do not understand the ideas in mathematics class, I ask someone to help me.
Intrinsic Goal Motivation	In mathematics class, I prefer to learn about things that really challenge me so I can learn new ideas.
Peer Learning	When studying for my mathematics class, I often try to explain the ideas to someone else.
	I try to work with someone else to complete the assignments in mathematics class whenever possible.
Self Efficacy	I am certain I can master the skills being taught in mathematics class.
Self Regulation	When I study for mathematics class, I set goals for myself in order to get the most out of the time I spend studying.
Task Value	I think that I will be able to use what I learn in this mathematics class in life or other classes.

¹ In retrospective, we have observed some test-anxiety in the test phase of the cycles, which we did not anticipate.

We adapted this questionnaire from its original form, which used *course* for referring to the subject of instruction. We reworded all the questions to refer to *mathematics class* in the pretest and *computer task* in the posttest. The questions were also reworded to fit the vocabulary and understanding of sixth grade students. Since the pretest was intended as a covariate, it removed the effect of motivation in mathematics class that existed before the experiment, while the posttest focused on the effects of the treatment. In the introduction to the MSLQ posttest, we emphasized that our computer environment would not replace regular mathematics instruction in the classroom. Rather, it should be looked upon as a tool to further one's learning after classroom instruction. We presented this test on the computer to simplify data gathering and evaluation.

A pretest - posttest measure like the MSLQ does not usually provide insights into the student activities during an experiment. To capture students' thoughts and feelings during the study we used a tool called *Experiences that Energize*, originally developed and used by Brophy (Brophy 1998, 2003), to give us insight into the motivational state of learners while they used our system.

Experiences that Energize

Experiences that Energize (ETE) (Brophy 1998, 2003). This measure determines motivation during an activity and is comparable to intellectual *flow* (Csikszentmihalyi 1990). Subjects repeatedly reported their current energy level on a single seven point Likert-style question while they were involved in learning activities (Figure 40). For example, a teacher may interrupt a class session several times and asks students to write down their ETE score. When plotted as graph of average ETE scores against time, peaks show those interventions that energize the students during instruction. High responses imply heightened intellectual flow, which implies increased motiva-

tion, attention, and enjoyment of the reported experience, while low responses imply boredom and disinterest.

Table 4. Experiences That Energize Measured while Students Use the System.

Problem Read	You are starting or resuming to make your Smart Tool. Please rate your energy level.
Quiz Entered	You are going to try your Smart Tool. Please rate your energy level.
Quiz Finished (QD)	You have just finished trying your Smart Tool. Please rate your energy level.
Test Graded (TD)	You have just finished solving real problems. Please rate your energy level.
Test Re-viewed (TR)	You have just finished reviewing your problems. Please rate your energy level.
Test Summary Seen (TS)	You have just finished looking at the summary. Please rate your energy level. After this, you can go on in the cycle and read your next problem introduction.

We were faced with the alignment problem when trying to use this measure in an effective way. In classroom instruction, all students are simultaneously exposed to the same experience, which makes it easy to detect peaks in the graph of the average of this measure versus time. However, students work at their own pace and in their own way in a learning environment, so finding collective peaks in time does not work. For our research, we identified interesting transition points and asked students at each of these points to rate their energy level (Table 4). This allowed us to collect cumulative data at well-defined points, so we could graph the ordered occurrence of these transition points against the average ETE score.

A disadvantage of Experiences that Energize is that they may disrupt flow. ETEs that occur during an activity may interrupt a student's thinking and can be annoying. Hence, we carefully

chose to use ETEs only when the student transitioned between activities, especially when they transitioned between phases of a cycle. At these times, ETE questions were displayed to the student using the interface shown in Figure 40. The instructions that students received to report ETE values can be found in Appendix B.

Knowledge Test

We evaluated student learning of declarative and procedural knowledge with a multiple-choice test that we used for pretest and posttest (see Appendix C). This test covered the curriculum material presented in the learning cycles, and required good understanding of concepts to achieve a high score. Some items could be looked upon as near transfer problems. The test measured the ability of the students to use graphs (read points, add points, and solve problems using the graph), answer standard mathematics word problems, and understand the relationship between slope and speed of a line. An example question is illustrated in Figure 39. Problems between pretest and posttest were varied slightly (changed numbers or slopes) to avoid the problem of memorization of answers.

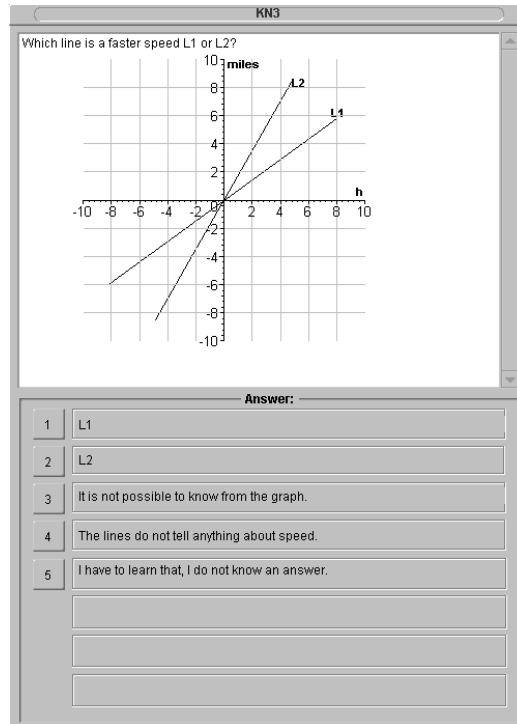


Figure 39. Knowledge Pretest Example: Which Line is a faster Speed?

The questions were designed to decrease student guessing. Questions about graphs show all four quadrants of a graph to illustrate problems in pretest and posttest (see Figure 39), while our graph tool in the learning environment used the first quadrant only. This required students to think beyond the first quadrant problem solving skills that they had acquired during the main study. To avoid that students can guess correct solutions, we designed the answer-choices to the questions to cover up to eight possible answers. For example, a question about reading a point on a graph provided eight answer choices that exhausted all possibilities.

The pretest contained 11 and the posttest had 16 questions (see Appendix C). The posttest added five questions to the pretest. The extra questions were not used in the statistical analysis of results for knowledge gain. However, the extra questions could be included in an analysis to compare treatment groups with the pretest as covariate. All questions were graded by awarding

one point for a correct answer and zero points for any incorrect answer. In situations where we thought students could receive partial credit, we did give a score of 0.5. Two questions that related to material in the fourth cycle, which was not covered in the experiment due to time constraints were removed from the tests¹. We did not weigh questions for difficulty and importance.

Online Testing Interfaces

As discussed, our agents communicated with their student-teachers using a multiple-choice dialog interface. This framework (Figure 27, top six objects) was modified to enable online pretests and posttests. The MSLQ questions and ETEs shared a common interface (Figure 40), while the knowledge test used a vertical layout that allowed us to display longer answer choices (Figure 39).

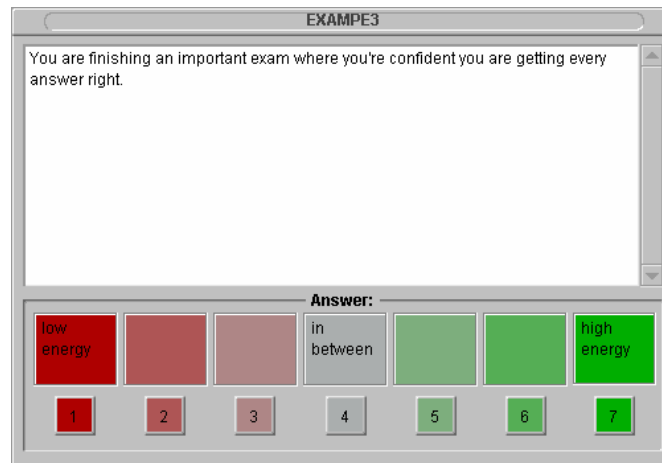


Figure 40. Electronic Likert Testing Interface for MSLQ, ETE (this picture), and Knowledge Tests

The online testing interface performed automatic scoring of tests. Each multiple-choice answer was associated with a variable and a score. The last question of a test notified the dialog

¹ The pretest had originally 13 questions.

controller through the agent communication interface to score the dialog. Then, each variable was summed, and written to the log-file, from where it could be copied onto a spreadsheet.

Near Transfer Task

After students finished all the posttests, we conducted a transfer test, where students created a flow-rate graph by running a real experiment in another domain. This was introduced as an extra cycle in the learning environment. This cycle provided a blank graph tool (without units or numbers on axes), and the students did not have the simulation tool to generate data points from which the graph could be constructed. Instead, the students had to conduct a real experiment, collect relevant data, and then plot it to determine the rate line. We told students that the teachable agent, Billy, would not be available for this part of the study. Instead, the mentor agent, Ms. Mathie described the problem to students of both groups.

The problem described a situation where the company's engineer, Larry, had to repair a broken engine, which would require a new fuel injection nozzle. Students had to conduct experiments to find the flow rate of fuel through a chosen nozzle and plot this information on a graph. Students were provided a notepad, a stopwatch, a 100 ml measuring cylinder, and a water nozzle, simulating the flow of the fuel-injection nozzle (see Figure 41) to collect data.

The notepad included a blank three-column table without units. Students were told to record values on the notepad. They received instructions on how to use the stopwatch, and how to read a water level on the measurements cylinder. No further instructions were given. Students worked individually at their own pace to collect the data and make entries into their table. When they finished this task, they returned to their computers to draw the graph and used it to compute the flow rate. The students were instructed to write their answer into their notebooks.

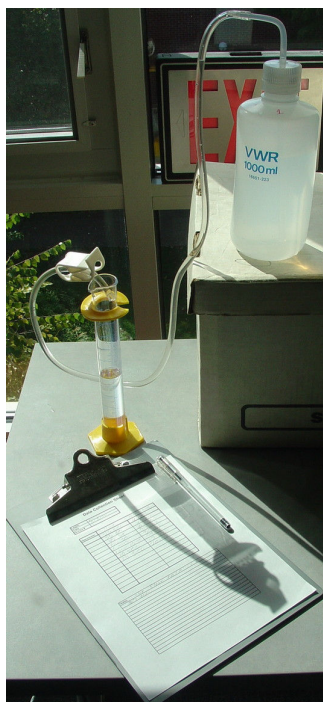


Figure 41. Photo of Equipment Supplied to Students in the Transfer Test (without Stopwatch)

We graded the transfer test and generated a composite score calculated by summing the scores for the three separate tasks: the quality of the produced table, the labels students put on their graphs, and the quality of the resulting line. Each of these tasks was assessed with several item scores that awarded one, half, or zero points for a correct, partially correct, or incorrect answer. Table 5 gives an overview of our grading scheme. Half points were rarely awarded except for a few borderline cases, for example, when students used the graph unit on the right axis in the wrong spot of the graph label, without compromising the functionality of the graph.

Table 5. Near Transfer Task Grading Scheme

Table		Graph Labels		Line Quality	
Table Label	1	Graph Label	1	Line (see text)	4
X-Axis Label	1	Graph Unit	1	(0,0) part of line	1
Y-Axis Label	1	X-Axis Label	1	Y-intercept ≈ 0	1
X-Axis Unit	1	Y-Axis Label	1		
Y-Axis Unit	1	X-Axis Unit	1		
Data points (see text)	5	Y-Axis Unit	1		
		Min. X (0 or 1)	1		
		Min. Y (0 or 1)	1		
		Max. seconds	1		
		Max. ml	1		

Students got a maximum of 10 points for generating a correct table. The table was graded by the following criteria: presence or absence of a table label, and labels and units for x and y measurements. Additionally, students received 5 points for measuring five or more data points, or one point for each non-duplicate data entry in their table.

The second part of the transfer score was the labeling of the graph on the computer graph tool. Students could get a maximum score of 10 points for a correctly labeled graph. We awarded one point for each correct axis label, axis unit, and minimum (value of 0 or 1) for each axis. Because the maximum for each axis depends on the collected data, we have allowed values between 60 and 120 as correct solution for each axis.

The third task assessed the line quality, and students were awarded a maximum of 6 points for a correct line. Students received one point if the line included the origin of the graph, and a second point if the line had a reasonable (extrapolated) y-intercept within five units around the origin. Participants received 4 points, if their graph was a straight line, and the line was created from the data they had recorded. 3 points were awarded for a straight line that did not fit the data collected. Students received only 2 points if their graph was not a single straight line. Finally,

they got at least 1 point for any marginal attempt to draw a line. Students who drew horizontal, vertical, interrupted, or no lines received zero points. The maximum score a student could obtain for the transfer task was 26 points.

Survey

Before students solved the posttest problems, we surveyed their experiences during the study. The survey had 14 questions (see Table 6), and assessed maturation, interactions with the staff, self-reported perception of how much they learnt, and whether they had fun. They were also asked to rate their experiences with specific components and agents in the system.

Table 6. Survey Questions

1.	During the experiment, how many times did you study materials about graphs or rate problems when you were not using our computers?
2.	During the experiment, how many times did you ask other persons about graphs or rate problems outside the experiment, and they helped you figure things out?
3.	How much help from the research staff did you need to complete the cycles?
4.	How often did the research staff solve a math problem for you?
5.	How did you like the voices and speech in the learning software?
6.	Did you like Ms. Mathie?
7.	Was Ms. Mathie helpful in figuring things out?
8.	Did you like Billy?
9.	Did you like the simulation?
10.	Did you like the graph-tool?
11.	How much have you learned while using our system?
12.	Rate the overall difficulty of the problems and tasks:
13.	How much fun was learning with this software?
14.	Was Billy a good student?

Questions asking for how many times something occurred coded 1 for never, 2 through 6 for 1, 2, 3, 4, and 5 times, and 7 for more often. Other questions used a standard seven point Likert ranking, like the MSLQ.

User Action Traces

We analyzed the student's action traces that our system recorded in a log-file for each student. An especially interesting and informative measure was the number of modifications students made in the graph tool, especially how often students added points, changed lines, and read the graph (successfully or not). The system also traced how often a student succeeded or failed in running the simulation, and recorded quiz and test scores. The learning environment also saved the full state of the graph after each modification, which allowed us to replay how a student created a graph as a movie, or analyze a student's graph at specific times during the experiment. This feature was useful in obtaining the students' graphs of the near transfer test. Extraction and counting of these measures was done automatically with UNIX tools (egrep, wc), because every action or score was tagged uniquely.

Procedures

Before the experiment, we obtained the permissions from the Internal Review Board at Vanderbilt University and the local Metropolitan School District to conduct the experiments in two 6th grade mathematics classrooms. With help of the teachers, we integrated our experiment into regular classroom instruction. After parents and students signed the consent forms to participate willingly in our experiment, an initial introductory lecture discussed the experiment and presented the anchoring context. We explained to the students our reasons for conducting this re-

search study, and gave students a broad overview of what they may learn during the study. We also made clear to the students that they would be assigned to one of two types of systems. We explained to students that to the extent possible they should avoid talking about the details of their system or their own work with other students, because otherwise, it may corrupt the results of our study. At the end of session one, we gave a short demonstration of the learning environment (without the teachable agent) and showed students the introductory movie "Working Smart," one of the twelve Adventures of Jasper Woodbury (Cognition and Technology Group at Vanderbilt 1997).

After the introduction, the students were split into groups of 11 students or less, and we assigned times when they could work on the system¹. We pulled students from regular classroom activities to participate in our experiment. Each student used the system for ten sessions of 45 minutes each. The first session was used for the pretests (O₁, O₂), and the last two sessions for the posttests (O₄, O₅, and O₆). Subjects worked individually on separate laptops, and were strongly urged not to look at each other's work. All participants had enough time to complete all cycles before advancing to the posttests.

When students finished all cycles they were asked to take the posttests and the transfer task in the next scheduled session. After finishing the experiment, students in the alternate condition had the opportunity to use the teachable agent system.

¹ The experiment itself was limited to groups of less than 11 students at a time, because we had a limited number of laptops.

Summary

In this chapter, we introduced the experimental design for this research. To show how learning by teaching influenced students, we split students into two groups, one that learned by teaching an agent and another that learned on their own. Students in both groups learned from the teacher agent, Ms. Mathie, who presented domain knowledge in interactive dialogues that were identical for both conditions. To focus our findings only on the effects of learning by teaching, we carefully designed the system to keep all feedback from and interactions with the system that were unrelated to our treatment equal between groups.

To assess motivation among our students we used modified versions of the Motivated Strategies for Learning Questionnaire (Pintrich et al. 1993, 1993) using a pretest-posttest design. Modifications were made to reduce the reading level of the questionnaire and account for the intended motivating tasks of our study (mathematics class, computer task). In addition to this measure, we used Experiences that Energize (Brophy 1998, 2003) to attempt to capture the motivational state of students when they were using the system.

To assess knowledge gains, we designed a multiple-choice pretest, posttest, and a near transfer test. The knowledge test included traditional word problems, some of which were direct applications of the graphs that students created before; others required additional manipulations to derive the answer. The near transfer test involved creating a graph from real measurements generated by an experiment that students conducted to find the flow rate of a nozzle. Students measured time and milliliter values with a stopwatch and a measurement cylinder, recorded them on paper, and created a graph in the computer environment. At the end of the study, we conducted a

small survey, which gave us direct feedback on what students thought about components of our environment.

CHAPTER VI

RESULTS

This chapter discusses the results of our experimental studies with the teachable agents system. In the previous chapter, we presented the detailed design of our experiments. We expected that learning by teaching would have a positive influence on learning and motivation of students. To study this, we used four measures: the knowledge test, the near transfer task, the Motivated Strategies for Learning Questionnaire, and Experiences That Energize. We used these measures to validate our hypotheses, which are summarized in Table 1 in the previous chapter. We were interested to find if all students learn (H_{KGAIN}), and if learning, motivation, and transfer (H_{KNDIFF} , H_M , H_{ETE} , and H_T) differ between treatment groups.

All hypotheses were tested using the parametric normal linear model (also called the general linear model) as it is implemented in SPSS™ version 11.5. We used this model to perform the analysis of variance (ANOVA) and the analysis of covariance (ANCOVA). The statistical assumptions made for the knowledge test are discussed in this chapter. The tests on the data for the remaining analyses are discussed in Appendix E.

Knowledge Test Results

In this section, we discuss learning of procedural and declarative knowledge that we measured with our knowledge tests. To explore our hypothesis that all students learned by using our system (H_{KGAIN}) we evaluated knowledge gain from pretest to posttest using a repeated within subjects

measure¹. To show a treatment effect, we looked at knowledge gain difference between conditions (H_{KDIFF}). Because the time that students spent using our system and the amount of formative feedback during the quiz could influence knowledge test results, we begin our discussion with these topics in the next subsections. The time on task is especially relevant, because the experiment controlled for the opportunity to learn (students should complete all cycles), and therefore it became harder to control for time.

Time on Task

This section discusses issues of the time that students spent on our system: First, we look at the time to complete all tasks. Second, we look for time differences that can be attributed to a particular group or class. We also asked the teachers to rate problem solving speed of their students to verify that our treatment groups were equivalent in this respect.

Time to Complete all Cycles

The time that students spent on each cycle, i.e., the time from when they entered the teach phase to the time they finished reviewing the summary in the test phase, was recorded in a log file (Figure 42). It is interesting to note that the experimental group took on average 387 minutes to complete the curriculum ($\sigma = 87$), while the alternate group took 319 minutes ($\sigma = 84$). In other words, it took on average 68 minutes more for the experimental group to complete the tasks of all three cycles. The difference in time between groups for cycle one is 30 minutes, for cycle two 50 minutes, and for cycle three there is no difference between the conditions. The students in the alternate condition who finished early were given the opportunity to work on previous cycles.

¹ For the repeated measure analysis, we have selected only the subset of 11 equivalent questions between pretest and posttest.

This was done in an attempt to balance the time between the groups, but only one student accepted this offer, and this student also worked on the system only for a very short time. We did not ask students for the reason as to why they did not want to go back, but one likely cause was that students did not want to miss out on material being covered in their regular mathematics class.

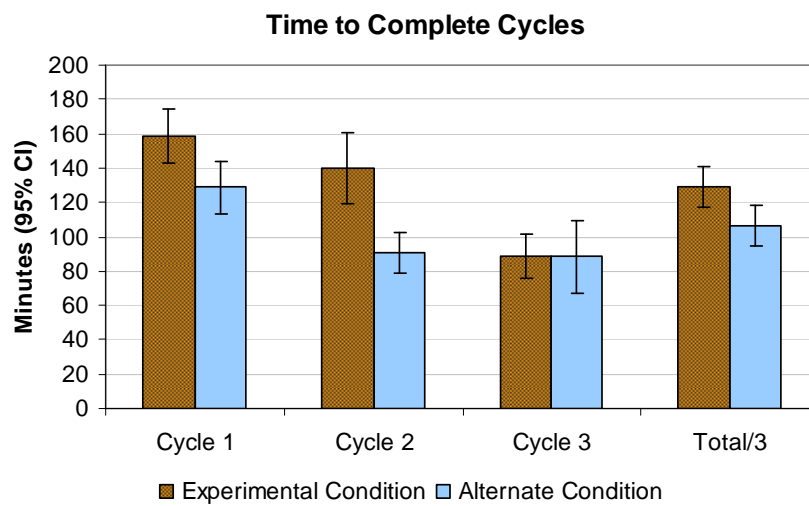


Figure 42. Time to Complete each Cycle and Average Time per Cycle¹

The task of teaching the agent, Billy, in the experimental condition probably was the biggest reason for the time difference. The dialogues for teaching Billy were quite involved. For example, teaching Billy in cycle one required 20 exchanges between the student and Billy. The numbers for cycles two and three were 27 and 17, respectively². In addition, it turned out that the quiz interface for the experimental group was harder to use, therefore, students would require more time in the quiz phase. The time data clearly indicates that our teachable agent group required

¹ We have divided total time (sum of cycle 1 to 3) by three to show the results in a single figure.

² The alternate condition had five dummy dialogues with Billy per cycle, which did not balance the time difference.

more time to complete each of the cycles. The decrease in the average time students spent per cycle can be partially explained by the fact that students became more familiar with the interfaces and tools, and they became more proficient in managing the dialogue structure.

Class Differences

It is interesting to note that most of the time disparity between conditions in the first two cycles was caused by students from one class (Figure 43). In the last cycle, the differences decreased. A likely cause for the reduction in time difference (regression towards the mean) for cycle 3 was students realizing they had only a few sessions left to finish their study. Overall, we assume that the causes for the time disparity can be attributed to the differences in the environment, time management skills of the students, and the teaching styles employed in the two classes.

Overall, looking at the time differences in Figure 43, but especially those for class Q, give a good indication of the time penalty attributed to the learning by teaching task. Our estimated extra time required for teaching is in the range of 20-25 minutes per cycle.

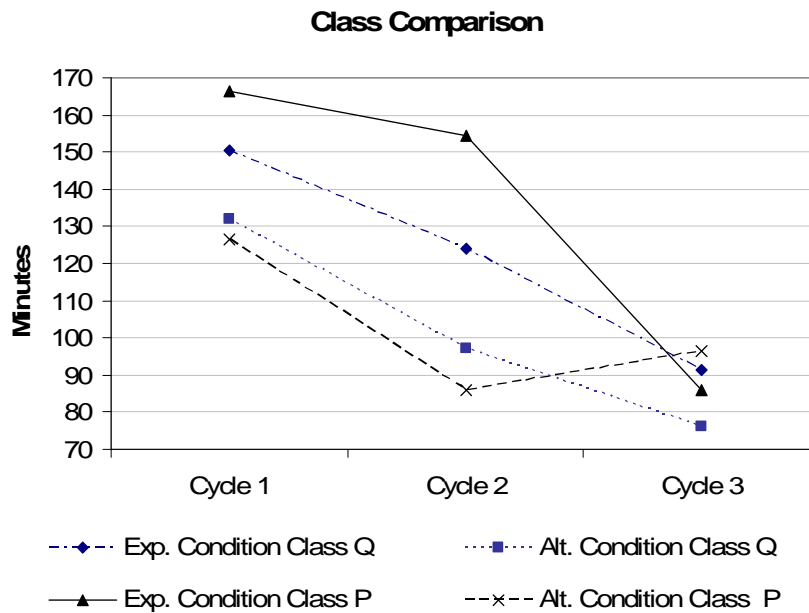


Figure 43. Time to Complete each Cycle depends on Classroom

Group Equivalence - Teacher Rated Problem Solving Speed

To ensure that the observed time difference did not stem from non-equivalent group assignment (which is unlikely in the case of a fully random assignment), we asked the mathematics teachers to estimate their students' problem-solving speed that they had observed during regular mathematics class on a scale of 1-20. The averages of these ratings for both conditions were very close (11.17 vs. 11.14), and confirmed the equivalence of experimental and alternate condition in respect to problem solving speed.

Correlation of Knowledge Scores versus Time

Because students of our experimental group took more time to complete the cycles, the natural question is whether the additional time spent on the system resulted in additional learning. We analyzed if there was a positive relationship between the time and knowledge scores.

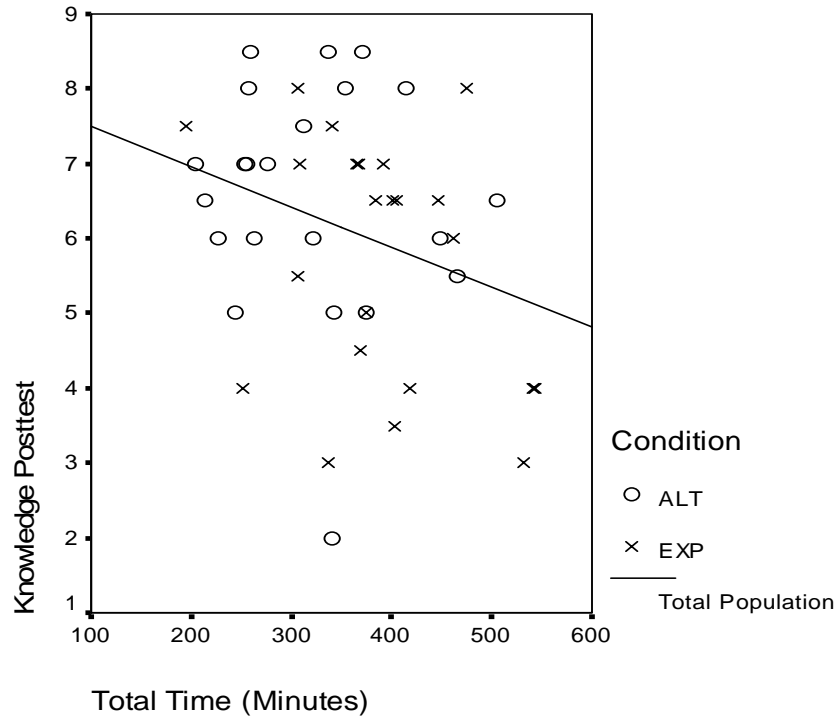


Figure 44. Scatter-plot of Knowledge Score versus Time

We plotted the posttest scores versus total time (Figure 44). The regression line shows a slight negative relationship: In other words, the longer students took on task, the lower their score in the test phase. The result shows a very low negative Pearson Correlation in Table 7. This confirms that the additional time spent on the system did not result in better learning. The more likely interpretation of the time on task result is that weaker students took more time to solve the problems.

Table 7. Pearson Correlation of Knowledge Score versus Time

		Knowledge Posttest	Total Time (Minutes)
Knowledge Posttest	Pearson Correlation	1	-,302(*)
	Sig. (2-tailed)	.	,044
	N	45	45
Total Time (Minutes)	Pearson Correlation	-,302(*)	1
	Sig. (2-tailed)	,044	.
	N	45	45
* Correlation is significant at the 0.05 level (2-tailed).			

Quiz Behavior

Student performance during the quiz phase is an indicator of how well students have participated in the experiment. We looked at two variables that describe student’s quiz behavior: number of correct answers generated, and frequency of asking questions in each cycle.

Although, students could game the system and could get their answers right in both conditions by trial and error or using teacher answers¹, the failure to answer multiple questions in this phase suggests low participation of students and could likely affect knowledge test scores. We analyzed the records, and found that almost all students succeeded in obtaining check marks for all the questions. One student showed poor participation in all three cycles, but his knowledge test scores were average. Two other students struggled only in one cycle. Thus, the number of correctly answered quiz questions did not provide evidence for any group or treatment differences. Hence, we further looked at how often students requested quiz questions, which told us how much they practiced.

¹ Randomizing problems on failure avoids trial and error, but it creates a moving target problem during learning by teaching. Therefore, we decided to randomize only on request by the student. The experimental condition could try different dialogue choices to teach the agent until they got the corresponding quiz question right. The alternate condition could use the teacher’s answer and type it in. However, students knew that they could not cheat in the test phase.

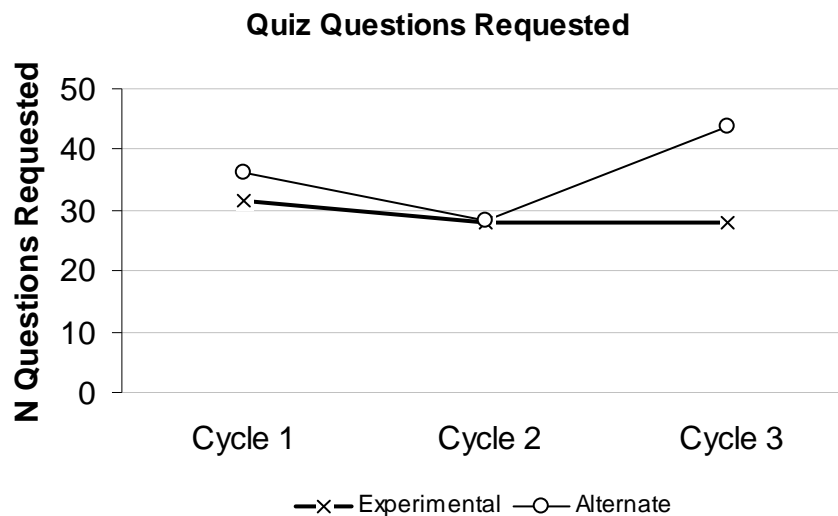


Figure 45. Quiz Questions Requested by Students in each Cycle

We looked at how often students requested questions in the quiz phase of each cycle. There were 11 to 13 questions in each quiz. Figure 45 shows how often students requested a question, not how often they solved it (correctly or incorrectly). Students in the alternate condition requested more questions in cycle three. The means for experimental and alternate condition were as follows: Cycle one, 31.8 ($\sigma = 13.7$) and 36.0 ($\sigma = 14.5$), cycle two, 28.2 ($\sigma = 12.1$) and 28.5 ($\sigma = 14.3$), and cycle three, 28.35 ($\sigma = 13.5$) and 43.64 ($\sigma = 27.5$). The spike in cycle three stems mostly from class P's alternate condition (compare with Figure 43) and coincides with the extra time that these students spent on the system. This subgroup displayed on average 52 quiz questions that is about 20 requests above average. We do not think that this difference indicates a treatment effect, because it occurs in only one cycle and stems from only one classroom. Therefore, we think that students probably preferred continuing with the experiment rather than go back to their regular math class.

Combining these findings with the time results allowed us to exclude time and extra practice as major confounding factors for test results. Therefore, the additional activities and dialogues that the experimental group needed to teach the agent explains most of the time differences between the groups.

Knowledge Test Analysis

We included all subjects in this evaluation, and only used the 11 questions in our analysis, that were common to knowledge pretest and posttest. Students could score a maximum of 11 points in each test. We evaluated the repeated measure knowledge gain (H_{KGAIN}) for all subjects regardless of treatment, and the differences between treatment conditions (H_{KDIFF}) using a normal linear model. The descriptive statistics are summarized in Table 8.

Table 8. Descriptive Statistics of Knowledge Pretest and Posttest

	Condition	Mean	Std. Deviation	N
O ₁	EXP (X ₁)	4.783	1.7309	23
	ALT (X ₂)	4.932	1.5142	22
	Total	4.856	1.6118	45
O ₄	EXP (X ₁)	5.913	1.9751	23
	ALT (X ₂)	6.614	1.5192	22
	Total	6.256	1.7826	45

To verify the assumptions of the normal linear model, we performed two tests: (1) the test for normality of the input data, and (2) the test for equal variance between groups. At first, we plotted histograms for the knowledge pretest and posttest, and then we overlaid a normal curve on the histograms (see Figure 46 and Figure 47). In both figures, we saw a deviation from normality in the center of the distribution. However, the Shapiro-Wilk test (Table 9) confirmed that

our data was normal. The Kolmogorov-Smirnov test was included for reference only (we did not have a large enough sample size to use this test).

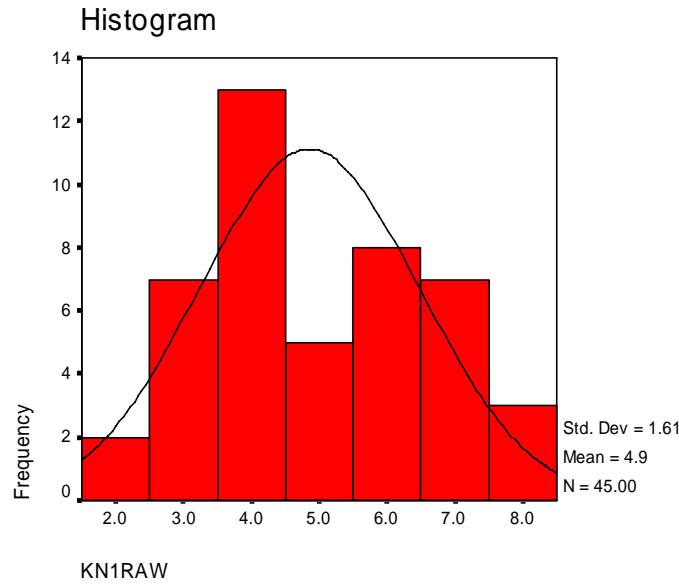


Figure 46. Histogram of the Knowledge Pretest (O_1)

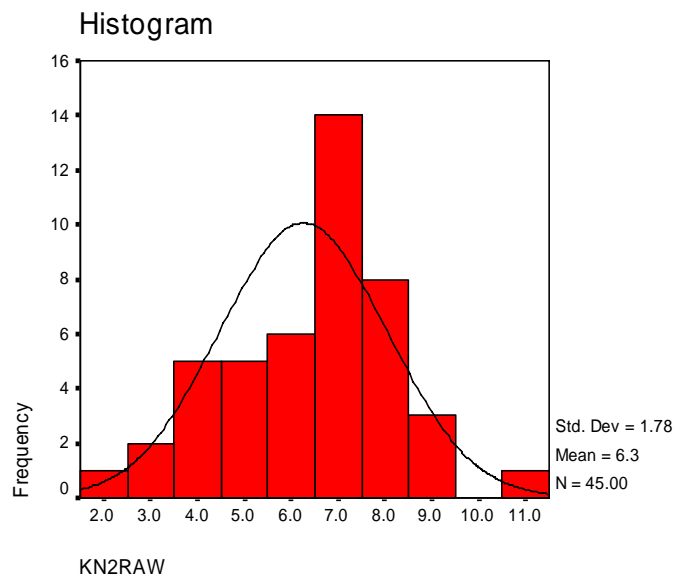


Figure 47. Histogram of the Knowledge Posttest (O_4)

Table 9. Normality Tests for H_{KNDIFF}

	Kolmogorov-Smirnov			Shapiro-Wilk		
	Statistic	Df	Sig.	Statistic	df	Sig.
Knowledge Pretest O_1	.191	45	.000	.957	45	.092
Knowledge Posttest O_4	.132	45	.047	.966	45	.200

The results of the Levene Test for equal variance of the dependent variable produced the results shown in Table 10. The results were not significant; therefore, the assumptions of our statistical model were not compromised.

Table 10. Levene's Test of Equality of Error Variances

	F	df1	df2	Sig.
Knowledge Pretest O_1	.735	1	43	.396
Knowledge Posttest O_4	1.313	1	43	.258

Tests of within subjects effects on the repeated measure knowledge, O_1 vs. O_4 (Figure 33 in Chapter V) showed significant learning for all participants with $F(1,44) = 16.491$ and $P < 0.001$, with a power of 0.978 for predictions at $\alpha = 0.05$. Thus, we rejected our null hypothesis $H_{\text{KNGAIN}0}$. This suggests that our systems helped students of all conditions learn domain material.

Tests of between subject effects (O_4 of the experimental group vs. O_4 of the alternate group) did not produce a significant difference between the means for our two conditions $F(1,44) = 1.31$ and $P = 0.257$. Thus, we failed to reject our null hypothesis ($H_{\text{KDIFF}0}$ in Table 1, of Chapter V). Therefore, this experiment did not conclusively demonstrate that learning by teaching helped

students to do better on the knowledge test. As a next step, we looked at differences between individual questions on the knowledge test.

Individual Extraordinary Questions of the Posttest

To study the issue of how much students learnt in a little more depth, we looked at three questions of the posttest, which produced scores with very large differences between the groups. The results were not analyzed for statistical significance. The number of correct pretest and post-test answers is shown in Table 11.

Table 11. Number of Correct Answers by Group on the Knowledge Tests

	A	B	D	G	H	K	L	M	N	O	U	V	W	X	Y	Z
Pre EXP (X₁)	21	15	7	0	4	5	6	10	21	10	-	-	-	-	13	-
Pre ALT (X₂)	21	15	11	0	3	3	15	9	18	8	-	-	-	-	11	-
Post EXP (X₁)	19	12	7	3	11	9	15	14	22	9	15	15	10	12	17	18
Post ALT (X₂)	20	17	6	2	13	8	15	16	20	15	12	15	3	6	18	18

When looking at individual questions of the knowledge posttest, three of the 16 posttest questions showed a strong bias towards one of the groups (O, W, and X in Table 11). For the other questions, the scores were about the same for both groups. We asked students (KN_O in Appendix C), "If we make a distance-time graph for a car driving at 70 miles/hour. The line in a graph shows the car's ... [time, distance, speed, acceleration]."¹ Students in the alternate condition provided six more correct answers to the question than the experimental condition. Distance

¹ In cycle two, students in both conditions answered the quiz question, "A constant speed in a graph is ... [a straight line, a curve, depends on the question]". Additionally, Billy could be taught, "A straight line without bends and edges is a single speed."

and time (stated in the question) were the most prevalent among the wrong answers. We could not find any reasons for this outcome.

The experimental condition was more successful in solving the other two questions. We asked (KN_X in Appendix C), "What are coordinates?", with the following options: (1) Some numbers to find a location; (2) The fewest numbers needed to find a location; (3) exactly two numbers to find a location: X and Y-coordinate¹. The experimental group outscored the alternate group 10 to 3. Although both groups answered this question as part of the quiz, the activity of teaching it to Billy has probably led to better understanding of the concept of coordinates in graphs.

So, why did the experimental condition learn this concept better? Students in the experimental condition requested the question related to this concept on average 3.74 times, compared to 3.05 in the other condition. Statistically this difference is not significant. The teacher's explanation of the correct solution and instructional material were the same for both conditions. Many dialogues mentioned coordinates, but did not focus on the concept that coordinates are the fewest numbers to that define a point in a two dimensional graph. The remaining difference is the activity of teaching the agent through the dialogue. We think that the most likely reason for this difference is the insight stated by Billy after being taught correctly: "So, that is why we write down points with two numbers." It is likely that the strategy of the teachable agent to reinforce what the student taught by appearing to have an insight led to better understanding of the concept.

¹ Additionally, learners in the experimental condition had the option to teach the agent the following three answers: (1) Coordinates are points in the graph that give a location. If taught, Billy would say on the quiz/if asked, "A point in the graph is a coordinate." (2) A coordinate helps to find a location on a line. Billy would say, "A coordinate helps to find a line." (3) Coordinates are the fewest numbers needed to find a location. Billy would say, "Coordinates are the fewest numbers that I need to tell someone to find a location. So, that is why we write down points with two numbers."

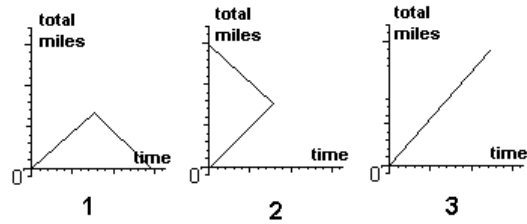


Figure 48. Illustrated Multiple Choices for a Posttest Question

The experimental condition also performed better on the question (KN_W in Appendix C), "A car drives at 45 miles/hour to another city 50 miles away. It then turns around and drives back without stopping. Which graph shows this best (see Figure 48)?" The score of this question was 12 to 6 in favor of the experimental group. Therefore, these students may have gained a deeper understanding of the covariation of time and distance.

For the remaining questions, the difference was only 1-2 points in favor of one or the other condition. Almost all students learned how to read a point from the graph (question A; see Table 11), identify axes (N), and distinguish lines that represented faster and slower speeds (Y). These questions showed a ceiling effect. Slightly fewer learners succeeded in writing a point into the graph (B). Seventy percent of the students were able to identify the line graph that represented a car that stopped for refueling correctly (V). The teacher agent taught this topic. Overall, neither group did very well in calculations of slope.

Transfer Test Results

Our primary hypothesis was that students in the experimental condition would develop a deeper understanding of the domain, and do better on transfer test problems that were described in the previous chapter. Both groups used the transfer enhancing methods of learning by exploration

and discovery (e.g., Miller, Lehman, and Koedinger 1999), situated cognition, and anchored instruction (Cognition and Technology Group at Vanderbilt 1993).

In the transfer test, we removed outliers and replaced missing values. We removed one outlier from the whole experiment, who was exceptionally fast in solving all tests, and used the same choice for all answers in the knowledge and motivation tests. One student of the experimental condition was absent, so we excluded this student from the analysis. After analysis with a box-plot, we excluded extreme values (two from the experimental condition, one from the alternate condition) from our data. These extremes occurred because some of the students faked their measurement data¹, or they did not record any data for unknown reasons (instructions on how to obtain the data were made explicit). As a result, we had 19 scores from the experimental condition and 21 from the alternate condition in our final analysis. We did not analyze flow-rate results (in ml/min) because almost all students produced their answer from their tables (took the value for one minute, interpolated, guessed, etc.) instead of reading it from their graphs, which was not the way to obtain the answer² that we had in mind when we designed the test.

We employed a normal linear model to test between subject effects of the total transfer score (without covariates). A Shapiro-Wilk test (Table 21) indicated a deviation from normality for the experimental group's total score, but the histogram (Figure 57) did show a normal distribution with the outliers included. We did not think this warranted dropping the normal linear model, which is robust against deviations from normality. To be sure we cross validated the results with a non-parametric Mann-Whitney U Test. Other data that helped to assert the statistical model's assumptions are included in Appendix E.

¹ A few students found it more convenient to interpolate or extrapolate measures often beyond the range of possible values (e.g. 100 ml) instead of performing the measures. Faked values that could be identified as such did not count in the table task, and led to wrong lines in the graph task, which led to low-score outliers.

² Only in about four cases, the value could have come from the graph. In other cases, reading the graph at 1 minute did not yield the result that students gave as answer.

We found the difference between total transfer test scores of the groups to be significant at the 0.05 level: $F(1,39) = 4.507$, $P = 0.04$ and power = 0.543 at $\alpha = 0.05$. Thus, we rejected our null hypothesis H_{T0} , and concluded that learning by teaching agents did result in better transfer skills, which indirectly implies that this helped students gain a deeper understanding of concepts¹. The non-parametric Mann-Whiney test confirmed the results ($P = 0.04$).

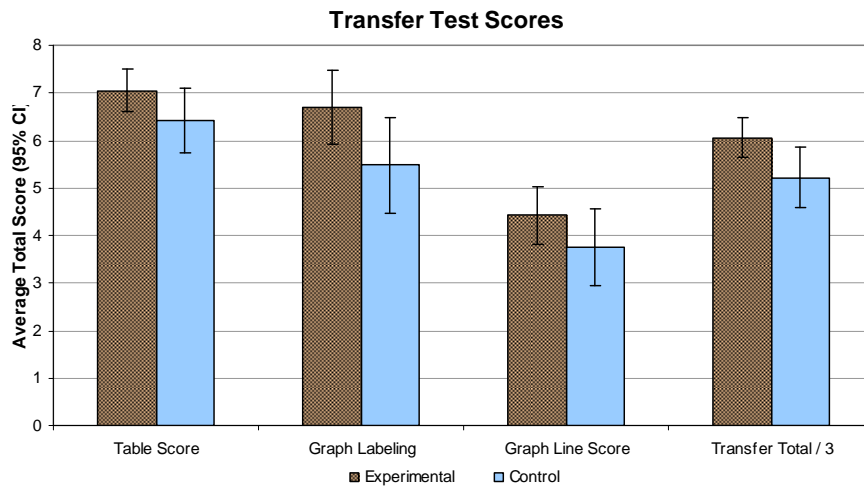


Figure 49. Transfer Test Scores by Task²

Figure 49 splits the transfer test scores by sub-task for each group. The sub-tasks were (1) creating a table, (2) labeling the graph, and (3) drawing a line. We could see advantages for the experimental condition in all three tasks. Students in the experimental condition performed slightly better in labeling the tables (not shown; included in the Table score), which is a task involving transfer from graphs to tables.

¹ Using time as covariate for transfer did not change the result. The P-value remained at 0.04.

² We divided total score (sum of all three tasks) by three to show the results in a single figure.

Table 12. Descriptive Statistics of the Transfer Task

Transfer Task	Condition	Mean	Std. Deviation	N
Table	EXP (X_1)	7.053	.9703	19
	ALT (X_2)	6.405	1.5939	21
	Total	6.713	1.3582	40
Graph Labels	EXP (X_1)	6.711	1.7185	19
	ALT (X_2)	5.476	2.3742	21
	Total	6.063	2.1549	40
Graph Line	EXP (X_1)	4.421	1.3464	19
	ALT (X_2)	3.762	1.8949	21
	Total	4.075	1.6701	40
Total	EXP (X_1)	18.184	2.8490	19
	ALT (X_2)	15.643	4.4557	21
	Total	16.850	3.9471	40

The descriptive statistics for the transfer test sub-tasks are summarized in Table 12. We observed the most pronounced effect for the experimental condition in labeling their transfer test graphs. This effect most likely resulted from repeatedly teaching the agent these ideas in cycle two and cycle three, and this may have made the importance of this task clear. To provide solutions with correct units the agent required units on the graph's axes to be correct. This shows that learning by teaching works well in helping students gain a good understanding of domain concepts, and at the same time fosters good learning behaviors among students.

Motivation Test Results

In this section, we present the analyses of the Motivated Strategies for Learning Questionnaire (MSLQ) and Experiences That Energize (ETE). Details of these measures were presented in the previous chapter.

Motivated Strategies for Learning Questionnaire (MSLQ)

To see if learning by teaching agents influenced motivation and learning strategies of students, we compared the Motivated Strategies for Learning Questionnaire results between the two experimental conditions. For the analysis, we used the normal linear model with MSLQ scores from the pretest as covariate (O_5 between conditions, with covariate O_2 , see Figure 33 in chapter V).

We removed a single influential outlier from the data. The student scored the highest pretest score and the lowest posttest score by selecting the same answer-choice during each test. A box-plot also identified this data point as outlier. Inclusion of this measurement would have corrupted our analysis, as it is not robust against measurement errors in the pretest. No other modifications were made.

Motivated Strategies for Learning Questionnaire Posttest

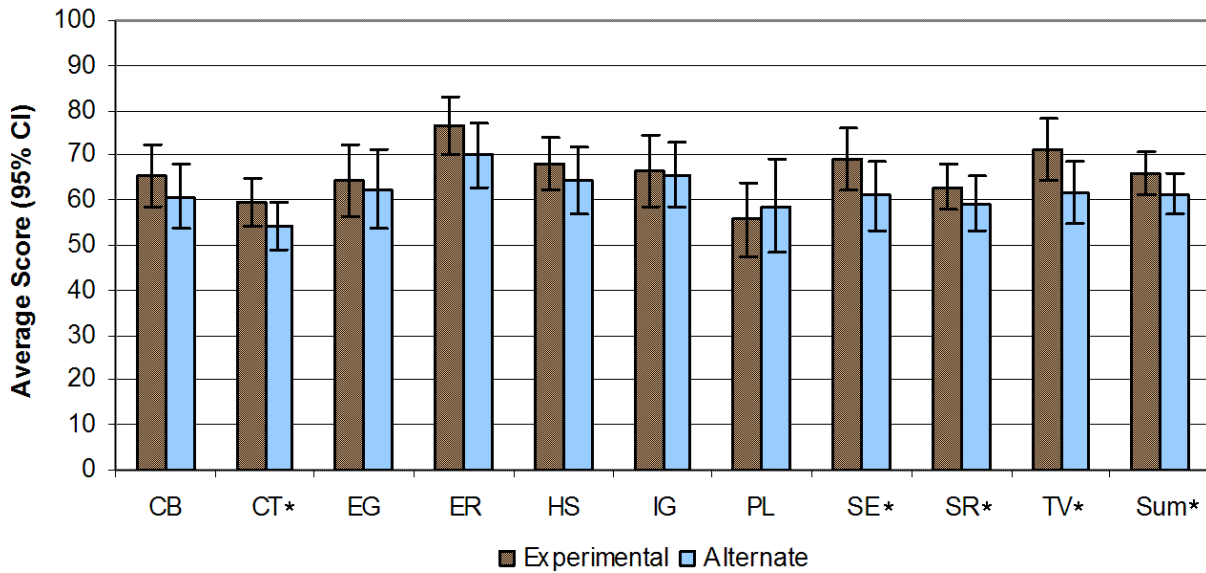


Figure 50. Motivated Strategies for Learning Questionnaire Posttest Comparison of Conditions¹

Figure 50 describes the MSLQ posttest results in a bar chart that shows 95% confidence intervals. Significant results are starred. The following abbreviations for MSLQ variables were used in the chart: control belief (CB), critical thinking (CT*), extrinsic goal motivation (EG), effort regulation (ER), help seeking (HS), intrinsic goal motivation (IG), peer learning (PL), self-efficacy (SE*), self-regulation (SR*), task value (TV*), and total score (Sum*). For example questions, see Table 3, Chapter V.

The total score for the MSLQ posttest was significant in favor of our experimental learning by teaching condition. We have analyzed this variable (O_5 in Figure 33) with a normal linear model with the MSLQ pretest (O_2 in Figure 33) as covariate. Our experimental condition scored on average of 249.3 points ($\sigma = 42.2$) and our alternate condition 231.7 ($\sigma = 40.2$). Tests of between subjects effects were significant at $F(1,43) = 7.75$ and $P = 0.008$ with an observed power of 0.78 at $\alpha = 0.05$.

¹ We have divided total score (sum of all variables) by the number of variables to show the results in a single figure.

When analyzing individual variables with the same method and their pretest counterparts as covariates we obtained significant individual results from an increased perceived *task value*, with $F(1,43) = 7.45$ and $P = 0.009$ with an observed power of 0.76 at $\alpha = 0.05$, increased perceived *self-regulation* $F(1,43) = 6.49$ and $P = 0.015$ with observed power of 0.7 at $\alpha = 0.05$. Additionally, *self-efficacy* was significantly increased with $F(1,43) = 4.416$ and $P = 0.042$ with an observed power of 0.537 at $\alpha = 0.05$, and perceived *critical thinking* improved with $F(1,43) = 4.354$ and $P = 0.043$ with an observed power of 0.53 at $\alpha = 0.05$.

Interestingly, working with a social agent seemed to impose an opposite trend on peer learning, as we can see in Figure 50 (PL). Although, in our experiment this effect is very small and not significant, its trend should be observed in future experiments. Future work could explore if a more advanced "peer-like" agent can reduce the desire of learners to work with human peers.

In an informal pretest to posttest comparison of the motivation means, MSLQ test scores fell slightly over time. However, we did not make a statistical comparison between pretest and posttest, due to slight differences in the measures. We observed this drop in motivation also in several preceding studies on the teachable agent Betty's Brain, when pretest and posttest had exactly the same questions. Probably this reflects high expectations and interest of the students when they begin the experiment, but they are only partially met.

Experiences That Energize (ETE)

To trace students' motivation when using our system, we asked them to rate their energy levels with ETES. Students gave their ratings before teaching, before quizzing, after quizzing (QD), after taking the test (TD), after reviewing the test (TR), and after seeing the test summary (TS) on a seven-point scale (see Table 4 in chapter V). We repeated this for each cycle and graphed the re-

sult in Figure 51. Because students repeatedly entered the teach phase and the quiz phase of each cycle, sometimes on multiple days, we did not analyze these two ETE results.

From the remaining data, we removed one outlier from the experimental group, and replaced some missing values. The outlier scored exceptionally low (40 below average of 61 with $\sigma = 14.3$ and 15 below the next subject in the group) and was identified by a box-plot. We had missing values in one of the three cycles. Therefore, we replaced them by the average of the other two values of the same variable of the same subject. This form of replacement preserves an eventual bias of a subject in answering the questions, because many students preferred one or the other end of the seven point Likert range. This became evident by looking at the four introductory questions, which were formulated to produce extreme high or low answers (see Appendix B). Most missing values occurred in the third cycle: seven in the experimental condition, three in the alternate condition. Of the remaining four replacements, three occurred in the test summary of cycle one in the experimental condition, and one in the test summary of cycle two of the alternate condition.

First, we looked for general differences in the sum of all ETEs (QD+TD+TR+TS) between conditions. The scores were the average over all cycles. The experimental condition had a mean score of 63.0 ($\sigma = 11.6$), while the alternate condition had a mean of 54.3 ($\sigma = 18.7$). We used a normal linear model with the sum of the four initially given norming ETEs as covariate¹. Our analysis showed significant differences between conditions with $F(1,42) = 7.326$ and $P = 0.01$ with a power of 0.75 at $\alpha = 0.05$.

¹ The covariate was the sum of the ETE norming questions. When looking at the ETE results, it became apparent that students had different views about what is energizing and what is not. The norming questions were designed to produce a pattern of maximum, minimum, maximum, minimum Likert score. However, some students found copying a grammar lesson from the board moderately energizing, or did not use the most negative score at all. This bias of individual students was captured and removed by the covariate.

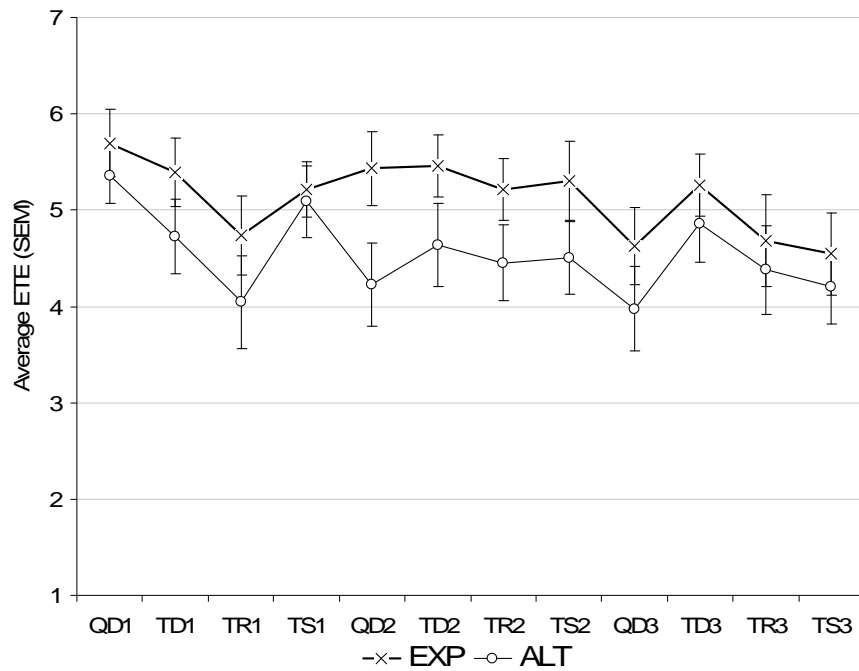


Figure 51. ETE Time Series for cycle 1, 2 and 3 (Standard Error Mean Bars)

In Figure 51, we see a time series of experiences that energize (ETEs) over all three cycles of our experiment. From the graph, we see that students in the experimental condition continuously ranked themselves higher than students in the alternate condition. Scores during cycle two are individually significant between groups at $\alpha = 0.05$, although we do not provide a detailed analysis.

In all three cycles, ETE scores fell during the test review (TR), which showed that students reacted adversely to mostly negative feedback for incorrect answers from the teacher agent. Remarkably, in the first two cycles scores improved after receiving summative feedback and before entering a new cycle (TS). We believe that this reflects the relief that students felt when they were done with the test combined with anticipation for the next cycle. Cycle number one was a

relatively easy task, cycle number two presented moderate challenging task appropriate for the grade level, and cycle number three was a very hard task for 6th grade students.

Our ETE scores show that students felt more energized during the task of learning by teaching in the quiz phase and test phase. We think that this can be attributed to Billy taking blame for being unintelligent or failing questions, which diverts the student's attention from their own performance.

Survey Results

We asked survey questions to get a general understanding of how much students liked interacting with our system, and to assess potential interferences with our study. Figure 52 shows the results of our survey with individually significant questions starred¹ when performing a comparison by group.

¹ Changes were not hypothesized. Significance testing was provided upon request. One in 20 of the results may be significant by chance.

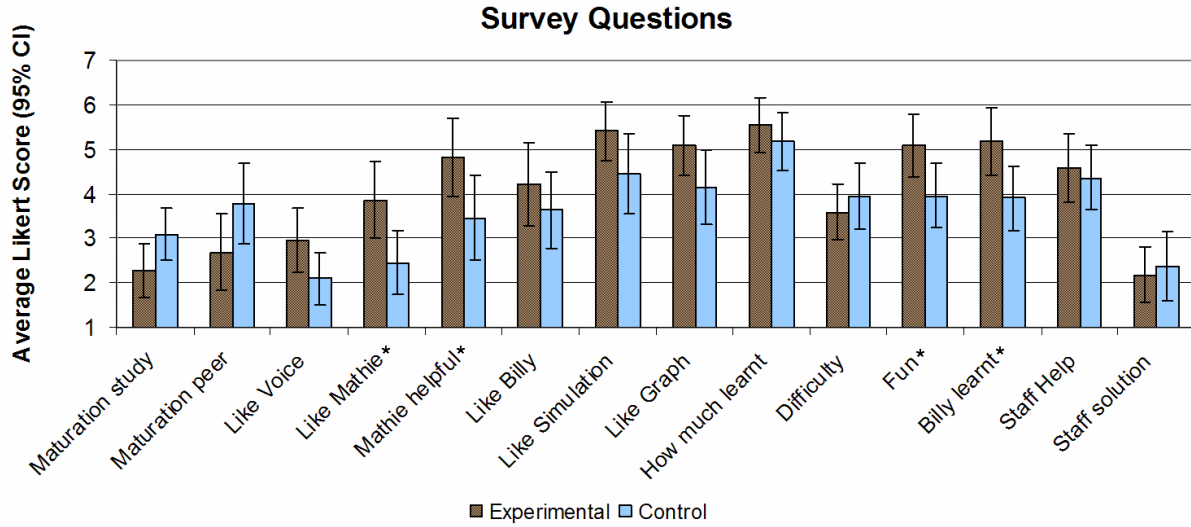


Figure 52. Survey Results

We determined study maturation by asking students how often they have looked at or asked peers about materials related to our study outside of our study. Interestingly, we found that the alternate condition used outside resources more often than the experimental condition. From this, we can claim that the better knowledge and transfer results for the experimental group cannot be attributed to maturation. An explanation for this result might be that students in the alternate condition felt more compelled to succeed in tests for themselves, while students who learn by teaching do not feel as bad if their teachable agents fail to answer questions. This issue should be addressed in further research.

When we asked students how they liked the voice in the system, we saw relatively low scores. We think the reason for this is that students reacted to problems in navigating through the dialogues, which let agent-speakers often repeat the same text. During the experiment, students were amused by the English accent of one speaker and expressed good understandability of the voices.

When we asked students how they liked specific agents and components of the system (Mathie, Billy, the simulation, and the graph), students in our experimental condition consistently reported higher scores than students in our alternate condition. It was interesting that students in our experimental group liked the teacher agent, Ms. Mathie, better ($F(1,43) = 6.11$, $P = 0.018$, power = 0.67) and found her to be more helpful ($F(1,43) = 4.28$, $P = 0.045$, power = 0.53) even though her dialogues were identical for both conditions. Additionally, students of the learning by teaching condition reported an on average higher score when asked how they liked the simulation and graph tool than the alternate condition.

When asked about how much students thought that they had learnt, both groups responded with high ratings that were almost equal. The difficulty of the system was rated average by both groups, but students in the learning by teaching condition reported that they had more fun. Students in the experimental condition rated significantly higher when asked how much fun they had ($F(1,43) = 4.9$, $P = 0.032$, power = 0.58). The question about how much students thought that Billy had learnt lacks a matching situation in the control condition (Billy could not learn), and cannot be analyzed for comparison.

Finally, we tried to assess interferences with our experiment. We found, that students were helped on average three times during the experiment, and the staff provided on average one solution of a problem during this time. However, because these interferences affected both groups equally we do not expect them to have influenced our results.

Summary

Table 13 presents a summary of our overall results. The summary compares students' performance, and motivation between two conditions. In the experimental condition, students learned

from the mentor agent, and had the goal to teach what they learned to the teachable agent so Billy could perform well on tasks that he had to perform in the company where he was employed as an intern. The problems that the teachable agent had to solve for the company were presented as quiz and test questions in the learning environment. In the alternate condition, students learned from the mentor agent to improve their own performance on the same tasks that the agent had to perform for the company. The learning environment in the alternate condition did not have Billy as a teachable agent. In the following paragraphs, we summarize the results of our study.

Table 13. Executive Summary of Study Results

Time	We concluded that our teachable agent condition required more time to complete all the material and teach the agent. We estimated that the experimental group spent up to 20% more time to complete each cycle.
H _{KGAIN}	We concluded that our system helped students learn domain material in both conditions.
H _{KDIFF}	We did not find a significant difference between the means for our two conditions on the knowledge test. Thus, we did not to reject our null hypothesis and concluded that learning by teaching did not lead to improved learning for the alternate condition.
H _T	Learning by teaching agents improved transfer.
Transfer Task Details	We observed the most pronounced difference in performance between the two groups in the task of labeling their transfer test graphs. This effect most likely could be attributed to the repeated teaching the agent this topic in cycle two and cycle three, which could have made the importance of this task clear.
H _M	The total score of the MSLQ posttest was significant in favor of our experimental learning by teaching condition. Learning by teaching a social agent was beneficial for motivating students.
MSLQ Details	Increased perceived <i>task value</i> , with P=0.009, increased perceived <i>self-regulation</i> P=0.015, <i>self-efficacy</i> was significantly increased with P=0.42 and perceived <i>critical thinking</i> improved P=0.43
H _{ETE}	Our analysis showed significant differences between conditions with P=0.01. Our ETES showed that students of the experimental group felt better than students of the other group, for example, more energized, during the task of learning by teaching in quiz phase and test phase. We attributed this to Billy taking the blame for failing questions in the test phase. This took some pressure of the students, who were not discouraged because of their own performance.
Survey	When we asked students how they liked specific components of the system, students in our experimental condition consistently reported higher scores than our alternate condition. Students in our experimental condition liked the teacher agent, Ms. Mathie, better and found her more helpful, even though the dialogues were identical in both conditions. Students of the learning by teaching condition have perceived simulation and graph tool better than the alternate condition. Students in the learning by teaching condition reported that they had more fun working with the system.

Our comparison of pretest versus posttest scores for the knowledge test determined that students *in both conditions* showed significant learning gains. However, a *comparison of our treatment groups* showed no significant learning differences. It turns out that the way the interactions between the students and the two systems were structured, students teaching our software agents spent more time in each cycle than the students in the alternate condition. This could be mainly attributed to teaching and correcting tasks that required extra time. Students had to navigate and answer many dialogues to teach the agents correctly. We would have liked the students in the experimental group to reflect more on the material they were teaching and gain deeper understand-

ing of domain material, but it is not clear that this was fully achieved. There was no marked improvement in the posttest results between groups; however, we did see significant differences on the near transfer task.

To study transfer, we made both groups solve a near transfer task. Students in the learning by teaching condition demonstrated significantly better performance over students in the alternate condition. This implies that like the other studies we have conducted (Biswas et al. 2005; Leelawong et al. 2002; Leelawong et al. 2003) the task of teaching others helps the students develop a better long-term understanding of the primary domain concepts. For example, students in our experimental condition performed better in the general skill of labeling the graph. This skill was fostered by teaching the teachable agent how to label the graph, and by the fact that wrong labeling had consequences on the agent's performance. Students in the experimental condition realized that an incompletely labeled or incorrectly labeled graph was ambiguous for Billy; therefore, he could not use the graph correctly, if he could use it at all.

When we looked at motivational differences between the two treatment conditions, we found that students in our experimental condition were significantly better motivated. The factors where the experimental condition showed higher motivation were task value, self-regulation, self-efficacy, and critical thinking. However, their urge to work with peers on the learning by teaching task showed an opposite trend. This could indicate that the teachable agent was accepted as a substitute for a peer by students. Students probably perceived increased task value because they experienced the fantasy context of helping the agent, Billy, with mathematics learning and problem solving. Increased self-efficacy suggested that students felt more capable of dealing with their own problems. The effects on self-regulation and critical thinking occurred probably due to many reflective tasks one had to undertake during teaching.

The pattern that students in our experimental condition had higher motivation was replicated in measures taken during the use of the learning system (ETEs), and in the survey that we conducted at the end of the study. We found that students in our experimental condition felt significantly more energized during the quiz phase and the test phase. In addition, our survey found that students in our experimental condition generally liked the components of our system better, and reported that they had more fun than the students in the other condition. This also confirmed positive motivational influences of learning by teaching.

Our interpretation is that an agent that demonstrated independent performance on evaluation and testing tasks seems to have taken some pressure of our students. Students felt that test results did not directly reflect their own performance. On the other hand, the feedback seems to have helped them in longer-term learning, as we can envision by their improved performance on the transfer task. Thus, working with a shared representation, where the other agent takes on some responsibilities (e.g., test taking) seems to have positive effects on motivation and learning.

CHAPTER VII

CONCLUSIONS

The overall goal in this thesis was to develop an intelligent learning environment that adopted an explanatory and constructivist learning approach, and helped students learn and improve their problem solving activities in the mathematical domain on distance-rate-time problems. The learning environment implemented for this thesis combines the STAR-Legacy cycle structure (Schwartz, Lin et al. 1999), variants of Smart Tools (Owens et al. 1995), and Adventure Player (Crews et al. 1997) into a single learning environment that implements a learning by teaching approach. Smart Tools provided the interactive representations that users create for problem solving based on their knowledge or understanding of domain concepts. Adventure Player provided a problem-solving environment with planning and simulation tools for developing and verifying solutions to complex problems. Adventure Player was linked to the Rescue at Boones Meadow episode of the Adventures of Jasper Woodbury series that adopted an anchored instruction approach for teaching and learning (Cognition and Technology Group at Vanderbilt 1997). The outer cycle structure into which all of the above tools were embedded was adapted from the STAR-Legacy system to provide a structured sequence of increasingly difficult problems. Each cycle in our learning environment draws problems from a global anchoring context, organizes learning activities, and arranges formative and summative feedback during learning. We extended this environment with software agents that could be taught, to pursue the goal to improve motivation and the ability to transfer beyond the set of problems that students were taught to solve.

Our implementation combined three loosely connected frameworks: a simulation framework, a Smart Tool (representation) framework, and social agents based on an agent framework. These frameworks are linked externally by our learning cycle structure, and internally by the agent communication infrastructure, which allows agents to manipulate the environment and transfer information to each other. This helps the agents to demonstrate, observe, learn, and comment on the actions of the student-teacher who is really the learner. To implement mechanisms by which the student could teach a responsive teachable agent, we added an internal communication system that let all agents observe changes that learners made in the environment, and engaged students in relevant dialogues that were linked to the state of the environment and the previous actions. Our agents were designed to learn declarative knowledge about the domain, and procedures on how to draw and read graphs. Once taught, agents could explain their solutions to the student when asked, and this helped the students to reflect on their incorrect solutions to determine how to teach the agents better.

We used this environment to study the effects of learning by teaching social agents on domain novice middle-school students by paying specific attention to motivation, learning, and transfer. Although we could not establish that our treatment resulted in improved performance on solving word problems, we found significant influences on transfer and highly significant effects on motivation. Students that learned by teaching our agents experienced higher task value, self-efficacy, self-regulation, and critical thinking, when we asked them to evaluate their experiences with the Motivated Strategies for Learning Questionnaire. These motivational benefits were confirmed by the way students rated themselves with Experiences That Energize, and by our survey.

Overall, the results of our study confirmed that teaching and interacting with social agents influenced middle school students positively. Students seemed to experience their interactions

with the teachable agent as genuine social interactions; this was borne out of the increased task value ratings that we observed in the experimental condition. Increased self-efficacy seems to indicate that students felt more capable of dealing with their mathematical problems if they were asked to teach, than if they were asked to learn from the teacher agent. Improved self-regulation and critical thinking were a direct consequence of the reflection initiated by the teaching tasks. This confirmed that learning by teaching is a powerful tool for providing motivating learning environments that kept the students interest in their learning and problem-solving task. The transfer test showed that learning by teaching also has advantages in helping students apply their learned knowledge in different tasks. This reflects most prominently in knowledge that was directly taught to the agent.

Although the benefits of learning by teaching were illustrated by this study, our approach increased the time that students had to spend on learning the same material than they would need in a more traditional setting (i.e., by being taught). This additional time makes this approach more costly and only worthwhile for teaching topics that students traditionally find hard to learn. This way, teachers and students could benefit from the motivation that the system generates among its users. In addition, students could use an online version of the system that allows them to solve homework outside of class.

Future work on this system's design should focus on improving the dialogue interaction with students. Like game environments, students might use buttons or menus to teach the agent or ask it queries without the tedious task of navigating a dialogue structure. However, our strategy of providing a uniform structure worked well after students became used to it. Initially, future systems should make better use of the functionality that is provided by the dialogue system, by creating more dialogues that the teachable agent initiates directly upon observing actions of the stu-

dent. Such dialogues are harder to design, but might enhance the user's experience of the agents dramatically. Alternatively, an agent could be taught procedural tasks not only through dialogues and demonstration, but also by graphical representations, such as, flow-charts. A visual representation of internal knowledge structures could help, too.

Another interesting topic is that students who learn by teaching often focus on the performance of their agent, and less on their own. Therefore, learning by teaching systems must let students also use the knowledge that is taught to the agent. Our system tried to balance and interleave using knowledge for problem solving and teaching it, but there may be better approaches to help improve learning gains. It is also possible that sequencing effects exist, and teaching followed by learning has different characteristics than learning followed by teaching an agent. In addition, letting students create tests and quiz questions for the agent may give this approach another boost.

In this sense, we hope that learning by teaching agents will become a useful addition to standard classroom instruction. The demonstrated benefits in transfer of learning and motivation make the approach a worthy target for implementation, commercialization, and for future research.

APPENDIX A

MOTIVATED STRATEGIES FOR LEARNING QUESTIONNAIRE

In the following appendix, we document our modified versions of the Motivated Strategies for Learning Questionnaire (MSLQ). The MSLQ was introduced by Pintrich et al. (Pintrich et al. 1993, 1993). All questions were answered online with an interface like in Figure 40.

Table 14. Modified Motivated Strategies for Learning Questionnaire (MSLQ) Pretest

I1	These questions will help us both find out how you feel about learning mathematics. On the following pages is a series of statements about how you learn mathematics. Mark your answers by telling how much you are like each statement. It will take about 20 minutes. Select 1 to go on.
I2	Here is an example to help you get started. A statement says I like mathematics . Think about that statement, and select a number from 1 to 7 that is closest to how much you like mathematics. If you don't like mathematics at all, select choice 1. If you like mathematics a lot select 7. If you like math most of the time, choose 6 or 5. If you like mathematics some of the time choose 4. If you do not like math most of the time, choose 3 or 2. Now, try it out below.
I3	There are no "right" or "wrong" answers. The only correct answers are those that are true for you. Whenever possible, let the things that have happened to you help you make a choice. Choose an answer because that is what you actually do or feel, not because it is what you should do or should feel . Select 1 to go on.
MSLQ1	In mathematics class, I prefer to learn about things that really challenge me so I can learn new ideas.
MSLQ2	If I study well, then I will be able to learn the ideas in mathematics class.
MSLQ4	I think that I will be able to use what I learn in this mathematics class in life or other classes. I believe that I will get an excellent grade in this mathematics class.
MSLQ5	I believe that I will get an excellent grade in this mathematics class.
MSLQ6	I am certain that I can understand the most difficult ideas presented in the readings in this mathematics class.
MSLQ7	Getting a good grade in mathematics class is the most important thing for me right now.

MSLQ9	It is my own fault if I do not learn the ideas in mathematics class.
MSLQ10	It is important for me to learn the ideas in mathematics.
MSLQ11	The most important thing for me right now is improving all of my grades, so my main concern in mathematics class is getting a good grade.
MSLQ12	I am confident that I can learn the basic ideas in mathematics class.
MSLQ13	If I can, I want to get better grades in mathematics class than most of the other students.
MSLQ15	I am confident that I can understand the most complex ideas presented by my mathematics teacher.
MSLQ16	In mathematics class, I prefer to study things I am curious about, even if they are difficult to learn.
MSLQ17	I am very interested in mathematics.
MSLQ18	If I try hard enough, I will understand the ideas in this mathematics class.
MSLQ20	I am confident that I can do an excellent job on the assignments and tests in mathematics class.
MSLQ21	I expect to do well in mathematics class.
MSLQ22	The most important thing for me in mathematics class is trying to understand the ideas as good as possible.
MSLQ23	I think what I study in mathematics class is useful for me to learn.
MSLQ24	When I have a choice about assignments and projects in mathematics class, I choose assignments and projects that I can learn from, even if that means it will be more difficult to get a good grade.
MSLQ25	If I do not understand the ideas in mathematics class, it is because I did not try hard enough.
MSLQ26	I like mathematics.
MSLQ27	Understanding the ideas in mathematics class is very important to me.
MSLQ29	I am certain I can master the skills being taught in mathematics class.
MSLQ30	I want to do well in mathematics class because it is important to show my ability to my family, friends, teachers, and others.
MSLQ31	When I consider the difficulty of this mathematics class, my mathematics teacher, and my skills, I think I will do well in mathematics class.
MSLQ33	During mathematics class, I often miss important information because I am thinking of other things.
MSLQ34	When studying for my mathematics class, I often try to explain the ideas to someone else.
MSLQ36	When reading for my mathematics class, I make up questions to help focus my reading.
MSLQ37	I often feel so lazy or bored when I study for mathematics class that I quit before I finish what I planned to do.
MSLQ38	I often find myself questioning things I hear or read in mathematics class in order to decide if I believe them.
MSLQ40	Even if I have trouble learning the ideas in mathematics class, I try to do the work on my own, without help from anyone.
MSLQ41	When I become confused about something I am reading for mathematics class, I go back and try to figure it out.
MSLQ44	If the ideas in my mathematics book seem difficult to understand, I change the

	way I read my mathematics book.
MSLQ45	I try to work with someone else to complete the assignments in mathematics class whenever possible.
MSLQ47	When a theory or conclusion is presented in mathematics class or in the mathematics book, I try to decide if there is good supporting evidence.
MSLQ48	I work hard to do well in mathematics class, even if I do not like what we are doing.
MSLQ50	When I study for my mathematics class, I like to discuss the ideas with someone else to make sure I understand.
MSLQ51	I think of the ideas in mathematics class as a starting point for developing my own ideas about mathematics.
MSLQ54	Before I read a new section in my mathematics book, I often skim it to see how it is organized.
MSLQ55	I ask myself questions to make sure I understand what I have been studying in mathematics class.
MSLQ56	If I need to, I change the way I study in order to get a better grade on tests and assignments in mathematics class.
MSLQ57	I often find that I have been reading for mathematics class, but I do not know what the reading was all about.
MSLQ58	When I do not understand something in my mathematics class, I ask my teacher to help me.
MSLQ60	When work in mathematics class is difficult, I give up or only study the easy parts.
MSLQ61	When I am studying for mathematics class, I try to think through a topic and decide what I am supposed to learn from it rather than just reading about it.
MSLQ66	I like to play around with ideas of my own that are related to what I am learning in mathematics class.
MSLQ68	When I do not understand the ideas in mathematics class, I ask someone to help me.
MSLQ71	Whenever I read or hear facts or conclusions in mathematics class, I think about if or why they are true.
MSLQ74	Even when the mathematics book is dull and uninteresting, I manage to keep reading until I finish the assignment.
MSLQ75	I try to find other students in mathematics class whom I can ask for help if I need it.
MSLQ76	When studying for mathematics class, I try to find out what ideas I do not understand well.
MSLQ78	When I study for mathematics class, I set goals for myself in order to get the most out of the time I spend studying.
MSLQ79	If I get confused while taking notes in mathematics class, I make sure I sort it out afterwards.
END	Congratulations. You are done with questions about your learning style. We will evaluate the questions and give the result to you at the end of the experiment.

Table 15. Modified Motivated Strategies for Learning Questionnaire (MSLQ) Posttest

I1	On the following pages are a series of statements about our math learning software . You have answered similar questions about mathematics class before you have started using our system. Imagine that you would use this learning software during mathematics class while learning graphs with your teacher.
I2	There are no "right" or "wrong" answers. The only correct answers are those that are true for you. Whenever possible, let the things that have happened to you during the experiment help you make a choice.
MSLQ1	With this learning software, I prefer to learn about things that really challenge me so I can learn new ideas.
MSLQ2	If I use this learning software right, then I will be able to learn ideas.
MSLQ4	I think that I will be able to use what I have learned with this learning software in life or other classes.
MSLQ5	I believe that I will perform excellent with this learning software.
MSLQ6	I am certain that I can understand the most difficult ideas presented by this learning software.
MSLQ7	Performing well with this learning software will help me make a good grade in mathematics class.
MSLQ9	It is my own fault if I do not learn ideas from this learning software.
MSLQ10	It is important for me to learn ideas from this learning software.
MSLQ11	The most important thing for me right now is improving all of my grades, and this learning software will help me in mathematics class.
MSLQ12	I am confident that I can learn the basic ideas in this learning software.
MSLQ13	If I can, I want to do better with this learning software than most of the other students.
MSLQ15	I am confident that I can understand the most complex ideas of Ms. Mathie.
MSLQ16	In this learning software, I prefer to try things I am curious about, even if they are difficult to learn.
MSLQ17	I am very interested in what I have learned from this learning software.
MSLQ18	If I try hard enough, I will understand the ideas in this learning software.
MSLQ20	I am confident that I can do an excellent job on the quizzes and tests in this learning software.
MSLQ21	I expect to learn well with this learning software.
MSLQ22	The most important thing for me in using this learning software is trying to understand the ideas as completely as possible.
MSLQ23	I think what I study with this learning software is useful for me to learn.
MSLQ24	When I have a choice about tasks in this learning software, I choose tasks that I can learn from, even if I do not get a grade for it.
MSLQ25	If I do not understand the ideas in this learning software, it is because I did not try hard enough.
MSLQ26	I like learning mathematics with this learning software.
MSLQ27	Understanding the ideas in this learning software is very important to me.
MSLQ29	I am certain I can master the skills being taught in this learning software.

MSLQ30	I want to do well with this learning software because it is important to show my ability to my family, friends, teachers, and others.
MSLQ31	When I consider the difficulty of this learning software, and my skills, I think I will do well in solving problems.
MSLQ33	During using this learning software, I often miss important information because I am thinking of other things.
MSLQ34	When learning with this learning software, I often would like to discuss the ideas with someone else.
MSLQ36	When learning with this learning software, I make up questions to help me understand.
MSLQ37	I often feel so lazy or bored when I learn from this learning software that I quit before I finish what I planned to do.
MSLQ38	I often find myself questioning things I hear or read while using this learning software in order to decide if I believe them.
MSLQ40	Even if I have trouble learning ideas from this learning software, I try to do the work on my own, without help from anyone.
MSLQ41	When I become confused about something I am reading in this learning software, I go back and try to figure it out.
MSLQ44	If the ideas in this learning software seem difficult to understand, I change the way I read them.
MSLQ45	I try to work with someone else to complete the assignments in the learning task whenever possible.
MSLQ47	When a result or solution is presented in this learning software, I try to decide if there is good supporting evidence.
MSLQ48	I work hard to do well with this learning software, even if I do not like what I am doing.
MSLQ50	When I use this learning software, I would like to set aside time to discuss the ideas with someone else to make sure I understand them.
MSLQ51	I think of the ideas in this learning software as a starting point for developing my own ideas about mathematics.
MSLQ54	With this learning software, if I read new resources, I often skim them to see how they are organized.
MSLQ55	I ask myself questions to make sure I understand what I have been studying while using this learning software.
MSLQ56	If I need to, I change the way I work in order to solve problems and assignments with this learning software.
MSLQ57	I often find that I have been looking at resources or listening to the teacher while using this learning software, but I do not know what it was all about.
MSLQ58	When I do not understand something while working with this learning software, I ask the teacher or research staff to help me.
MSLQ60	When working with this learning software seems difficult, I give up or only do the easy parts.
MSLQ61	When I am working on this learning software, I try to think through a topic and decide what I am supposed to learn from it rather than just reading about it.
MSLQ66	I like to play around with ideas of my own that are related to what I am learning from this learning software.

MSLQ68	When I do not understand the ideas in this learning software, I ask someone to help me.
MSLQ71	Whenever I read or hear facts or conclusions while using this learning software, I think about if or why they are true.
MSLQ74	Even when the some tasks in this learning software are dull and uninteresting, I manage to keep going until I finish the assignment.
MSLQ75	I would like to find other students working with this learning software whom I can ask for help if I need it.
MSLQ76	When working with this learning software, I try to find out what ideas I do not understand well.
MSLQ78	When working with this learning software, I set goals for myself in order to get the most out of the time I spend using the system.
MSLQ79	If I get confused with this learning software, I make sure I sort it out afterwards.
END	Thank you for answering all questions for us, we appreciate your help. Now, please ask the research staff for what you can do next.

APPENDIX B

EXPERIENCES THAT ENERGIZE

This appendix lists all Experiences that Energize (ETE) used during the experiment. Table 16 lists the introduction to ETEs with four norming questions. The ETEs used in our evaluation have been discussed on page 137.

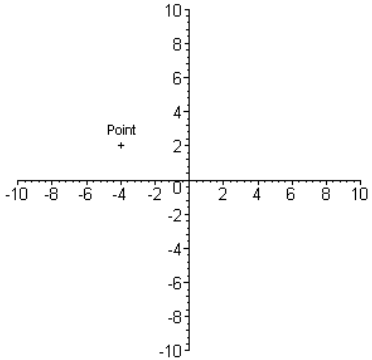
Table 16. Experiences that Energize

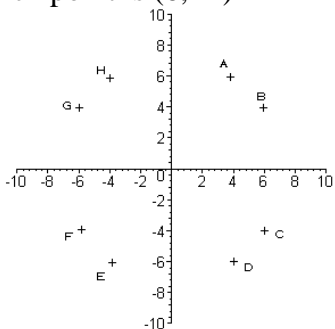
Introduction	Have you ever noticed that some things energize you, and other things seem to drain your thinking energy? For example, some people say they have a great deal of energy, when they think they are about to discover something new. But, their energy drops way down, when they have to do something that is boring, or they have no idea what to do.
Introduction	Next there are four example situations that I would like you to read. If you think the situation would energize you and make you ready to think more, circle a rating on the high side. If you think the situation would drain your energy and make you not want to think any more, circle a rating on the low side. If you think the situation is somewhere in between give it a middle score.
Example 1	You are about to figure out the answer to a problem that no one else had been able to solve.
Example 2	You are copying a grammar lesson from the board.
Example 3	You are finishing an important exam where you're confident you are getting every answer right.
Example 4	You are writing a paper that you do not think is very good.
End	When you use the system, every now and then a window will pop up and ask you similar questions. Please rate your energy similar to how you did it here. Thank you.

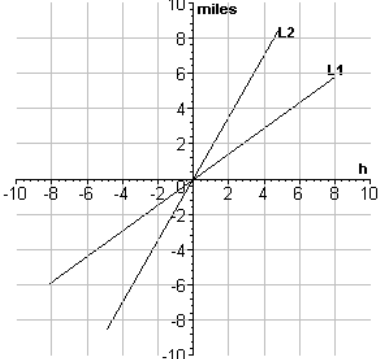
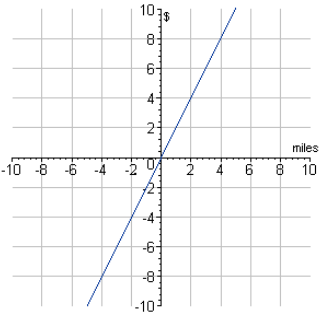
APPENDIX C

KNOWLEDGE TESTS

Table 17. Knowledge Pretest

I1	<p>On the following pages you will answer math-questions No teacher will see your answers or results. There are no grades for these questions. These questions are only for our information so that we can tell if you have learned something when using our system. You will possibly not be able to answer more than one third of the questions. There is nothing wrong with that, because you will learn most answers to these questions later, when you use the program. Do not be scared to say: "I have to learn that." if you do not know the solution.</p>
I2	<p>If you do not understand any instruction, raise your hand now. Otherwise start your questions ...</p>
KN14	<p>In a graph the x-axis is always ... (if it is not marked differently)</p> <ul style="list-style-type: none"> • Horizontal (from left to right) • Vertical (from top to bottom) • I have to learn that, I do not know an answer.
KN1	<p>Read the point from the graph.</p> <div style="text-align: center;">  </div> <ul style="list-style-type: none"> • (2,4) • (4,2) • (2,-4) • (-2,4) • (4,-2) • (-4,2) • (-2,-4)

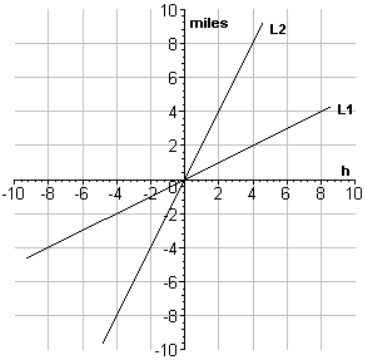
	<ul style="list-style-type: none"> • $(-4, -2)$
KN2	<p>Which point is $(6, -4)$</p> 
KN15	<p>If a car drives 180 kilometers in 3 hours what is its speed?</p> <ul style="list-style-type: none"> • 60 kilometers/hour • 183 kilometers/hour • 540 kilometers/hour • 60 hours/kilometer • 183 hours/kilometer • 540 hours/kilometer • I have to learn that, I do not know an answer.
KN12	<p>You have a stopwatch. Cars are passing by at exactly 55 miles/hour. You use the stopwatch to find how long the cars take for different distances. Then, you plot distance-time points in a graph. What would the graph look like?</p> <ul style="list-style-type: none"> • All points are scattered over the graph. • All points are on the same spot. • When the points are connected they lie on a straight line • When the points are connected they lie on a curve. • I have to learn that, I do not know an answer.
KN13	<p>If we make a distance-time graph of a car driving at 55 miles/hour, the slope of a line in a graph shows the car's ...</p> <ul style="list-style-type: none"> • Speed • Distance • Time • Acceleration • Steepness • I have to learn that, I do not know an answer.
KN3	<p>Which line is a faster speed L1 or L2?</p>

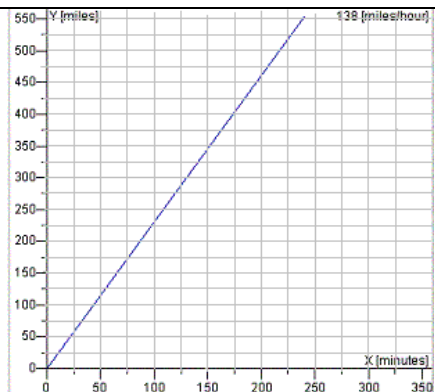
	 <ul style="list-style-type: none"> • L1 • L2 • It is not possible to know from the graph. • The lines do not tell anything about speed. • I have to learn that, I do not know an answer.
KN4	<p>Find the slope of the line in the graph.</p>  <ul style="list-style-type: none"> • -2 \$/mile • 1/2 \$/mile • 2 \$/mile • -1/2 \$/mile • -2 miles/\$ • 1/2 miles/\$ • 2 miles/\$ • -1/2 miles/\$
KN7	<p>What is the slope of the equation (function): $y = 7 - 5 * x$</p> <ul style="list-style-type: none"> • 0 • -7 • 7 • -5 • 5 • The equation (function) does not show the slope. • I have to learn that, I do not know an answer.
KN8	<p>P and Q are two points on a line. How do you calculate the slope of a line? There are multiple right answers. Choose the one that you know best.</p> <ul style="list-style-type: none"> • slope = x / y • slope = y / x

	<ul style="list-style-type: none"> • slope = run/rise = right/up • slope = rise/run = up/right • slope = $(Q_x - P_x) / (Q_y - P_y)$ • slope = $(Q_y - P_y) / (Q_x - P_x)$ • I have to learn that, I do not know an answer.
KN10 (removed) ¹	<p>Mike makes a trip to his aunt in Atlanta. He reads that the trip is 250 miles from his odometer when he has arrived. He has been driving for 5 hours, but made a 45 minutes stop for lunch. In between, he drove at various speeds between 45 and 70 miles/hour depending on speed limits. Can you calculate the average speed?</p> <ul style="list-style-type: none"> • 45 miles/hour • 50 miles/hour • 55 miles/hour • 60 miles/hour • 65 miles/hour • 70 miles/hour • To calculate average speed I need exact speed information. • I have to learn that, I do not know an answer.
KN9 (removed)	<p>How can you calculate average speed (AS) of a trip? There are multiple right answers. Choose the one that you know best.</p> <ul style="list-style-type: none"> • $AS = TotalTime / TotalDistance$ • $AS = TotalDistance / TotalTime$ • $AS = (ArrivalTime - StartTime) / (ArrivalDistance - StartDistance)$ • $AS = (ArrivalDistance - StartDistance) / (ArrivalTime - StartTime)$ • I have to learn that, I do not know an answer.
KN11	<p>Convert the speed 72 kilometers / hour to a slope in the unit meters / second. A kilometer has 1000 meters and an hour has 3600 seconds. The result is ...</p> <ul style="list-style-type: none"> • $72 / 3600 = 2 / 100 = 1/50$ meters/second • $3600 / 72 = 100 / 2 = 50$ meters/second • $72 * 1000 / 3600 = 20$ meters/second • $3600 / (72 * 1000) = 1 / 20$ meters/second • $3600 / 1000 = 3.6$ meters/second • $3.6 / 1000 = 0.0036$ meters/second • I have to learn that, I do not know an answer.
END	<p>Congratulations. You are done with the math questions. Please do not feel bad if you could not answer a lot.</p>

¹ Cycle 4 was removed, so the pretest questions relating to cycle four were removed.

Table 18. Knowledge Posttest

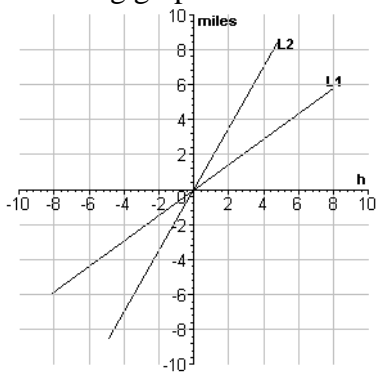
I1	<p>If you have finished all three cycles early and would like to review a previous cycle please raise your hand and let somebody from the research staff know. You can also redo any "Real Customers" if you wish.</p>
I2	<p>On the following pages you will answer math questions.</p> <p>No teacher will see your answers or results.</p> <p>These questions are only for our information so that we can tell if you have learned something when using our system.</p> <p>Please give every question a fair try.</p>
KN_Z	<p>If you draw the ordered pair $P = (-4, -8)$ into the graph, it is ...</p>  <ul style="list-style-type: none"> • On the faster line L1. • On the slower line L1. • On the faster line L2. • On the slower line L2. • On the origin. • Not on any line.
KN_N	<p>In a graph the Y-axis is always ... (if it is not marked differently)</p> <ul style="list-style-type: none"> • Horizontal (from left to right) • Vertical (from top to bottom)
KN_U (extra)	<p>My son is leaving Atlanta in the Cessna 172 heading to Nashville. That is 225 miles away. They fly 138 miles/hour. How many minutes from now will I have to pick him up? Use the graph tool from cycle one:</p>



- 575 minutes
- 225 minutes
- 172 minutes
- 138 minutes
- 100 minutes

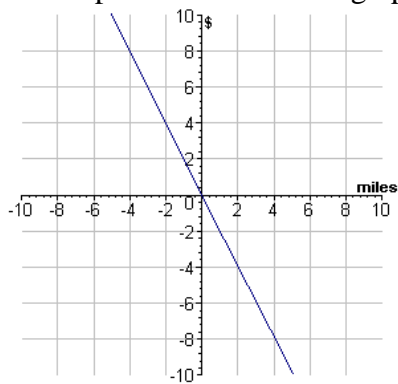
KN_Y

The following graph shows two lines. Which line is a faster speed?



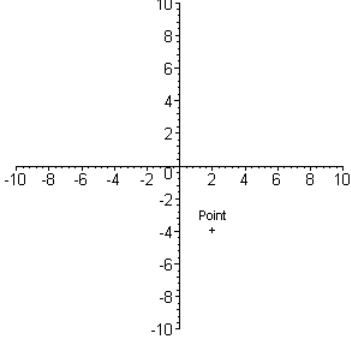
KN_D

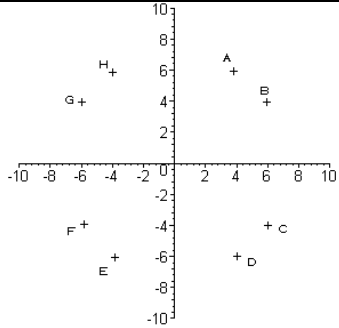
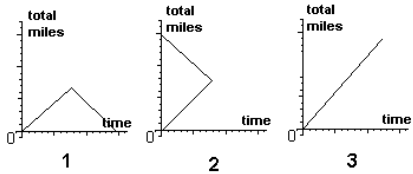
Find the slope of the line in the graph.

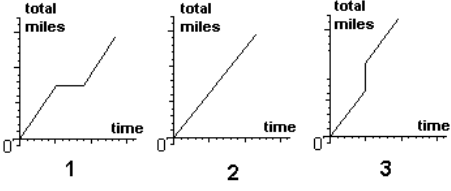


- -2 \$/mile
- 1/2 \$/mile
- 2 \$/mile
- -1/2 \$/mile
- -2 miles/\$
- 1/2 miles/\$
- 2 miles/\$

	<ul style="list-style-type: none"> -1/2 miles/\$
KN_H	<p>P and Q are two points on a line. How do you calculate the slope of a line? There are multiple right answers. Choose the one that you know best. Q_x is the x-coordinate of Q and P_y is the y-coordinate of point P, and so on.</p> <ul style="list-style-type: none"> slope = $x / y = 3/2$ slope = $y / x = 2/3$ slope = run/rise = right/up = $3/2$ slope = rise/run = up/right = $2/3$ slope = $(P_x - Q_x) / (P_y - Q_y) = (3-6)/(2-4) = -3/-2 = 3/2$ slope = $(P_y - Q_y) / (P_x - Q_x) = (2-4)/(3-6) = -2/-3 = 2/3$ slope = $(Q_x - P_x) / (Q_y - P_y) = (6-3)/(4-2) = 3/2$ slope = $(Q_y - P_y) / (Q_x - P_x) = (4-2)/(6-3) = 2/3$
KN_L	<p>You have a stopwatch. Cars are passing by at exactly 55 miles/hour. You use the stopwatch to find how long the cars take for different distances. Then, you plot distance-time points in a graph. What would the graph look like?</p> <ol style="list-style-type: none"> All points are scattered over the graph. All points are on the same spot. When the points are connected they lie on a straight line When the points are connected they lie on a curve.
KN_A	Read the point from the graph.

	 <ul style="list-style-type: none"> • (2,4) • (4,2) • (2,-4) • (-2,4) • (4,-2) • (-4,2) • (-2,-4) • (-4,-2)
KN_O	<p>If we make a distance-time graph of a car driving at 70 miles/hour, the line in a graph shows the car's ...</p> <ul style="list-style-type: none"> • Distance • Time • Acceleration • Speed • Steepness
KN_M	<p>If a car takes 3 hours for 180 kilometers what is its speed?</p> <ul style="list-style-type: none"> • 60 kilometers/hour • 183 kilometers/hour • 540 kilometers/hour • 60 hours/kilometer • 183 hours/kilometer • 540 hours/kilometer
KN_G	<p>What is the slope of the function: $y = - 5 * x + 7$</p> <ul style="list-style-type: none"> • 0 • -7 • 7 • -5 • 5 • The function does not show the slope. • I have to learn that, I do not know an answer.
KN_B	<p>Which point is (-4, 6)</p>

	
KN_K	<p>Convert the speed 72 kilometers / hour to a slope in the unit meters / second. A kilometer has 1000 meters and an hour has 3600 seconds. The result is ...</p> <ul style="list-style-type: none"> • $72 / 3600 = 2 / 100 = 1/50$ meters/second • $3600 / 72 = 100 / 2 = 50$ meters/second • $72 * 1000 / 3600 = 20$ meters/second • $3600 / (72 * 1000) = 1 / 20$ meters/second • $3600 / 1000 = 3.6$ meters/second • $3.6 / 1000 = 0.0036$ meters/second
KN_X (extra)	<p>What are coordinates?</p> <ul style="list-style-type: none"> • Some numbers to find a location. • The fewest numbers needed to find a location. • Exactly two numbers: x and y.
KN_W (extra)	<p>A car drives at 45 miles/hour to another city 50 miles away. It then turns around and drives back without stopping. Which graph shows this best?</p> <div data-bbox="326 1136 735 1308" style="text-align: center;">  </div> <ol style="list-style-type: none"> 1. The distance increases and goes back to zero. 2. The time increases and then goes back to zero. 3. It is a straight line. We cannot see if a car turns around in a graph.
KN_V (extra)	<p>A car drives at 45 miles/hour to another city 50 miles away. Half way it makes a stop at a gas-station for 10 minutes. Which graph shows this best?</p>

	 <p>1 2 3</p>
END	Thank you. You are done with the math questions. Please do not feel bad if you could not answer all of them.

APPENDIX D

TRANSFER TEST WORKSHEET

Date:	
Nozzle:	
Nickname:	

Flow Rate	
-----------	--

Notes:

APPENDIX E

NORMAL LINEAR MODEL ADDENDA

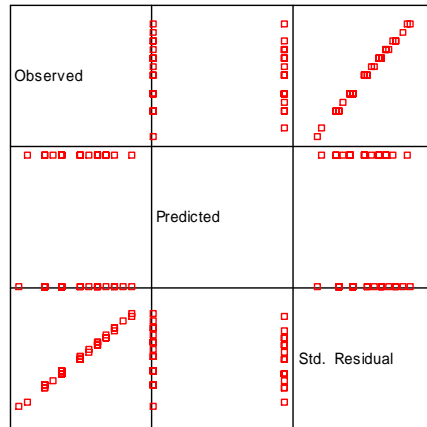
Knowledge Test (Analysis of H_{KNGAIN})

Table 19. Box's Test of Equality of Covariance Matrices¹ for Knowledge Tests

Box's M	.488
F	.154
df1	3
df2	357043.82
Sig.	.927

¹ Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.
Design: Intercept+GROUP Within Subjects Design: TIME

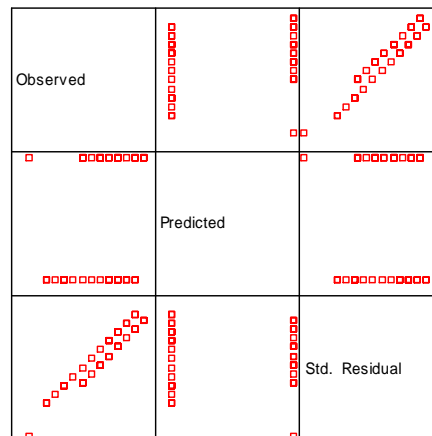
Dependent Variable: KN1RAW



Model: Intercept + GROUP

Figure 53. Residual Plot Knowledge Pretest

Dependent Variable: KN2RAW



Model: Intercept + GROUP

Figure 54. Residual Plot Knowledge Posttest

Knowledge Test (Analysis of H_{KNDIFF})

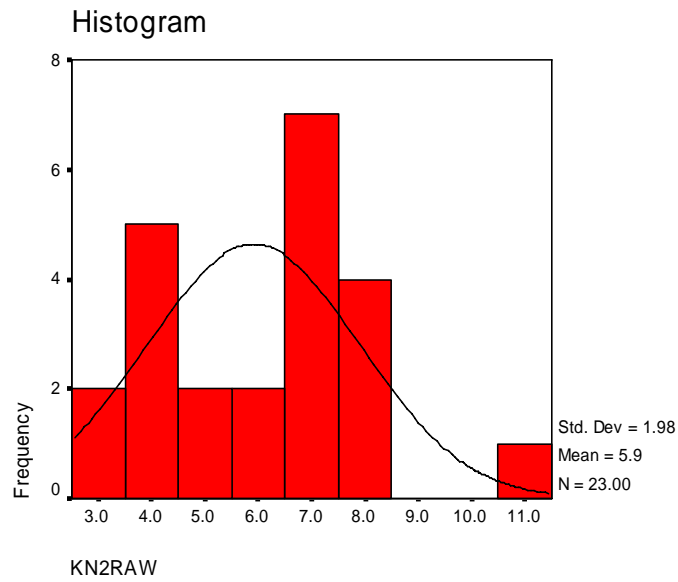


Figure 55. Histogram of the Knowledge Posttest of the Experimental Condition

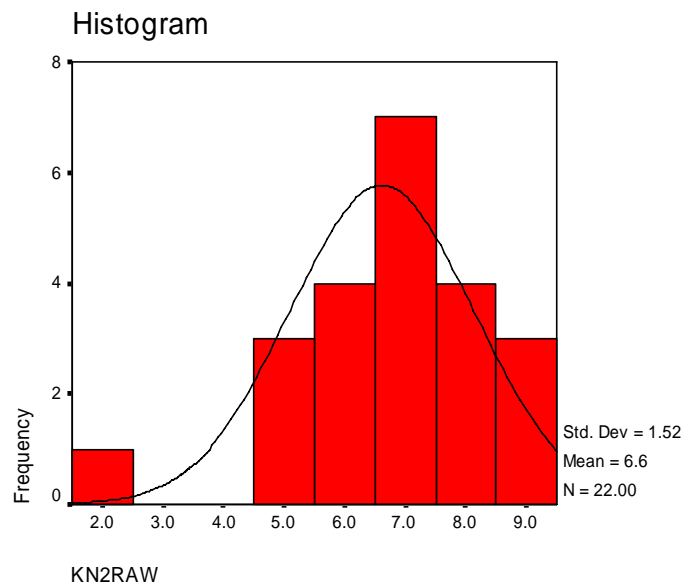


Figure 56. Histogram of the Knowledge Posttest of the Alternate Condition

Table 20. Normality Tests for H_{KNGAIN}

	Condition	Kolmogorov-Smirnov			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
KN1RAW	EXP	.196	23	.022	.949	23	.278
	ALT	.185	22	.048	.966	22	.611
KN2RAW	EXP	.139	23	.200	.935	23	.140
	ALT	.161	22	.141	.894	22	.023

Transfer Test

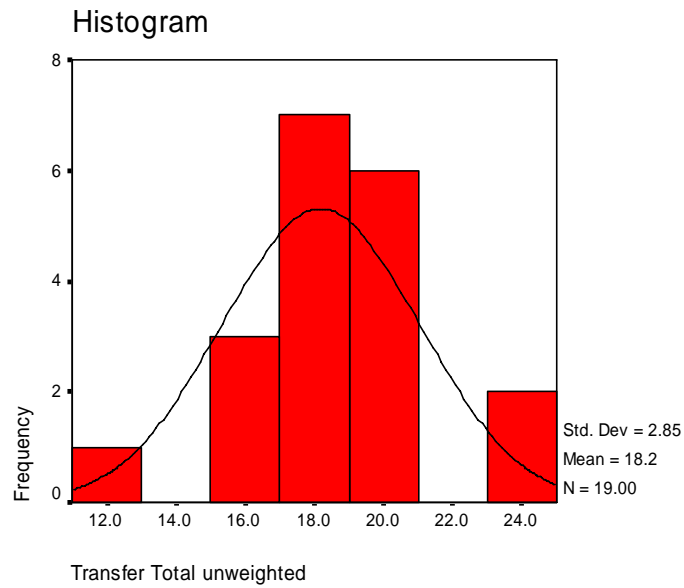


Figure 57. Histogram of the Transfer Test of the Experimental Condition

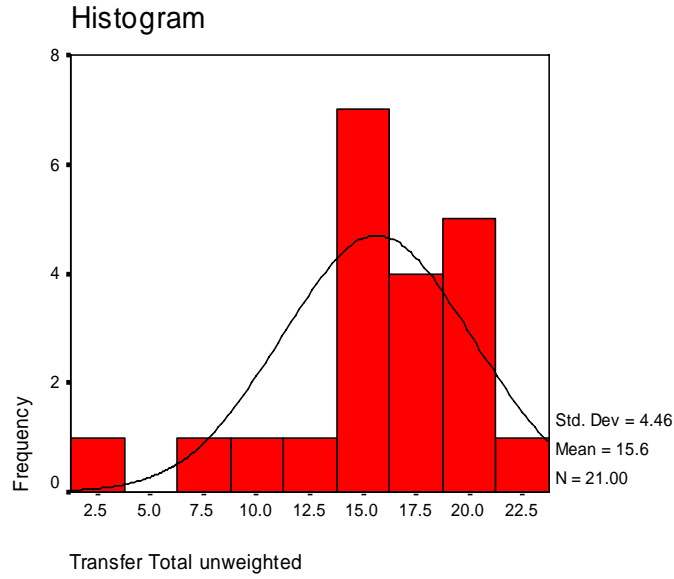


Figure 58. Histogram of the Transfer Test of the Alternate Condition

Table 21. Normality Tests for the Transfer Test Analysis

	Condition	Kolmogorov-Smirnov			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Transfer Table	EXP	.311	19	.000	.859	19	.010
	ALT	.265	21	.000	.777	21	.000
Transfer Graph Labels	EXP	.146	19	.200	.954	19	.466
	ALT	.118	21	.200	.957	21	.459
Transfer Graph Line	EXP	.195	19	.055	.873	19	.016
	ALT	.180	21	.074	.885	21	.018
Transfer Total	EXP	.128	19	.200	.952	19	.419
	ALT	.166	21	.136	.903	21	.039

Table 22. Box's Test of Equality of Covariance Matrices¹ for Transfer Test

Box's M	7.213
F	2.267
df1	3
df2	399135.67 7
Sig.	.079

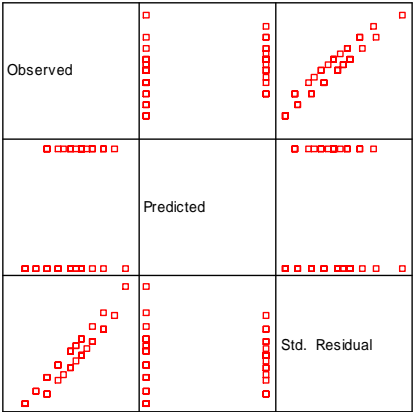
Table 23. Levene's Test of Equality of Error Variances² for Transfer Test

	F	df1	df2	Sig.
Transfer Graph Labels	1.967	1	38	.169
Transfer Total	1.957	1	38	.170

¹ Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups. Design: Intercept+GROUP Within Subjects Design: TIME

² Tests the null hypothesis that the error variance of the dependent variable is equal across groups. Design: Intercept+GROUP

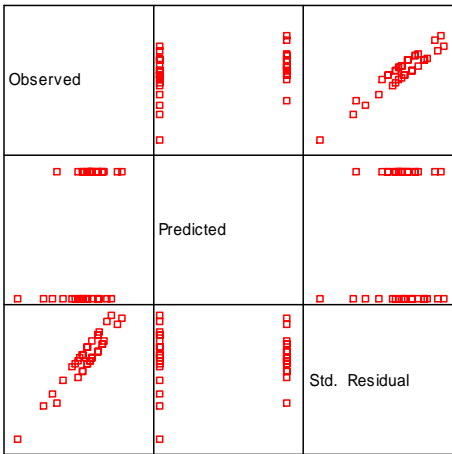
Dependent Variable: Transfer Graph Labels



Model: Intercept + GROUP

Figure 59. Residual Plot for Transfer Test (Graph Labels)

Dependent Variable: Transfer Total



Model: Intercept + GROUP

Figure 60. Residual Plot for Transfer Test

MSLQ

Histogram

For GROUP= EXP

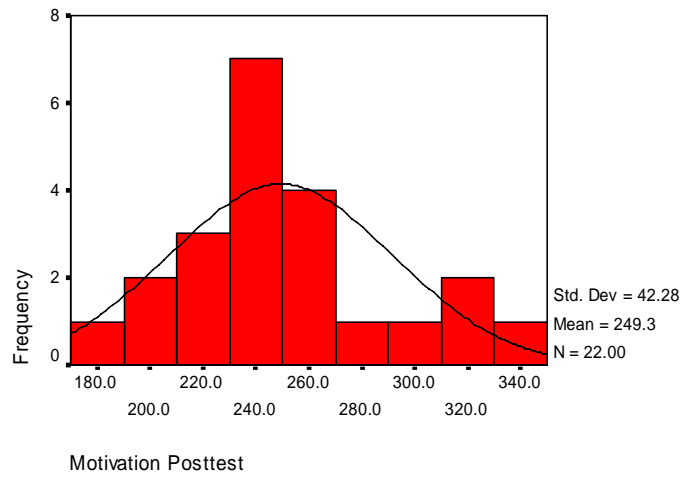


Figure 61. Histogram of the MSLQ of the Experimental Condition

Histogram

For GROUP= ALT

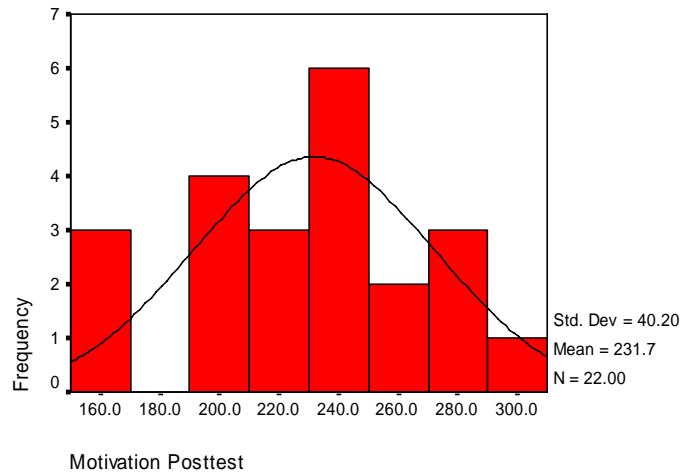


Figure 62. Histogram of the MSLQ of the Alternate Condition

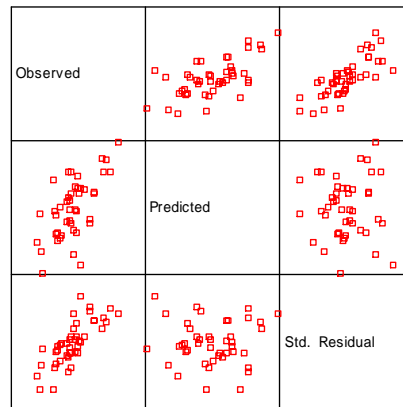
Table 24. Normality Tests for the MSLQ Analysis

	Condition	Kolmogorov-Smirnov			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Motivation Pre-test	EXP	.135	22	.200	.950	22	.317
	ALT	.094	22	.200	.975	22	.818
Motivation Posttest	EXP	.191	22	.035	.938	22	.178
	ALT	.086	22	.200	.974	22	.791

Table 25. Levene's Test of Equality of Error Variances¹

F	df1	df2	Sig.
.013	1	42	.910

Dependent Variable: Motivation Posttest



Model: Intercept + MSLQ1 + GROUP

Figure 63. Residual Plot Motivation Posttest

¹ Tests the null hypothesis that the error variance of the dependent variable is equal across groups. Dependent Variable: Motivation Posttest; Design: Intercept+MSLQ1+GROUP

BIBLIOGRAPHY

- Aimeur, Esma, H. Dufort, D. Leibu, and C. Frasson. 1997. Some Justifications for the Learning by Disturbing Strategy. Paper read at AI-ED 97, World Conference on Artificial Intelligence and Education, at Kobe, Japan.
- Aleven, Vincent A. W. M. M., and Kenneth R. Koedinger. 2002. An Effective Metacognitive Strategy: Learning by Doing and Explaining with a Computer-Based Cognitive Tutor. *Cognitive Science* 26:147–179.
- Allen, Vernon L., Ed., ed. 1976. *Children as Teachers: Theory and Research on Tutoring*. New York: Academic Press.
- Anderson, John R. 1983. *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.
- . 1995. Cognitive Tutors: Lessons Learned. *The Journal of the Learning Sciences* 4 (2):167-207.
- Bandura, A. 1977. *Social Learning Theory*. Englewood Cliffs, NJ: Prentice Hall.
- Bandura, A., and D. H. Schunk. 1981. Cultivating Competence, Self-Efficacy, and Intrinsic Interest through Proximal Self-Instruction. *Journal of Personality and Social Psychology* (41):586-598.
- Barron, B. J., D. Schwartz, Nancy Vye, A. Moore, L. Zech, J. Bransford, and Cognition and Technology Group at Vanderbilt. 1998. Doing with Understanding: Lessons from Research on Problem- and Project-Based Learning. *Journal of the Learning Sciences* 7:271-311.
- Biswas, Gautam, Thomas Katzlberger, John Bransford, and Daniel Schwartz. 2001. Extending Intelligent Learning Environments with Teachable Agents to Enhance Learning. In *Artificial Intelligence in Education*, edited by J. D. Moore, C. L. Redfield and W. L. Johnson. San Antonio, TX: IOS Press, Amsterdam, Netherlands. Available from <http://citeseer.ist.psu.edu/biswas01extending.html>.
- Biswas, Gautam, D Schwartz, J. Bransford, and TAG-V. 2001. Technology Support for Complex Problem Solving: From SAD Environments to AI. In *Smart Machines in Education*, edited by Forbus and Feltovich. Menlo Park, CA: AAAI Press.
- Biswas, Gautam, Daniel Schwartz, Krittaya Leelawong, Nancy J. Vye, and TAG-V. 2005. Learning by Teaching a New Agent Paradigm for Educational Software. *Applied Artificial Intelligence* 19:363-392.
- Boud, D., ed. 1985. *Problem-Based Learning for the Professions*. Sydney: HERDSA.
- Boud, D., and G. Feletti, eds. 1991. *The Challenge of Problem-Based Learning*. New York: St Martin's Press.
- Bransford, John D. 1979. *Human Cognition: Learning, Understanding, and Remembering*. Belmont, CA: Wadsworth.
- . 1990. Anchored Instruction: Why We Need It and How Technology Can Help. In *Cognition, education and multimedia*, edited by D. Nix and R. Sprio. Hillsdale, NJ: Lawrence Erlbaum Associates.
- . 2004. *Star Legacy Audiovisual Quicktime Presentation*. Vanderbilt University 1990 [cited October 8 2004]. Available from <http://iris.peabody.vanderbilt.edu/bransford.htm>.
- Bransford, John D., L. Ann Brown, and R. Rodney Cocking. 2000. *How People Learn*. Expanded ed. Washington, D.C.: National Academy Press.

- Brophy, Jere. 1998. *Motivating Students to Learn*. Boston: Mc Graw Hill.
- Brophy, Sean P. 1998. Learning Scientific Principles through Problem Solving in Computer Supported and Laboratory Environments. Ph. D. Thesis, Education, Vanderbilt University, Nashville.
- . 2003. Experiences that Energize. Personal Communication with the Author.
- Brown, A.L., and J.C. Campione. 1996. Psychological Theory and the Design of Innovative Learning Environments: On Procedures, Principles, and Systems. In *Innovations in learning: New environments for education*, edited by L. Schauble and R. Glaser. Mahwah, NJ: Erlbaum.
- Brown, J. S., and R. Burton. 1978. Diagnostic Models for Procedural Bugs in Basic Mathematical Skills. *Cognitive Science* 2:155-192.
- Brown, J. S., A. Collins, and S. Duguid. 1989. Situated Cognition and the Culture of Learning. *Educational Researcher* 18 (1):32-42.
- Brown, J. S., and K. VanLehn. 1980. Repair Theory: A Generative Theory of Bugs in Procedural Skills. *Cognitive Science* (4):379-426.
- Bruner, J. 1966. *Toward a Theory of Instruction*. Cambridge, MA: Harvard University Press.
- Bruner, J. S. 1961. The act of discovery. *Harvard Educational Review* 31:21-32.
- CarnegieLearning. 2004. *Corporate Websiete* 2004 [cited November 18, 2004]. Available from <http://www.carnegielearning.com>.
- Chan, Tak Wai, and Chih Yueh Chou. 1997. Exploring the Design of Computer Supports for Reciprocal Tutoring. *International Journal of Artificial Intelligence in Education* 8:1-29.
- Chi, M. T. H. 1997. Self-Explaining: The Dual Processes of Generating Inferences and Repairing Mental Models. In *Advances in Instructional Psychology*, edited by R. Glaser. Mahwah, NJ: Lawrence Erlbaum Associates.
- Chi, M. T. H., M. Bassok, M. W. Lewis, P. Reimann, and R. Glaser. 1989. Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems. *Cognitive Science* 13 (2):145–182.
- Cognition and Technology Group at Vanderbilt. 1993. Anchored Instruction and Situated Cognition Revisited. *Educational Technology* 33 (3):52-70.
- . 1997. *The Jasper Project: Lessons in Curriculum, Instruction, Assessment, and Professional Development*. Mahwah, New Jersey: Lawrence Erlbaum Associates.
- Cohen, A. 1990. *Language Learning*. New York: Harper Collins.
- Cohen, Jiska. 1986. Theoretical Considerations of Peer Tutoring. *Psychology in the Schools* 23 (2):175-186.
- Collins, A., J. S. Brown, and S. E. Newman. 1989. Cognitive Apprenticeship: Teaching the Craft of Reading, Writing and Mathematics. In *Knowing, Learning, and Instruction: Essays in Honor of Robert Glasser*, edited by L. B. Resnick. Hillsdale, NJ: Erlbaum.
- Cook, Thomas D., Donald T. Campbell, and Thomas H. Cook. 1979. *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Chicago: Rand McNally.
- Crews, Thaddeus R. 1995. Adventureplayer: A Microworld Anchored in a Macrocontext. PhD, Computer Science, Vanderbilt University, Nashville, TN.

- Crews, Thaddeus R., Gautam Biswas, Susan Goldman, and John Bransford. 1997. Anchored Interactive Learning Environments. *International Journal of AI in Education* 8.
- Csikszentmihalyi, Mihalyi. 1978. Intrinsic Rewards and Emergent Motivation. In *The Hidden Costs of Reward*, edited by M. R. Lepper and D. Greene. Hillsdale, NJ: Lawrence Erlbaum Associates.
- . 1990. *Flow*. New York: Harper & Row.
- Davis, Joan M., Kittaya Leelawong, Kadira Belyne, Robert Bodenheimer, Gautam Biswas, Nancy Vye, and John Bransford. 2003. Intelligent User Interface Design for Teachable Agent Systems. Paper read at International Conference on Intelligent User Interfaces, January 12-15, at Miami, Florida.
- Detterman, D. K. 1993. The Case for the Prosecution: Transfer as an Epiphenomenon. In *Transfer on Trial: Intelligence, Cognition, and Instruction*, edited by D. K. Detterman and R. J. Sternberg. Norwood, NJ: Ablex.
- Dewey, John. 1933. *How We Think*. Boston: D.C. Heath.
- . 1938. *Logic: The Theory of Inquiry*. New York: Holt: Rinehart and Winston.
- Doignon, J.-P., and J.-C. Falmagne. 1999. *Knowledge Spaces*: Springer-Verlag.
- Doignon, Jean-Paul, and Jean-Claude Falmagne. 1985. Spaces for the Assessment of Knowledge. *International Journal of Man-Machine Studies* (23):175-196.
- . 2004. *Assessment and LEarning in Knowledge Spaces (ALEKS) 2003* [cited October 28, 2004]. Available from http://www.aleks.com/about/Science_Behind_ALEKS.pdf.
- Elby, Andrew. 2000. What Students' Learning of Representations Tells us about Constructivism. *Journal of Mathematical Behavior* 19 (4):481-502.
- Etzioni, O., and D. S. Weld. 1995. Intelligent Agents on the Internet: Fact, Fiction, and Forecast. *IEEE Expert* 10 (4):44-49.
- Finin, Tim, Yannis Labrou, and James Mayfield. 1997. KQML as an agent communication language. In *Software Agents*, edited by J. Bradshaw. Cambridge: MIT Press.
- Flavell, J. H. 1979. Metacognition and Cognitive Monitoring: A new Area of Cognitive-Developmental Inquiry. *American Psychologist* 34:906-911.
- Flavell, J. H., P. H. Miller, and S. A. Miller. 1993. *Cognitive Development*. Englewood Cliffs, NJ: Prentice Hall.
- Forbus, Kenneth D., and Peter B. Whalley. 1994. Using Qualitative Physics to Build Articulate Software for Thermodynamics Education.
- Franklin, S., and A. Graesser. 1996. Is It an Agent or Just a Program? A Taxonomy for Autonomous Agents. Paper read at Third International Workshop on Agent Theories, Architectures, and Languages, at New York.
- Frasson, Claude, Thierry Mengelle, Esma Aimeur, and Guy Gouardères. 1996. An Actor-based Architecture for Intelligent Tutoring Systems. In *ITS'96 Conference, Lecture Notes in Computer Science*. Montréal: Springer Verlag.
- Gaustard, Joan. 1993. Peer and Cross-Age Tutoring. *ERIC Digest* 79.
- Glaser, R. 1992. Expert Knowledge and Processes of Thinking. In *Thinking Skills in Sciences and Mathematics*, edited by D. F. Halpern. NJ: Lawrence Erlbaum Associates.

- Gust, Helmar, Christoph Peylo, Claus Rollinger, and Wilfried Teiken. 2005. *The Virtual Campus Prolog Learning Environment*. AIED'99 Poster 1999 [cited September 12, 2005]. Available from <http://citeseer.ist.psu.edu/239290.html>.
- Haskell, Robert E. 2001. *Transfer of Learning*. London, UK: Academic Press.
- Hietala, Pentti, and Timo Niemirepo. 1998. The Competence of Learning Companion Agents. *International Journal of Artificial Intelligence in Education* 9:178-192.
- IRIS. 2004. *Star Legacy Modules*. Vanderbilt University 1999 [cited October 8, 2004]. Available from <http://iris.peabody.vanderbilt.edu/slm.html>.
- Johnson, W. Lewis, and Jeff W. Rickel. 2000. Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. *International Journal of Artificial Intelligence in Education* 11:47-78.
- Judd, C. H. 1908. The Relation of Special Training and General Intelligence. *Educational Review* (36):42-48.
- Kafai, Y. B. 1995. *Minds in Play: Computer Game Design as a Context for Children's Learning*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Kant, Immanuel. 1781. *Critique of Pure Reason*.
- Katz, L.G., and S.C. Chard. 2000. *Engaging Children's Minds: The Project Approach*. 2nd ed: Ablex.
- Kearsley, Greg. 2004. *Explorations in Learning & Instruction: The Theory Into Practice Database* (WWW Version 2.2) 2002 [cited September 23, 2004]. Available from <http://tip.psychology.org>.
- Klahr, David, and Milena Nigam. 2004. The Equivalence of Learning Paths in Early Science Instruction: Effects of Direct Instruction and Discovery Learning. *Psychological Science* 15 (10):661-667.
- Koedinger, Kenneth R. 2001. Cognitive tutors as modeling tool and instructional model. In *Smart Machines in Education*, edited by Feltovich. Menlo Park, CA: AAAI Press.
- Lander, D., A. Walta, M. McCorriston, and G. Birchall. 1995. A Practical Way of Structuring Teaching for Learning. *Higher Education Research and Development* 14:47-59.
- Langer, E. J. 1975. The Illusion of Control. *Journal of Personality and Social Psychology* (32):311-328.
- Lave, J., and E. Wenger. 1990. *Situated Learning: Legitimate Peripheral Participation*. Cambridge, UK: Cambridge University Press.
- Lawler, R. W. 1982. Designing Computer Microworlds. *Byte* (7):138-160.
- Leelawong, Krittaya, Joan Davis, Nancy Vye, Gautam Biswas, Dan Schwartz, Kadira Belyne, Thomas Klatzberger, and John Bransford. 2002. The Effects of Feedback in Supporting Learning by Teaching in a Teachable Agent Environment. Paper read at Fifth International Conference of the Learning Sciences, October 23-26, at Seattle, Washington.
- Leelawong, Krittaya, Karun Viswanath, Joan Davis, Gautam Biswas, Nancy J. Vye, Kadira Belyne, and John. B. Bransford. 2003. Teachable Agents: Learning by Teaching Environments for Science Domains. Paper read at The Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence, August 12-14, 2003, at Acapulco, Mexico.
- Leelawong, Krittaya, Yingbin Wang, Gautam Biswas, Nancy Vye, and John Bransford. 2001. Qualitative Reasoning Techniques to Support Learning by Teaching: The Teachable Agents Project. Paper read at Fifteenth International Workshop on Qualitative Reasoning, at San Antonio, Texas.

- Lepper, Mark R., and T. W. Malone. 1987. Intrinsic Motivation and Instructional Effectiveness in Computer-Based Education. In *Aptitude, learning, and instruction: Conative and affective process analyses*, edited by R. E. Snow and M. J. Farr. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Lepper, Mark R., Maria Woolverton, Donna L. Mumme, and Jean-Luc Gurtner. 1993. Motivational Techniques of Expert Human Tutors: Lessons for the Design of Computer Based Tutors. In *Computers as Cognitive Tools*, edited by S. P. Lajoie and S. J. Derry. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Lippitt, Peggy. 1976. Learning Through Cross-Age Helping: Why and How. In *Children as teachers: Theory and research on tutoring*, edited by V. L. Allen. New York: Academic Press.
- Malone, T. W., and Mark R. Lepper. 1987. Making Learning Fun: A Taxonomy of Intrinsic Motivation for Learning. In *Aptitude, learning, and instruction: Conative and affective process analyses*, edited by R. E. Snow and M. J. Farr. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Mayer, Richard E. 2004. Should There Be a Three-Strikes Rule Against Pure Discovery Learning? *American Psychologist* 59 (1):14-19.
- McClelland, D. C., J. W. Atkinson, R. W. Clark, and E. L. Lowell. 1953. *The Achievement Motive*. New York: Appleton.
- Mengelle, T., Frasson, C. 1996. A Multi-Agent Architecture for an ITS with Multiple Strategies. Paper read at CAL-ISCE.
- Merriam-Webster. 2005. *Websters Dictionary*. Online Edition ed.
- Michie, Donald, Andrew Paterson, and Jean Hayes-Michie. 1989. Learning by Teaching. Paper read at 2nd Scandinavian Conference on Artificial Intelligence '89, at Tampere, Finland.
- Miller, Craig S., Jill Fain Lehman, and Kenneth R. Koedinger. 1999. Goals and Learning in Microworlds. *Cognitive Science* 23 (3):305-336.
- Modell, H.I., and J.A. Michael, eds. 1993. *Promoting Active Learning in the Life Science Classroom*. New York: New York Academy of Sciences.
- Mullis, Ina V. S., Michael O. Martin, Eugenio J. Gonzalez, Kelvin D. Gregory, Robert A. Garden, Kathleen M. O'Connor, Steven J. Chrostowski, and Teresa A. Smith. 1999. *Trends in Mathematics and Science Study: International Mathematics Report*: International Study Center, Lynch School of Education, Boston College.
- Nichols, David. 2005. *Issues in Designing Learning by Teaching Systems* 1994 [cited September 3, 2005]. Available from <http://citeseer.ist.psu.edu/148715.html>.
- Nichols, David Martin. 1994. Intelligent Student Systems: An Application of Viewpoints to Intelligent Learning Environments. Ph. D., Computing Department, Lancaster University, Lancaster, UK.
- NIST/SEMATECH. 2004. *e-Handbook of Statistical Methods*. Carroll Croarkin and Paul Tobias n. d. [cited December 2, 2004]. Available from <http://www.itl.nist.gov/div898/handbook/>.
- Obajashi, Fumiaki, Hiroshi Shimoda, and Hidekazu Yoshikawa. 2000. Construction and Evaluation of a CAI System Based on 'Learning by Teaching' to Virtual Student. *Information Processing Society Japan Journal* 41 (12):21.
- Ohlsson, S. 1986. Some principles of intelligent tutoring. *Instructional Science* 14:293-326.

- Owens, Stephen, G. Biswas, M. Nathan, L. Zech, J. Bransford, and S. Goldman. 1995. Smart Tools, A Multi-Representational Approach to Teaching Functional Relationships. Paper read at International Conference on Artificial Intelligence in Education, at Washington D.C.
- Palincsar, A. S., and A. L. Brown. 1991. Reciprocal Teaching of Comprehension-Fostering and Monitoring Activities. *Cognition and Instruction* 1:117-175.
- Palthepe, Srinivas, Jim E. Greer, and Gordon I. McCalla. 1991. Learning by Teaching. Paper read at The International Conference on Learning Sciences, at Illinois, USA.
- Papert, Seymour. 1980. *Mindstorms*. New York: Basic Books, Inc.
- Pea, R. R., and D. M. Kurland. 1984. On the Cognitive Effects of Learning Computer Programming. *New Ideas in Psychology* 2 (2):137-168.
- Piaget, Jean. 1953. How Children form Mathematical Concepts. *Scientific American* 189 (5):74-81.
- . 1954. *The Construction of Reality in the Child*. New York: Basic Books.
- Pintrich, Paul A., David A. F. Smith, Theresa Garcia, and Wilbert J. McKeachie. 1993. A Manual for the Use of the Motivated Strategies for Learning Questionnaire.
- . 1993. Reliability and Predictive Validity of the Motivated Strategies for Learning Questionnaire (MSLQ). *Educational and Psychological Measurement* 53 (3):801-813.
- Quinlan, J.R. 1986. Induction of decision trees. *Machine Learning* 1 (1):81-106.
- Reif, Frederick, and Lisa A. Scott. 1999. Teaching Scientific Thinking Skills: Students and Computers Coaching Each Other. *American Journal of Physics* 67 (9):819-831.
- Reusser, Kurt. 1993. Tutoring Systems and Pedagogical Theory: Representational Tools for Understanding, Planning and Reflection in Problem Solving. In *Computers as Cognitive Tools*, edited by S. P. Lajoie and S. J. Derry. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Rickel, Jeff, Rajaram Ganeshan, Charles Rich, Candace L. Snider, and Neal Lesh. 2000. Task-Oriented Tutorial Dialogue: Issues and Agents. Cambridge: Mitsubishi Electric Research Laboratories.
- Salomon, Gavriel, and D. Perkins. 1988. Teaching for transfer. *Educational Leadership*:22-32.
- Schank, Roger C., and Daniel J. Edelson. 1989. Discovery Systems, Artificial Intelligence and Education. Paper read at Proceedings of the 4th International Conference on AI and Education.
- Schwartz, D., S. Brophy, X. Lin, and J. Bransford. 1999. Software for Managing Complex Learning: Examples from an Educational Psychology Course. *Educational Technology, Research and Development* 47 (2):39-59.
- Schwartz, D., X. Lin, S. Brophy, and J. Bransford. 1999. Toward the Development of Flexibly Adaptive Instructional Designs. In *Instructional-Design Theories and Models: New Paradigms of Instructional Theory*, edited by C. Reigeluth. Mahwah, NJ: Erlbaum.
- Schwier, R., and E. Misanchuk. 1993. *Interactive Multimedia Instruction*. New Jersey: Educational Technology Publications.
- Scott, Lisa A. 1991. Design and Assessment of an Interactive Physics Tutoring Environment. Ph. D. Thesis, School of Education, University of Pittsburgh, Pittsburgh.

- Self, John. 1990. Theoretical Foundations for Intelligent Tutoring Systems. *Journal of Artificial Intelligence in Education* 1 (4):3-14.
- . 1991. *Formal Approaches to Student Modelling*. Lancaster: Lancaster University.
- Sison. 1998. Student Modelling and Machine Learning. *International Journal of Artificial Intelligence in Education* 9:128-158.
- Sternberg, Robert J. 1995. *In Search of the Human Mind*. Fort Worth, TX: Harcourt Brace & Company.
- Stipek, Deborah J. 1988. *Motivation to Learn*. Englewood Cliffs, NJ: Prentice Hall.
- Thompson, P. W. 1994. The Development of the Concept of Speed and its Relationship to Concepts of Rate. In *The Development of Multiplicative Reasoning in the Learning of Mathematics*, edited by G. Harel and J. Confrey. Buffalo: NY: SUNY Press.
- . 1994. Students, Functions, and the Undergraduate Mathematics Curriculum. In *Research in Collegiate Mathematics Education*, edited by E. Dubinsky, A. H. Schoenfeld and J. J. Kaput: American Mathematical Society, Providence, RI.
- Torp, Linda, and Sara Sage. 2002. *Problems as Possibilities: Problem-Based Learning for K-16 Education*. 2nd ed: Association for Supervision and Curriculum Development.
- Tripathi, A. N. 1979. Memory for Meaning and Grammatical Structure: An Experiment on Retention of the Story. *Psychological Studies* 24 (2):136-145.
- VanLehn, K. 1990. *Mind Bugs*. Cambridge, MA: MIT Press.
- Vosniadou, S., and W. F. Brewer. 1989. The Concept of the Earth's Shape: A Study of Conceptual Change in Childhood. *Unpublished paper*.
- Vye, Nancy J., D.L. Schwartz, J.D. Bransford, B.J. Barron, L.K. Zech, and Cognition and Technology Group at Vanderbilt. 1997. SMART Environments That Support Monitoring, Reflection, and Revision. In *Metacognition in educational theory and practice*, edited by D. Hacker, J. Dunlosky and A. Graessar. Mahwah, NJ: Erlbaum.
- Vygotsky, Semyonovich L. 1978. *Mind in Society*. Cambridge, MA: Harvard University Press.
- Weerasinghe, A., and A. Mitrovic. 2002. Enhancing learning through self-explanation. Paper read at Proc. ICCE 2002.
- Weigel, Van B. 2002. *Deep Learning for a Digital Age: Technology's Untapped Potential to Enrich Higher Education*. San Francisco: Jossey-Bass.
- Wenger, Etienne. 1987. *Artificial Intelligence and Tutoring Systems*.
- White, Barbara Y., Todd Shimoda, and John R. Frederiksen. 1999. Enabling Students to Construct Theories of Collaborative Inquiry and Reflective Learning: Computer Support for Metacognitive Development. *International Journal of Artificial Intelligence in Education* 10 (2).
- Whitehead, A.N. 1929. *The Aims of Education*. New York: MacMillan.
- Wilson, K. 1992. Discussion on Two Multimedia R & D Projects: The Plaenque Project and the Interactive Video Project of the Museum Education Consortium. In *Interactive Multimedia Learning Environments*, edited by M. Giardina. Berlin: Springer-Verlag.

Wilson, Robert A., and Frank Keil, eds. 2001. *The MIT Encyclopedia of the Cognitive Sciences (MITECS)*. Cambridge, MA: MIT Press.

Zimmerman, Barry J., and Dale H. Schunk, eds. 2001. *Self-regulated Learning and Academic Achievement : Theoretical Perspectives*. Mahwah, N.J.: Lawrence Erlbaum Associates.

Zuckerman, M., J. Porac, D. Lathin, R. Smith, and E. L. Deci. 1978. On the Importance of Self-Determination for Intrinsically Motivated Behavior. *Personality and Social Psychology Bulletin* (4):443-446.