AN ASSESSMENT AND EVALUATION OF COMPUTER SCIENCE EDUCATION

Jennifer R. Parham
University of Montana
Computer Science Dept. SS401(MCSI01)
Missoula, Montana 59812-5256
jrparham@cs.umt.edu

ABSTRACT
This research shows an audience of fellow computer science teachers that there is benefit in testing college student's scientific reasoning skills and that the students increase their logical thinking skills in computer science classes when they are presented with real world problems, engaged in group activities, and given hands-on experiments.

Students' scientific reasoning skills from several different computer science courses are analyzed and tested before and after taking the computer science course to measure their improvement in logical thinking. The higher order thinking skills test results are compared, and the overall class and individual student improvement between the classes that use different teaching strategies are analyzed. Also, the test results are compared with students' ability to perform adequately in a computer science course. This information is valuable to advisors when placing students in computer science courses.

In the assessments of students' entering knowledge and evaluations of students' thinking improvement, the students' individual higher order thinking skills as well as their overall level of cognitive development are analyzed. Through the analysis of their individual skills and level of thinking, one can expand conclusions about the students' and teacher's performance over time.
INTRODUCTION

Society is becoming increasingly more dependent on research and development in the mathematical and computer sciences for technological advances; however, education systems are not producing enough students interested in such jobs. According to the national science foundation's science and engineering 2002 indicators, 30% of bachelors and 35% of master's students major in a science and engineering field. However, less than 8% of the US's total workforce is employed in a science and engineering job, and less than 3% actually do science, while the other 5% perform duties such as administration, management, etc. Positively, the 2002 indicators reveal that half of the computer science and math majors work in science and engineering fields (National Science Board of Education, 2002).

Not only are institutions not producing enough students interested in the nation's need for science and engineering but the indicators show that many more freshman have the intent to major in science and engineering disciplines than actually finish. The problem is that few freshmen have the intent to major in computer science and/or mathematics versus all other scientific and engineering disciplines, but jobs for computer specialists are predicted to increase more than any other science and engineering job between the years 2000 - 2010 (National Science Board of Education, 2002). Many CS students have previous degrees, and students return to school to major in computer science to increase their chances of finding a well paying job. The lack of students working in science and engineering fields is a direct result of colleges, high school, and middle school educators either not sustaining interest from students to enter scientific disciplines or improperly preparing students for the next level of learning and reasoning needed to progress through future stages of the sciences. As one author points out, "A third of all eighth-grade ..., while fewer than half of U.S. high-school seniors showed the consistent grasp of fractions, decimals, and percentages expected of competent seventh-graders" (Wilson and Bennett, 1994, p. 5). Dr. Mark Cracolice, from the University of Montana Chemistry department shows evident results when he administered higher order thinking tests to the University of Montana Chemistry freshmen; 75% of the students have insufficient formal thinking skills that should be fully developed by adolescence. Later research finds very similar results in the introductory computer science classes. Since the Regan's Nation at Risk document (1983), high school graduates' higher order thinking has remained an issue among educational leaders (p. 8). Formal thinking skills are what enable us to understand and apply computational and logical reasoning that is used in the mathematical sciences, such as computer science.

Higher order thinking skills are the skills that allow you to logically reason and can be categorized into two main subcategories, concrete and formal thinking. These subcategories are derived from Jean Piaget, who developed a philosophy of cognitive development among children. According to Piaget, an individual should develop their concrete thinking during their later childhood (8-12) and formal thinking during their later adolescence (13-16), with transitional periods between (Richmond, 1970, p. 99). Transitional periods are referred to as low-level and high-level transitions. Within the concrete and formal thinking levels, there are many different types of skills. The only thinking skill associated with concrete thinking is conservation, such as conservation of weight, conservation of length, conservation of area, etc. However, there are several different types of reasoning skills classified under formal thinking.
including control and exclusion of variables, ratio and proportion, compensation and equilibrium, correlation, probability, combinatorial, and hypothetico-deductive. Some skill types are used more than others in computer science, but an overall formal level of thinking is required. The more common skills used in computer science are combinatorial, hypothetico-deductive, control and exclusion of variables, and correlation reasoning.

Following are examples of questions testing higher order thinking skills. The student is presented with a question and multiple choices to choose from sometimes followed by multiple reasons for the answer. The lowest level of higher order thinking is the concrete level, and an example of a conservation of volume question from the concrete level is "Ryan has sixteen wooden blocks of identical size and shape. He arranges eight of the blocks into the shape of a square. He then arranges the remaining eight wooden blocks into the shape of a rectangle. Which of the following statements is true?" There are several types of skills from the formal thinking level, but a simple example illustrating combinatorial reasoning is "Brian is making a five-course meal for his girlfriend. The courses are appetizer (A), entree (E), main course (M), vegetable (V), and dessert (D). He knows that the appetizer is served first, and the dessert is served last, but he is unsure about what order to use for the entree, main course and vegetable. How many different ways can Brian serve the meal, if he uses each course only once?"

The astonishing large percentage of University of Montana students that do not show the development of formal thinking skills presents a problem to university instructors. College instructors are uneducated in helping to develop these skills, and traditional institutional teaching is designed around rote lecturing, assignments, and tests for those students who have already developed the formal thinking necessary to understand and perform adequately on their own. College instructors need to address this problem and help prepare our students to become future scientists and self-directed learners.

Before one can help correct the problem, one must determine there is a problem among a variety of our students. The research applies to CS195, a beginners' FORTRAN programming class, CS471, an upper level scientific computing class, and CS101, a beginners' Visual Basic programming course, to use as an assessment and evaluation. First in each course, the students are tested for higher order thinking skills. The introduction FORTRAN course is used for the controlled experiment and incorporates new pedagogical methods into the curriculum. Then, the students are retested to assure the instruction facilitates progress. The class is new to the course listings, so there are only 12 students who took the experimental FORTRAN class. However, the scientific computing and Visual Basic courses are offered yearly and use a more traditional method of teaching computer science. Therefore, there is always a high enrollment in the Visual Basic course with 46 students in this research. Even though there is a difference in the number of students enrolled in the experimental versus the traditional introduction course, the pedagogical methods in this research apply to any classroom size with the appropriate planning. This research paper provides a description of the approaches taken to assess the issues in computer science education and help assist in finding an effective way of teaching computer science.
BACKGROUND

Despite the research conducted on effective ways of educating students at the primary, secondary, and higher education levels, there has not been much research in applying new methodologies of teaching to computer science and assessing the effect the new method of teaching has on students' learning. The research in this paper is only reflective at the higher education level, but the same techniques can be applied when teaching computer technology at any educational level. Adults and children learn by examples and experiences, and a teacher's job is not to pour information about a subject into students' head so the information can be lost through their ears, but rather facilitates the students' ability to learn and think on their own in the future, when they are presented with problems and challenges within a given subject. The overall aim of education must be to nurture the power of thought (Wees, 1971, p. 59). Computers are the best example of a subject that presents more problems and challenges when nobody is around to help. Therefore, computer related subjects are taught best through hands-on experience that exposes students to some likely problems before they become stranded on their own. "Application breeds learning" (Negal, 1994, p. 21).

Many new approaches to teaching, which have all led to increased enrollment and class grades, usually contain application of the concept, group discussions/class work, and hands on experience (National Science Board of Education, 2002). One reason for approaching teaching from many angles comes from the idea that all people learn differently, and therefore teaching needs to be treated as a variable always capable of changing. The way to approach the diversity in learners is with variety of teaching (Draves, 1997, p. 5). Active learning, collaborative learning, inquiry based learning, and discovery learning pedagogies naturally lend themselves to research intensive and experimental disciplines, such as computer science. According to national education interests, recommended science and engineering reforms include a high priority on undergraduate education and research, making faculty aware of new teaching methods, and incorporating interdisciplinary teaching into the curriculum. Institutions currently teach disciplines as they have been taught for years, without incorporating new scientific findings or new methods for the way scientific research is conducted today. This style of teaching does not help prepare students for life outside of college, and instead, it hinders students' ambitions and competency.

CASE STUDIES

A computational science FORTRAN programming, CS195, course is the first test set selected for collecting data, conducting research, and analyzing the results of the research. After data is collected from the first test set using progressive teaching strategies, the data is compared to other computer science classes from different levels that use other methods of teaching. An upper level scientific computing course, CS471, is chosen to compare the skills held by students already attaining a scientific degree versus those just beginning in a scientific field. The other computer science class selected for analysis is a beginners' Visual Basic course, CS101 that is for the non-computer scientist. This course has a wide variety of students varying in levels and disciplines.
Selecting the Test Sets

First, an experimental test set is chosen for implementation of alternative teaching strategies in a classroom of mixed grade levels and disciplines, while the other test sets are not controlled and more traditional. Both intro-level classes contain a variety of students from different disciplines, except the upper level scientific computing course. The two introductory CS courses are chosen to show differences in learning using progressive teaching, and the upper level scientific computing course is used to analyze the differences between students with known computer programming knowledge containing students with mild interests in learning about computer science.

Collecting the Data

The students in the experimental FORTRAN computation science course, CS195, are given a test of scientific reasoning. The test of scientific reasoning tests students for higher order thinking skills, and the test given before the students began CS195 is a revised version of Lawson's 1978 test (Lawson, 2000). An online version of the clinical tests is developed as a tool for conducting this research. Not only does this help in future testing, but also this allows many questions from different tests to be combined for more variety. For congruency, both tests given before and after the course have a total of fourteen possible points, and the total questions answered correctly out of fourteen categorizes a student's level of thinking: 11-14 (Formal), 7-10 (High Trans), and 3-6 (Low Trans), 0-2 (Concrete). The data is used for a prediction and assessment of the students' performance in the class due to the affect the new teaching method has on the development of the students' thinking skills.

The students from the upper level scientific computing and beginners' Visual Basic test sets are given an updated version of the online higher order thinking skills tests (University of Montana, 2002). The online test is a combination of several scientific reasoning tests combined. All the higher order thinking skill types are tested except hypothetico-deductive reasoning. This skill is not tested because there are not enough hypothetico questions for the current design of the online test. The current online version of the HOTS test presents a student with a medium level question from a random skill, and if the student answers the question correctly, then he/she is presented a hard level question of the same skill type. The student completes a skill type when he/she answers two of any level (easy, medium, or hard) correctly with two failed attempts at a harder level, and if the student fails to answer two of all the levels correctly, then he/she is not rated at any level for the skill. The higher order thinking skills test is given to the scientific and Visual Basic classes to analyze the thinking skills held by upper and lower level computer science students and how the students' skills are influenced by the computer science class and instruction.

Conducting the Research

To find out if there is an effective way to teach computer science other than using traditional methods, new teaching practices are implemented into the CS195 FORTRAN
classroom. Three new additions are made to the classroom instruction. One addition to the computer science course is the addition of hands-on experience through a lab experiment conducted each week in the classroom. Another addition to the course is the weekly group discussions. The group/class discussion uses the whole class to solve a problem. The problems proposed in this class are unique because they are tailored to meet the needs of students' scientific backgrounds. The CS class is taught using an interdisciplinary approach combining math, CS, and other sciences revealing the interdisciplinary nature of computer science.

Within most scientific disciplines, there is a lab associated with the course, which introduces students to the concepts covered in class through hands-on experience. Currently, computer science does not have a lab with the courses, yet computer science is as experimental and hands-on as any other scientific discipline. Many computer science concepts cannot become concrete unless students are given the hands-on experiences that make the concepts clearer. Also, students learn through their own experiences. Labs in computer science help students overcome the obstacles of the computer science terminology and hardships. The labs allow the students to find places where their knowledge is missing or misconstrued, and they are a way for the students to have fun and stimulate their interests in computer science before they become frustrated and give up.

Another approach to active learning in the classroom is group discussion and interaction among other students and the teacher. Professors are there to help the students become better people through increasing their ability to think for themselves in a discipline. Do not hesitate to express your interest in getting to know others or to admit that you have much to learn from them (Negal, 1994, p. 179). The group discussions not only break communication barriers between the students and teacher but also can help a teacher assess what the students do not correctly understand. Another important issue conveyed through group discussion is the fact that there is more than one way to do the same thing on a computer. Therefore, group discussion to a solution can permit students to see opposing viewpoints on issues (Negal, 1994, p. 121).

The last addition to the FORTRAN class is the application of programming concepts to real world problems. Wees (1971) states that "Force breeds fear" (p. 16). Therefore, applying CS to already known interests helps to stimulate rather than suppress the student's excitement. To many students, computer science is already foreign and the concepts in computer science are very abstract until they are applied to a problem of interest to the student. Application is accomplished in the lecture and group discussion, but it can also be put into projects that the students are able to create based on their interests. Ask the children what they would like to do and guide them into ways of accomplishing those things with an eye to further learning (Negal, 1994, p. 95). This helps a student become interested in a topic and increases the student's understanding and ability to progress in the subject on their own. The adult's interest in solving problems within their older time perspective makes adults more concerned with specific, narrow topics of relevance than broad, generalized or abstract subjects (Draves, 1997, p. 9). Computer science is inherently interdisciplinary, but it is also very theoretical and mathematical. Therefore, do not want to confuse this research with pure, theoretical computer science education, which is intended for those students wanting to study the raw science of
computers and how the operate. We are discussing computer science education for those students looking for the application.

**Analysis of the Data**

The data collected in this research is used to assess the students' knowledge, predict the students' ability to handle a course requiring formal thinking skills, and assess the students' improvement in thinking ability under a controlled teaching environment. Data from the CS195 students' higher order thinking skills tests is collected in the beginning of the FORTRAN course to determine the level of thinking developed among a diverse group of individuals, who do not have any previous instruction in computer science. After teaching FORTRAN by applying new techniques for teaching, the students' higher order thinking skills are retested to see if there is any improvement in their thinking skills after taking the class. Also, each skill that is tested is analyzed to see the exact type of thinking skills that increased among the remaining CS195 students. Not only does this test provide data about student's developed thinking skills, but the test results serve as a prediction about how well a student might perform in a computer science class that requires certain formal thinking skills. The following percentages for different levels are taken from the scoring of the Lawson test. The percentages are as follows: 78.6-100% (Formal), 50-71.5% (High Trans), and 21.4-42.9% (Low Trans), 0-14.3% (Concrete).

The data collected from the upper level scientific computing and Visual Basic courses are used to assess the students' logical thinking at different levels of computer science courses, predict student outcomes for two types of computer science courses, and compare students' developed formal thinking skills after taking the course. Differences between the students' scientific reasoning skills before and after a course are compared against the experimental programming course. The analysis is used to assess and evaluate new teaching strategies across our introductory level computer science courses that contain students from many different disciplines, inside and outside science. Data reveals that the scientific computing students developed their formal thinking skills prior to entering into CS471, and all the students remained and passed the class that have formal thinking. It isn't surprising that the CS471 students, who are in an upper level college science course, score so high on their logical thinking skills.

The data collected from the HOTS test before starting the experimental FORTRAN class provides a good indication of the overall level of thinking among the students. Most of the students in the class have not developed their formal thinking skills, but none of the students fell below the high transitional level. There is about an equal amount of students that fell into the higher end versus the lower end of the high-level transition stage. This is as expected because the course is promoted to students that already have an interest in science. However, the majority of our college students need to be at the formal thinking level, but if they are not at the formal thinking level, then one hopes they are in the higher end of the high transitional period.
The figure below shows the skills developed by a typical introduction to computer science course, CS101, at the University of Montana. In an introductory level computer science course, only about 20-25% of the students have developed their formal thinking skills. These are similar to the results from the experimental programming course, but the percentages for levels below the formal thinking level are very different. The Visual Basic, CS101, course is more representative of the average liberal arts college student wanting to learn about computer science, and these results show that about 11% of those students are in the lowest levels of higher order thinking, which are the concrete and low-level transition stages. As with the experimental course, more students took the test in the beginning than who retested at the end of the semester. However, compare the overall level in the beginning of the class to the end of the class and see the overall level remains roughly the same among the students. A lesser percentage of remaining students are in the formal or low transitional levels, but a greater percentage of the students are in the high transitional level.

To gain a better understanding of the affects the alternative teaching have on the students, only the students that took both tests are compared to get accurate percentages for the levels of thinking skills before and after either class. The data from the remaining students before and after FORTRAN shows that there is indeed a 29% increase in the class's formal level of thinking, as well as a movement of developed skills from the lower end to the higher end of the high transitional. This shows that either the teaching is effective on developing thinking skills or taking a computer science course stimulates a student's higher order thinking that is previously suppressed. In either case, the results show a positive outcome for the experimental course. However, the traditional Visual Basic course did not show any gain in the remaining students' level of reasoning skills. In fact, it looks as if there is a decline in the overall class's high transitional level and an increase in percentage in the low transition level.

Next, did the students' level of thinking or types of thinking skills influence either their decision to continue with the class or the grade they receive in the end of the class? According to the results shown below, students with formal thinking skills are more inclined to stay and pass a course requiring logical thinking versus students without formal thinking. Also, one can
see that the closer students are to the formal thinking level, the higher the chances are of success.

The results reveal a high correlation between a student's level of thinking and their ability to succeed, however the figure below does not show such a distinct correlation between the types of thinking skills developed and the student's ability. Also, the results below for the experimental course, CS195, show that students without conservation and control of variables skills are 100% likely to either drop or fail a computer science class, and a majority of the students that lack proportional and hypothetico-deductive reasoning skills have an increased likelihood to drop/fail. Students have an increased likelihood to drop if they do not have conservation skills, but there is not an indication that students without certain skills will drop/fail. It makes sense that students without conservation skills, which make up the concrete thinking level, might not be able to handle a course requiring a much higher level of thinking.

The data below shows the actual drop/failure rate versus the predicted drop/failure rate for the experimental-CS195 and traditional-CS101 courses. The predicted value was based on the
student's level of thinking, and if a student had reached the formal level of thinking or fell within the higher end of the high-level transition (9-14 or 60%), then he/she was predicted to have the skills needed to stay in the class. As you can see from the figures below, the predicted percentage of students that would either stay or drop/fail is very close to the actual percentage of students, who did stay or drop/fail. However, even though these results give us an indication of the overall class performance based on the student's level of thinking, this does not give us an idea about whether the particular students predicted to stay or drop/fail actually did so. The last set of results show the actual percentage of students that either stayed or dropped/failed out of the predicted students to stay or drop/fail. The actual percentage of students that either stayed or dropped/failed from those predicted is 75-80%, and this shows a strong correlation between students that have a higher versus a lower level of thinking and their ability to handle a scientific class requiring logical thinking.

According to the figures below, conservation and control are the two skills over half the class have developed before entering the experimental course, and the types of skills developed by the remaining students before and after CS195 are not much of an improvement, except in the proportion and correlation skills. This shows that not any one type of skill hinders the student's performance in such a class, but their overall level of thinking has much more of an effect. However, in the skill used most often, hypothetico-deductive reasoning, there is the greatest distance between the number of students that had the skill developed from the initial versus the remaining FORTRAN students. This may show that this skill is needed more to succeed in a computer science course. The second skill that is developed by more of the remaining students is conservation, which is the first higher order thinking skill learned and part of the concrete level of thinking developed before formal thinking. From the remaining students in CS195, proportion and correlation reasoning are the only two skills that increased after the class.

There is not any improvement in the Visual Basic class's overall level of scientific reasoning. However, the individual skills tested in CS101 and their applications to computer science are analyzed. Conservation, probabilistic, control, and proportional are the skills that over half the class had developed before CS101. There is an increase in the percentage of
remaining students with probabilistic, conservation, and combinatorial skills. This emphasizes the importance of conservation skills and combinatorial skills needed to succeed in a computer science course. However, the only skills that increase among the remaining students are control and proportional reasoning. All other skills decline among the Visual Basic students.

DISCUSSION & FUTURE WORK

This research proves to be a worthwhile experiment in computer science education. The research demonstrates that the higher order thinking skills tests can be used for assessing a student's knowledge and/or evaluating a student and class's improvement over time. The results show a direct correlation between the student's logical reasoning skills and his/her performance in a computer science course. Also, the research shows the overall class and student's learning after taking a class in computer science that should enhance his/her logical thinking ability. However, not all computer science classes need to participate in this type of research, because the current research illustrates that the students and classes benefiting the most from computer science education research are the introductory level computer science courses and students. The introductory level area shows the most need for improvement from the overall class and student's scores on the logical thinking tests. Therefore, attention is focused toward the results from the two introductory courses, Experimental Fortran-CS195 and Traditional Visual Basic-CS101, in the discussion and future work.

Discussion

The higher order thinking skills tests can aid our advisors with placing students in the correct computer science classes for their ability that allows the student to grow and gain more confidence to continue. Ideally, a university would have multiple introductory level computer science classes that meet the interests of a variety of students with different learning abilities. This not only reduces the number of students in the introduction to computer science classes, but this solution might capture the interests students have for computers and computer science. Maybe, a more realistic approach is to separate the computer science department into theoretical and applied, and therefore, students have a better idea of the type of CS class they are entering and the material covered. There is a reason why such a variety of people take the introduction to computer science course, and the reason is because the students are intrigued by the subject itself and/or they feel the subject can aid them in their discipline of study. However, instead of seeing a non-CS student's curiosity for computers being kindled, one tends to see quite the opposite. Very rarely do most of the students who take a traditional intro-level
course, like CS101, take another computer science in the future. Even though computer science is not the discipline of every intro-level computer science student, how many of the students actually reuse their programming skills taught in a course like CS101? However, how many of those students could benefit from knowing how to combine the computer science skills with their discipline? This is why it is beneficial to know where to place students in computer science courses and have several computer science courses available to meet the variety of interests among students. Also, computer science educators must try to meet the needs of different student's cognitive development using variable learning methods in the classroom. As proposed in this research, this can be accomplished through different ways of promoting active learning in the classroom, such as hands-on lab experiments, class and group discussions, teamwork, etc.

If a plan is made to change the way introductory level computer science courses are taught and organized, then there must be a way to measure the progress of the new learning approach and the student's cognitive development. This research illustrates how scientific reasoning tests are a good measurement of a student's logical thinking level, and therefore, these tests are used as a good indication of a student's improvement after taking a computer science course. Presumably, a student is not going to regress in his/her logical thinking after taking a scientific course, but a student's logical thinking ability may progress because the course helps to develop skills that are not previously existent and/or expresses the already existent skills. In either case, the educator of the science class is doing his/her job if the scores of the student's test results increase. Therefore, the higher order thinking skills test can be used to determine whether the students from a class increase their scientific reasoning skills and analyzes increases in students' scores after taking classes with altered pedagogies.

Future Work

There are many directions for this research, but ultimately, this research paves the way for future research using tests for assessing and evaluating computer science education. Currently, a problem exists with capturing and keeping students in introductory level computer science courses, and this research proposes that the future direction of our introduction to computer science courses is toward active and interdisciplinary learning approaches. It may be necessary to split CS into two separate fields with one being a theoretical approach and the other being an applied program. Computer science is in fact theoretical, and there must remain the study of the science of computers for computer science to continue. A new approach to computer science is needed to accommodate the interdisciplinary and applied nature of CS in today's society. Therefore, a student is prepared today for his/her tomorrow.

For a radical movement in computer science education to take place, more research needs to be conducted on measuring the amount that students learn using different teaching strategies in the same course. Even though this research gave results for a variety of computer science classes using different teaching methods, the research lacked varying teaching experiments on the same computer science course to see if there is a difference in the class's improvement employing the different pedagogies into the same class. Conclusions that students
benefit more from an active learning approach versus more traditional teaching methods are speculations until more research is completed in this area.

Not only are progressive teaching strategies associated with the future of computer science education, but also change for the introductory level computer science curriculum is envisioned. A movement toward the creation of many introductory level computer science courses and/or separation between theoretical and applied needs to take place to meet the needs and interests of students from a variety disciplines with a variety of scientific reasoning. Ideally, every discipline should have an introduction to computer science course tailored especially for those students majoring in the specific discipline. However, until the critical mass of professors from all the other disciplines become literate in computer programming, then the computer science department is going to have to take the lead in creating this mass of expertise. To emphasize the importance this movement has on the student's learning and career opportunities, more research needs to be conducted on how a student learns differently, what a student does in the future with his/her computer programming knowledge, and whether a student took the traditional introductory level computer science class, like CS101, versus an experimental computer science class tailored to the student's interest, such as CS195.

BIBLIOGRAPHY


