University of Central Florida
STARS

Electronic Theses and Dissertations

2007

Eino: An Intelligent Tutor For The University Of Central Florida Infinity Web Applets

James Hollister
University of Central Florida

 Part of the Computer Engineering Commons
Find similar works at: https://stars.library.ucf.edu/etd
University of Central Florida Libraries http://library.ucf.edu

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation
Hollister, James, "Eino: An Intelligent Tutor For The University Of Central Florida Infinity Web Applets" (2007). *Electronic Theses and Dissertations*. 3206.
https://stars.library.ucf.edu/etd/3206

EINO: AN INTELLIGENT TUTOR FOR THE UNIVERSITY OF CENTRAL FLORIDA INFINITY WEB APPLETS

by

JAMES HOLLISTER
B.S. University of Central Florida, 2005

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2007

© 2007 James Hollister

ABSTRACT

This study investigated the various methods involved in creating an intelligent tutor for the University of Central Florida Infinity Web Applets (UCF Infinity Web Applets). After conducting research into various methods, two major methods emerged and they are: solving the problem for the student and helping the student when they become stymied and unable to solve the problem. A storyboard was created to show the interactions of the student and system along with a list of features that were desired to be included in the tutoring system. From the storyboard and list of features, an architecture was created to handle all of the interactions and features. After the initial architecture was designed, the development of the actual system was started. The architecture underwent a multitude of changes to conclude with a working system, EINO. The final architecture of EINO incorporated a case based reasoning system to perform pattern recognition on the student's input into the UCF Infinity Web Applets. The interface that the student interacts with was created using flash. EINO was implemented in three of the labs from the UCF Infinity Web Applets. A series of tests were performed on the EINO tutoring system to prove that the system could actually perform each and every one of the features listed initially. The final test was a simulation of how the EINO would perform under a set of given cases. Test subjects with the same educational level as the target group were chosen to spend an unlimited time using each of the three labs. Each of the test subjects filled out a survey on every lab to determine if the EINO system produced a helpful output.

Dedicated to my parents for their constant love and support.

TABLE OF CONTENTS

LIST OF FIGURES vii
LIST OF TABLES viii
CHAPTER 1: INTRODUCTION AND BACKGROUND 1
1.1 Intelligent Tutors 2
1.1.1 ANDES 2
1.1.2 Quantum 2
1.2 The Infinity Project 3
1.3 EINO 4
1.4 Summary 5
CHAPTER 2: LITERATURE REVIEW 6
2.1 Procedural Help in ANDES 6
2.2 Other Tutoring Systems 8
2.2.1 ANDES Lessons Learned 8
2.2.2 QUANTUM Lessons Learned 14
2.3 Conversational Case Based Reasoning 15
2.4 Case Based Reasoning Tutor 17
2.5 Data Mining Tutor 18
2.6 Storyboarding 19
2.7 A Multi Agent Architecture 20
2.8 Summary 21
CHAPTER 3: PROBLEM DEFINITION AND HYPOTHESIS 23
3.1 General Problem Statements 23
3.2 Specific Problem Statements 24
3.3 Hypothesis 26
3.4 Contributions 26
CHAPTER 4: THE APPROACH 27
4.1 The Development Environment 27
4.2 EINO General Function 28
4.3 Goals and Cases 34
4.4 EINO Initiation 36
4.5 Class Diagram 38
4.6 Summary 40
CHAPTER 5: IMPLEMENTATION 41
5.1 UCF Infinity Web Applets Source Code 41
5.2 Step One of Implementation EINO Tutoring System 43
5.3 Step Two of Implementation EINO Tutoring System 45
5.4 Step Third of Implementation EINO Tutoring System 46
5.5 Step Fourth of Implementation EINO Tutoring System 47
5.6 Summary 48
CHAPTER 6: TESTING 50
6.1 The Not On Test 51

6.2 The Single Goal Test 52
6.3 The Group Goal Test 53
6.4 The Just in Time Hint Test 54
6.5 The Case Test 55
6.6 The Questions Test 56
6.7 The Real Life Simulation 59
6.8 Summary 62
CHAPTER 7: SUMMARY, CONCLUSION & FUTURE WORK 64
7.1 Summary 64
7.2 Conclusion 66
7.3 Future Work 67
REFERENCES 69

LIST OF FIGURES

Figure 1: ANDES Interface 10
Figure 2: EINO Block Diagram 29
Figure 3: Problem with Volume Slider 32
Figure 4: Off and On Buttons 33
Figure 5: The Lab is Not On 34
Figure 6: Try Adjusting 35
Figure 7: UCF Infinity Web Applets Class Diagram 37
Figure 8: EINO Class Diagram 39
Figure 9: Pixel Mapper 42
Figure 10: The General Help Button 57
Figure 11: Example General Help Menu 57
Figure 12: A Sample Question 58
Figure 13: The Survey 60

LIST OF TABLES

Table 1: Results of the Not On Test..... 51
Table 2: Results of the Single Goal Test 52
Table 3: Results of the Group Goal Test 53
Table 4: Results of the Just in Time Hint Test..... 55
Table 5: Results of the Case Test..... 56
Table 6: Results of the Questions Test 59
Table 7: Overall Results of Helpfulness of EINO 61
Table 8: Helpfulness Score for Lab 2-1..... 62
Table 9: Helpfulness Score for Lab 4-2-4-1 62
Table 10: Helpfulness Score for Lab 6-2-1..... 62

CHAPTER 1: INTRODUCTION AND BACKGROUND

As technology in the classroom changes and expands, the way one interacts with technology must also undergo change. As our tools have become more sophisticated, we as a people have adapted our tools, we have gone from writing implements such as stones and chisel to paper and pens to today's technological marvel–the computer. This progression of tools has enabled our students to complement their education by using the computer. Homework assignments are no longer expected to be mundane repetitive work. Students today can and do have the capability of completing labs and simulations from home. In a classroom setting, a student with a problem always has the teacher available to answer a question or to make corrections so that he/she can get back to the right track. There is no place for the student to go if, while they are working at home, have a question or problem that arises. This is the area where an intelligent tutor can be useful. It provides the student a place to turn to for help, when they become confused, as they try to complete a lab assignment. Having a place to turn, when in trouble will encourage more students to try a lab or other assignments at home. This will potentially raise a student's interest in learning. They will feel more confident tackling problems at home, as well as at school, and we can expectedly to see a rise in the educational level of the students.

1.1 Intelligent Tutors

1.1.1 ANDES

A well known intelligent tutor in operation today is ANDES. When first released, ANDES was only a physics intelligence tutor. The design architecture has since been applied to other mathematical fields such as geometry and physics. The core of the ANDES system is a Bayesian network used to predict how the student will solve the presented physics problem. The output of the Bayesian network is called a "solution graph."[1] A mix of rules and context-based reasoning is used to make ANDES act intelligently. In order to use the ANDES system, the student must learn how to operate with the system. The student must first draw all of the force vectors for the given problem. After drawing the force vectors, the student must enter a given series of equations that may be used to solve the given problem. If the student enters a wrong vector or equation, that vector or equation turns red. At any point in the process, the student can ask ANDES for help in solving the given problem.

1.1.2 Quantum

Another system, the Quantum tutor takes an entirely different approach than the well-known ANDES system. In the Quantum system, the student provides the question and the tutor will solve the problem. This process means that there are an endless number of questions that the Quantum system could potentially solve. The focus of the Quantum system has been in the field of mathematics and chemistry, primarily focusing on the balancing of chemical equations. The Quantum system utilizes a mix of forward and backward reasoning in conjunction with rule-based reasoning in order to solve each equation that the student may present to the system[2].

The first step in using the system is to enter the equation. The Quantum system converts the equation into chemical symbols. Afterwards, the Quantum system solves the equation in a step-by-step manner as the student observes. The student is able to ask questions about the procedure and theory of solving the presented problem.

These competing intelligent tutors show two different ways of analyzing problems and drawing conclusions. This in turn may demonstrate different problem-solving strategies to students who may then utilize more complex problem-solving behaviors on their own when presented with difficult problems. These two tutoring systems also demonstrate that there may be multiple approaches to tutoring a student. The first approach, used by the ANDES system, is to let the student try to solve the problem by themselves and then help them when the student makes a mistake or becomes confused. This approach applies the philosophy that one learns best by performing the actual action of solving the problem. This approach works well if the student has an idea on how and where to start solving the problem. The second approach, used by Quantum system, is to show the student how to solve the problem from the onset of receiving the problem. This approach works well if the student can not begin solving the problem. This approach assumes the student has attempted to solve the same problem by himself or herself and is only using the tutor when they are at a complete loss in the solving process.

1.2 The Infinity Project

The Infinity Project was created as an attempt to get high school students more interested in engineering studies, more specifically Computer and Electrical Engineering[3]. A few professors at Southern Methodist University (SMU) noticed a drop in the number of engineering students and created a year long class that high schools could offer to their students. The class,

the Infinity Project, was comprised of a textbook and a lab manual that the professors developed and produced. In to order complete all of the labs in the Infinity Lab Manual, the school must also purchase a lab kit, that includes a circuit board with the different components for each lab, along with speakers, and computer software that was used to control the circuit board[3]. In 2005, the University of Central Florida (UCF) became enthusiastic about the project. To support the project and potential university students, UCF hired a few graduate students to create a number of the lab experiments in a web format. This was done so that the students had the option of spending more time engaged in the activities that went beyond what the teacher had done in class with the lab kits on the experiments. An idea that was generated while the graduate students were working on the web experiments was the addition of a built-in web tutor to help the students if they become stuck or had questions regarding the experiments.

1.3 EINO

This was when "EINO", pronounced "I Know", was created. EINO is the intelligent tutor built into the web experiments created by UCF. EINO is able to assist students when they become perplexed or have further questions about the experiments being conducted. In conjunction with a student being able to call on the tutor, EINO follows the student through the experiment, offering "just in time hints" to students to help them complete the lab simulation. When a student has trouble comprehending or understanding a major topic, EINO explains the information to the student and follows up by asking the student a few questions to ensure that the student understands the information and material. EINO allows the student to complete the experiment in any order that the student sees fit, as long as the key points of the experiment are

met. This also indicates that EINO is keeping a record of which key points in which experiment the student has accomplished.

1.4 Summary

Within this paper, there is a discussion of the research and creation of the intelligent tutor, EINO. Chapter Two includes a review of some earlier research papers involving intelligent tutors and the results that the development team produced at the University of Pittsburgh while working on the ANDES intelligent tutor[4]. In Chapter Three, there is a discussion of the contemporary problem that led to the development of EINO and an explanation as to why EINO is the solution to this problem. Chapter Four covers the high level design of EINO. Chapter Five presents the actual implementation of EINO and discusses how it was implemented. In Chapter Six, the results of the implementation of EINO are discussed. Chapter Seven is the conclusion and future potential upgrades to EINO are suggested.

CHAPTER 2: LITERATURE REVIEW

2.1 Procedural Help in ANDES

The ANDES tutoring system has been continuously under revision since its development, and individuals have used it for a number of years with a great deal of success[4]. The first paper that is of interest is titled “Procedural help in Andes: Generating hints using a Bayesian network student model” by Abigail S. Gertner and Cristina Conati and Kurt VanLehn (1998)[1]. This manuscript describes how the ANDES tutoring system determines how to respond when the student/user requests help. As stated above, the ANDES system uses a Bayesian network to create a student model. The student model consists of information containing the student’s general knowledge, the student’s specific knowledge, and the student’s problem-solving approach to the current problem. The student model also contains what help the student has received in the past and what problems the student has already completed. The student model is prepared from multiple components. Two of the most important components are the “solution graph” and the Assessor. Without these two mechanisms, the student model and the hint system would not function properly.

A key factor in providing hints for the students in the ANDES system is the “solution graph.” A “solution graph” is generated by the problem solver and contains all of the rules, facts, strategies, and goals needed to solve the presented problem. The problem solver is a rule-based system that processes a single rule each time through a loop until all rules that apply to the problem have been processed. When this process is completed, the results are in an ordered

"solution graph." Naturally, the problem solver is only run once with the given problem as the "solution graph" for a given problem will not change.

The Assessor is a Bayesian network and is recreated from the "solution graph" every time a student selects a problem. This means the basic structure of the Assessor Bayesian network is the same as the "solution graph." The difference is that the Assessor has the probabilities that the student knows the given rule, fact, strategy, or goal. There are five kinds of nodes used in the Assessor. The five nodes are: Rule, Fact, Goal, Rule Application, and Strategy. The Rule nodes are used to determine if the student has the ability to apply the corresponding rule. The Fact nodes are used to determine the probability that the student has the knowledge of the corresponding fact. The Goal nodes are used to determine the probability that the student has been working toward the corresponding goal. The Rule Application nodes are used to determine the probability that the student can apply current knowledge and form a new fact, rule, or goal. The Strategy nodes are used to determine the probability that the student could choose a different strategy to solve the given problem. Once a step has been detected by the action interpreter, the corresponding node is set to "true" as the system is now sure that the student has the knowledge necessary to solve the problem. When "help" is requested of the ANDES system, the probabilities in the Assessor are used to give the most help that the student needs to solve the problem.

There are two types of suggestions that could possibly be given to a student[1]. The first type of suggestion would include conceptual help, which help involves helping the student with the theory behind why to perform the particular operation at that specific time. The second type of suggestion has to do with procedural help. This is the most common type of help needed.

Procedural help is the answer to the question of "what do I do next?" Since the ANDES system allows the student to solve the presented problem in any manor the student wants, ANDES needs to determine which way the student is solving the problem. This is to make sure that the hint given to the student is helpful. The way that ANDES achieves this task is by performing a depth first search of the "solution graph." This will provide all possible ways to solve the presented problem. The next task done by the ANDES system is to determine which solution path the student appears to be attempting. Once done, the ANDES system can present a procedural hint that is helpful to the student.

An earlier version of the ANDES hint system was tested in the Fall of 1997[1]. The results of those tests showed that ANDES had trouble presenting hints that were relevant to what the students were thinking. This finding led to the development of the version discussed in the paper.

2.2 Other Tutoring Systems

2.2.1 ANDES Lessons Learned

The first version of the ANDES system, ANDES1, was originally a project between the Office of Naval Research and the Navy's academic institutions designed to increase student learning and to forge a closer relationship between these two organizations[4]. The University of Pittsburgh, hoping to help the students, joined the project soon after. The original idea behind ANDES was to give the Navy's students a place work on physics homework and a place to turn for help with the intensive homework problems. Thus, ANDES was created to assist in this task. An important paper from the ANDES system development team is "The Andes Physics

Tutoring System: Lessons Learned" by Kurt VanLehn, Collin Lynch, Kay Schulze, Joel A. Shapiro, Robert Shelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill (2005). This paper is a summary written that encompasses the ANDES system project. This particular paper covers everything about the development of the ANDES system including the history, the point of view of the student, the architecture, the evaluations of the ANDES system, and most important aspect about the ANDES project: the lessons learned from the project itself.

One of the main challenges of the earlier system was to create a tutor that was as comprehensive as possible. The first version of ANDES used Bayesian networks and student modeling as described in "Procedural help in ANDES: Generating hints using a Bayesian network student model" by Abigail S. Gertner and Cristina Conati and Kurt VanLehn. After the ANDES system was tested at the Naval Academy, the ANDES system underwent a major overhaul and ANDES2 was born. The area of ANDES that was revised dealt with how the system generated hints. In the new version of the ANDES system all of the Bayesian networks were eliminated from the code and replaced by two new algorithms. The new version of ANDES has been through the initial testing and has shown excellent results. At this point, the development team has been adding extra features into the system like automatic problem grading.

While using the ANDES system, the student views a screen that is divided into four sections in a cube shape, two on top of two. The upper left square has the largest area. This is where the problem is presented and the student is to draw a force diagram. There is also a crude picture of the problem in this section. The square in the lower left is where the tutor displays the messages and hints to the student. There is not much else happening in this section. The upper

right square contains all of the variables and variable definitions that the problem gives and the student creates. The square in the lower right is where the student enters all of the needed equations to solve the presented problem. See Figure 1 for a picture of the ANDES interface. If the student enters any incorrect information, that equation or vector turns red. While correct equations and vectors turn green. This allows the student to know if they made a mistake and to know if they are on the right track to solve the presented problem. These four squares are at the focused presentation of the material.

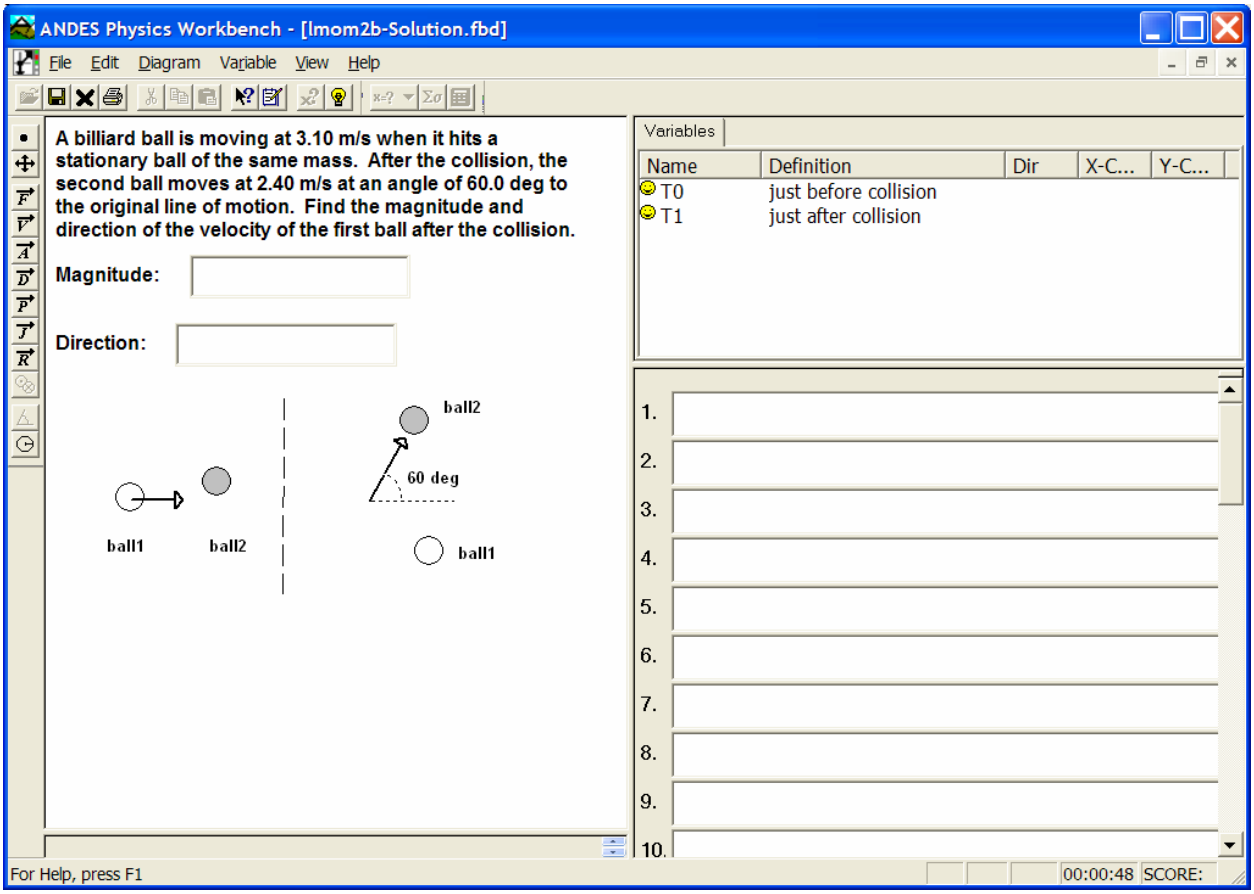


Figure 1: ANDES Interface

In the ANDES intelligent tutoring system, there are two different types of information that the student could enter[4]. The first type of information is the equation. An equation has the following form “fw_y=m*g.” The second type of information the student could possibly enter is the non-equation. A non-equation is everything that is not an equation, for example, force vectors. When any information, equations and non-equations, is entered into ANDES, the first step ANDES takes is to check whether the information is syntactically correct. If the syntax or structure is not correct, then the information cannot be verified by ANDES; therefore, it is automatically assumed to be incorrect. The next step, Step Two, depends on the type of information that has been entered. If the information is a non-equation, ANDES searches through the “solution graph”. The equation or vector information turns green if and only if an exact match is located in the system. Otherwise the equation or vector information is marked wrong by changing the color to red. If the information is an equation, then ANDES uses a new technique called “Color by Numbers.”[4] This new technique first checks the equation to make sure the units are correctly entered. In the “solution graph”, every variable needed or used to solve the problem is defined with the correct value. The next step in the “Color by Numbers” technique is to take the correct values from the “solution graph” for that problem and plug them into the equation that the student provided. If the equation balances, then it is turned green. Otherwise the equation or vector is marked wrong by turning it red. The “Color by Numbers” technique does not function properly all the time. If the student decides to divide by zero, this technique will not detect the error. However, this does not arise that often in real life problem-solving.

Therefore, after the entered information has turned red, the student should be asking himself/herself, “what’s wrong with the information I just entered?” To get the answer to that question, all the student is required to do is click on the “What’s Wrong” help key. The help information that appears is a rule-based system. Each piece of data that is incorrect is fed into the rule system, and the rule that matches the error with the highest priority is executed.

One of the most helpful components of the intelligent tutoring system is the option of how to proceed in solving the problem so that the student may utilize a “What’s next?” opportunity[4]. This is a “solution graph” for the problem that contains all the possible ways to solve the problem that has been presented. The “Next Step” help first recognizes which steps the student has already completed. The program must then recognize which way the student intends to solve the problem. If a branch point has not been reached yet, the ANDES system may ask the student which way they would like to solve the problem. If the student has completed steps on both solutions, the solution with the most completed steps is chosen. After a solution path has been chosen, the hint presented to the student is about the next step in the solution path.

For every problem that is entered into the ANDES system, a “solution graph” is generated automatically[4]. Studies have shown that experts in the field of physics tend to solve problems by using major principles. The major principles, like Conservation of Energy or Newton’s Second Law, can be used to reach the final answer by a variety of different means. This indicates there is more than one way to solve a physics problem, and all possible solutions need to somehow be in the “solution graph”. Each major principle method is programmed into the ANDES system. Once a problem has been entered into the system, ANDES looks at the

three different items to determine which principles could be used to solve the presented problem. The first item is examined is the equations that have already been created; the second item is composed of the variables in the problem that are already known. The last and most important item is the variables that ANDES is solving. If a major principle method meets all three items, then it is used to solve the problem. The ANDES system checks all of the major principles: this is to ensure that no solution is missed.

The ANDES system was tested and evaluated for a total of five different semesters at the U.S. Naval Academy between 1999 and 2003[4]. In a semester, all physics students take the same final exam and use the same textbook. The only situational differences are the instructor and the homework problems assigned to the students. Some sections used ANDES for homework problems while the other sections were assigned problems that were of similar difficulty to those within the ANDES system. On the midterm exam in all five semesters, the students using the ANDES system consistently outscored the non ANDES students. The final exam results did not demonstrate as significant a difference as the midterm exam, but on average the students using the ANDES system outscored the non ANDES students. The results on the final exam could have been skewed by the fact that the ANDES system at the time only covered about seventy percent of the final exam. After the final exam, the ANDES students filled out a questionnaire about the efficacy of ANDES in answering the student's questions and functioned in general. The result obtained from students indicated that the students had mixed experiences with the ANDES system.

The ANDES system development team has laid out a streamlined three phase process on how they would create an ANDES system. In the first phase, the student interface and

immediate color feedback would be created[4]. This allows the student to become accustomed to the interface and the available tools. Another factor that is needed for the success of this phase is the addition of numerous problems into the system. A professor wants a system that could be used for the whole semester and not just one week. The second phase involves creating the "Why what I entered is wrong" help[4]. In this phase, the error handlers are added in providing the student with help on figuring out why their input is wrong. These error handlers are be created using the logs from the phase one system and should be thoroughly tested to ensure no errors are misclassified. The third phase involves adding in the "What's Next" help[4]. With the implementation of phase 3 comes an important design decision. There are two different ways of to implementing the "solution graph". The first option is to generate an automatable system like the ANDES system. This heuristic has a high level of difficultly, but no feasible solution is ignored or missed. The second option is to have physics experts generate the "solution graph". This option could have missing solutions in the "solution graph" as experts are known to problem solve differently than non-experts. By following these three phases, one could create an ANDES system for their own personal use in any discipline that they may have interest.

2.2.2 QUANTUM Lessons Learned

The Quantum tutoring system is a private endeavor and as of yet has failed to produce many papers on the tutoring system. The tutoring approach used in the Quantum tutoring system, is one of solving the problem for the student and thus this approach was found to be unsuitable for the EINO tutoring system and therefore was not pursued to any further extent.

2.3 Conversational Case Based Reasoning

Based on the research that has been reviewed from the Office of Naval Research and the University of Pittsburgh, it has been shown that intelligent tutoring systems help students achieve a better understanding of coursework and better grades in their classes. Using that information, it is hypothesized that there are other systems that one may be able to develop to further enhance student learning. This project will explore another possible approach that could replace the ANDES help system with Case Based Reasoning (CBR). One way to describe CBR is to think of a filing cabinet that is filled with multiple folders[5]. The first step in a case based system is to find the closest case, or problem, in the filing cabinet to the one being presented to the system. In the next step, the system reuses and/or revises the output from the closest case and applies it directly to the one presented by the student. The last step would be to save the new case and the subsequent output for reuse later. The whole process can be described in three words and those words are: retrieve, reuse, and retain[5]. A case based system would provide a way to generalize the ANDES system by eliminating major parts of the "solution graph". Proof that Case Base Reasoning could handle this task can be found in "Dialog Learning in Conversational CBR" by Mingyang Gu and Agnar Aamodt. This paper was presented at the *Flairs* conference in May of 2006. The purpose of the Conversational CBR system is to guide a user to create a problem description by asking questions. This particular paper covers the framework for the Conversational CBR system. The framework is followed by the implementation and results of the Conversational CBR system.

A conversational CBR system functions a little differently than a normal CBR system[6]. In a conversational CBR system, the user first enters the target case or problem just as they

would in a normal CBR system. The conversational CBR system selects several similar cases and identifies multiple features from the case. The conversational CBR system presents the cases along with some questions to find out more information about the problem. It is at this point that the user has the option to select a case or a question. If a case is selected, the system terminates. If a question is selected, the system reevaluates the selected cases and presents a new group of cases that are similar to the one just presented. This process continues until there are no more questions or a case is selected. In a conversational CBR system, there are two CBR systems being utilized. The one that is found in all CBR systems, while the other system is designed to handle the dialog between the user and the system. The cases in both CBR systems are subsequently divided into three sections. The first section is the problem description, this section has the problem needed to be solved and the state of the surroundings. The second section is the solution; an essential part of a case is having the solution. The third section is the output. How did the solution affect the surroundings? Was the problem corrected?

In the implementation of the application CBR system; a global feature weight was used to select the cases that are closest to the input or test case[6]. The dialog system uses a similar idea to rank the questions.

The conversational CBR system was evaluated using thirty-two of the thirty-six data sets from the UCI Machine Learning Repository. The four were left out due to the length of the runtime. Over ninety percent of the datasets showed at least an eight percent improvement verses other CBR systems. The tradeoff for the improvement lies within the CPU time needed to execute the system and the amount of memory needed for the system to run[6]. The conversational CBR system uses more CPU time and memory space than other CBR systems.

2.4 Case Based Reasoning Tutor

An example of a case based reasoning tutor can found at North Dakota State University[7]. A team of professors has created a variety of virtual environments for educational purposes. One of the virtual environments, the DollarBay, was created to assist students in gaining some experience in retailing merchandise. Other virtual environments included cellular biology, history and geology. These virtual environments were created using LambdaMOO, which is an object oriented language[7]. After the various virtual environments were running properly, a decision by the design team was made to add a tutor into the system to help students who were having trouble understanding the concepts. A case based reasoning system (CBR) was soon added. The tutor is created each time a student enters the virtual environment. In any CBR system, the strength of the system is derived from the design of the case library and the ability to find similar cases.

All cases of the tutoring system are divided up into one of three different groups of cases. For example, in the DollarBay virtual environment, the first group of cases is the historical cases. This group contains all of the cases of actions that a player has made when the simulation is finished and the student transfers into the "Hall of Fame." The second group is the case library, this group has a copy of every distinctive case. The third group of cases is the prototypical cases, this case group is made up of possible players. These possible players will be matched up to how the player is currently interacting with the system, over cautious, over spender, etc. The tutor uses the last group of cases and all the active cases to determine how to interact with the student. A case is selected from the prototypical case group and the active case group. The tutor does not appear to the student until after the user has been in the system for twelve

hours[7]. The tutor presents different messages based upon how the student is doing in the virtual environment; an example of a message is “Try to keep more cash on hand” if the student has a low cash reserves.

To evaluate this tutoring system, the design team simulated sixteen players in the virtual environment. Some of the simulated players received tutoring while others did not. After a period of seven days, which would have been eleven simulated weeks, the players were scored. The top three players after the simulation had ended, all had had tutoring[7]. The bottom two players had no tutoring. This demonstrated that the tutor functions properly and provides helpful information to a struggling student and thus is able to help the student perform better.

2.5 Data Mining Tutor

Case based reasoning (CBR) is not the only option for an intelligent tutoring system. The Data Mining Tutor, or DaMiT, uses a multi dimensional approach. Data mining is the science of generating models based upon data used to predict future data[8]. The main goal of the DaMiT system was to create a system that is additive to the user and not the other way around[8]. In most systems, like the ANDES system, a user is forced to learn how the system needs the information entered in order to work with the system and have it solve problems. This is not true with DaMiT program. Some tutoring tasks do not need the ability to be multi dimensional. An example of this is if the goal of the system is to memorize text; only one dimension will work well. However, if the goal of the system is focused on research, development, and experimentation like in data mining, then a multi-dimensional approach with a tutor would save computing time and effort on the part of the student[8]. Multi-dimensional approaches not always mean multiple copies of a tutor but could also refer to the multiple

presentation of the knowledge. This could be done by incorporating videos into the tutor along with a text based dialog, adding in information while the user learns the process by performing the action assists learning to occur. This is why interaction between the tutor and the user needs to be planned out in detail, a unique technique to help plan out this interaction is known as storyboarding.

2.6 Storyboarding

Storyboarding is a design tool that lets one picture how the end product could and/or should look and is used mainly for movies and television. This same concept has been adapted for the e-learning environment, but the current storyboarding concepts are just replacing software design documents. In order for a student to learn, an interaction must occur with a teacher over some form of media. This media could be a simple conversation, a book, a video, or any way a professor may choose to utilize a learning environment. Current storyboarding concepts are inadequate to handle the use of all of these mediums[9]. Consequently, a re-creation of storyboarding must take place. The first item needed in the re-creation is a node: nodes represent an entire episode or just a scene in the episode[9]. To make this simpler, an episode will require a subgraph explaining the episode scene by scene. A scene could include a video that is playing, a picture or pictures that are being displayed, or a document opening up for the user. By having nodes represent scenes and episode, this allows storyboarding to be scaleable. In most videos, whether it is on television or in the movies, there are multiple scenes and between each scene a transition is needed. This is also true in storyboarding, where transitions are represented by edges in storyboarding**.** To connect a node and the edges, comments are used. This allows the creator to express knowledge about the interaction between the teacher

and the user - with the scalability of the new storyboarding, building a complete storyboard could take a lifetime. Every scene could be broken down into even more details.

2.7 A Multi Agent Architecture

All of the architectures discussed thus far have focused on a single program to tutor. This is not always the case. A tutor generally consists of four components[10]. The first component contains the curriculum, where all the expert knowledge is contained. The second component is the student model: the mapping of what the student knows and understands. The third component is the tutor. This is what actually controls presentation of knowledge to the student. The fourth and final component is the interface. In 1996, Nkambou, Gauthier and Lefebvre proposed a tutor architecture that only contained three main components[10]. The interface was not considered a main component. All of these components are completely different; therefore, the whole architecture is difficult to build. Roger Nkambou and Froduald Kabanza saw this difficulty and expanded this architecture in 2001 to incorporate a numerous agent approach[10]. All tutoring systems should have a prerequisite plan established as to how it will to respond to the student/user. This common task between all of the components is the key element of the agents. Several agents are placed together in a multilayered configuration. Therefore, all agents are working toward a common goal of planning how to respond to the user. The Roger Nkambou and Froduald Kabanza architecture shows that a distribution of tasks is possible in a tutoring system[10].

2.8 Summary

Within this chapter there is a discussion of some of the research relevant to the design and development of the EINO intelligent tutoring system. The first two papers describe two different parts of the ANDES tutoring system. The procedural help paper discussed the beginnings of the ANDES system. The initial design of the system is described in great detail. The initial design of ANDES had some components that functioned perfectly while other components were not as ideal. The major element that malfunctioned was the Bayesian network. The Bayesian networks were nonexistent in the final design. The final design of the ANDES system is discussed in the lessons learned paper. The lessons learned paper also described some problems that have occurred in the years of development and how the problems can be avoided. A streamline approach to the development of ANDES is also laid out in the paper. These two papers are important because they allow one to fully see how the ANDES works and the many problems encountered in the development.

The next two papers presented discussed tutors that already use case based reasoning. These two tutoring systems teach students vastly different material but each has the same basic logic behind the operation each tutor. This paper demonstrates that it is possible to create an intelligent tutoring system that functions acceptably using case based reasoning system. This is also the current design of the EINO system which also employs case based reasoning.

The data mining approach is similar to case based reasoning. Both approaches, data mining and case based reasoning, use past data in order to predict what actions the student will perform. It is a possibility for the EINO system and for this reason that the Data Mining Tutor is discussed.

The storyboarding paper lays out an approach for creating a detail storyboard of the interactions between the student and the system. This technique is used so the creator of the intelligent tutor does not become lost in the various interactions of the system and student. A storyboard has been created for the EINO system.

The multiple agent paper presents a completely different approach to an intelligent tutoring system. This is not a common architecture that one creates. The architecture has merit to the design and development of the EINO system as a creative architecture can spark a new design.

CHAPTER 3: PROBLEM DEFINITION AND HYPOTHESIS

3.1 General Problem Statements

As all of technology in the classroom changes and expands, the way one interacts with technology must also undergo changes. As our tools have become more sophisticated, we as a people have adapted to our tools by making them more efficient for us to use; we have gone from a horse drawn carriage to today's automobile and magnetic levitation transport (Maglev) trains. This advancement is true for all of tools that are known to man. We no longer have to seek ways to create fire because once fire was discovered to be a useful tool; the means to create fire was passed down by one generation teaching the knowledge to another generation. This is why every great and useful discovery significantly affects education. Just look how far we have come in documenting our knowledge. Writing implements such as stones and chisel to paper and pens to today's technological marvel – the computer. Of course, without the computer there would be no internet. This progression of tools has enabled our students to complement their education with sophistication by using the computer. Homework assignments are no longer expected to be mundane repetitive work. Students today can and do have the capability of completing labs and simulations from home. In a classroom setting, a student with a problem always has the teacher available to answer a question or to make corrections so that he/she can get back to right track. Where is the student to go if, while they are working at home, have a question or problem that arises? The internet has become a repository for anyone to place their ideas and / or discoveries, most of the time without proof. This leads us to our first problem, which is:

- *To create a place for a student to turn to when they need help.*

Modern artificial intelligence techniques have advanced to the point where one could simulate a person's reactions in a particular environment based upon information provided by the person. This allows us to create a teacher to answer questions on a specific topic. The creation of this teacher is known as intelligent tutor system. It provides the student a place to turn for help when they become baffled as they try to complete a lab assignment. Having a place to turn when in trouble will encourage more students to try a lab or other assignments at home. This will undoubtedly raise the student interest in learning. They will feel more confident tackling problems at home as well as at school and we can expectedly see a rise in the educational level of students.

3.2 Specific Problem Statements

The specific problems guiding this research lie in creating a tutoring system. The Infinity Project was created as an attempt to get high school students more interested in Engineering studies, more specifically Computer and Electrical Engineering. A few professors at Southern Methodist University (SMU) noticed a drop in the number of engineering students and created a year long class that high schools could offer to their students. The class, the Infinity Project, was comprised of a textbook and a lab manual that the professors developed and produced themselves. In order to complete all of the labs in the Infinity Lab Manual; a school must also purchase a lab kit that includes a circuit board with the different components for each lab, speakers, and computer software that was used to control the circuit board. The University of Central Florida, UCF, looked over the Infinity Project and became enthusiastic about the project. Wanting to support the project and upcoming university students, in every way they

could, UCF hired a few graduate students to create a number of the lab experiments in a web format. This was done so that the students had the option of spending more time engaged in the activities that went beyond what the teacher had done in class with the lab kits on the experiments. Each web experiment is contained within its own java applet. Each of these applets has three main parts. When first loaded the applet displays the first of three tabs. This first tab is the theory tab and is the first main component. All of the relevant information about the experiment is presented in the theory tab. The second main component and second tab is the displays a schematic of how each component is connected in the experiment. The final main component and tab is the experiment itself. The question: "Where is the student to go if they require help?" can have multiple answers depending on how one defines "help." From the papers describing the ANDES system, we learned there were two different types of help. The most recognized type of help is conceptual, which involves helping the student with the theory behind what is occurring in the particular experiment. This leads us the next problem statement, which is:

- *To provide conceptual help with particular experiments that is presented differently for the theory tab.*

The second type of help is procedural help, which involves helping the student answer the question, "What do I do next?" For this type of help, the system must monitor the student step-by-step in order to prevent a help step that the student has already done. Consequently, this eliminates repeating known information could lead to frustration and failure for the student as they may no longer show an interest in the topic. A full procedural help with the web experiments would not make sense, as the student already has a lab manual that provides step-

by-step instructions on the experiment. What is needed is help with the interface as there are differences between the web experiments interface verses what is shown in the lab manual. This is where the students might need help and leads us the next problem statement, which is:

- *To provide help with the interface to the user by offering just in time hints.*

3.3 Hypothesis

By providing a case based reasoning and rule-based help system into the web experiments created by UCF, students will have an online site to reference in case they require any help with an experiment and be more willing to attempt the experiment.

3.4 Contributions

1. A case based reasoning intelligent tutor
2. An implementation of the tutor in the UCF web experiments

CHAPTER 4: THE APPROACH

Within this chapter there is an attempt to clarify, on an advanced level, how the intelligent tutor, EINO, functions during operation. One of the most challenging aspects of designing an intelligent tutoring system is designing the tutoring system to function in conjunction with the University of Central Florida Infinity Web Applets (UCF Infinity Web Applets). Throughout this chapter, we will explore the approach taken during the development of EINO, the intelligent tutor.

4.1 The Development Environment

Integrating an intelligent tutor into an existing system is a two sided coin. On one side is the challenge of building EINO around classes and components that already exist. This restricts the designer in how the inputs are collected by the tutor. On the other hand, the programming language is already predefined. EINO will be built using the same language as the UCF Infinity Web Applets, which were programmed in Java. Borland's JBuilder 2005 was chosen as the development environment for two reasons. The first reason is that JBuilder possesses all the features needed during the development of EINO. The second reason is the familiarity with Borland's JBuilder. When developing a new system it is important to be familiar with the development environment. This helps in that one can spend more time on EINO instead of learning a new development environment. Knowing the software used to create EINO is only half of the picture. One must also look the specifications of the machines in which EINO was developed. The specifications are as follows:

- Intel based processor at least one point eight Gigahertz
- One gigabyte of ram
- At least eighty gigabyte hard drive
- One hundred twenty eight megabyte video card

The project was designed to work on the same operating system that the UCF Infinity Web Applets operates. With this thought in mind EINO can run on the same operating systems as the UCF Infinity Web Applets. The operating system used is Microsoft Windows XP or Microsoft Windows 2000. Java must be installed on the machine in order for the UCF Infinity Web Applets and EINO to work.

4.2 EINO General Function

The simplest way to look at a new project and understand how it works is to look at a block diagram. A block diagram for the standard operation of EINO while running is shown below, Figure 2. After EINO is initialized, the interface is not active until help is requested by the student or EINO can offer advice. The first and simplest block diagram to look at is the one for help requests made by the student. In this situation, help was requested by the student. EINO responds by opening a webpage with the EINO general help menu. The page displays a list of components used in the lab. The student is able to click on the component they are having problems with in the lab. EINO presents the needed information to the student. If the topic is considered a “big” topic, EINO follows up by asking a few questions about the topic to ensure the student understands the information. With the “small” topics, EINO simply presents the information requested. The general help menu list included an option for simple lab issues.

EINO can present a step-by-step guide of how to fix certain issues that might cause the lab not to function, for example, the sound malfunctioning. The webpage with the EINO general help menu is only activated when the student requests help.

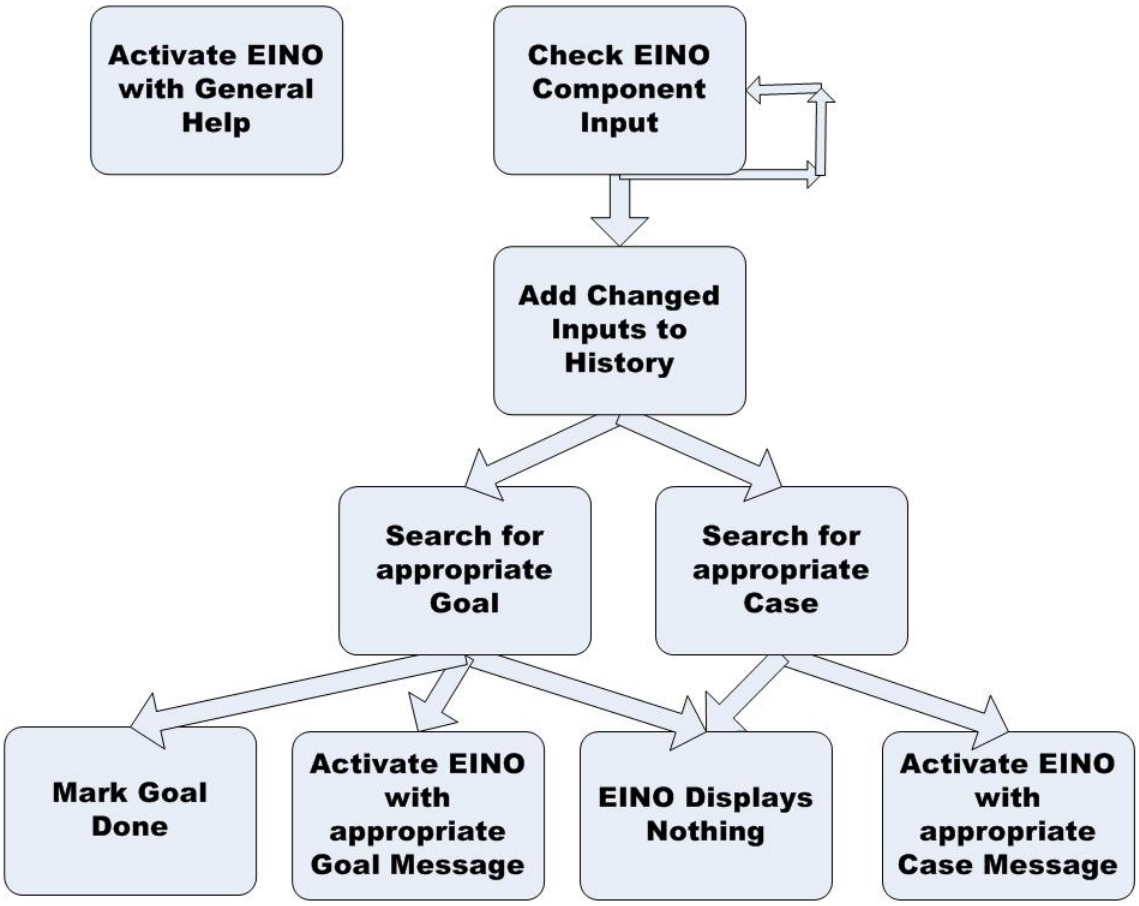


Figure 2: EINO Block Diagram

The second way the EINO interface can be activated is shown in the second part of the block diagram, Figure 2. While a lab is open, EINO checks to see if any of the inputs for that lab have changed roughly every second. Each component in the UCF Infinity Web Applets is different. A slider can change values very quickly while a microphone button has no numeric

values. The only value that a button has is if the button is pressed or not. Since a slider can change values so rapidly, EINO requires the slider to maintain the same value for a count of three or roughly three seconds. This way EINO knows that the student has stopped moving the slider and the value is not an intermediate value while the student was simply changing values. If any lab input has changed since the last check, the changed component's label and value is added into a history list. In the case of a button which has no numeric value, it is given a default value of one. The history list contains the last ten input changes made during the lab. A list of the last ten input changes is enough to figure out what the student is trying to accomplish in the lab. After the item is added into the history list, EINO begins the first of two different searches.

This first search involves looking though the goals for the lab. Every lab is designed to work as if the student was performing the lab in class. This insinuates the student is expected to be following a lab manual in order to complete the lab. How would EINO know that the lab has been completed by the student? The first possible solution to this problem is to require the student to complete the lab step-by-step according to the lab manual. This would make the labs inflexible and discourage the students from practicing the lab on their own time as it would limit their ability to experiment with the lab. The other solution is to divide the step-by-step lab instructions into a set of goals. If the goals are arranged properly, the student will accomplish the same tasks but this will allow them the flexibility of completing the step-by-step lab instructions in any order that the student may deem appropriate. A single goal could be anything that deals with the input into the lab, like setting the volume to a value of fifty eight. EINO is configured to use the second approach. The goal search looks through history list to see if the last action meets any unfinished goals. A goal may require that another goal be met

before it able to be considered. In other words, certain goals will need to be done in a set sequence. This makes the goal search more difficult to perform. EINO gives a higher priority to the goals that must be done in order than a single goal. If the student is in the middle of a set of goals and makes a mistake, EINO leaps to the aid of the student with a "Just in Time" hint of what the student should do next. When a goal or a goal set has been met, EINO marks those goals as done and they are removed from the search list until all of the goals are met. The other possible action for EINO is that after the goal search is done is to stop and do nothing. EINO is built to stay in the background until needed by the student. If help is not required by the student, EINO hides in the background.

The second search the EINO performs involves looking through the cases for the lab. The concept behind the case based reasoning (CBR) system in EINO is to recognize patterns in the student's actions and produce help to the student before they ask for it. A single case contains a history list that has a pattern that EINO is looking for and a number of the EINO webpage to display for the student. The best way to explain a CBR system is to imagine a filing cabinet in a doctor's office with each case acting as a file. A new patient walks in to see the doctor with a list of symptoms. The physician, after generating a list of the symptoms, realizes that he has already seen almost the same symptoms with a previous patient. The physician goes to his filing cabinet and looks for the previous patient's file and uses the same treatment that worked on the previous patient on the new patient. EINO acts as the physician in the scenario above. The student presents a list of actions through use of the lab and EINO looks through the previous cases to find one close enough to the students' work. Then EINO presents the help that was needed in the previous attempt. If no match is made, then EINO will wait until the

student's actions match something in EINO's database. A simple output for EINO can be seen in Figure 3. In this case, the student is having difficulty with understanding what effect the volume slider has on the lab.

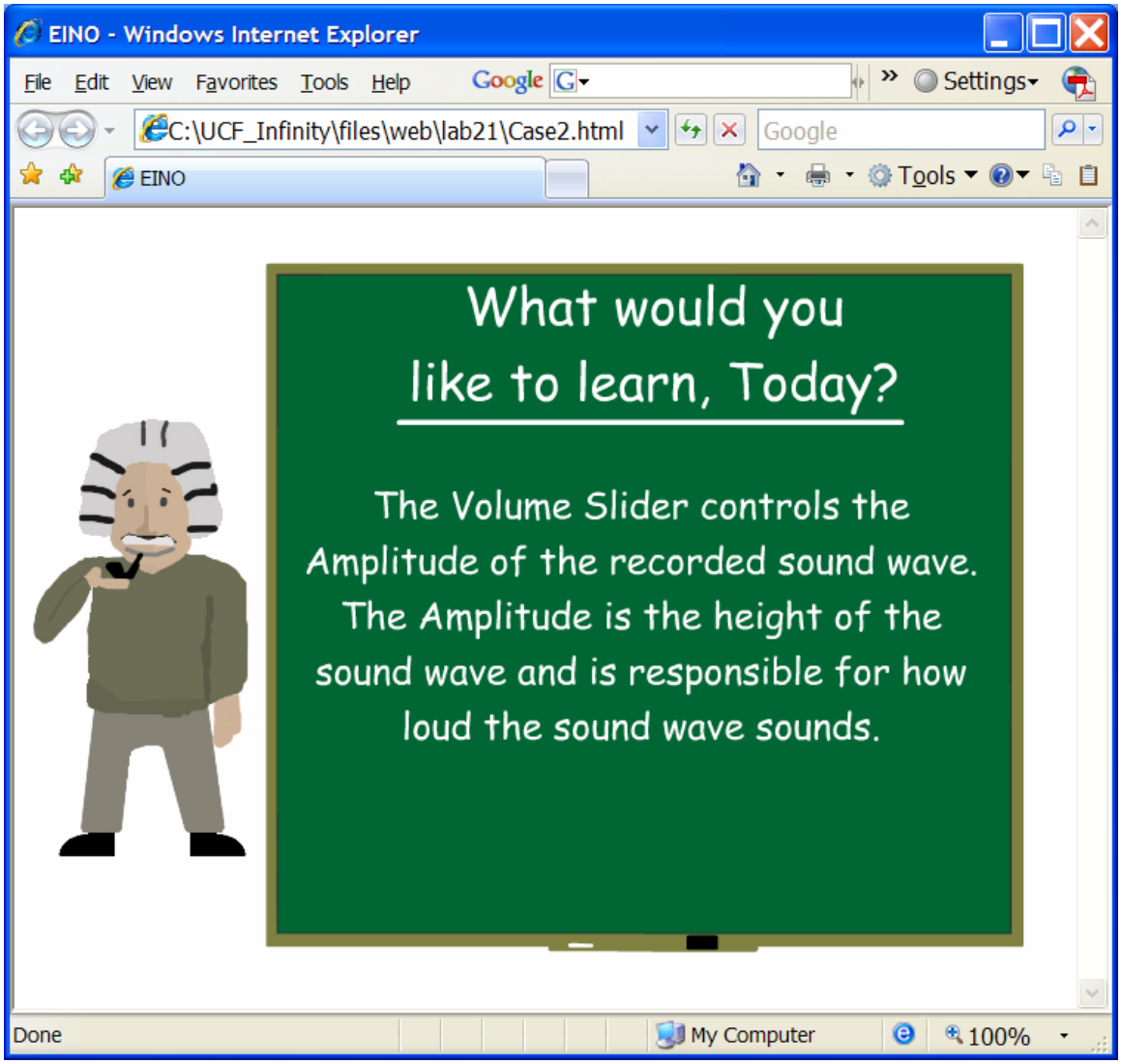


Figure 3: Problem with Volume Slider

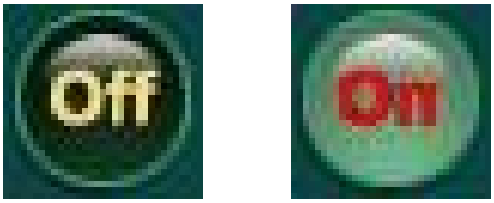


Figure 4: Off and On Buttons

A third way to generate an output from EINO is to change an input while the lab is off. Figure 4 shows an enlarged version of both the Off and On buttons. If the lab is in the off state, then the On button is displayed. Likewise, if the lab is in the on state, then the Off button is displayed. Why would it make a difference if the lab was either on or off? While in the Off state, some of the inputs for the labs will not function as the student would anticipate it to operate, if at all. This is simply due to the fact that some of the connections of the lab components are not made until the lab is turned on. These "malfunctions" may cause the student to question what is happening and ask for help. Therefore, EINO leaps to action and will give a gentle reminder that the lab is not turned on and should be turned on to function appropriately. See Figure 5 for this message.

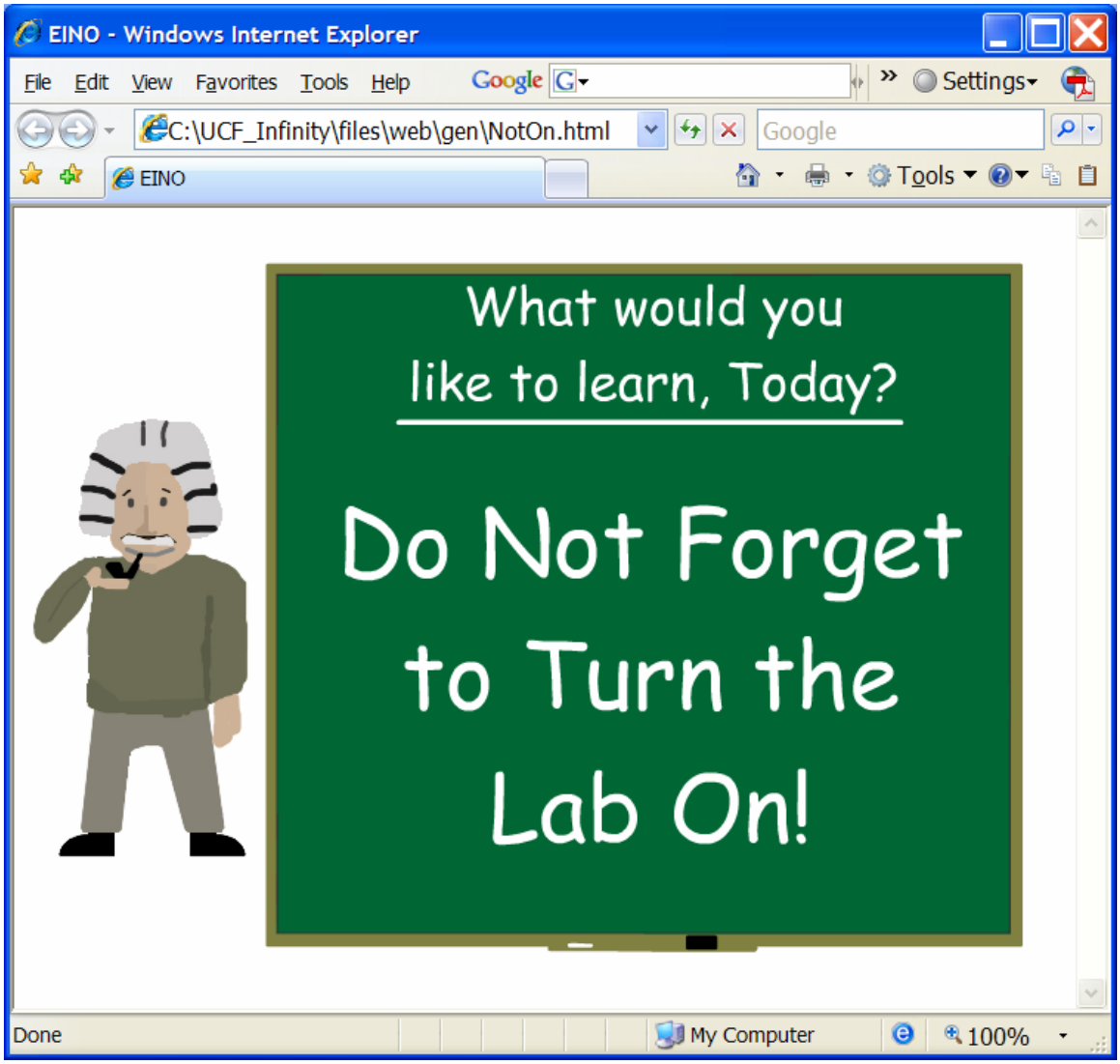


Figure 5: The Lab is Not On

4.3 Goals and Cases

A common question asked about EINO at this point is if goals and cases are so similar then why not make all the goals and cases and have just one search instead of the two. This separation was done for two reasons. First of all, the goal search is harder to process due to the fact that a goal could be a single move or a group of moves. This indicates a search must give a

higher preference to a group of goals versus the single goal all the while keeping all options open until the next move is made. In the goal search, a list of possible goals is made and maintained until the group of goals is finished or the list contains only single item goals. The second reason the goals and cases are separated is that EINO offers "Just in Time" help. This system's main focus is to offer help to the student to complete a group of goals.

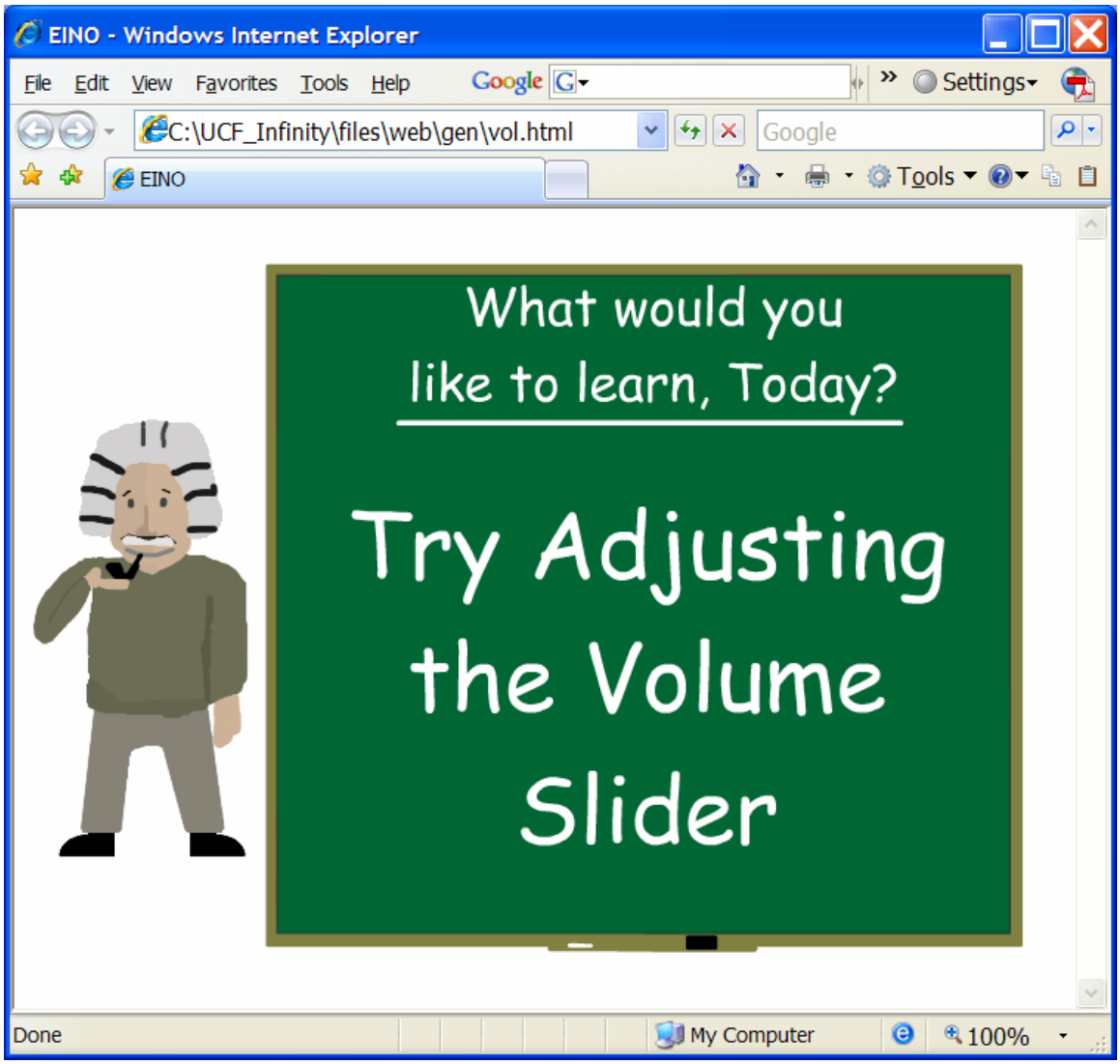


Figure 6: Try Adjusting

Imagine a student working their way though a large group of goals. At the last step the student misreads or mis-clicks on an input. Instead of throwing out all the hard work that the student performed, EINO forgets the last move made by the student, recommends a course of action, see Figure 6, and gives the student one chance to fix the mistake. This leads into the third reason that EINO has both cases and goals. A group of goals must be executed in order or else you did not perform those steps to the letter. A case can be close enough. Therefore, there is an exactness issue between cases and goals. If cases were all goals, then the freedom that is found in a close enough match is lost. If goals were all cases, then the student might not be completing the entire lab according to the lab manual.

4.4 EINO Initiation

EINO is initiated at the same time as the UCF Infinity Web Applets. This is the earliest moment that EINO can be loaded. By having EINO ready at this time, it allows the student to request help as soon as the lab loads. The first step in the initiation of EINO is to load the goals. The goal set for each lab is stored within a file. Each goal is read and then stored within an array. The second step is to load the cases used in the lab. Again, the case set for each lab is stored within a file and is stored within an array. The final step for the initiation of EINO is to start the thread. The thread is what allows EINO to check the inputs every second. Without the thread functioning properly EINO will not be able get inputs and match goals and cases.

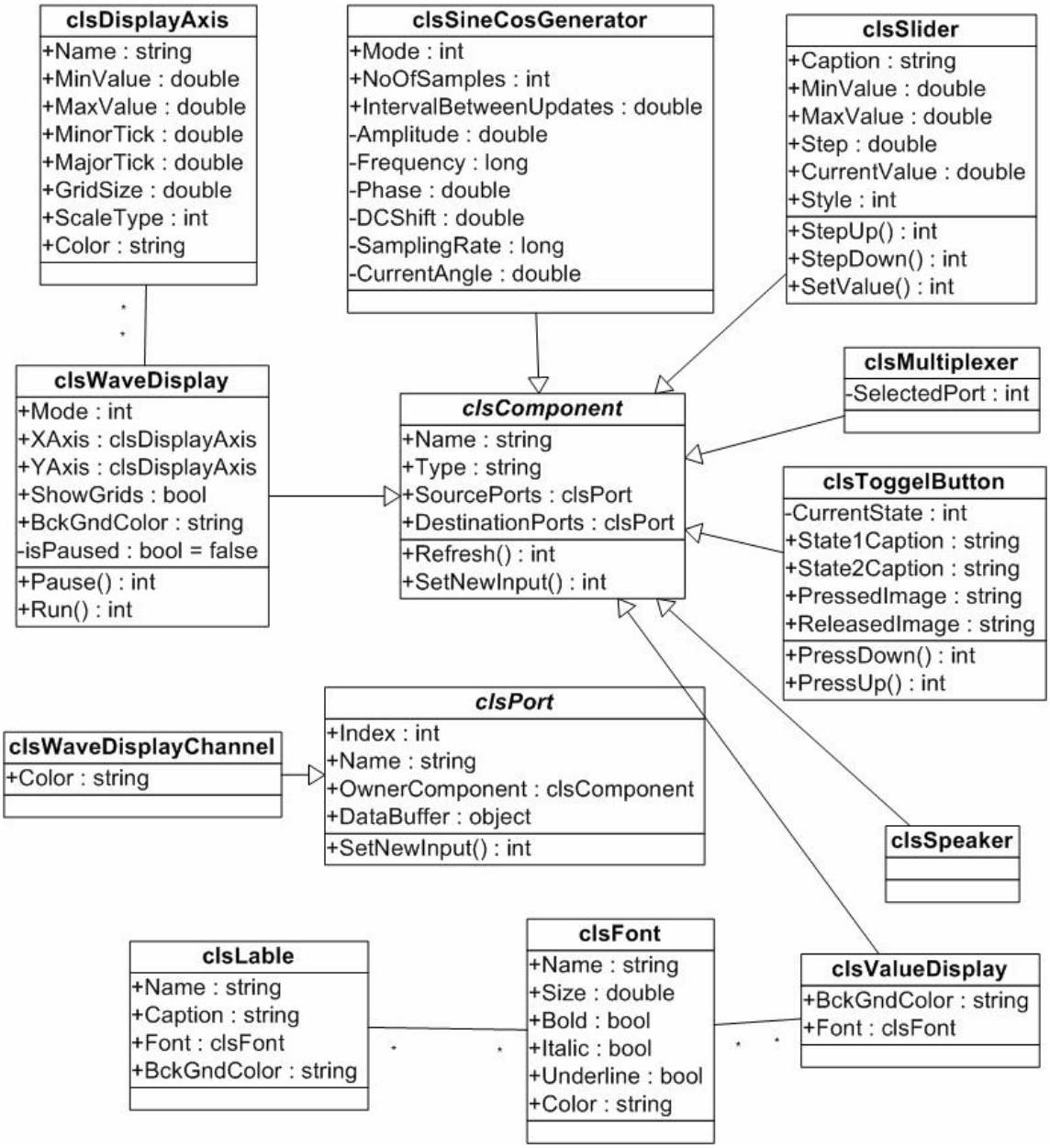


Figure 7: UCF Infinity Web Applets Class Diagram

4.5 Class Diagram

Since EINO was built in Java, an object oriented approach was taken in the design of EINO. In order to make EINO function better within the UCF Infinity Web Applets, EINO's design is similar to the UCF Infinity Web Applets. The UCF Infinity Web Applets class diagram is in Figure 7. The base object in the UCF Infinity Web Applets is the clsComponent class. The class contains variables needed in all of the components within the Web Applets. The input values that EINO checks are not within the base component but within the super class. EINO contains a modified version of the Web Applets super classes in order to tell if the input has changed. Figure 8 contains the class diagram for EINO. EINO also contains classes in order to manage the cases and goals. In the middle of the class diagram (see Figure 8), is the main class in the EINO system. The top part of the EINO class, lists the variables and variable types used in the class. The bottom part of the EINO class, lists the functions and the return type used in the class. In the class diagram, there many other classes that are shown and are used to store variables and functions that will be reused throughout the system. This information is required in order to recreate the EINO system. The class diagram will be discussed in further detail in the next chapter.

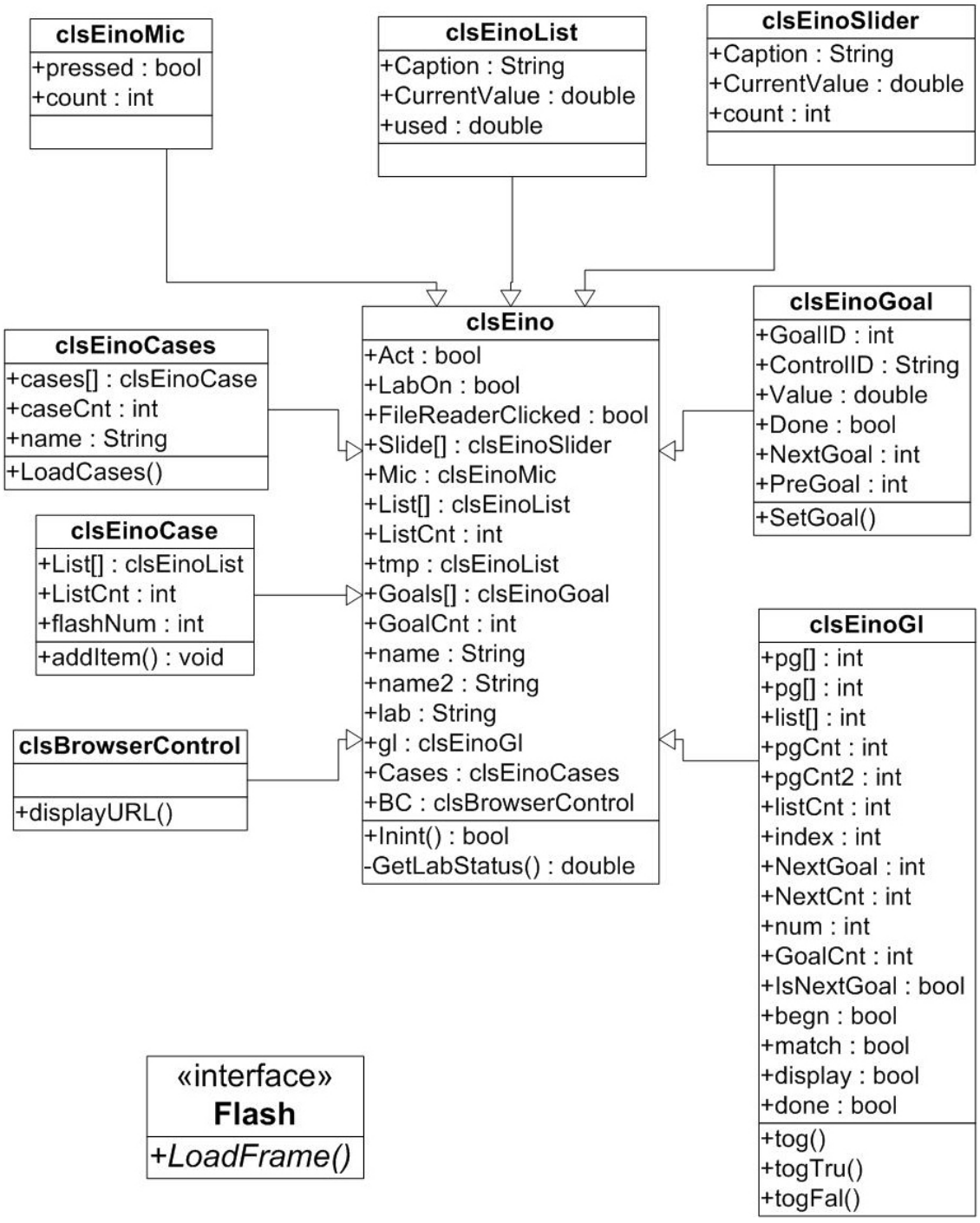


Figure 8: EINO Class Diagram

4.6 Summary

This thesis uses a mixture of case based reasoning (CBR) and a rule-based system (RBS) in order to produce an intelligent tutoring system in Java. Within this chapter, there is an explanation of the approach taken in the development of EINO. In this chapter, a step-by-step explanation of the how EINO tutoring system functions on a higher level. The first step is the initiation of the tutoring system. This is where the cases and goals are loaded and stored into the EINO database. The second step is to generate an output from the tutoring system; which can be completed in four different ways. The first way is the student requesting help. The second way is by EINO identifying a pattern in the student's actions and open up an appropriate case. The third way to generate an output from EINO, is to, in the middle of completing a group of goals and make a mistake before finishing. The fourth way to generate an output from EINO is to adjust an input while the lab is in the off state. The next chapter will cover the implementation of the EINO tutoring system in greater detail.

CHAPTER 5: IMPLEMENTATION

Within this chapter there is a discussion of the implementation of the EINO tutoring system within the University of Central Florida Infinity Web Applets (UCF Infinity Web Applets). An overview of the UCF Infinity Web Applets is given including some details of the source code. EINO has already been discussed at an upper level thus far and will be discussed in further in terms of programming.

5.1 UCF Infinity Web Applets Source Code

The UCF Infinity Web Applets were designed to emulate a hardware device using an object orientated approach in java. For the class diagram, see Figure 7. The most basic class that is used often is the clsComponent. The clsComponent class is the base class that is used as the starting point for all the components that are created in the UCF Infinity Web Applets. Almost all the other classes inherit attributes from the clsComponent class. In the UCF Infinity Web Applets, there are four different types of classes. The first type of class is the output components. These components produce an output for the student. This output could be visual, like an image or audio, like a sound wave produced by the speakers. The output component sometimes has a rudimentary interactivity, if they have anything at all. If the student clicks on a data point on a wave output, then the value at that data point are displayed. The image outputs have no need for any interactivity because what is important for this output is the image itself.

The second type of classes in the UCF Infinity Web Applets is the input components. These components allow the student to enter data into the system whether the data is in the form of an audio signal, or an image or an adjustment of a value. The audio signal is entered through

use of a microphone. The image is entered through the use of a file reader built into one of the classes and the adjustment of values is done by use of a slider or the pixel mapper. The slider is a simple bar that moves left or right. If moved to the left, the value is decreased and likewise if a bar is moved to the right, the value is increased. All of the data needed for the slider is store in its own class called clsSlider. The most obvious value stored in the class is the current value of the slider. The clsSlider class contains the maximal value that the slider could have along with minimum value for the slider. A slider also has a step value, which tells the system at what intervals to move the current value up or down. A string value is used to display what value the slider is controlling because there can be several sliders in each lab. The other type of component used for the adjustment of values is the pixel mapper. The pixel mapper is a completely different type of input when compared to the slider. The pixel mapper looks like graph or chart with a single line running through the graph.

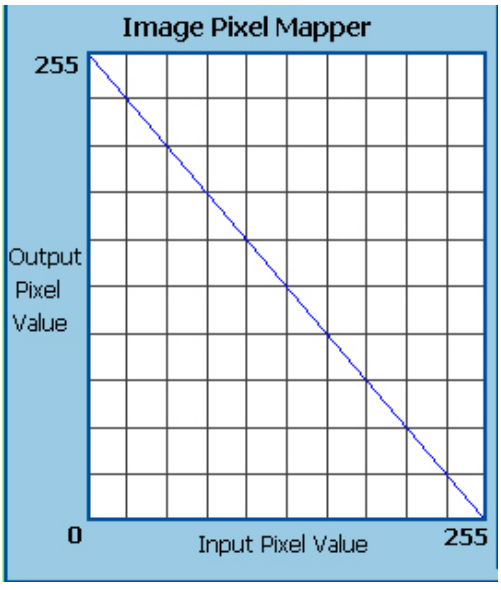


Figure 9: Pixel Mapper

The bottom part of the graph, the X axis, represents the pixel values from zero to two hundred fifty five. The left side of the graph, Y, axis, is also numbered from zero to two hundred fifty five and represents the new remapped values. See Figure 9 for picture of the pixel mapper. In the picture, zero has been remapped to two hundred fifty five. This means any place in the original picture that had a value of zero is now given a value of two hundred fifty five. The new values are stored within the clsPixelMapper class to allow several pixel mappers to be used in a lab.

The third type of class in the UCF Infinity Web Applets is the calculation classes. These classes are used to perform some type of calculation on the data in the lab. A good example of a type of calculation done often in the UCF Infinity Web Applets is a Fast Fourier Transform or FFT. The FFT algorithm produces the frequency of a signal. This calculation is performed in real time throughout several labs.

The fourth type of class in the UCF Infinity Web Applets is the lab class. These classes must be called in order for the lab to run. All of the components needed for the lab are created and connected to each other in this class. This signifies there is no input data stored in the lab class; just a set of connections.

5.2 Step One of Implementation EINO Tutoring System

The implementation of the EINO tutoring system was broken down into a four step process. The first step was to find a way to access the data from the inputs. This step was harder then it gives the impression due to the architecture of the UCF Infinity Web Applets. There is no data in the lab class, where EINO is created, just a set of connections. A thread was found to be the most dependable way to create a loop inside the lab; the thread also allows a

delay to be added into the checking of input data. A student can only change data at a certain rate and it serves no purpose to look for a changed input data ten times faster then the student can change it. EINO's delay is roughly one second. Inside the thread are certain EINO functions that are used to check the values in the input components. In essence, there are two types of input components: there are buttons and non buttons. The button input components are the microphone button and the file reader button. EINO only requires of these inputs to be notified when the button is pressed. The functions that check to see if a button is clicked, take in a copy of either the microphone class or the file reader class. In both classes, there is a Boolean variable that is set to true if the button has been clicked. After checking if the button has been pressed, the Boolean variable is reset to false. The non button input components include the sliders and the pixel mapper. With these components, the value of the component matters. The functions that check the values of these components takes in a copy of the component and stores it in a simplified version of the slider class or pixel mapper class that mostly contains the values. The next time the values are checked, the new values are compared to the old values. If they have changed, the new values are stored. Otherwise, a count is augmented until a preset number on times before relaying the information of the component being modified. This count makes sure that the change detected is not some middle value as the student makes changes to component. The count currently being used in EINO is equal to a two second delay. Once an input component change is detected, the name of the component and the value is stored within a list of the last ten input component changes made by the student. It is this list that the goals and cases use as a search space.

5.3 Step Two of Implementation EINO Tutoring System

The second step in the implementation of the EINO tutoring system was to create the goal system. The goal system is designed in a way that when the student meets all the goals, the student has completed most of the steps in the lab manual. The size of each lab is different and therefore the amount of goals that each lab can process varies. A single goal consists of five items. The first item is a goal number. This number is used as a reference to the goals place in the goal array. The second item is the caption. The caption is the name of the input component. Every name is different. It does not matter how many versions of the same components there are in the lab. The third item is the value; as discussed above, the buttons have no values. They were all assigned a default value of "1.0" in the lab. The last two items deal with the idea of if the single goal is part of a group. The fourth item is the next goal. If the goal has a next goal, meaning it is part of a group, the next goal number is listed. Otherwise, the next goal item is defaulted to a <-1>. The fifth and last item is the previous goal. If the goal has a previous goal, meaning it is part of a group, the previous goal number is listed. Otherwise, the previous goal item is again defaulted to a <-1>. All of the lab goals are stored within a text file.

The goal search was the most difficult to create. Originally, the goal search was to be of the case based reasoning (CBR) system. The problem that occurred is that the case based reasoning system preferred the single goal over the group of goals. If all the goals were single goals, then the CBR system and the goal search could be combined. The goal search initially starts off by creating a list of possible unfinished goals that the user is trying to complete. If the list of possible goals contains only single goals, then the first goal on the list is marked as

completed. If the list of possible goals contains at least one group, then the goal search goes into a secondary search mode. The secondary search is not started until a new input component is added to the list. In the secondary search mode, the next goal item of the possible goals list is compared to the newest item in the history list. A new possible goal list is created. The secondary search used until the possible goals list only contains one goal set. Once the goal set has been completed, all the goals in the set are marked as completed and the whole goal search starts from the beginning. If the possible goals list is empty, the first entry from the previous possible goals list is used suggest a course of action to student in the form a "Just in Time" hint. After the "Just in Time" hint has been shown, the student has one chance to perform the action that the system suggested. If the student does not perform the action that the system suggested, then the goal search immediately performs the phase one goal search of the last item added to the history list and the process begins all over.

5.4 Step Third of Implementation EINO Tutoring System

The third step in the implementation of the EINO tutoring system was to create the CBR system. The CBR system is used to detect patterns in the student's inputs into the system and determine if the student requires help to solve the problem. The number of cases in each lab can vary. A case is made up of three items. The first item is the number of items in the case. This number is the minimum number of items that must be in the history list before a comparison is made. The second item that creates a case is the case number; this number is used as a reference to the case's place in the case array. The third and last item is a list of items. This list of items is a possible history list. Finally, all of the lab cases are stored within a text file.

The case search is fairly easy to program. Every time an item is added to the history list a case search is performed. Each case is presented one at a time. The first step in the case search is to check the minimum number. If the number of items in the history list is equal to or greater than the case's minimum number, then step two is performed. Otherwise, the search for that case is aborted. In step two, a comparison is made between the history list and the list of items in the case. If the comparison shows that the history list and the list of items in the case are at least a ninety percent match, the appropriate EINO help is displayed. If no cases match the history list, then nothing happens.

5.5 Step Fourth of Implementation EINO Tutoring System

The fourth and last step in the implementation of the EINO tutoring system was to create an interface that was easy to read and use by the student. Flash was chosen for this task as it offers a clean interface and reuse of components already built. A short animation was designed and built to give every lab's help a sense of uniformity. The main menu contains a list of component buttons in the lab that the student might have a question about and a link for help if the lab is not functioning properly. The "small" topic components buttons link to a different frame in the flash to display relevant information about the topic to the student. The "big" topic components buttons also link to a new frame in the flash to display the relevant information. However, after the student is done with the information, questions relating to the given information are presented to the student one by one. If the student misses the question, the answer is given and a new question is presented. Otherwise, the student has answered the question correctly and is then taken back to the main menu. The flash movie has one variable that is set on the load of the web page. The variable, "md," instructs the flash movie what frame

to load first. This means EINO can instantly pop up with a help message like “Do not forget to turn the lab on” without the student fumbling around a menu system.

5.6 Summary

The EINO intelligent tutoring system is pieced together around the UCF Infinity Web Applets. The design of the UCF Infinity Web Applets makes it easy to put multiple labs together with recreating every component over and over. This design means that the implementation of EINO must work around the restrictions imposed by the design. One of the hardest parts in the creation of EINO was to find a way to check values of components intermittently in the runtime of the lab. This was a necessary first step in the creation of EINO. The second step in the implementation of EINO was to create a way to know if the student had completed the lab. This step is where the goal system becomes useful. The goals are read from a file and loaded into a goal array. As a changed input is detected a goal search is initiated to look for possible goals the student is working towards. The third step in the implementation of EINO was to create a way to detect if the student needs help before the student asks for it. A CBR system was implemented to perform pattern recognition on the student’s inputs into the system. Again the cases are loaded from a file and a case search is initiated as a changed input is detected. The reason the goals and cases both are loaded from a file is that the file allows for the quick addition of goals or cases. The last step in the creation of EINO is to create a clean interface the student can easily read and interact with. Flash was used as it solves both of these problems. The flash movie also can be programmed to control how the student interacts with the interface. By following the four step progress, the EINO tutoring system can be recreated to work with any system that has the same architecture as UCF Infinity Web Applets. If the

system is not compatible with the UCF Infinity Web Applets architecture, implementing a version of EINO is possible but not recommended. A new subject can be applied to EINO as long as the architecture is similar to the UCF Infinity Web Applets.

CHAPTER 6: TESTING

Within this chapter there is a description of a series of experiments designed to confirm the hypothesis stated in Chapter 3. There will be a set of seven tests performed on the EINO intelligent tutoring system. The main factors that need to be validated are:

1. The functionality of the EINO intelligent tutoring system
2. The helpfulness of the EINO intelligent tutoring system

The seven tests that are to be performed tests performed on the EINO intelligent tutoring system are:

1. The Not On Test
2. The Single Goal Test
3. The Group Goal Test
4. The Just in Time Hint Test
5. The Case Test
6. The Questions Test
7. The Real Life Simulation

The functionality of the EINO intelligent tutoring system is tested to determine if EINO can perform as it is expected to perform. This requirement is tested in all of the seven test situations that will be performed. The helpfulness of the EINO intelligent tutoring system is tested to determine if EINO is truly helping the student understand the basic concepts of physics. EINO may be providing comments to the student, yet if the comments do not make a difference in the students' understanding of the material, then EINO may be doing more harm then good in

helping the student to study and understand the basic material. Each of the following tests was performed on the same version of code. In all of the tests, only one copy of each experiment was running at a time. The student is only expected to be working on one lab at a time and therefore is only to have one lab running at a time.

6.1 The Not On Test

The first test presented is the Not On Test. The purpose of this test is to determine if EINO can sense whether or not the lab is in the on state. Without this functionality the student may become disoriented due to the fact that the student is providing input into the system and nothing is occurring. The procedure order to perform this test is as follows. The applet is started. The experiment tab is click to display the experiment. Each lab has three tabs. On start up, a tab with the theory is displayed. In order to get to the experiment the experiment tab must be clicked. By default, the lab starts off in an off state. With the lab still in the off state, the user clicks on an input component.

Table 1: Results of the Not On Test

Components	Results
Lab 2-1	
Slider	**Output Displayed**
Microphone Button	**Output Displayed**
Lab 4-2-4-1	
File Reader	**Output Displayed**
Pixel Mapper	**Output Displayed**
Lab 6-2-1	
Keyboard Button	**Output Displayed**
Slider	**Output Displayed**

By changing the input component with the lab in the off state, this test is testing the action of the user forgetting to turn the lab on. This action by the user should trigger a response by EINO to inform the user that the lab is in the off state. See Figure 5 for EINO's response. Table 1 illustrates the results of the Not ON test. This test helps prove functionality of the EINO intelligent tutoring system.

6.2 The Single Goal Test

The second test performed on the EINO intelligent tutoring system is the Single Goal Test. This test is designed to test whether or not EINO can detect the completion of a single goal. Without this functionality EINO will not be able to determine if the student has completed the steps as described in the lab manual. The procedure order to perform this test is as follows: the goal file for each lab is changed to provide a single goal for each input component to test for. Each lab is started one at a time.

Table 2: Results of the Single Goal Test

Components	Results
Lab 2-1	
Slider	**Single Goal Matched**
Microphone Button	**Single Goal Matched**
Lab 4-2-4-1	
File Reader	**Single Goal Matched**
Pixel Mapper	**Single Goal Matched**
Lab 6-2-1	
Keyboard Button	**Single Goal Matched**
Slider	**Single Goal Matched**

The experiment tab is click to display the experiment. Each lab has three tabs. On start up, a tab with the theory is displayed. In order to get to the experiment the experiment tab must be clicked. The on / off button is clicked by the student in order to change the state of the lab from off to on. Each input component is automatically changed. EINO only responds to new inputs provided by the student. This is why an input change is required. EINO responds by simple text output stating which goal is matched. Table 2 shows the results of the Single Goal test. This test again assists in proving the functionality of the EINO intelligent tutoring system.

6.3 The Group Goal Test

The third test performed on the EINO intelligent tutoring system is the Group Goal Test. This test is designed to test whether or not EINO can detect the completion of group goals. Again this functionality is crucial for EINO to determine if the student has completed the steps as described in the lab manual.

Table 3: Results of the Group Goal Test

Labs	Results
Lab 2-1	**Group Goal Matched**
Lab 4-2-4-1	**Group Goal Matched**
Lab 6-2-1	**Group Goal Matched**

The procedure order to perform this test is as follows. The goal file for each lab is changed to provide only group goals for each lab as the test is opened. Each lab is started one at a time. The student is only working on one lab at a time, according to the guidelines established in EINO

and therefore only to have one lab running at a time. The experiment tab is clicked to display the appropriate experiment. Each lab has three tabs.

When the lab first begins, the theory tab is displayed. The experiment tab needs to clicked to display the experiment. The on / off button is clicked in order to change the state of the lab from off to on. Each input component is changed automatically in an order that is defined in the goals file. EINO responds via outputting a simple text line stating which goals were met in the experiment. Table 3 shows the results of the Group Goal Test. The functionality of the EINO intelligent tutoring system is tested in this test.

6.4 The Just in Time Hint Test

The fourth test performed on the EINO intelligent tutoring system is the "Just in Time Hint" test. The purpose of this test is to determine if EINO can offer pertinent "Just in Time" hints to the student. A "Just in Time" hint is offered when the student is working through a group of goals and instead of a successful outcome to the problem, the student makes a mistake. EINO may then suggest to the student an input component to change. This test will help prove the functionality of the EINO intelligent tutoring system. The procedure order to perform this test is as follows. The goal file for each lab is changed to provide only group goal for each lab as the test is initially opened. Each lab is started one at a time. The student is only working on one lab at a time, according to the guidelines established in EINO. The experiment tab is clicked to display the appropriate experiment. Each lab has three tabs. On start up, a tab with the theory of the presented experiment is displayed. In order to get to the experiment, the experiment tab must be clicked. The on / off button is clicked in order to change the state of the lab from off to on. Each input component is changed automatically in an order that is defined in

the goals file except for last goal on the list. A different input is changed instead of the last one on the goal list. EINO responds via outputting a help message suggesting which component to change. See Figure 6 for a picture of a possible output. Table 4 shows the results of the "Just in Time Hint" test.

Table 4: Results of the Just in Time Hint Test

Components	Results
Lab 2-1	
Slider	**Just in Time Hint Offered**
Microphone Button	**Just in Time Hint Offered**
Lab 4-2-4-1	
File Reader	**Just in Time Hint Offered**
Pixel Mapper	**Just in Time Hint Offered**
Lab 6-2-1	
Keyboard Button	**Just in Time Hint Offered**
Slider	**Just in Time Hint Offered**

6.5 The Case Test

The fifth test performed on the EINO intelligent tutoring system is the Case Test, which is used to test effectiveness of the case based reasoning system employed in EINO. The procedure order to perform this test is as follows: the case file for each lab is changed to provide simple cases for each of the input component in the lab. The cases tested are the repeated use of a single component. Each lab is begun one at a time as stated in the initial conditions. The experiment tab is clicked to display the experiment. Each lab has three different tabs. On start up, a tab with the theory is displayed. In order to get to the experiment the experiment tab must be clicked. The on / off button is clicked by the student in order to change the state of the lab

from off to on. Each input component is changed one at a time repeatedly for the number of times stated in the case file. EINO responds by presenting the student with information about the component. See Figure 3 for a sample output. Table 5 contains the results of the Case Test. By proving the case based reasoning system in EINO also demonstrates the working functionality of EINO. This indicates that the EINO concept is feasible and is partly proved due to the success of the case based reasoning system.

Table 5: Results of the Case Test

Components	Results
Lab 2-1	
Volume	**Case Information displayed**
MSec to Display	**Case Information displayed**
Microphone Button	**Case Information displayed**
Lab 4-2-4-1	
File Reader	**Case Information displayed**
Pixel Mapper	**Case Information displayed**
Lab 6-2-1	
Keyboard Button	**Case Information displayed**
Encoder	**Case Information displayed**
Decoder	**Case Information displayed**

6.6 The Questions Test

The sixth test performed on the EINO intelligent tutoring system is the Questions Test. This test is designed to verify that questions are asked after EINO presents a “big” topic to the student. These questions are designed to help the student comprehend the subject matter.



Figure 10: The General Help Button

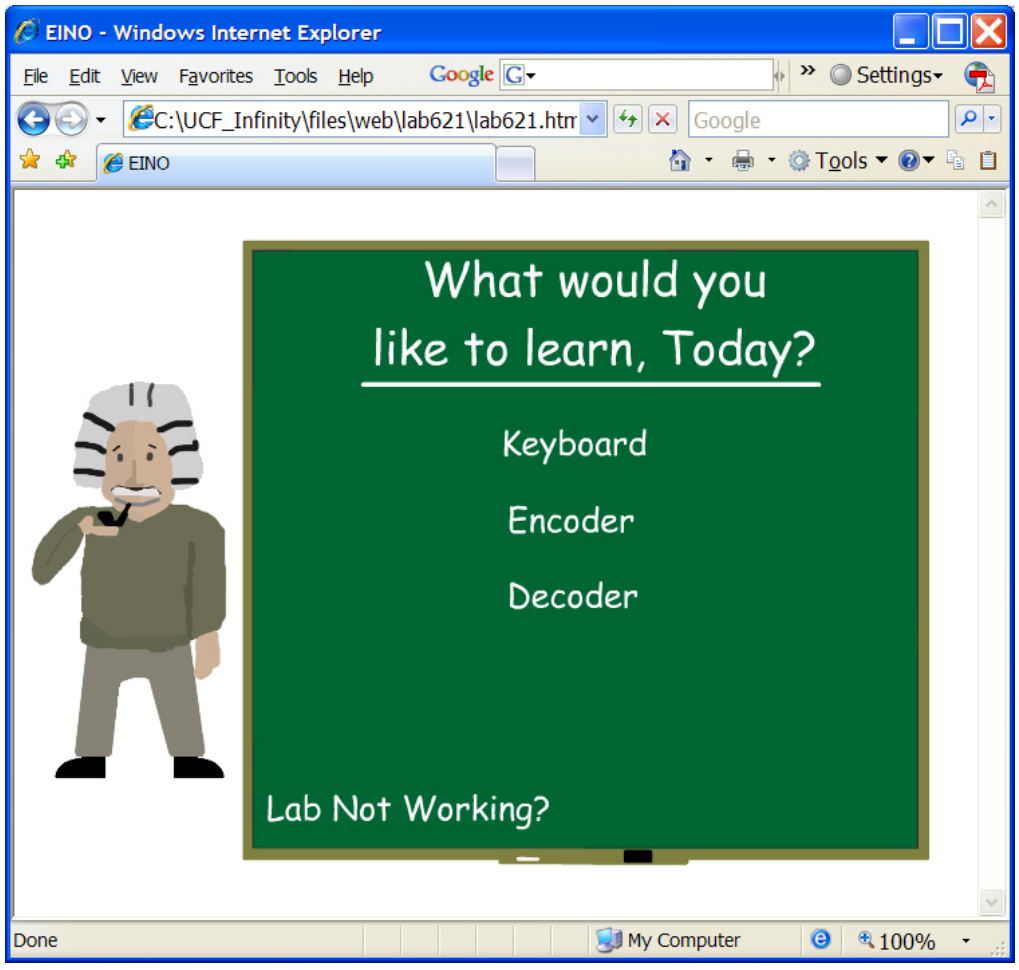


Figure 11: Example General Help Menu

Without the added questions, the student might not fully grasp a concept that could be causing difficulty. To perform this test, each lab is started one at a time as stated in the initial conditions. In the lower right part of the lab is a general help button. See Figure 10 to check what this button looks like. By clicking on this button, EINO displays a general help menu, see Figure 11, of components used in the lab.

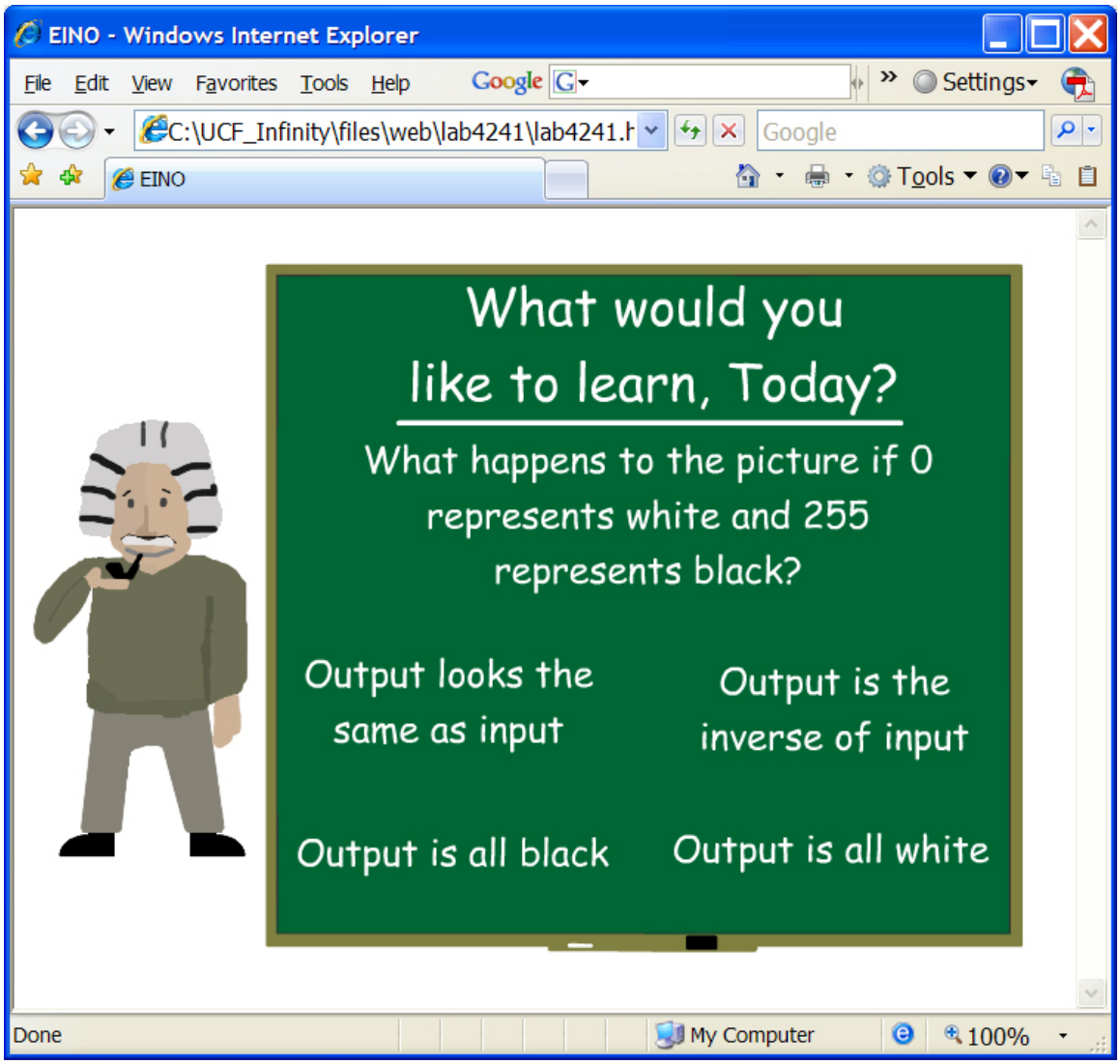


Figure 12: A Sample Question

A big topic button, like the pixel mapper, is pressed. EINO presents some key information about the topic. After the information is presented, questions should be asked until the student answers them correctly. See Figure 12 for a sample question. Table 6 shows the results of the Questions test. The functionality of the EINO intelligent tutoring system is tested in this test.

Table 6: Results of the Questions Test

Components	Results
Lab 2-1	
Spectrum	**Questions Asked**
Lab 4-2-4-1	
Pixel Mapper	**Questions Asked**
Lab 6-2-1	
Encoder	**Questions Asked**
Decoder	**Questions Asked**

6.7 The Real Life Simulation

The seventh and final test performed on the EINO intelligent tutoring system is the Real Life Simulation. This test provides information on the helpfulness of EINO along with how well EINO functions in the actual world of the classroom or lab. This test was verified by five test subjects who consented to test the functionality of each lab. Most of the test subjects were within the first two years of their college education. The reason these students were chosen is that these are the type of students that are the targeted audience of the EINO intelligent tutoring system. The procedure to perform this test was standardized as following: every test subject was given a copy of the step-by-step instructions for each lab from the lab manual and unlimited time to “play” with each lab.

Lab:_____ Name:_____

Did Eino activate? Yes / NO How many times?_____

What did you do to activate Eino?

_____
_____
_____
_____

What was the message?

_____
_____
_____
_____

Was it helpful? On a scale of 0 to 10 with 10 being the best.

_____
_____

If not helpful, what could be done to make Eino better?

_____
_____
_____
_____

Was Eino easy to understand?

_____
_____

Do have any previous knowledge of the subject? If so, how do you acquire this knowledge?

_____
_____
_____
_____

Figure 13: The Survey

After the test subjects finished each lab a brief survey about the helpfulness of EINO was administered. The survey is seen below, Figure 13, consisted of eight (8) questions. From these completed EINO surveys, it was discovered that the most common help message presented to the student is "the lab is Not On" message, Figure 5. There were two significant questions on the survey that dealt with the helpfulness of EINO. The first question dealt with the understandable of EINO. Was EINO easy to understand? Ninety three percent of the surveys responded that EINO was easy to understand. The other seven percent did not fill out the rest of the survey due to the fact that the test subject did not trigger any response from EINO on the one lab. The second question dealt strictly with the helpfulness of EINO. The question asked the test subject to rank the helpfulness of EINO on a scale of one to ten with ten as the best. The overall results of this question can be seen in Table 7.

Table 7: Overall Results of Helpfulness of EINO

EINO Helpfulness Score	Percent
10	53%
9	13.3%
8	13.3%
7	7%
0	13.3%

This signifies that almost eighty percent of the test subjects gave the helpfulness of EINO an eight or higher. There were two zeros that were given by the respondents. The first zero was given because the test subject never triggered a response from EINO on one of the labs. The other zero was given because one student thought that EINO kept offering help a little too often and found it annoying rather than helpful. The EINO helpfulness score can be broken down by

each lab. Table 8 shows the helpfulness score for lab 2-1. Table 9 shows the helpfulness score for lab 4-2-4-1 and Table 10 shows the helpfulness score for lab 6-2-1.

Table 8: Helpfulness Score for Lab 2-1

EINO Helpfulness Score	Percent
10	40%
9	20%
8	20%
7	20%

Table 9: Helpfulness Score for Lab 4-2-4-1

EINO Helpfulness Score	Percent
10	80%
8	20%

Table 10: Helpfulness Score for Lab 6-2-1

EINO Helpfulness Score	Percent
10	40%
9	20%
0	40%

6.8 Summary

The EINO intelligent tutoring system had two chief factors to prove. The first factor is the functionality of the system. Does the EINO system perform as expected? This factor is proven in the Not On Test, Single Goal Test, Group Goal Test, Just in Time Hint Test, Case Test, and the Questions Test. Each of these tests were designed to test part of the functionality of the EINO system. When the results of the tests are combined, a fully functioning EINO

intelligent tutoring system is proved. The second factor to prove is the helpfulness of the EINO system. The premise resides in the necessity of a fully function tutoring system. However, there is no single way to prove a system is helpful as the term helpful has different meaning to every individual. The way that was used to prove the helpfulness of the EINO system is by having multiple test subjects fill out a survey. Almost eighty percent of the test subjects gave the helpfulness of EINO an eight or higher on a ten (10) point scale. Another part of the helpfulness of the EINO system is the comprehension level of the system. There could be a chance that the EINO system is helpful yet difficult to understand. This survey also attempted to ascertain how easy EINO is to understand to students involved in the use of the program. According to the survey ninety-three percent of the test subjects responded that EINO was easy to understand. The tests have proved that the EINO intelligent tutoring system functions properly and that a good percentage of users find EINO “helpful”. These findings, validate the two factors stated at the beginning of this chapter regarding the functionality and usefulness of the EINO intelligent Tutoring system for use in the classroom or lab situation.

CHAPTER 7: SUMMARY, CONCLUSION & FUTURE WORK

A summary of the research is conducted in this chapter by readdressing the problem statements made in chapter three and examining them in detail. The possible solutions to the problems raised are listed along with their usefulness. Conclusions are then drawn from the testing completed in Chapter 6. The last section of this chapter proposes future directions and work that could extend the work described in this thesis.

7.1 Summary

Overall, the concept of an intelligent tutoring system utilizing a computer is not a new idea. In fact, the ANDES system has been in continuous development for over ten years and is still undergoing changes. With each new development and application of an intelligent tutor, the design and process of these systems must also change.

Chapter Three stated the problems that this thesis has addressed. This section reviews these statements and discusses how these problems were resolved. As discussed in chapter three, there are two key types of help or instruction. The first type of help that is offered is conceptual assistance. Conceptual help involves helping the student understand the theory behind what is occurring in the particular experiment. The EINO intelligent tutoring system solves the problem of providing conceptual help to the student in several of ways. The first way is the general help button. At a click of a button the student has access to a list of the major components in the particular lab. Then by clicking on the subject matter the student is having problems with solving, the student has a resource of conceptual help in any experiment. The first resource the student has access to is the theory tab, which is displayed by default on start up of

the lab. The second resource is the general help button and the third resource is the Infinity textbook, which has all the theory for all of the labs in a hard copy format, in the order which they are displayed. All three ways presents the information to the student a different way. A second way EINO provides conceptual help is with the case based reasoning system. The case based reasoning system in EINO is looking for patterns in the students' input into the system. Imagine a student working through an experiment and the student is struggling to understand what the purpose of a certain component is. The student may keep changing the certain component in order to see the effects of what the component does. After a predefined amount of changes, EINO displays a message of describing the component and what the component does. This is the second way EINO provides conceptual help to the student.

The second type of assistance is procedural help. Procedural help involves helping the student answer the question "What do I do next?" Procedural help within the University of Central Florida Infinity Web Applets (UCF Infinity Web Applets) is a tricky matter. The student will be using a lab manual. The lab manual provides for the student a step-by-step instruction on how to complete the lab. Even with the lab manuals, a student could have problems. This is why EINO still provides some procedural help in the form of a "Just in Time" hint. A "Just in Time" hint is produced when EINO knows what group of goals the student is working through and the student makes a mistake by changing the wrong input. The "Just in Time" hint will suggest to the student to change a certain input component. For an example, see Figure 6.

When both of these help systems are combined to function as one, the student is left with place to turn for a full range of help. Thus, EINO solves the original problem statement of

creating a place for a student to turn to when they need help and there is no instructor or lab aide available.

7.2 Conclusion

The thesis statement, in chapter three, stated the hypothesis as: *By providing a case based reasoning and rule-based help system into the web experiments created by UCF, students will have an on-line site to reference in case they require any help with an experiment and be more willing to attempt the experiment.* The experiments performed support the hypothesis completely. There were, however, results that were not anticipated. The most common message from EINO that the test subjects encountered during the real life simulation was "The Lab is Not On." This common message leaves one asking is the on / off button just one more step that the student must pass through in order to access the intelligent tutor? And if so, could and should this step be eliminated? Also worth noting, the real life simulation was one of the first times that the student user was an individual outside the design team. This fact created some interesting situations during the real life simulation. The most telling quote from a student/tester was "… the interface [UCF Infinity Web Applets] had many design problems because matching of the attempt to match the lab manual. However, EINO was able to balance the difference."

The EINO intelligent tutoring system is not perfect. The biggest problem that has been discovered in EINO involves the manner in which EINO delivers messages to the student. Each message is displayed in an Internet Explorer (IE) window. Every time EINO delivers a message, a new IE window is opened. This means the old message is still left opened in the background. This flaw could possibly confuse the student. In the student's mind EINO could

be suggesting to try two items at the same time, since there are two items opened. The problem that has occurred is a simple one, the student has neglected closing an open window with the earlier message. This is only a minute possibility as most messages from EINO only show the required information. In the real life simulation none of the test subjects had a problem with the leaving of EINO's messages open.

7.3 Future Work

What can be done in the future to make a more efficient version of EINO? The most obvious answer would be to create a better interface for EINO. The flash interface works well but this design involves combining two very different programming languages. The whole process can become overwhelming and confusing to the creator. The solution for this can be accomplished by incorporating the interface and web applet into one program instead of the two window approach used by EINO. Although the two window approach works, it can become frustrating switching back and forth between windows. When both windows are combined, the tutoring messages would seem more fluent from the lab and could respond to the student's actions more rapidly. The lab's interface would have to be expanded to allow room for a dialog area in which EINO could communicate effortlessly with the student. This would also allow for a redesign of the layout of each of the lab's interface which would allow the lab's interfaces to become more uniform. Another option for the interface would be the use of Open GL. Open GL would allow a three dimensional professor to guide the student through the process. This upgrade would have no affect on the artificial intelligence of EINO but would better the overall appearance of the system. The student might be more willing to follow a three dimensional

professor than a two dimensional one that is located in another window simply because it looks better.

Implementing a version of EINO into a system that does not follow the UCF Infinity Web Applets architecture could be possible but not recommended. EINO was designed to fit within already built and already functional components. This challenge led to the piecewise design of the EINO system. A more concise design with greater potential can be created if the tutoring system and the web applets are built at the same time. However, the EINO system has proved to function well in its current form and design.

REFERENCES

[1] A. S. Gertner, C. Conati, and K. VanLehn, "Procedural help in Andes: Generating hints using a Bayesian network student model," presented at Fifteenth National Conference on Artificial Intelligence, 1998.

[2] Q. Simulations, "Quantum Tutors - Learning Principles of AI Technology," 2002.

[3] G. Orsak, S. Wood, S. Douglas, D. M. Jr., J. Treichler, R. Athale, and M. Yoder, *The Infinity Project Engineering our Digital Future*. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[4] K. VanLehn, C. Lynch, K. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill, "The Andes Physics Tutoring System: Lessons Learned," *International Journal of Artificial Intelligence in Education*, vol. 15, 2005.

[5] A. Aamodt and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," *AICom - Artificial Intelligence Communications*, vol. 7, pp. 39 - 59, 1994.

[6] M. Gu and A. Aamodt, "Dialog Learning in Conversational CBR," presented at Nineteenth International Florida Artificial Intelligence Research Society, Melbourne Beach, Florida, 2006.

[7] P. M. Regan and B. M. Slator, "Case-based Tutoring in Virtual Education Environments," presented at Collaborative Virtual Environments Proceedings of the 4th international conference on Collaborative virtual environments, 2002.

[8] K. P. Jantke, G. Degel, G. Grieser, M. Memmel, O. Rostanin, and B. Tschiedel, "Technology Enhanced Dimensions in e-Learning," presented at International Conference on Interactive Computer Aided Learning, 2004.

[9] K. P. Jantke and R. Knauf, "Didactic design through storyboarding: standard concepts for standard tools," presented at ACM International Conference Proceeding Series; Vol. 92 Proceedings of the 4th international symposium on Information and communication technologies, 2005.

[10] R. Nkambou and F. Kabanza, "Designing Intelligent Tutoring Systems: A multiagent Planning Approach," *ACM SIGCUE OUTLOOK*, vol. 27, pp. 46-60, 2001.