ABSTRACT

AN EVOLVING APPROACH TO LEARNING IN PROBLEM SOLVING AND PROGRAM DEVELOPMENT: THE DISTRIBUTED LEARNING MODEL

by
Idania Espinosa

Technological advances are paving the way for improvements in many sectors of society. The US education system needs to undergo a transformation of existing pedagogical methods to maximize utilization of new technologies. Traditional education has primarily been teacher driven, lectured-based in one location. Advances in technology are challenging existing paradigms by developing tools and educational environments that reach diverse learning styles and surpass the boundaries of current teaching methods.

Distributed learning is an emerging paradigm today that has promise to contribute significantly to learning and improve overall academic success. This research first explores various systems that provide different modes of learning. The problem domain of this research is the difficulty novice programmers' face when learning to program. This paper proposes how distributed learning can be used in a teaching environment to enrich learning and the impacts for the given problem domain.

AN EVOLVING APPROACH TO LEARNING IN PROBLEM SOLVING AND PROGRAM DEVELOPMENT: THE DISTRIBUTED LEARNING MODEL

by
Idania Espinosa

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer Science

May 2002

APPROVAL PAGE

AN EVOLVING APPROACH TO LEARNING IN PROBLEM SOLVING AND PROGRAM DEVELOPMENT: THE DISTRIBUTED LEARNING MODEL

Idania Espinosa

Dr. Fadi Deek, Dissertation Advisor — Date
Associate Dean, College of Computing Sciences and Associate Professor, NJIT

Dr. James McHugh, Committee Member — Date
Acting Chair, Computer Science Department and Associate Professor, NJIT

Dr. Qianhong Liu, Committee Member — Date
Assistant Professor, Computer Science Department, NJIT

BIOGRAPHICAL SKETCH

Author: Idania Espinosa

Degree: Master of Science

Date: May 2002

Undergraduate and Graduate Education:

- Master of Science in Computer Science
  New Jersey Institute of Technology, Newark NJ, 2002.

- Bachelor of Science in Computer Science
  New Jersey Institute of Technology, Newark NJ, 1999.

Major: Computer Science

This thesis is dedicated to my father. He instilled in me discipline and the desire to achieve my goals. Cancer took his life when I was a junior in high school. This event led me closer to God and to discover my eternal Father. I want to thank God for my family, for all the events in my life, and for guiding me always in the right path.

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to my research advisor, Dr. Fadi Deek, for his valuable support in providing resources, insight, and professional advice during my research and throughout my undergraduate career. I would like to express gratitude to Dr. James McHugh and Dr. Qianhong Liu for serving as members of my review committee.

I would also like to acknowledge the Optical Networking Group of Lucent Technologies for affording me the opportunity to pursue my Master's degree.

I would like to thank my family and friends for their continued support, love and encouragement throughout the program. Finally, I would like to thank God for with him all things are possible.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Objective

Novice programmers have difficulties in breaking down a given problem, designing a working solution, and debugging a program. Different systems are discussed that have attempted to assist and facilitate understanding of programming. The systems that are discussed are collaborative, non-interactive, distance learning, and natural programming. [12][13][14][16]

Many programming environments, software applications, and teaching tools have been developed over the years to address this issue. Along with the advancement in information and communication technology, the method of learning has also evolved with the times. Formal education accounts for only 19 percent of students' time. In order to achieve improvements in learning, the scope of learning needs to be extended outside of the domain of traditional schooling. Distributed learning among social educational environments (i.e. family, work, schools, and extracurricular groups) can enhance students' learning. This paper identifies the improvements in programming classrooms resulting from the use of a distributed learning approach. It also identifies existing issues that cannot be resolved with distributed learning.

1.2 Novice vs. Expert Programmer

Several factors play a role in developing problem solving skills and program development in students. Achieving these skills are based on the internal cognition of the individual, the instructor abilities, environmental factors and the programming language. Why do novices have such a difficult time learning to program? This question is answered from different angles: student, teacher, and programming language.

Novice programmers normally find programming frustrating and difficult. They may not have the sufficient preparation to grasp concepts that are abstract and complex. Novices are not familiar with designing and testing logical structures for solving problems by computer. They lack an adequate mental model about the internals of a computer and how it operates. Novices tend to program using real-world strategies and experiences. [46] They tend to incorporate non-programming experiences in learning to program. For example, in studies done to analyze students' programming, researchers found that students use goal/plan-merging strategy (e.g. reusing variables in loop statements). Although this is useful for conserving resources, skills are needed to apply this in programming. Also, these models are sometimes inappropriate because they do not take into account computer limitations. [10][16][17][21]

Students also have difficulties learning how to program because the instructor drives the speed of teaching programming concepts. Therefore, if students do not have a clear understanding, they tend to fall behind and find it difficult to play catch up. Students may fail to understand a concept given in

textbooks because they failed to learn a previous concept correctly. When examples in the text are based on previous examples, students have difficulty understanding the concept because they failed to understand the previous concept. [27]

The following barriers have been identified as obstacles to learning critical thinking skills in science: lack of motivation, limited knowledge in science, inability to understand abstract relationships from theory, and difficulty keeping track of a complex discussion. [48]

When expert programmers encounter unexpected problems during programming, they possess the skills necessary to resolve the possible causes of errors. Experts think about problems in a more abstract manner. They have the ability to visualize "commonalities and differences among various problems and programs." [22] Novices often times are unable to proceed with the necessary steps for debugging a problem. Their level of thinking remains at a low-level in terms of individual statements, whereas an expert might think in terms of high-level algorithms and devise an end-to-end solution.

These reasons prohibit the novice from learning the essential skills in programming, such as developing a concrete mental model of how the computer operates during program execution, debugging a problem, and finding relationships between abstract programming concepts.

1.3 Contrasting Teaching Methods and Learning Styles

Inappropriate teaching methods used to teach programming are also a factor that hinders students' ability to learn. In a study of college science instruction, Sheila Tobias defines two tiers of entering college students. [23] The first group is composed of those who go on to earn science degrees. The second group includes those who have the initial intention to pursue science degrees and the ability to do so but instead switch to non-science related fields.

Tobias's study concludes that the instruction in introductory science courses is responsible for deterring many students in the second tier. Some of the target areas include:

1. failure to motivate interest in science by not establishing its relevance to the students' lives and personal interests
2. students passivity in the classroom
3. emphasis on competition for grades rather than cooperative learning
4. focus on algorithmic problem solving as opposed to conceptual understanding.

The emphasis in introductory CS1 and CS2 courses has mainly been on language construct and programming syntax. [12][18] The mentality has been 'programming-in-the-small' whereby small programming assignments were given to teach computer concepts. Students mainly worked individually as a means to gain fundamental understanding, which resulted in developing individualistic work habits. Group assignments were given often times too late in the course to offset these habits. [36] In the past, writing small programs was a viable means of

developing applications. This no longer is possible. Corporations require software development teams to develop and maintain products. Introductory CS curriculums need revisions to pass down the necessary programming skills for today's fast paced market. This includes team-oriented assignments, early exposure to large programs, focus on the software development process, and software reusability.

1.3.1 Traditional Teaching Methods

There also exists the debate between proponents of reform teaching and those in favor of traditional teaching methods. "Our vision would replace a future based around lectures, seminars, tutorials and personal tutoring by a high tech/high-touch future based around the Internet, action learning and large group workshops". [5][12] Traditional methods often involve lecture-based classes. It has been found that when instructors lecture, it is easy to disconnect and lose interest in the material after an extended period of time.

Some of the weaknesses and disadvantages of lecture-based classrooms are provided below. [50]

1. Lectures hinder learning by placing students in a passive role.
2. Lectures require an effective speaker.
3. Lectures are not suited for complex, detailed or abstract material.
4. Lectures fail to teach students where and how to find new material.
5. Lectures fail to teach students how to solve problems through content application.

If abstract scientific concepts are not taught in a manner that will allow students to think and understand, they will not develop these cognitive skills. Student surveys find instructors teaching in a repetitive manner and their learning too passive. Students reported spending a lot of time note-taking in class rather than learning the material. Also, much study time was spent memorizing rather than in activities such as analysis, application, and evaluation. [50]

The materials and tools used in classrooms also play a role in the effectiveness of learning. Jeff Salvage, author of "The C++ Coach", notes that in his experience teaching introductory programming course for information systems students, they were "intimidated by the hard-core approach previously taken to teach C++." [41] He found that even though he incorporated real life system examples and fun examples that students could follow, students complained about the manner and complexity in which textbooks were designed.

1.3.2 Learning Styles

Educational research shows that students have different styles of learning. These learning styles can be identified based on the type of information students prefer to receive: sensory or intuitive, visual or verbal, inductive or deductive, actively or reflectively, and sequentially or globally. [2] Instructors should mold their teaching styles to that of students' learning styles in order to effectively maximize learning in classroom.

Sensing learners tend to retain information like facts and observations. They process information more effectively through sights, sounds, and physical sensations. They are detailed oriented and prefer solving problems using well-

defined steps. They are practical and complain when information is not related to the real world. Intuitive individuals, on the other hands, are more imaginative and thus prefer abstract concepts and information that is open to interpretation. They rely on memory, ideas, and their own insights. Sensing learners tend to get lower grades than intuitive learners do on lectured courses.

Visual learners retain information through images more so than verbally. They perceive information more effectively via diagrams. If a lecture does not incorporate some visual representation of a concept, visual learners have the tendency to forget the material. Research shows that most students in science classes are visual learners. [2] Lectures are mostly verbal with abstract concepts, formulas, and few diagrams. Verbal learners learn based on written and spoken words.

Inductive learners learn based on first specific scenarios and then more general principles. Students tend to use given facts and observations to infer principles. Deductive learners, on the other hand, prefer general principles first and then deduce consequences. Research shows that inductive learning promotes a deeper understanding and retention of information. Classrooms tend to use deductive approach. [43]

Active learners prefer group discussions and exchanging ideas with others. Reflective learners tend to think things through and then experiment. They prefer individual work or in pairs. In a lectured course, both groups are actually neglected and enforce a passive approach to learning. Studies show

that students in an active environment excel in comprehension, memory retention, and problem solving. [29]

Sequential learners gain an understanding of information in small, connected pieces. They solve problems efficiently, but may lack an understanding of the larger problem domain. Global learners' approach is all-or-nothing. Initially, they may appear to be slow students, but once they grasp the larger picture, they can make connections that sequential learners do not identify.

1.4 Programming Language Construct

Another hypothesis for the difficulties that beginner programmers experience is that programming solutions are expressed in ways not familiar or natural to the novice. Research shows that "programming languages make the task more difficult than necessary because they have been designed without careful attention to human-computer interaction issues." [37] The way in which programmers formulate algorithms and manipulate data is not the same as that of every day life situations. Modern programming language paradigms do not associate with the natural tendencies of solving a problem. "The mismatch between the way programmers think about a solution and the way it must be expressed in the programming language makes it more difficult not only for beginners to learn how to program, but also for people to carry out their programming tasks even after they become more experienced." [37]

In the fields of Psychology of Programming and Empirical Studies of Programmers, programming is classified as, "a process of transforming a mental

plan that is in familiar terms into one that is compatible with the computer." This process of transformation has not been carefully analyzed and so existing programming languages today contain flaws that make it difficult to learn. Hoc and Nguyen-Xuan have determined that many difficulties and bugs are created because the distance between the mental plan and the computer's interpretation of this plan is too large. This concept has been labeled as closeness of mapping: "The closer the programming world is to the problem world, the easier the problem-solving ought to be ... Conventional textual languages are a long way from that goal." [25]

1.4.1 Programming Abstractions

Deficiencies in problem solving are a problematic area for novice programmers. Payne observed that mental models are generated for daily encounters with computers. Unlike this belief, programming language designers have developed languages, such a Scheme or SML, based on mathematical abstraction alone. Students who are learning these languages develop their own metaphors for understanding purposes. The advantage is that once they have a mental model, they can visualize the behavior and attributes clearly.

When programmers begin a project, they drive off of a set of requirements that define the behavior of the program. They may understand the required graphical interface and external devices necessary, but may not have an understanding of the abstract modeling that make up the program. This phase is the analysis and design steps. This consists of creating and transforming abstract models into conceptual tools. Class diagrams, sequence diagrams, and

other visual tools have been deemed useful in understanding the problem domain.

These abstract problems need to be analyzed in a manner that novice programmers can understand and solve across different levels of abstractions. Scientist and mathematicians have analyzed the experience of creativity and have concluded that people tend to form "mental images as a way of relating a new problem in an unstructured domain to [an] existing experience." [4] Kaufmann shows that individuals' imaginations and visual creativity are related to one's ability for interpretation. He also notes the importance of the role language plays in manipulating familiar concepts and the role images play in manipulating unfamiliar ones.

1.4.2 Programming Syntax

The syntax of a language plays a role in a programmer's ability to use the language. Just like the English language has rules of grammar to formalize written communication, computer languages too have a set of grammar rules. This has evolved in the past couple of decades from a few specialists understanding zeros and ones of computers to people interfacing to computers in a reasonable manner.

The following describes some of the usability issues in today's programming languages. [34] Designing a programming language has resulted in providing various ways of performing the same operation. This added flexibility of programming choices has led to many problems. C programming

language has 16 levels of precedence, some are left associative and others are right associative. (e.g. a=b--=c=+d*e+++f=-g=h++).

As mentioned earlier, instead of providing an interface closer to the user's language, computer scientists had to learn the bits and bytes of a computer. The learning curve was much longer than presently. Yet, further advancements in programming language design are still necessary until users have minimal problems in communicating with computers. Currently, C++ uses "void" to mean nothing, "char" to mean a byte, and a '+' to mean many things. Arrays begin at zero, but people tend to begin counting from one. The following is a more extensive list of ambiguities of a language.

- "==" used for equality vs. "=" used for assignment.
- 3/4 should evaluate to .75, but it evaluates to zero.
- In PERL, variables, arrays, hashes, and functions are allowed to all have the same name: 'foo(@foo[$foo])'.
- Object oriented programming is a hard way to think for many people.

It is important to be consistent in a language. In C++, the word static signifies at least three different things. In some cases, C++ declarations end with ";" and in other cases it ends with ",". In function calls, parameters can be passed by value, reference, and name. Some types are automatically type-casted (e.g. int to float) while others are not. Java has a completely different way of handling data based on whether variable types are reference types or primitive types. See the following example. [34]

```
MyRecordType x = new MyRecordType(); // valid to use "new" operator
x.removePhone();   // valid to call member method of object
x = x + 1;         // illegal to use primitive operations on objects
int z = new int(2); // illegal to use the "new" operation on primitives
int y = 3;         // have to use direct assignment to create values
y = y + 4;         // valid to use primitive operations to modify value
y.add(5);          // illegal, primitives do not have member methods
```

Today, the manner in which programming languages are constructed results in error-prone coding. C and C++ do not have any error checking for out of bound array assignments or incorrect pointer arithmetic. Typographical errors, not detecting by the compiler, result in the program behaving incorrectly e.g., "=" for "==", "x+=7" vs. "x=+7". Giving the responsibility to the programmer to perform memory management in C and C++ results in memory leaks problems. Interpreted languages detect problems only at run time.

These inconsistencies and ambiguities in programming languages today play a negative impact in programmers' ability to learn the language with ease.

CHAPTER 2

EXISTING APPROACHES TO AIDING NOVICE PROGRAMMERS

There are endless proposals, software packages, development tools, and learning environments that exist today that attempt to provide a solution to aiding novice programmers in problem solving and program development. [11][15] The following sections describe and analyze various learning systems and serve as a microcosm for each specific domain. The systems for discussion include a collaborative environment, an individual cognitive system, the natural programming project, and distance learning.

2.1 Collaborative Learning

Collaborative learning is a term used today that has come to signify a wide variety of things. In order to understand the cognitive effects of collaborative learning, one must understand the context of usage. Many terms and definitions exist today that blur the definitions. Among the many terminologies are cooperative learning, problem-based learning, and team learning. Cuseo, Smithand, and MacGregor attempt to distinguish these terms. [7][44] The general definition of collaborative learning is a situation in which two or more people learn or attempt to learn something. [19]

The broadness of meaning allows for exploration of the dimensions of learning based on group size, time, and level of interaction. Examples of variety of scales range from a small number of students working on a project together (face-to-face interactions) for a semester to a group of professionals developing

an entire software system (face-to-face meetings, videoconferencing, email) for a period of years. Much research has been done on the effectiveness of collaborative learning for small-scale groups between two to five people. Contrary to this, little research has been done on computer-supported collaborative learning (CSCL), which normally averages 40 people for a period of a year. Evolving research shows the notion of 'scale' changes. A group can be viewed as a unit and an individual can be seen as a group. According to Minsky, distributed cognition treats a group as a single cognitive system; the individual is seen as the distributed system. [31] Piaget, Mead, and Vygostky believe the notion that thinking is a result of dialogues with oneself. [19] According to Hoppe and Ploetzner, collaborative learning is encouraged in CSCL by identifying subgroups. Learners are matched based on different criteria (e.g. skills/ knowledge complement each other).

Learning includes any collaborative activity performed within an educational context. It includes activities such as studying, working on assignments, problem solving and brainstorming. One study shows the activity to be problem solving where learning is actually the side effect of problem solving. Another situation involves learning based on acquiring or gaining experience from a professional community for a period of time.

Collaborative learning should not be viewed as a single means for learning. Just as individuals learn because they perform activities such as reading, thinking and answering questions, collaborative cognitive systems learn because they perform activities together such as explaining information, arguing,

and answering each other's questions. Individual cognition is not suppressed during peer interactions. In fact, these extra activities triggered by interactions would cause internal cognitive processing.

The field of collaborative learning studies these activities that trigger learning among subjects. At the present, there is no guarantee that these learning mechanisms occur in collaborative interactions. Schools today enforce group work by providing constraints such as limiting group size, giving common grade for group work, etc. This does not necessarily constitute collaboration because there is no guarantee that the instructions set forth by instructors will be adhered to or that groups of learners will contribute to the end solution.

Four categories have been identified to increase group interactions. [19]

1. Set up pre-conditions
2. Define the collaboration agreement with a scenario based on roles
3. Define interaction rules in the group
4. Monitor interactions

The first category involves the initial setup. This includes identifying group size, gender, common and differing viewpoints of group members. In order to effectively cause interactions, the criteria for selecting groups need to be defined. Some research has already begun to study and find results to set up these pre-conditions; findings show complexity in combining factors and base this on the vastness of differing projects. [20]

The second category defines specific roles to group members. This includes grouping individuals that will trigger a conflict or compliment learner's

knowledge. The third category increases interactions by defining rules (e.g. everyone must provide his/her input). Lastly, a facilitator needs to monitor and assist interactions. His purpose is to direct groups in a productive direction and ensure that everyone is adhering to pre-conditions, rules, and participating in discussions, etc.

Collaboration is classified into three aspects of learning: a situation, an interaction, and a learning mechanism. [19] By situation, collaboration occurs between people that are at the same level. Their skill set should be comparative and workload should be divided evenly. A teacher/student relationship is not considered collaborative.

The second aspect of collaboration is interaction. It is effective based on the degree to which the interactions influence the peers' cognitive thinking. If a group has simply divided the work evenly, each working on his part, and then come together to combine their work, no additional cognitive thinking has developed. The work was based on individual cognition alone. Collaborative learning therefore involves synchronous communication. In terms of communication tools, a chat room and instant messenger is considered synchronous. E-mail is considered asynchronous. Another factor in peer collaboration is negotiating. Many times there is someone in the group who imposes his viewpoint without allowing others to contribute. This can hinder learning and lead to frustration and dissatisfaction. The negotiation process is a means for peers to develop a shared solution or mutual understanding. In addition, peers may misunderstand each other due to differences in opinions,

language barriers, and miscommunication. This should not be eliminated from a collaborative environment because it leads to learning through explaining one's views and reformulating one's statements. 'Explanation-based learning' is viewed as very important in cognitive science because individuals reinforce their views by elaborating explanations and proof on their views. [32][52] At an extreme, when misunderstanding is high among group members, learning is not possible.

The third aspect of collaboration is learning mechanisms. The mechanisms specific to individual cognition are extended to group interactions since individuals are a component of group interactions. One common mechanism is induction – the process of deriving general principles from particular facts or instances. Groups would integrate models developed by individuals into a common abstract representation. [49] Another mechanism is cognitive load, which means that collaboration involves a horizontal division of labor that reduces the processing performed by each member. On the other hand, cognitive load of group members increases as they interact with each other by explaining ideas. This balance is necessary for optimal learning.

Internalization and appropriation are characteristics specific to collaborative situations. Internalization is the process of transferring "tools from the social plane (interactions with others) to the inner plane (reasoning)..." [19] One general case is of a child learning from an adult. The adult performs a task with the child and the child later understands and performs the same task. Few studies have been performed for symmetric situations and so it is unclear how

this process is fully achieved. Appropriation is another learning mechanism whereby an individual "reinterprets his own action or utterance under the light of what his partner does or says next." [19] This involves reformulating a previously defined concept with the influences of a peer.

2.1.1 History of Belvedere – Collaborative System

Belvedere started off as a project designed to support critical thinking skills with the use of hypertext systems and graphical user interfaces (GUI) for young students. The focus of the Belvedere system was to develop the cognitive and motivational limitations of beginners as found by psychological studies and testing with 12-15 years olds and in 10^{th} grade classrooms in city public high school.

Belvedere addresses some of students' limitations in problem resolution. It uses diagrams to represent abstract relationships that enable users, who normally would have difficulty recognizing these abstractions in scientific theory, to obtain a concrete understanding. The diagrams (box-and-link representation) help users identify different kinds of relations. An online advisor is used to assist users who find it difficult to focus on important aspects of a complex problem. The advisor highlights an area of the diagram that may require more attention. To address the issue of students' lack of motivation and limited knowledge domain, such as in science, Belvedere is designed to support group work. The aim is to provide peer motivation. [48]

One of the key objectives of the Belvedere system is to learn critical inquiry skills. This is achieved by gaining understanding in the area of study,

identifying the problem domain, identifying possible solutions and evidence to support the solution, drawing conclusions, getting feedback from outside individuals, and evaluating the results of the feedback. [47]

Belvedere is a knowledge-based educational software that enables collaboration between users and represents educational materials shareable between different applications across the Internet. The software is designed so that resources reside on servers. Netscape and Java technology are the means to deliver the user interface. Belvedere has a database that students can access simultaneously. This inquiry database contains problem statement, proposed hypotheses, and material for and against the hypothesis. This database has a Java GUI for students to access. The database is also accessible via HTML based interfaces and represents records in tabular format.

Belvedere software enables separate computers to display the same document. Students are not able to modify an object that another person is working on. A lock mechanism is put in place so that when a user starts to modify the object, no one else can modify it. When the user is done editing the object, the lock is released and others' screens are refreshed.

Belvedere was designed with the aim of providing a user-friendly environment. The learning curve in using Belvedere is very small. The interface resembles a drawing program whereby many of the tools are automated. For example, shapes are created with a default size. Objects and their associated links retain their connection if the object is moved. The automated advisor also provides additional information to the user. Colors are available to represent

different viewpoints. Line thickness can be altered to represent the importance of a relation.

The Belvedere GUI allows for multiple users to view the same screen. See diagram below.

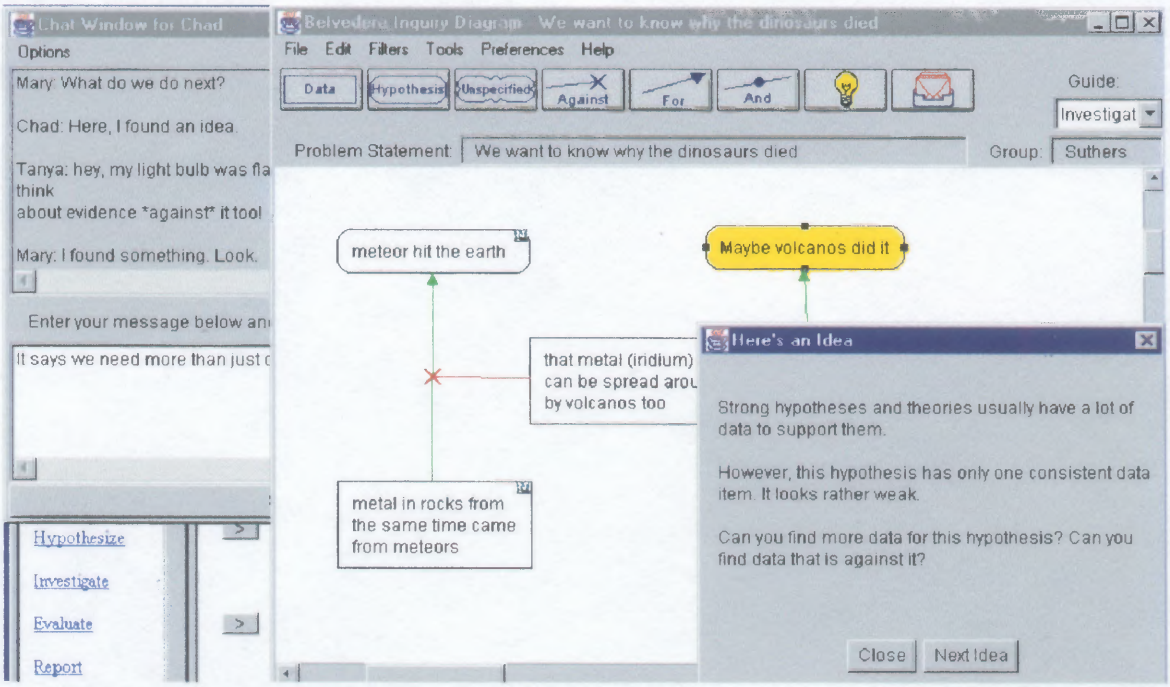

Figure 2.1 Example of Belvedere GUI.

The screen contains the statements and lines to represent relationships between statements. The GUI also has a chat window so users can communicate and exchange ideas. A dialog box, known as the "Coach", is located on the lower right corner of the diagram and provides assistance to students, if needed. It is triggered based on students' demand. When the "Coach" needs to provide critical advice, a light bulb begins to flash. The "Coach" also offers hints based on principles, such as consistency, maximizing a theory, and alternative theories. [48] The inquiry diagram is another part of the

Belvedere GUI that contains the problem statement. The user can add, delete, or modify statements (hypothesis and evidence used to prove/disprove hypothesis). The user can also add and remove relationships between statements. Belvedere also has the capability to go to the Internet for information retrieval purposes. Students can type text into their diagrams or copy information from the Internet.

2.1.2 Example of Belvedere

A study of the Belvedere system was conducted to 10^{th} grade students in an urban high school. The area of discussion was whether HIV is the cause of AIDS. Teachers along with developers of the software developed a database for theories of the cause of AIDS. Eight sessions were conducted with teams of two to three students who worked on their own computers. Although everyone had their own computer, they were sitting close enough to see the team member's monitor.

The following diagram is representative of two thirty-minute sessions.

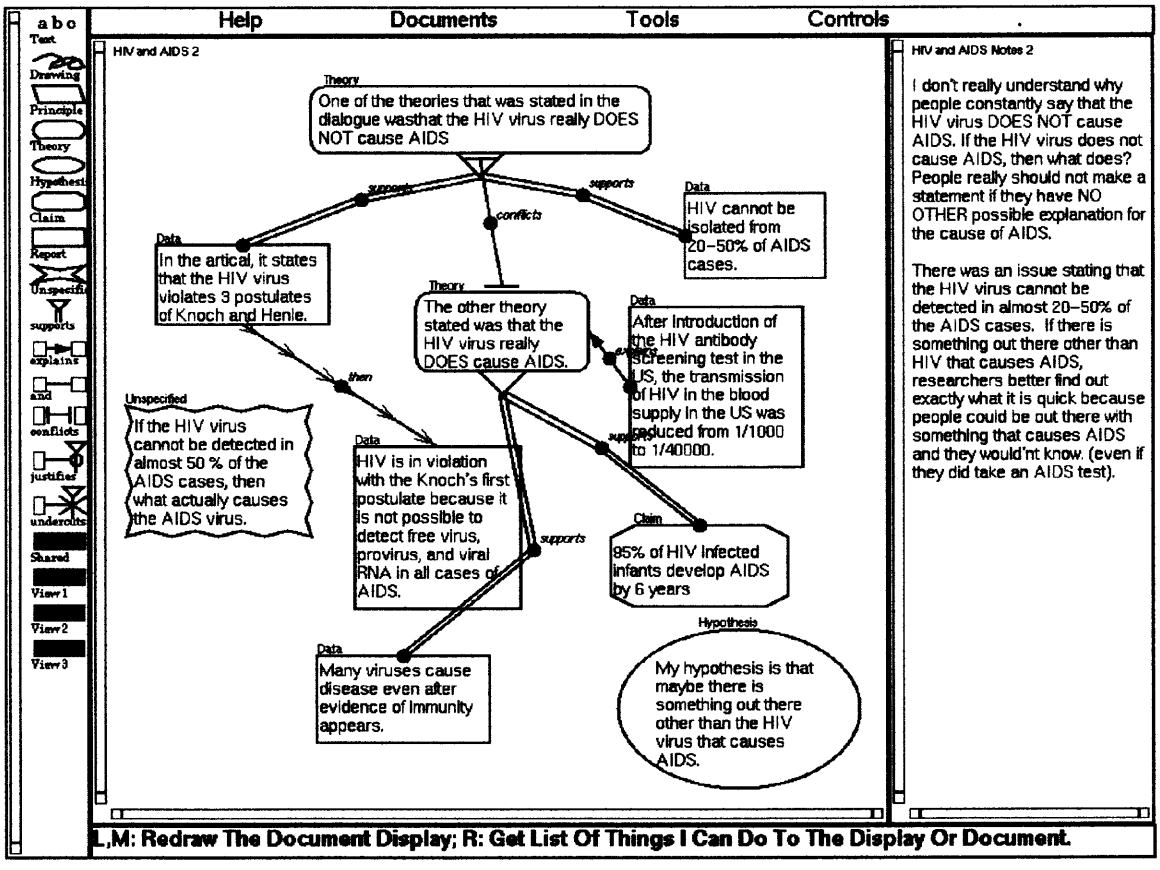

Figure 2.2 Example of Belvedere Diagram on AIDS topic.

The study showed that most of the students created diagrams regarding information on the subject matter. They were able to incorporate ideas into their diagrams. They used information based on knowledge and personal experience, in addition to online material. When discussing this issue with the teacher, they were able to present their opinions.

It was also observed that collaborative work aids in developing many hypotheses. In contrast to a lecture approach where only one perspective is discussed, peer work allowed for multiple ideas to be explored and discussed.

They typically found that more verbal proposals were made rather than entered into the Belvedere software. Peer coaching was also found to be helpful when students persisted on an issue for an extended period of time. Also, students were observed to have complemented each other on their responses. [47]

Working in a social environment also proved to be negative. In a team, one of the members wished to focus on a single hypothesis and prove it true, while another member suggested to brainstorm and find many hypotheses. The latter member at one point got frustrated and challenged the other member's argument. This suggests that a cooperative group may not make the most out of an individual's knowledge and experiences.

2.1.3 Analysis of Belvedere

The Belvedere system is a collaborative tool used for the development of critical thinking skills of students by engaging them in discussions. There are many advantages to the system. Peer motivation, the capability to chat with others via a chat interface, access to online materials, and the "Coach" interface are all benefits of the system. The ease of use of creating diagrams and documenting relationships between statements allow users to develop more concrete representations of abstract concepts. It also assists in identifying new ways to support or negate an argument. Students are able to switch between joint and independent work without losing track of discussions is another benefit.

In the past, to promote collaborative learning, students sometimes had to share a single computer. This was problematic because not everyone had the

chance to interact with the computer, therefore making them a passive learner. Belvedere software enables separate computers to display the same document.

Interoperability and reusability is another key issue that the Belvedere system has addressed. This system has been enhanced to support client platforms on Mac OS, Solaris, and different versions of Windows. It has also been enhanced to have a Java GUI. The Belvedere system has been modeled to add components without affecting existing modules.

Some of the disadvantages that were found in the Belvedere system include the manner in which the shared information is displayed and the content of the shared material. The diagram formations are useful in showing certain relationships between arguments, but the current system is not able to represent mathematical and scientific models for discussion. This cannot fit into the existing text-based diagrams. Another disadvantage is that the "Coach" cannot answer specific questions. The intervention of a professor or moderator is necessary to address users' concerns. This makes the software not entirely stand-alone. This system can be used only for small-scale groups in a classroom setting. It is not scalable.

2.2 Individual Cognitive System

Learning in classrooms has traditionally been individualistic where the focus is on individual performance. Grades are primary based on individual factors such as homework assignments, examinations, and projects. The norm has been to work

alone. The academic community has not modeled teamwork in the educational process. The following system models learning for the individual.

2.2.1 History of Tinker

Tinker is an example-oriented environment for beginner programmers. The idea for developing this software was based on the principle of human learning. Teaching is said to be a learning experience not only for the student, but also the teacher. The lesson can be successfully achieved when abstract knowledge and concrete examples are intertwined. Concrete examples allow both the student and teacher to test their understanding and reduce the reliance of short-term memory. In the same manner that students learn by examples from teachers, the machine should learn by demonstrated examples based on the programmer. This differs from the idea that programming involves the machine to follow a systematic sequence of abstract rules as implemented by the programmer. The immediate goal is achieved when Tinker formulates a generic program based on the given examples, and the programmer in turn attains an understanding of the program based on provided examples. The intent of the software is to achieve the same benefits as in teaching via examples in programming. [8]

Tinker reacts much like a slow student, starting by remembering the examples shown and the steps the programmer performs. Since it does not have a student's capacity to decide for itself what is useful from one example that may be important for future examples, the programmer must decide. At this point, Tinker can create a function for handling such instances.

2.2.2 Example of Tinker

The following example shows how a novice programmer uses Tinker for learning. [8] The student will show Tinker how to build a stack of elements. It is assumed a function exists to move an element onto another element. The simplest example is moving element 1 on top of element 2. Another example is moving element 1 on top of element 2, but there exists element 3 on top of element 1.

Figure 2.3a Simplest Example.

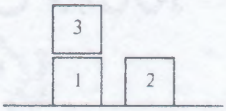

Figure 2.3b Complex Example.

Tinker is useful for creating functions. Based on examples, it generates a set of actions. If arguments are given, the set of actions can do something different based on the input. The menu used by programmers is seen below.

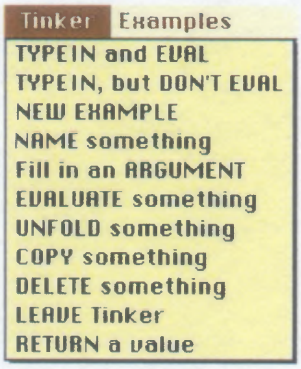

Figure 2.4 Main Menu on Tinker.

The programmer is able to perform the above operations. The first two menu items are used to add new Lisp expressions into the Snapshot window. The third menu item provides a new example to an existing function. The other

operations modify existing operations. The Snapshot window is used to interface with the programmer. See illustration below.

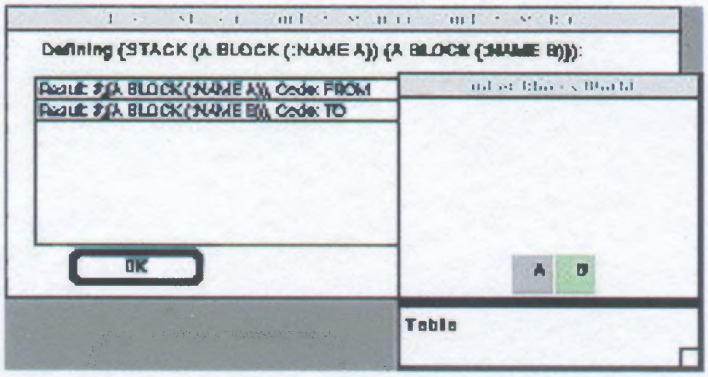

Figure 2.5 Snapshot Window.

Elements A corresponds to the FROM argument of the function. Element B corresponds to the TO argument of the function. The basic operation that is relied upon is Move-Block that simply moves one block on top of another. When the programmer inputs Move-Block and selects the first two lines, the result is Block-A is moved on top of Block-B. Internally, Tinker does "Result: Move-Block FROM TO".

In the more complicated case where element 3 is blocking element 1 to move to element 2, the programmer must show Tinker how to remove the obstacle in a way that it can develop a general program to satisfy this condition. It does this by finding a relationship with the obstacle element and the other elements. The programmer would enter Result: #,(1 'block :name ' 3), Code: (Above From). The function "Above" returns either the element above it or NULL if nothing exists. Therefore, for this case, two steps are needs to reach the end result. Tinker can create a recursive function to handle the case where there

exists many elements on top of the FROM element. This is based on the above two examples.

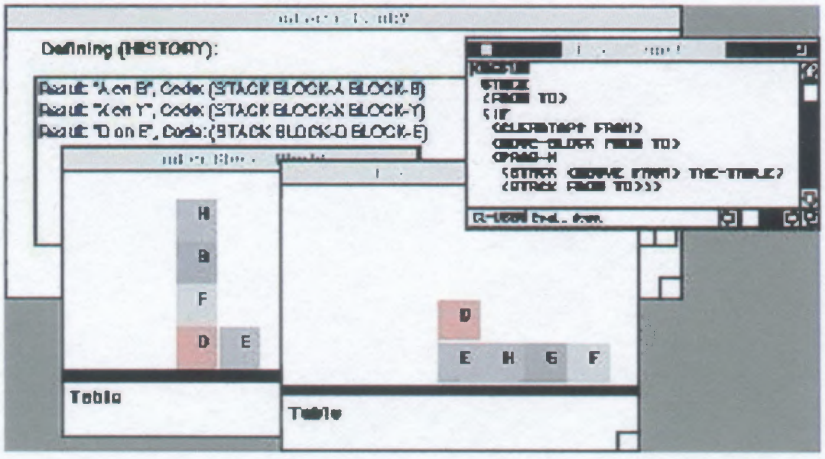

Figure 2.6 Recursively Moving Obstacle to a Table.

Once this has been achieved, Tinker has a complete program to solve the complicated case of moving Element 1 on top of Element 4 even with the other elements being on top.

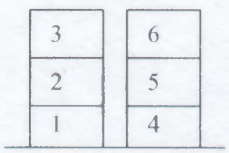

Figure 2.7 Complex Example to Remove Obstacle Elements.

2.2.3 Analysis of Tinker

Tinker is a teaching tool for individuals through the use of examples for building understanding and skills in programming. One of the main benefits of the software is that it allows for beginners to learn the basic concepts of functions, recursion, conditional statements, and loops in incremental steps. The use of examples allows novice programmers to develop their minds to deriving different

cases for a particular program. Unlike current programming today where students learn only sunny-day scenarios, different examples can include also failure cases so that Tinker can develop a generic program.

One of the difficulties beginners experience is programming complex expressions involving nested functions. Tinker assists in this problem by allowing beginners to incrementally build up a complicated expression. The programmer can know the result of sub-expressions before using it as part of a larger expression. Another benefit of Tinker is that it constructs conditionals based on examples. Therefore, there is no untested code in the final program. Tinker also does not make any inferences. In case it finds any ambiguities based on input, it would prompt the user.

On the other hand, Tinker is not a potent teaching tool because its programming language is Lisp, which has primarily been used for Artificial Intelligence applications and not commercial applications. Also, the belief that just as teachers gain a deeper understanding while teaching students does not carry over to the programming world. Tinker relies on good examples to formulate generic functions. These examples are based on the novice's ability to provide good examples. Unless novice programmers have a clear understanding of the problem in question, the examples that are fed into Tinker may not be clear.

Individual cognitive systems have not proven to be sufficient for resolving the problems and limitations discussed in Chapter 1. Individual limitations continue to exist in solving complex problems and achieving an understanding of

abstract concepts. Learning mechanisms such as induction, deduction, and debates with oneself result from the individual. These mechanisms could potentially not reach the same capacity of learning as systems that involve group learning.

2.3 The Natural Programming Project

2.3.1 History

The Natural Programming Project began in the Human-Computer Interaction Institute at Carnegie Mellon University. Its goals are to develop principles, methods and language designs that would reduce the learning curve for non-professionals (novices) in programming. To achieve this, the project studied how non-programmers reasoned and understood programming language concepts. The project performed studies to identify how people naturally express programming concepts. It concluded with a set of guidelines for the design of future programming languages. The domains that are targeted to use these guidelines include new programming languages for children (educational software), "tailorable systems" configurable by users, creation of CGI scripts for web pages development, multimedia authoring, robot controllers, and more. The ultimate goal is to provide all computer users with the capability to automate their tasks through scripting. [33]

2.3.2 Study

Studies have been performed to obtain the natural expressions and thinking process of a general audience on a broad range of problem scenarios. One study examined children's solutions to a set of scenarios necessary to program a computer game. [37] Fourteen fifth graders were given tasks to solve using diagrams and text. Beforehand, they were taught some essential programming methods, such as variables, assignments, comparison of values, boolean logic, arithmetic, selection and iterations, and searching.

A set of nine scenarios from a PacMan game was chosen. Graphical depictions included images, animations and textual data. Their responses were based on the following criteria. [37]

1. an overall summary of the game
2. how the user controls PacMan's actions
3. PacMan's behavior in the presence and absence of other objects such as walls
4. what should happen when PacMan encounters a monster under various conditions
5. what happens when PacMan eats a power pill
6. scorekeeping
7. the appearance and disappearance of fruit in the game
8. the completion of one level and the start of the next
9. maintenance of the high score list

The study found that participants answered questions in an event-based manner. This suggests that an imperative language may not be the most natural choice. A characteristic of imperative languages is explicit control over program flow. Participants responded in a more reactive manner, without attention to the flow control of the game.

The study found that beginners sometimes confused their role while developing a program. Instead of thinking from the programmer's perspective, they took the role of the end-user. A large percentage responded, "When I push the left arrow PacMan goes left." Another discovery made was that the majority of the participants drew pictures when deriving their solutions. Sixty-seven percent included at least one picture.

Another conclusion that can be drawn is that most popular languages today require iterative operations on objects. Participants strongly preferred to use set and subset expressions to specify the operations in aggregate form. Ninety-five percent of the responses indicated a preference to using set and subset specifications. (e.g. "When PacMan gets all the dots, he goes to the next level."). Loops have been known as a problem area for novice programmers. Loops express, in a complicated way, operations that participants express easily using set operations. (e.g. "d5 moves down to d6, d6 moves to d7, etc., until d10 which is kicked off the high score list.") In the study, the raters found that all of the mathematical operations were expressed in a natural language form rather than mathematical notation.

In terms of data structures and data manipulation, participants inserted and deleted data elements without considering issues of storage space. Memory management is considered a problem area in many programming languages today, such as C and C++. Also, participants expected sorting to be a basic operation that can be used as part of their solutions.

The study concluded that programming languages used today contain many gaps. Novice programmers thus have difficulties when transferring their knowledge to the actual language. The constructs and features of a language do not match with the programmer's natural strategies. [37] The table below provides a more exhaustive list of conclusions discovered in the study.

Table 2.1 Results of study

1. Overall Structure		
Programming style 54% Production rules/events 18% Constraints 16% Other (declarative) 12% Imperative	**Perspective** 45% player or end-user 34% programmer 20% other (third-person) **Use Pictures** : 67% Yes	**Modifying state** 61% Behaviors built into objects 20% Direct modification 18% Other
2. Use of Keywords		
AND 67% Boolean conjunction 29% Sequencing	**OR** 63% Boolean disjunction 24% To clarify or restate a prior item 8% "Otherwise" 5% Other	**THEN** 66% Sequencing 32% "Consequently" or "in that case"
3. Control Structures		
Operations on multiple objects 95% Set/subset specification 5% Loops or iteration	**Complex conditionals** 37% Set of mutually exclusive rules 27% General case, with exceptions 23% Complex boolean expression 14% Other (additional uses of exceptions)	**Looping constructs** 73% Implicit 20% Explicit 7% Other
4. Computation		
Remembering state 56% Present tense for past event 19% "After" 11% State variable 6% Discuss future events 5% Past tense for past event	**Arithmetic operations** 59% Natural language style – incomplete 40% Natural language style – complete	**Insertion into a data structure** 48% Insert first then reposition others 26% Insert without making space 17% Make space then insert 8% Other
Tracking progress 85% Implicit 14% Maintain a state	**Randomness** 47% Precision 20% Uncertainty without using "random" 18% Precision with hedging 15% Other	**Sorted insertion** 43% Incorrect method 28% Correct non-general method 18% Correct general method

2.3.3 Analysis

The Natural Programming project is beneficial for the future of new programming language designs. The consideration for human interaction and human processing would greatly benefit programmers in the future to learn with ease. There are many advantages if the guidelines of the study are applied to language designs. The most positive achievement would be that average computer users may one-day use programming languages to customize applications for their own personal use. If languages are simplified for all to quickly grasp and learn, then programming environments, collaborative and distributed learning systems could be used as a means of learning skills in problem solving and program development.

The Natural Programming Project would not achieve solving complex and abstract programming problems. Rather, it facilitates and simplifies the learning process of a language. This method of learning would not be the end-to-end solution for novices. A mechanism is still needed to break down problems and expand programmers' knowledge of a software solution.

2.4 Distance Learning

Online, web-based communication is a key technological advancement that has made distance learning possible today. Information technology is paving the way for a renaissance in learning. Distance education is becoming a common way of teaching using technology-based delivery methods to reach wider audiences. It is believed that the interest in distance learning reflects the many calls for change

in higher education and learning. According to educational consultant and author Vicky Phillips, the number of distance learning students has jumped from 100,000 in 1990 to over one million students today. In 2000, over 5,000 accredited courses were offered over the Web. [26] According to the U.S. Department of Education, Web-Based Education Commission, more than 2 million students are expected to enroll in 2002 via distance learning. With the speedy growth of web technology, the reinvention and design of educational courses lags far behind.

Distance learning is a form of education that breaks the mold of the traditional classroom where the student and teacher are not in the same place. Students learn across the following various media: live broadcasts, two-way interactive videos, videotapes, wireless transmissions, fiber optic cable, and over the Internet. With today's technological advancements, it is possible for education to come to students with disregard to location barriers. This type of learning is becoming both a reality and a vision for future education.

The Internet and web-technology have increased the potential for a new and improved means of transmitting knowledge to students. These are not the only factors that have influenced the growth of distance learning today. According to [26], other factors include:

- Decrease in government subsidies of public institutions of higher education
- Increase in costs of higher education at public and private institutions
- Reductions in secure, long-term jobs

- Increases in credential requirements for entry to and continuing work in many jobs
- Rapid changes in information technologies
- Increases in online business
- Increases in attention to lifelong education

There are two aspects of distance learning: timing and medium of delivery. Timing can be synchronous or asynchronous. Synchronous communication means communication occurring at the same time. Examples of the media of delivery for synchronous learning are teleconferencing and video classrooms where the teacher can broadcast to students at multiple locations live. The advantage is the immediate interaction between student and teacher (just like traditional teaching). Students can ask questions using microphones. The disadvantage is the inconvenience for students to attend video classrooms. Since transmission speeds are still not adequate for quality video imaging, real-time video over the Internet has not been used for distance learning purposes.

In asynchronous learning, students learn at their own pace via the Internet or videotapes. Learning includes videotapes, exercises, and online exams. Access to the instructor is done through email. The obvious advantage is flexibility. The disadvantage in the case of online delivery is slowness of downloading materials (e.g. large files). Also, another disadvantage is having no access or limited access to an instructor.

Distance learning offers an array of benefits, including convenience, flexibility, and availability, regardless of the barriers of time, place or pace of

learning. [3] It provides convenient locations for students and professors. With technologies such as the Internet and videotapes, students can easily obtain access to classroom materials from home. Videoconferencing is another source of learning that is distributed from a single point to multiple remote sites. Students have the flexibility to participate whenever they want, and at their pace (e.g. view a videotape at night, answer questions via email in the morning).

Research studies have found that distance learning is equally or more effective than traditional classroom learning when technologies are used correctly for teaching purposes, when there is student-to-student interaction, and when feedback from teachers is given in a timely fashion. A study was conducted on several hundred undergraduate college students taking an introductory economics course at the University of California to compare the performance of online and traditional learners. [35] The distance learning course had CD-ROM based lectures, an electronic bulletin board, and an online discussion board. The study found that online learners performed as well as, or even better than, traditional learners regardless of race, gender, academic background, or computer knowledge. [35] Another study, conducted at California State University, found that students who participated in an online course had higher test scores than students who took the course in a traditional classroom setting. [3]

Another key benefit of distance learning is the variety of learning styles and teaching materials used for learning, which can offer students combinations of interaction and media. Students who prefer visual learning can focus on the

video aspect of DL, whereas students who have better listening skills can focus on the audio conferencing aspect of DL. For introverted students, DL could increase the level of interaction with other students via email or chat groups. A study found that the choice of preference for online students was individual work mainly because of their tight schedules. Many online students said they learned best by studying and researching alone; this was followed by participating in group discussions. One student commented, "Independent work gets the knowledge available, the discussions implant it." A small number indicated that their primary learning style was through group work. This group of learners can be seen as team players or socially outgoing.

The primary beneficiaries of distance learning are the nontraditional students. This group includes professionals who need to upgrade certifications or are preparing for a new career. People who have families or employment responsibilities and are unable to follow rigid course schedule on campus. Working mothers are furthering their education online and adding a "third shift" to their responsibilities, according to a study. A report by the American Association of University Women (AAUW) Educational Foundation has found that women make up the majority of students who pursue careers via distance learning. Sixty percent of these nontraditional learners are female and over 25 years of age. [26] The average distance learning student is 34 years old, employed part-time, has some college education and is female. Students who are most suited for DL are highly motivated, independent, active learners with good organizational and time management skills.

The possibilities of distance learning are innumerable. Currently, many higher education programs are working with their state to upgrade existing wide-area networks to serve primary and secondary schools and also public libraries. States are also assisting by helping to overcome geographic barriers. In Georgia and Kentucky, teachers, students, and staff members can connect to educational networks from anywhere in the state without incurring long distance charges. [28]

Disadvantages to distance learning also exist. Since DL involves access to educational resources, people's limited access to technology is an ongoing problem among certain areas in the U.S. and could potentially restrict these groups of people from participating in distance learning. Teachers also need to adjust to new learning styles to serve the needs of students. The need to mentor students from a distance can also be a challenge for both the student and the teacher. New courses offered via DL need to be accountable for following current standards of quality. [6]

The college community is currently facing issues on incorporating information technology (IT) in their programs. IT used in higher education will require decisions on how best to apply it. Society is changing whereby "students can build an additive degree program by taking courses either at different institutions or at the different campuses of one institution." [1] Sandra Weiss, former chair of the University System's Academic Council, defines this as course articulation – regarding this as important because individual universities are now becoming part of a 'global academic village'. Weiss explains that for DL courses,

"we need to identify comparable content across courses that would be acceptable for transfer and also grapple with our expectations regarding traditional 'face to face contact' between professor and student and among student themselves." [1] In the past, universities developed their curriculums independent of one another. In the case where a student is attaining a degree at one school, this would not matter. However, when a student is attaining a degree from among many schools, the schools need to have some commonality. This is known as ontological standardization – separate organizations in a given institutional area are required to uniform the most fundamental categories of their internal workings.

In the past, it has been common to transfer courses from community colleges to four-year colleges or across four-year colleges when switching majors. This is easier with online courses, where it is becoming possible for students to assemble their college education from different college programs. If proper standards are not but in place, the material students are learning across different schools are not equivalent, and intellectual diversity may be hindered. Standardization would be needed so that course content, grade, course materials can be similar across schools.

Standardization would have serious side effects. The curriculum may shift from faculty to accredited organizations or university administrators. Educational decentralizations and diversity, which have been established by geographic limitations in the US, now are threatened by the use of DL. Educators need to

weigh the benefits and consequences of standardization with that of educational diversity.

CHAPTER 3

MOVING TOWARDS DISTRIBUTED LEARNING

The systems that have been researched and analyzed in the previous chapter present a model of learning that is individual based, collaborative based, and distance learning based. Each system having advantages and disadvantages. A new model of learning has emerged which collectively brings together these models. This is distributed learning. This section describes this model of learning and also describes the solutions to existing problems in programming.

According to Syllabus Magazine in 1995, "Distributed learning is not just a new term to replace the other 'DL,' distance learning. Rather, it comes from the concept of distributed resources. Distributed learning is an instructional model that allows instructors, students, and content to be located in different, non-centralized locations so that instruction and learning occur independent of time and place. The distributed learning model can be used in combination with traditional classroom-based courses, with traditional distance learning courses, or it can be used to create wholly virtual classrooms." [40]

Distributed learning is leading to a new instructional paradigm based on the needs of learners and the electronic tools offered today. The following four models are shaping distributed learning: [9]

1. knowledge webs used to complement teachers, texts, libraries as sources of information
2. interactions in virtual communities used to complement face-to-face relationships in classrooms

3. virtual worlds used to extend learning-by-doing in real-world settings
4. sensory immersion techniques such as visualization used to help learners understand through illusion

Knowledge webs enable access to information that is distributed across the World Wide Web. Information retrieval is no longer restricted to barriers of distance, conflicting scheduling, and time. Educators and students are able to join distributed conferences that provide immediate network of contacts with expertise and responses to questions. Groupware tools are also used to enhance collaboration.

Knowledge webs provide an abundant source of information. When used in a correct manner, it can expand students' knowledge. The problem lies when this link between data and personal knowledge is missing. Raw information that is readily available does not mean that learners create a framework of ideas. In order to gain skills and master concepts, they need to generate mental models. Teachers need to facilitate this transformation. Saturating the student with information will not achieve learning. Rather, students will continue to memorize and regurgitate information and not gain the necessary skills.

Virtual communities provide a means to get to know people by exchanging ideas and experiences without the face-to-face interaction. [9] Educators can improve learning by developing new teaching methods for use with this infrastructure. Learning involves both the intellect and social aspects. Individuals who normally feel isolated because of their lack of social ability can open up and learn from others through chat groups, email or other forums. An

individual who cannot move forward with tackling a complex problem has the ability to inquire with others and share information. The possibility of developing virtual parent-teacher conferences would encourage parent involvement. Peer tutoring is another example of distributed learning used for virtual communities. Virtual interactions, enforced with GroupWare tools, can enable student-to-student relationships.

A virtual world is another feature for enhancing distributed learning. It adds to the experiences encountered in real life by providing a means of learning hands-on. Currently, the US Department of Defense is using virtual simulation to create virtual battlefields for developing military skills of new recruits. Distributed simulation can empower a broad range of educational uses. It continues to evolve based on users collaborative interactions.

Finally, sensory immersion in virtual reality is another attribute in distributed learning. It involves the user to feel as though he/she is within the synthetic environment. The following analogy can be used to describe virtual reality: a user dives into an aquarium window rather than looking into it. The use of sensory immersion is a powerful means to gain insight into abstract and intangible forms. [9] Visualization tools such as x-ray machine used by doctors and weather-detection systems used by meteorologist assist to recognize relationships that would otherwise be difficult. This needs to be carried over to the programming world to simplify abstract models and deepen learners' motivation.

3.1 Difficulties of Distributed Learning

Research has found that it is difficult to work effectively when confidence in team members' participation is lacking. This results when members do not get to know one another. This problem is escalated further when teams are distributed. Many times what ends up happening is that the team is not working as one but rather as a web of individuals. The figure below depicts the decrease in the ability for members of a distributed group to divide and execute their work over time. [51]

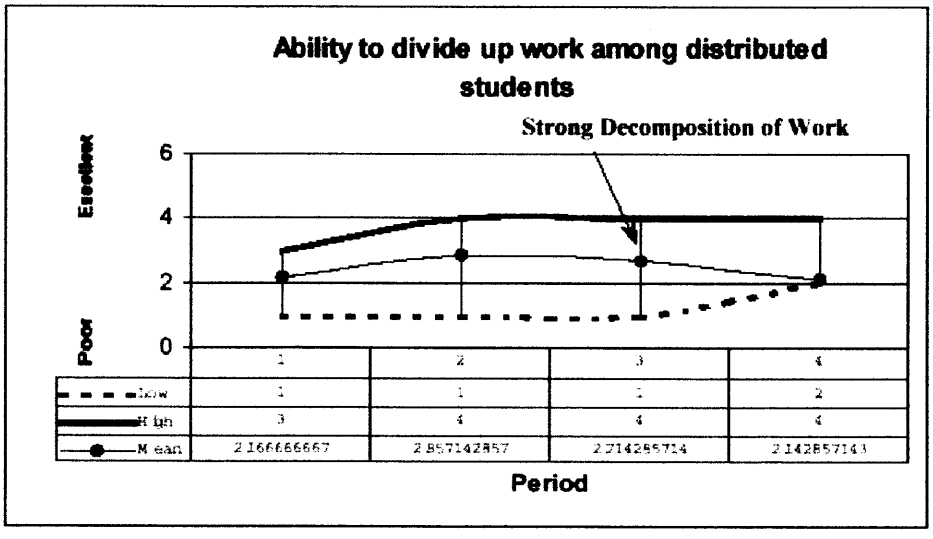

Figure 3.1 Distribute Work within Group.

Current tools are designed to support certain interactive communication such as conversations, disagreements, and graphs. The problem lies that these technologies are very narrow in the scope and so do not perform well in interactions not related to that scope. Achieving a task requires different types of interactions and therefore requires an environment to adjust to the needs of the constantly changing tasks.

As with collaborative learning, distributed learning does not guarantee learning among groups. It is based on the level of group interactions that trigger learning mechanisms. In order to increase the rate that interactions occur, careful design of the initial conditions is needed. This includes identifying the optimal group size, group grading policy, level of development experience among individuals, etc. Due to varying forms of interactions, it is difficult to set up conditions that guarantee the effectiveness of learning.

The instructor also serves a role in distributed learning. He/she is responsible for redirecting group work in the productive way and monitor members' activities. The goal is not for the teacher to provide the correct answers, but rather to intervene and provide guidance as to the direction students should take. Tools are currently being devised to manage group collaborative interactions.

Technology-based modes of learning, such as distributed learning, provide many advantages when technology is used appropriately to develop skills and maximize students' learning. Alternatively, if goals are shifted away from the student, results can be disastrous. For example, technology if misused to minimize costs, could lead to classes where lessons, notes, and assignments could be published on the web, video could replace lectures, and teaching assistants could answer students' questions. In essence, courses could become completely automated and not provide students with the proper skills.

3.2 Collaborative Tools used in Distributed Learning

An important factor in the effectiveness of a distributed team is the tools used to communicate between team members. These channels are necessary to facilitate people interacting at a distance and with time constraints.

The following diagram describes the collaborative tools used to communicate by a distributed team. [51] Based on the type of interaction needed, appropriate mechanisms are used. For example, if a student wants a written record of the information that is transmitted and does not necessitate immediate responses, a thread discussion would be the choice of interaction.

Distributed Interaction Mechanisms Mapping

- E-Mail
  - Asynchronous/Unstructured/Intentional/Non-committal
- Threaded Discussion
  - Asynchronous/Structured/Intentional/Non-committal
- Web Repository
  - Asynchronous/Structured/Intentional/Non-committal
- Channel (Push Technology)
  - Asynchronous/Structured/Intentional/Non-committal
- Buddy List
  - Asynchronous/Structured/Unintentional/Non-committal
- Chat
  - Synchronous/Unstructured/Intentional/Medium-commitment
- Audio/Video Conferencing
  - Synchronous/Unstructured/Intentional/Medium-commitment

Figure 3.2 Communication Mechanisms for a Distributed Team

The following figure provides group interaction activities used to maximize group effectiveness. [51] It is possible for members not to benefit from groups if each individual works independently. Each member may develop skills by

applying various interaction activities throughout the duration of the project (i.e. participation of group discussions and brainstorming sessions).

Group Interaction Activities

- Information Dissemination
  - Course Handouts, Readings, and Video Materials
- Design/Knowledge Building
  - Lectures, Group Discussions, and Brainstorming Sessions
- Group Co-ordination
  - Meeting Notifications, Agreements, and Responsibilities
- Group Cohesion
  - Social interaction to Create Shared Understanding
- Decision Making
  - Mechanism to Reach a Shared Vision, Goal or Direction

Figure 3.3 Interaction Activities of a Distributed Team

Interaction modes are used to characterize collaborative tools. Depending on the stage of the project, the mode of interaction changes. In the beginning of a project, frequent face-to-face meetings are necessary. As the project progresses, members can interact asynchronously via email or discussion groups. It is important that the technology matches the diverse dimensions of interaction.

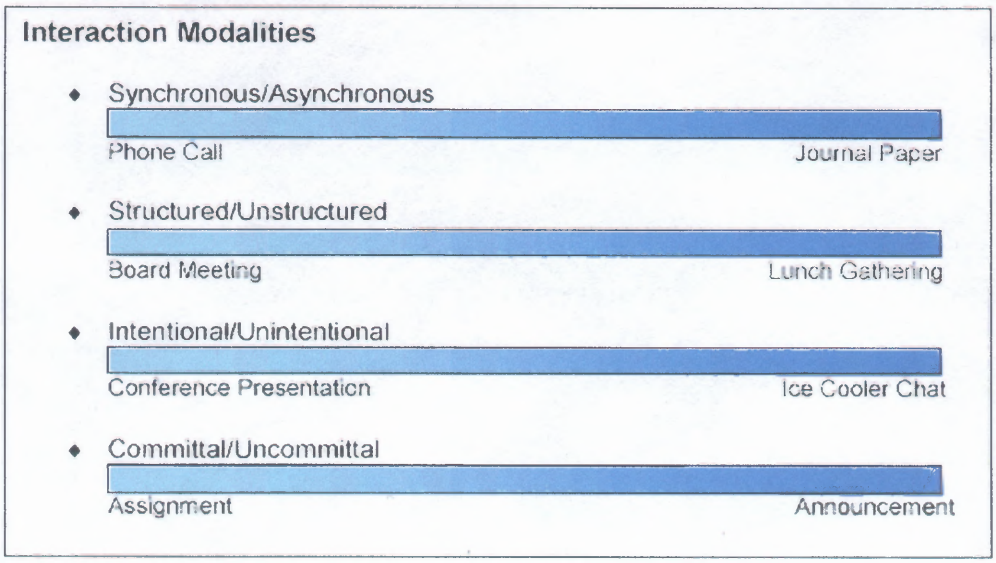

Figure 3.4 Interaction Modes of a Distributed Team

Virtual environments today are developed from the notion of physical world environments. Examples include a conversation, a meeting, a presentation, a collection of information, etc. The success of these virtual environments depends on how well it facilitates and supports individuals in completing their tasks. The environment needs to adapt to different situations based on individual's work style, background, social and organizational roles.

The environment includes the physical and virtual space, the tools, and support structure necessary to learn. A setting is necessary for information to be processed individually and collectively. In the physical realm, the challenge is designing such a system that balances flexibility of work and adaptability of individual work styles with structured interactions. Flexibility, adaptability, and structure are achieved by the services offered by the system, by the different levels of interaction, and by the different modes of interaction relative to time and

space. In the virtual realm, this same challenge continues to exist. Software applications, conferencing, messaging, and Web technology are actually modeled based on physical space.

In distributed learning, collaborative systems are used. They have been classified into four groups: session-centric or meeting-centric, document-centric, place-based and hybrid systems. [24] Session-centric systems are generally not persistent. They are based on a model that structures today's casual conversion or presentation. These tools support synchronous collaboration. When the session is over, there is no documentation saved on the collaboration. Information is lost for someone who may join the interaction later. Based on this model, factors such as interaction, floor control, and response time are taken into consideration. Some known applications that have integrated synchronous, meeting-focused tools for audio, video, and data conferencing and follow the model described above are Microsoft NetMeeting [30], desktop video teleconferencing, PlaceWare Auditorium [39], and MIT's CAIRO [38].

Document-centric systems are persistent. They model real-life information retrieval systems. It supports primarily asynchronous communication (with some tools for synchronous communication). Information is saved for future retrieval, but they fail to support real-time collaboration. Examples of such applications are messaging systems like electronic mail, bulletin board systems, Lotus Notes, and online forums.

The third category is place-based systems. The following properties are taken from physical space and are used to distinguish place-based systems from

other systems. They are persistent. Messages and information remain between logged sessions regardless of someone being present. They support a degree of peripheral vision, whereby a user has the ability or control to detect the presence of others. In real life, people who work together have a sense of awareness as to their team members' activities. In virtual space, a user can filter this out, yet monitor activities occurring in his/her surroundings. Another property is state. Stateful means that a user has the ability to modify the state of the environment (e.g. modify, add, remove a document in a room or interact with others).

Place-based systems provide a familiar environment where users can behave in a natural way with gestures when they communicate. This characteristic is known as behavioral framing. They support both synchronous and asynchronous communications. They are location independent, which means they can be accessed independent of one's location. They are location transparent, which means that interactions can occur without knowing anyone's physical location. These characteristics are actually an integrated framework of both session-centric and document-centric systems. Examples of such systems include MOO, MITRE's CVW, and Jupiter XeroxPARC.

Hybrid systems combine both the physical and virtual realm. Essentially, both physical and virtual space is intertwined. Something that was developed in a virtual space is projected onto physical and vice versa. Examples of this are Jupiter XeroxPARC and MITRE's CVW. These systems are assembled from existing tools such as MUDs, email, and desktop videoconferencing. In the case of place-based systems, MUD clients began to extend the command line

interface to GUI based interface. Other services combined the physical context (i.e. shared video and audio conferencing) with virtual context (i.e. representation of people, objects, and places in space).

In the past, tools were developed in modular components, satisfying one type of communication or one category (e.g. synchronous vs. asynchronous). The problem with this is the design and development of individual capabilities rather than an integrated toolset. This is necessary to satisfy the evolution of an end-to-end solution. As people are working through the various stages of software development cycle, the team evolves (changes in roles within a group) and the tools used change as well. Moving information between tools is time consuming and inefficient as individuals need to learn and transfer data across tools.

A Collaborative Virtual Workspace (CVW) is an integrated, object-oriented framework that supports multiple users. It can be considered a building with floors and rooms. People exchange information and interact with each other in a shared virtual space, using both synchronous and asynchronous tools.

It is designed to support dispersed team members by providing services for document sharing, audio, video and textual communications. The following are some of the services it offers: creates and maintains groups; retrieves documents, users, and rooms; notifies users of events occurring in environments. It also controls documents and provides locking rooms capabilities. The virtual workspace is persistent with people and documents that potentially existing in rooms, floors, and buildings. People can interact in rooms via audio or chat

boxes. Communication sessions (e.g. audio, video, and text) are established automatically regardless of others having the capability of being involved in session. Users are able to communicate privately by locking rooms. Users can add/remove documents of different types in rooms. Persistence is achieved through a database that stores documents for each room in virtual workspace. Only an authorized person has capability to delete documents. A check-in and checkout method is used to enforce modification of a document one user at a time. Users editing documents are also tracked. Freeware is used to provide the basic applications, such as audio, video, chat rooms, and whiteboard. This is done so that all users can access applications.

3.3 Programming in Distributed learning

Programming using distributed learning model serves as microcosm of the future of learning whereby learning can take place both inside and outside of classrooms. The boundaries have been removed and interactions with people from different locations and different times are possible. The use of distributed learning in computer classrooms provides a new direction for teaching. It resolves the many difficulties that have been observed by novice programmers in learning software development. Below is a summary of difficulties encountered by students and the provisions needed along with a distributed learning environment to resolve these issues.

1. Lack a mental model to visualize abstract relationships

2. Instructors in traditional learning hold an important role in transferring knowledge. If instructor is lacking social skills necessary for interacting with students, technical ability, or effective speaking, an impediment for learning results
3. Students fall behind in classrooms and miss fundamental concepts.
4. Lack of motivation
5. Limited knowledge in a scientific area which may result in difficulty in interacting and following complex discussions
6. Students passivity in class
7. Small programming assignments which tend to lead to individualistic work habits
8. Deficiencies with traditional teaching methods, esp. lectured-based teaching
9. Emphasis on competition for grades rather than learning
10. Programming language paradigms do not associate with natural tendencies of problem solving.

The difficulty of mentally representing abstract concepts is overcome by the use of visual modeling tools. Sensory immersion devices assist in visualizing abstract concepts. These devices have been successful in areas of science and medicine. Once this is extended to the computer world, novices will be better able to model abstractions.

Distributed learning provides alternate means of learning than traditional teaching methods. By allowing students to engage in collaborative learning

exercises and to use visual modeling tools, they can spread their depth of learning. The goal is to provide a more enriched curriculum that can guide students to achieve maximum learning potential. Interacting with students could enhance cognitive skills and complement individual cognition.

Distributed learning offers a solution to the third difficulty described whereby students fall behind in class. Virtual classrooms can provide additive means of learning, such as discussion forums, so in the event that a student falls behind, other sources of information are readily available to bring students up to speed on misunderstood materials. For students who prefer learning at their own pace, distance learning course can facilitate this.

Learning in groups stimulates members to achieve maximum potential and increase motivation levels. In addition to individual cognition, groups provide a balance for optimal learning when members interact with each other by explaining and exchanging ideas. Group work, when done correctly, can result in team members working efficiently together. This could minimize levels of competition by providing a better environment focused on group learning.

The fifth difficulty encountered by novice programmers is nearly eliminated with the advent of the Internet. Information becomes available at one's footsteps. In addition to existing sources, such as books and journals, distributing learning in the classroom offers a collective way of transferring this raw information into knowledge. On a different note, individuals who shy away from face-to-face discussions due to their limited knowledge in an area may openly ask questions

in an unthreatening environment such as online forums, threaded discussion, or email.

Distributed learning offers diverse tools targeted towards different learning types (i.e. sensory, inductive, active learners). By offering sensory immersion devices, sensory, visual, and intuitive learners could achieve a higher level of learning. Verbal learners continue learning based on reading materials and written communication with other students. Nevertheless, instructors need to continue to provide diversified teaching methods to reduce the passivity while teaching in traditional classrooms and virtual classrooms.

The seventh problem affecting novice programmers is not resolved with distributed learning. Revisions need to be made to CS curriculum to enhance programming skills for today's fast paced market. Enhancements include offering team-oriented assignments focused on the software development life cycle process and decomposing complex problems into subcomponents. These collaborative work assignments should be enforced and represent real-world work projects.

Distributed learning resolves the many problems faced with traditional teaching methods, esp. lectured based approach. Distributed learning offers an array of tools for these learners. This includes sensory immersion devices that facilitate visualizing abstract concepts. Distributed learning largely benefits active learners; they prefer group discussions and exchanging ideas. This is achieved via email, chat rooms, instant messaging, and through the use of virtual

classrooms. The same technologies are also effective for the reflective learner who first thinks through a concept and then interacts with others.

One problem that continues to exist in programming that distributed learning does not resolve is programming language design. Languages do not associate with natural tendencies of problem solving. Therefore, novices will continue to take longer to understand the fundamentals of computer internals.

CHAPTER 4

FUTURE WORK

Distributed learning has great potential to be the wave of the future in academic development. Learning without boundaries to time and distance. It offers a wider spectrum of instructors, advisors, and collaborators than in any single educational institution. It provides a means of attaining skills from remote sources and collaborating with dispersed team members. Technology provides the essential tools for establishing interactions, but these tools, if misused, could lead to adverse effects. A balance is necessary between virtual and direct communication within group members in order to develop and sustain a sense of community. Interactions based on phone conversations alone lack the vibrancy of face-to-face. Also, while technology-mediated communication (i.e. teleconferencing, digital and video) will open the doors of virtual interactions over various information mediums, it will not completely replace personal contact. In order to optimize the use of the distributed learning model, provisions must be made and new inventions devised which provide the best of both worlds for learning.

The future of distributed learning is via collaborative virtual workspaces and sensory immersion devices. [42] Further studies are necessary to measure the performance of dispersed team members using CVW. More research is also needed in the area of sensory immersion to study the impacts of these devices in the (virtual) classroom. High performance computing will gradually enable virtual

communities to make possible face-to-face interactions and sensory immersion to be an integrated part of everyday life.

Collaborative learning, in theory, allows students to stimulate each other in the learning process. Provisions are needed to ensure that maximum learning is achieved and that theory turns to practice. A quality audit of the curriculum (i.e. in Computer Science) is necessary to evaluate existing practices and reform existing standards. The provisions made forth should be based on evaluations of the role of the instructor, group formation, course grading, and teaching practices, among other areas of a curriculum.

REFERENCES

1. P. E. Agre, "Information Technology in Higher Education: The 'Global Academic Village' and Intellectual Standardization," *On The Horizon7* vol. 5, pp. 8-11, 1999.

2. W. B. Barbe and M. N. Milone, "What We Know About Modality Strengths," *Educational Leadership*, vol. 38, pp. 378-380, February 1981.

3. A. Barron, "A Teacher's Guide To Distance Learning," Florida Center for Instructional Technology, 1998. Retrieved November 9, 2001 from the World Wide Web: http://fcit.coedu.usf.edu/distance/default.htm

4. A. F. Blackwell, "Metaphor or Analogy: How Should We See Programming Abstractions?" *Proceedings of the 8th Annual Workshop of the Psychology of Programming Interest Group,* pp. 105-113, April 1996.

5. T. Bourner and S. Flowers, "Teaching and Learning Methods in Higher Education: A Glimpse of the Future," *Reflections on Higher Education (A Journal of the Higher Education Foundation)*, vol. 9, pp. 77-102, 1997.

6. T. Collins and S. Dewees, "Distance Education: Taking classes to the students," *The Rural South: Preparing for the Challenges of the 21st Century*, no. 17, Feb. 2001. Retrieved October 5, 2001 from the World Wide Web: http://srdc.msstate.edu/publications/distance_education.pdf

7. J. Cuseo, "Collaborative & cooperative learning in higher education: A proposed taxonomy," *Cooperative Learning and College Teaching*, vol. 2, pp. 2-5, 1992.

8. A. Cypher, *Watch What I Do: Programming by Demonstration*, MIT Press, Cambridge, Massachusetts, 1993.

9. C. Dede, "Distance Learning to Distribute Learning: Making the Transition," *Learning & Leading with Technology* ISTE (International Society for Technology in Education), vol. 23, no. 7, pp. 25-30, 1996.

10. F. P. Deek, "A Framework for an Automated Problem Solving and Program Development Environment," *Journal of Integrated Design and Process Science*, vol. 3, no. 3, pp. 1-13, December 1999.

11. F. P. Deek, K. Ho, and H. Ramadhan, "A Critical Analysis and Evaluation of Web-Based Environments for Program Development," *Journal of Internet and Higher Education*, vol. 3, no. 4, pp. 223-269, August 2001.

12. F. P. Deek, H. Kimmel, and J. McHugh, "Pedagogical Changes in the Delivery of the First Course in Computer Science: Problem Solving Then Programming," *Journal of Engineering Education*, vol. 87, no. 3, pp. 313-320, July 1998.

13. F. P. Deek and J. McHugh, "Problem Solving and the Development of Critical Thinking Skills," *Journal of Computer Science Education – ISTE SIGCS*, vol. 14, no. 1/2, pp. 6-12, April 2000.

14. F. P. Deek and J. McHugh, "Prototype Software Development Tools for Beginning Programming," *Journal of Computer Science Education – ISTE SIGCS*, vol. 14, no 3/4, pp. 14-20, April 2001.

15. F. P. Deek and J. McHugh, "SOLVEIT: An Experimental Environment for Problem Solving and Program Development," *Journal of Applied Systems Studies*, vol. 2, no. 2, 2001.

16. F. P. Deek and J. McHugh, "A Survey and Critical Analysis of Tools for Learning Programming," *Journal of Computer Science Education*, vol. 8, no. 2, pp. 130-178, August 1998.

17. F. P. Deek, J. McHugh, and S.R. Hiltz, "Methodology and Technology for Learning Programming," *Journal of Systems and Information Technology,* vol. 4, no. 1, pp. 25-37, June-July 2000.

18. F. P. Deek, M. Turoff, and J. McHugh, "A Common Model for Problem Solving and Program Development," *Journal of the IEEE Transactions on Education*, vol. 42, no. 4, pp. 331-336, November 1999.

19. P. Dillenbourg, *Collaborative-learning: Cognitive and Computational Approaches,* Pergamon Press, Oxford, England, 1999.

20. P. Dillenbourg, M. Baker, A. Blaye, and C. O'Malley, "The evolution of research on collaborative learning," In E. Spada & P. Reiman (Eds.) *Learning in Humans and Machine: Towards an interdisciplinary learning science*. Oxford, England, 1995.

21. D. Du Boulay, T. O'Shea, and J. Monk, "The black box inside the glass box: Presenting computing concepts to novices," *International Journal of Man-Machine Studies,* vol. 1, pp. 133-161, 1981.

22. K. Ehrlich and E. Soloway, "An empirical investigation of the tacit knowledge in programming," *Human Factors in Computing,* Norwood NJ, pp. 113-133, 1985.

23. R. Felder, "Reaching the Second Tier: Learning and Teaching Styles in College Science Education," *Journal of College Science Teaching,* vol. 23, no. 5, pp. 286-290, 1993.

24. T. Gallemore, "What is Space? (In Collaborative Virtual WorkSpace)," MIT-SPORG Research Group and the MITRE Corporation of Bedford MA, September 21, 1998. Retrieved November 9, 2001 from the World Wide Web: http://www.designformation.com/estudio/What_Use_Is_Space.pdf

25. T. R. Green and M. Petre, "Usability analysis of visual programming environments: a 'cognitive dimensions' framework," *Journal of Visual Languages and Computing*, vol. 7, pp. 131-174, 1996.

26. C. Kramarae, *The Third Shift: Women Learning Online*, AAUW Publications, Washington D.C., 2001.

27. B. W. Liffick and R. Aiken, "A Novice Programmer's Support Environment," *Proceedings of the SIGCSE/SIGCUE Conference on Integrating Technology into Computer Science Education*, Barcelona, Spain, June 1996. Retrieved November 9, 2001 from the World Wide Web: http://cs.millersv.edu/~liffick/sigcse/sigcse.html

28. J. B. Mathews, "Statewide Educational Networking: Trends and Issues Highlighted," Southern Regional Education Board, Atlanta, GA, April, 1998. Retrieved October 5, 2001 from the World Wide Web: http://www.sreb.org/programs/EdTech/pubs/98WAN.asp

29. W. J. McKeachie, *Teaching Tips: A Guidebook for the Beginning College Teacher*, 8th Edition, Lexington, Massachusetts, 1996.

30. Microsoft Windows Technology: Windows NetMeeting, 2001. Retrieved November 9, 2001 from the World Wide Web: http://www.microsoft.com/netmeeting

31. M. Minsky, *The Society of Mind*, William Heinemann Ltd., London, England, 1987.

32. T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli, "Explanation-Based Generalization: A Unifying View," *Machine Learning*, vol. 1, no. 1, pp. 47-80, 1986.

33. B. A. Myers, Natural Programming: Project Overview and Proposal, January 1998. Retrieved September 21, 2001 from the World Wide Web: http://www.cs.cmu.edu/~NatuProg/projectoverview.html

34. B. A. Myers, Usability Issues in Programming Languages, 1999. Retrieved September 21, 2001 from the World Wide Web: http://www-2.cs.cmu.edu/~NatProg/langeval.html

35. P. Navarro and J. Shoemaker, "Performance and Perceptions of Distance Learners in Cyberspace," *The American Journal of Distance Education*, vol. 14, no. 2, pp. 15-35, August 2000.

36. S. R. Oliver and J. Dalbey, "A Software Development Process Laboratory for CS1 and CS2," *Proceedings of the SIGSCE,* Phoenix, Arizona, vol. 26, no. 1, pp. 169-173, March 1994.

37. J. F. Pane, C. A. Ratanamahatana, and B. A. Myers, "Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems," *International Journal of Human-Computer Studies*, vol. 54, no. 2, pp. 237-264, February 2001.

38. F. Peña-Mora, K. Hussein, S. Vadhavkar, and K. Benjamin, "CAIRO: A Concurrent Engineering Environment for Virtual Design Teams," *Artificial Intelligence in Engineering*, vol. 14, no. 3, pp. 203-219, 2000.

39. PlaceWare WebConferencing, 2001. Retrieved November 9, 2001 from the World Wide Web: http://www.placeware.com

40. S. Saltzberg and S. Polyson, Distributed learning on the World Wide Web. Syllabus, September 1995. Retrieved December 8, 2001 from the World Wide Web: http://www.syllabus.com/archive/Syll95/07_sept95/DistrLrngWWWeb.txt

41. J. Salvage, *The C++ Coach, Essentials for Introductory Programming*, Addison Wesley, New York, 2000.

42. M. C. Salzman, C. Dede, and B. Loftin, "ScienceSpace: Virtual realities for learning complex and abstract scientific concepts," *Proceedings of IEEE Virtual Reality Annual International Symposium*, New York, pp. 246-253, 1996.

43. L. K. Silverman and R. M. Felder, "Learning and Teaching Styles in Engineering Education," *Engineering Education*, vol. 78, no. 7, pp. 674-681, April 1988.

44. B. L. Smith and J. T. MacGregor, What is collaborative learning? In A. S. Goodsell, M. R. Maher, B. L. Smith, and J. MacGregor, (Eds), *Collaborative Learning: A Sourcebook for Higher Education*, University Park, PA, 1992.

45. P. J. Spellman, J. N. Mosier, et al., Collaborative Virtual Workspace, 1997. Retrieved December 8, 2001 from the World Wide Web: http://www.mitre.org

46. J. C. Spencer, E. Soloway, and E. Pope, "Where the bugs are," SIGCHI Proceedings of the CHI '85 Conference on Human Factors in Computing Systems, San Francisco, CA. pp. 47-53, 1985.

47. D. Suthers and D. Jones, "An Architecture for Intelligent Collaborative Educational Systems," *Proceedings of AIED '97, 8th World Conference on Artificial Intelligence in Education,* Kobe, Japan, pp. 55-62, August 1997.

48. D. Suthers and A. Weiner, "Groupware for Developing Critical Discussion Skills," *Computer Supported Cooperative Learning*, October 1995.

49. D. L. Swartz, "The Emergence of Abstract Dyad Representations in Dyad Problem Solving," *The Journal of the Learning Sciences*, vol. 4 no. 3, pp. 321-354, 1995.

50. G. H. Turnwald, K.S. Bull, and D.C. Seeler, "From Teaching to Learning: Part II. Traditional Teaching Methodology," *Journal of Veterinary Medical Education*, vol. 20, no. 3, pp. 148-156, 1993.

51. S. Vadhavkar and F. Peña-Mora, "Geographically Distributed Team Interaction Space", *Proceedings of the Eighth International Conference (ICCCBE - VIII)*, vol. 1, pp. 373-379, August 2000.

52. K. Vanlehn, R. M. Jones, and M. T. H Chi, "A Model of the Self-Explanation Effect," *Journal of The Learning Sciences*, vol. 1, pp. 1-59, 1992.