How Students "Measure Up":

Creation of an Assessment Instrument for Introductory Computer Science

Adrienne Decker
Dissertation Proposal for the Degree of Doctor of Philosophy
Department of Computer Science and Engineering
University at Buffalo, SUNY

Abstract

Many recent innovations in the computer science curriculum have focused on the first-year computer science courses, and much work has been done to help determine what predicts success in the first year. However, many of these investigations lack an appropriately validated assessment instrument to confirm their findings. There are several assessment instruments available to computer science faculty, but each of them is flawed, therefore making them inappropriate for the task of assessment of the first-year computer science courses.

I propose to create an assessment instrument that can be administered to students during their first year of study of computer science. This instrument will be constructed using the guidelines given in Computing Curricula 2001 for programming-first introductory courses. This instrument will be assessed for its reliability and validity and administered to students in their first year of study in computer science. The creation of this instrument will enable computer science faculty to further study innovations in the curriculum for the first year computer science courses.

1. Introduction

The proposed work for this dissertation is the creation of an assessment instrument to be administered during the first year of study of computer science. The instrument will be designed so that it is language and paradigm independent.

Computers and computing began to emerge as a field in the middle of the last century. Colleges and universities began creating departments and degree programs in this field of study in the 1960s. As these departments grew in number, a group of the newly emerging faculty from these colleges and universities was formed under the auspices of the Association for Computing Machinery (ACM) to explore the various issues facing these institutions while developing these programs. This group produced a report outlining a curriculum for this new emerging discipline of computer science (Committee on Computer Science Curriculum 1968). Since that time, there have been several revisions made to reflect the changing times and trends in the field (Committee on Computer Science Curriculum 1978; ACM/IEEE-CS Joint Curriculum Task Force Curricula 1991).

The most recent of these has been Computing Curricula 2001, more commonly known as CC2001 (Joint Task Force on Computing Curricula 2001). CC2001 divides the curriculum into fourteen knowledge areas that permeate the entire discipline. Furthermore, the report layers the curriculum into introductory, intermediate, and advanced course levels. For each of these levels, the report recommends pedagogical approaches to the topics in each area. The approaches include many specific details that were not present in previous curricula.

Before CC2001, there was much debate in the literature about the approach, assignments, lab environments, and other teaching aids that were most appropriate for courses. Of special interest were the CS1-CS2 introductory courses, since these are the first courses that students are exposed to. CC2001 recognizes six approaches to the introductory sequence: three programming-first approaches (Imperative-first, Objects-first, and Functional-first) as well as three non-programming-first approaches (Breadth-first, Algorithms-first, and Hardware-first). The report does not recommend one over the other, but rather points out the relative strengths and weaknesses of each of them.

2. Problem Statement and Motivation

2.1 Problem

CC2001, as with the previous curricula, does not provide faculty with instructions for how to implement its suggestions and guidelines. This leaves faculty to take their own approaches to the material, and invent assignments, lab exercises, and other teaching aids for specific courses outlined in the curriculum. Whenever a new curricular device is conceived, its effectiveness must be determined: Does the innovation actually help students' understanding of the material? Research investigations conducted on new curricular innovations have employed measures based on lab grade, overall course grade, resignation rate, or exam grades (Cooper, Dann et al. 2003; Decker 2003; Ventura 2003).

The problem with using these types of metrics in a study is that often they are not proven reliable or valid. Reliability, or the "degree of consistency among test scores" (Marshall and Hales 1972), and validity, the ability of a test to "reliably measure what is

relevant" (Marshall and Hales 1972), are both essential whenever the results of these metrics are to be analyzed.

If a metric is reliable, then the results for a particular student for that metric must be reproducible. This can refer to the test-retest ability for a metric or to the metric's internal consistency. Reliability can be assessed using a time-sampling method (for test-retest ability), a parallel-forms method, or an internal-consistency method (Ravid 1994; Kaplan and Saccuzzo 2001). The most common time-sampling method is the test-retest method, where the same subjects take an exam at two different times and scores are checked for consistency. For a parallel-forms method, two tests are created that are designed to test the same set of skills. Students then take both forms of the exam, and their results are compared for consistency. For an internal-consistency method, the test is split into two halves, and the two halves are compared for consistency. With an internal consistency method, the test is only taken once, which saves time and resources for the researcher.

However, each method has its drawbacks. When using a test-retest method, there can be practice effect. The practice effect is the possibility that when students take an exam more than once, they will do better the second time simply because they have taken the exam before. This effect is not easy to address, so many researchers choose to measure reliability using some variant of the parallel-forms method or internal-consistency methods (Marshall and Hales 1972; Ravid 1994; Kaplan and Saccuzzo 2001). However, with parallel forms, there is a burden on the participants and administrators of the exam. The participants must take a very similar exam twice, and resources must be devoted to administering these two exams. To minimize practice

effect, this duplicate testing should occur on the same day (Marshall and Hales 1972; Ravid 1994; Kaplan and Saccuzzo 2001).

Validity can be assessed using the methods of face validity, content-related validity, or criterion-related validity. Face-validity evidence is gathered from the appearance of validity. For example, a test screening for suitable mechanics to fix cars for a dealership should have questions about cars and their component parts and would probably not include questions about the interpretations of famous literary works. This type of validity does not include an in-depth analysis of the test, but rather a quick read of the questions to assure that the test appears to be applicable to the domain (Marshall and Hales 1972; Ravid 1994; Kaplan and Saccuzzo 2001).

Content-related validity is like face validity in that it is a logical, rather than a statistical, way of assessing validity. With content-related validity, one must determine whether the construction of the test adequately assesses the knowledge it is supposed to. Expert judgment is often called in to assess the content-related validity of a measure (Kaplan and Saccuzzo 2001).

Criterion-related validity is the assessment of how well a particular metric corresponds with a particular criterion (Kaplan and Saccuzzo 2001). For example, the Scholastic Aptitude Test (SAT) is used by most colleges and universities as an indicator of how well a student will perform after high school. In order for one to use the SAT in this way, the SAT should provide results of criterion-validity testing to show the evidence that it is predictive of college performance, which can be viewed in the SAT Program Handbook (SAT 2003).

The investigations mentioned before (Cooper, Dann et al. 2003; Decker 2003; Ventura 2003) suffer a further drawback in that they do not specify how a particular grade is arrived at. For example, when using overall course grade as the success marker, one should know if there was a curve placed on the grades, or even the basic breakdown of what is considered "A" work. This problem persists even when using numeric percentage grades, such as 89%. Grading standards must be well documented to be valuable for assessing the quality of the study.

2.2 Motivation

With all of the claims of innovation in the CS1 curriculum, we need a way of assessing students' comprehension of the core CS1 material. The goal of this dissertation is to create a reliable and validated assessment instrument that assesses the knowledge of a student who has taken a CS1 class using one of the programming-first approaches described in CC2001. This assessment should be independent of the approach used for CS1, and should not rely on testing a student's syntactic proficiency with a particular language.

The main motivation for this work is the fact that there is no such assessment instrument available. There are many forms of assessment available at the end of the four years of undergraduate education that are available to computer science faculty. Two such examples are the Educational Testing Service's (ETS) Graduate Record Examinations (GRE) Subject Test in Computer Science (GRE 2003) and ETS's Major Field Test in Computer Science (ETS 2003). The GRE Subject Test is designed to assess a student's ability to succeed in graduate school, while the Major Field Test is designed as an overall outcomes test for the undergraduate curriculum. In either case, since the

tests are administered at the end of the student's program for the undergraduate degree, they are not practical sources of information about the students' knowledge at the end of their first semester of their undergraduate career. Furthermore, careful examination of the reliability and validity of these two exams gives us a better indication of their lack of applicability to this investigation.

The GRE subject tests across all disciplines have been shown to predict first-year graduate grade-point average moderately well, and are more predictive than undergraduate grade-point averages in half the cases (GRE Score Use 2003). The subject test scores have also been used in conjunction with GRE test scores and undergraduate grade-point average to help predict performance. Unfortunately, the data provided by ETS about the GRE Subject tests does not include information specifically about the predictive value of the computer science subject exam.

The ETS Major Field Test in Computer Science was created with the help of experts in the subject area. There is no indication, however, that the test corresponds to a particular curriculum or to the recommendations of CC2001 (Major Field Test 2003). The reliability reported for the test for the academic year 2001–2002 was 0.89 (Major Field Test 2002), which is deemed acceptable. However, the exam tests all of the following topics: programming fundamentals, software engineering, computer architecture, computer organization, operating systems, algorithms, theory, computational mathematics, and certain other specialized topics in computer science (Major Field Test Content 2003). This topic list is much larger than what is covered in any CS1, whether programming-first or non-programming-first. Even though there is a sub-section of the exam that students who have completed CS1 could complete, the breadth of the concepts

covered on the test would be overwhelming for students who had only completed one semester of study, even in a programming-first approach.

Also available is the Advanced Placement (AP) exam (AP 2003) in computer science, which students can take while in high school to show their knowledge of the material before entering the higher education setting. This test has been shown to be an effective instrument to gauge a student's readiness and abilities in the introductory computing courses. However, this exam has shortcomings as I will discuss in §5. The information about the reliability and validity for the AP exam has been collected by its creators. This information only tells us the results for the AP as a measure of student's knowledge of the material tested on the AP exam. It has not been shown to be reliable or validated for any other purpose, especially not as an assessment for introductory computer science at the college level using the CC2001 guidelines.

These assessment measures, which are taken before starting CS1 or after the end of four years of study, do not help us evaluate student's understanding of the core CS1 material immediately after completion of the CS1 course, nor do they provide a good source of comparison of curricular innovations for CS1. In order to promote further experimentation within the development of this course, a validated and reliable assessment instrument needs to be created.

3. Prior Investigations

3.1 Approaches to the Introductory Curriculum

Before CC2001, there was a long debate regarding the most acceptable way to teach the introductory computer science curriculum. The material presented here is in summary form and will be discussed further within the dissertation.

Marion (1999) gives a detailed description of what a CS1 course could be and what the goals are for the course. He does not try to influence the reader that his approach or presentation is the best, but rather that the goals are important. Fincher (1999) also offered a position about how to teach programming as she relates some of the possible approaches that one could take when teaching computer science. She indicates that it does not always have to be about programming, and also recommends that the field should have some consensus in regards to how to best teach programming.

Many others have also contributed to the myriad of approaches for teaching the introductory course sequence. Owens, et al. (1994) offered varying viewpoints on the different models for teaching CS1. Evans (1996) also offers a model for the CS1 curriculum that emphasizes using topics that pervade the entirety of the computer science domain, which has been labeled in CC2001 as the breadth-first approach to the introductory sequence.

Still others have argued that given the current generation's exposure to computers, the way we need to teach computing must change. They argue that the programs that students create must appeal to this generation's sense of how programs should be (Stein 1996; Guzdial and Soloway 2002). This somewhat echoes an approach given by Proulx, Rasala, & Fell (1996) as well as Alphonce & Ventura (2003), who both advocate an approach to CS1 that utilizes graphics and event-driven programming to motivate the core material in CS1. Reges (2000) advocates an approach to CS1 using graphical user interfaces, but gives a more conservative approach to their use in the course.

Approaches that have nothing to do with programming have also been explored. Among them are the Applied Apprenticeship Approach (Astrachan and Reed 1995) and

Pair Programming (Nagappan, Williams et al. 2003). Each of these ideas has merits, and, for some approaches, anecdotal evidence would suggest their success, but since no validated assessment instruments exist, they, too, are in question.

Another set of documented approaches relies heavily on language, but more importantly on paradigm, and on issues that arise when teaching a particular paradigm. A programming paradigm is a view of a particular program's main unit of computation. For example, when programming in Lisp, the main unit of computation is the function, and the paradigm is called functional programming.

Back in the days of heavy Pascal use, Pattis (1993) argued about the appropriate point in the curriculum to teach subprograms. Moving forward a few years, we see Culwin (1999) arguing how to appropriately teach object-oriented programming, followed by a strong course outline for an Objects-first CS1 advocated by Alphonce & Ventura (2002; Ventura 2003). These are among the many debates that have been discussed in the literature. For these as well as others, while there may be strong anecdotal evidence to support an approach, little empirical evidence, aside from Ventura (2003), that has been presented as to the real effect of these methodologies on learning the appropriate material for CS1.

3.2 Predictors Research

The need for accurate assessment instruments is again evident when one looks at the literature on predictors of success for CS1 (Kurtz 1980; Mazlack 1980; Leeper and Silver 1982; Evans and Simkin 1989; Hagan and Markham 2000; Wilson and Shrock 2001). For each of these studies, different factors were identified as possible reasons for success in a programming-first CS1 course. In each case, success was measured either by

overall exam score, laboratory exercise scores, programming assignment scores, or exam scores. None of these measures of success were validated or clearly explained. Furthermore, for many of the studies, the CS1 course had not been clearly defined. We cannot be sure that the outcomes that were expected of the students in some of these studies are in line with the recommendations of CC2001. For the studies that occurred before the publication of CC2001, it must be assumed that the researchers could not have anticipated CC2001 and therefore the courses will not reflect the recommendations it contains.

Even recent work done on a course that embraces CC2001's recommendations for an Objects-first CS1 uses only measures of overall course grade, exam grades, and lab grades in its study (Ventura 2003). The predictive values of the factors studied are given, as in the other work cited above, but it once again fails to convince that the students' level of success has been validated in some form.

3.3 Educational Objectives and Outcomes Assessment

Well-defined educational objectives and outcomes assessment measures for creating new curricula in CS1 are increasingly common (Parker, Fleming et al. 2001; Neebel and Litka 2002). These approaches require as much reorganization of the course contents and presentation as adoption of the CC2001 recommendations, but go a step further in giving not just tasks and skills needed, but ways to structure the course for better learning outcomes.

One set of educational objectives that has been explored in a CS1 course is that of Bloom's Taxonomy of Educational Objectives (Lister and Leaney 2003). Lister and Leaney use Bloom's Taxonomy as a way to structure the criterion used to grade the

students of the course. Students who would receive the minimum passing grade in the course are expected to have successful completed criterion that fall into the lowest two levels of Bloom's Taxonomy. Higher grades in the course are earned by completing criteria that are categorized at higher levels of the taxonomy. Missing, however, is a clear description of exactly what (if any) skills the student should come out of CS1 with. It is unclear whether students are required to understand such topics as iteration or selection to pass the course. The authors argue that, with this approach, CS2 must be modified to embrace Bloom's Taxonomy as well. When adapting the CS2 course using Bloom's Taxonomy the outcomes expected from CS2 seem to differ from the traditional set of topics that are normally associated with CS2.

Another approach to course-embedded assessment[*] was used at Slippery Rock University (Whitfield 2003). Their curriculum was designed so that each student would come out of the program having learned a well-defined set of topics and ideas. The courses at this university were designed to make sure that the appropriate material was presented to achieve these outcomes. However, the stated outcomes for the curriculum seem generalized and vague. It is difficult to see whether or not they coincide with CC2001's recommendations for CS1. Their assessment methods were not proven valid or reliable, nor did they indicate whether students were meeting the designated goal of success in the course.

[*] Course-embedded assessment is assessment that occurs within the course at semi-regular intervals. For example, midterm exams, graded homeworks, and quizzes all could be administered throughout the semester as course-embedded assessment instruments.

3.4 Assessment of Programming Skill for CS1

There has been one documented attempt at creation of an assessment for CS1. A working group from the Conference on Innovation and Technology in Computer Science Education (ITiCSE) created a programming test that was administered to students at multiple institutions in multiple countries (McCracken, Almstrum et al. 2001). The group's results indicated that students coming out of CS1 did not have the programming skills that the test assessed.

Among the positives of this attempt at assessment were that it included problems that were well thought out and that it made an attempt to define and cover all of the material that a CS1 student should have mastery of. Another positive was the fact that there were specific grading rubrics created for the problems, which helped lead to uniform scoring. The students were not restricted to a particular language or programming environment, so the students completed the exercises in whatever way was most comfortable to them.

However, the study was flawed, as recognized even by the members of the working group. The problems given had an inherent mathematical flavor that would have disadvantaged students with mathematical anxiety. They also admit in their analysis that one of the test questions "was undoubtedly difficult for students who had never studied stacks or other basic data structures" (McCracken, Almstrum et al. 2001). They also pointed out flaws in the presentation of the problems and the instructions for administering the exercises. Therefore, even with all the positives of this study, there is still room for improvement to make an assessment instrument that could be more true to the current flavors of CS1 as described in CC2001.

4. Researcher's Preliminary Work

4.1 Study of Performance in Non-Majors Course

Work was completed analyzing student's retention of object-oriented concepts in the CS2 course for non-majors at the University at Buffalo, SUNY (Decker 2003). One problem that grew out of this investigation was that of how to accurately assess student's knowledge in this area. The solution that was used was one that is commonly found in the rest of the literature, to simply use exam scores as the benchmark of success. The experiment was well received by the reviewing committee for the Consortium for Computing Sciences in Colleges Eastern Conference as well as the participants[†] as a true empirical investigation of student comprehension of basic object-oriented material in a CS1-CS2 sequence.

However, this study in itself suffers from the same problems as much of the literature in this area. The exams and tests that were administered were not proven to be reliable or valid. Furthermore, this experiment covered only the students' knowledge of object-oriented concepts, not general CS1 knowledge. In this dissertation, we seek a comprehensive assessment of CS1 knowledge that is independent of language or paradigm.

4.2 Advanced Placement Exam

There are two validated exams available that are predictive of success in an introductory computer science course: the Advanced Placement (AP) Computer Science

[†] This paper received two awards at the conference at which it was presented. The first was that of "Best Paper Finalist". From the four "Best Paper Finalists", this paper was chosen to be the "Best Overall Paper" for the entire conference.

A and Computer Science AB exams. Data was collected on AP test scores for students from the incoming freshmen classes of 2000–2002. The scores on the AP exam and the student's subsequent performance in CSE 115, our version of CS1, have been analyzed and will be discussed in §5.2.2.

4.2.1 Advanced Placement Exam Background

The AP exam is a nationally standardized test that is administered to students at the high school level after completion of at least one year of study in the subject area. The courses are designed to be equivalent to a college course. All of the AP exams have a section for free response and another for multiple choice questions (Board 2003). The Computer Science AP exam consists of four free response and forty multiple choice questions (Hunt, Kmoch et al. 2002).

The AP exam was developed with the recommendations for curriculum of the ACM and IEEE, which would indicate that it should follow the CC2001 recommendations (College Board 2003). However, the exam booklet does not indicate that the exam is based on any curricular models specifically, so it is not certain whether it follows CC2001.

There are two Computer Science AP Exams; the Computer Science A exam covers material that is closely correlated with a CS1 course, while the Computer Science AB exam covers material normally covered in both a CS1 and CS2 course.

4.2.2 Analysis of AP Exam Data

We looked for a correlation between a student's AP exam score and the final letter grade the student received in CSE 115. The AP exam is scored on an ordinal scale

of 1 to 5, with 5 being the highest grade attainable (AP 2003). Student's letter grades in CSE 115 can be A, A-, B+, B, B-, C+, C, C-, D+, D, or F. Students could have also resigned the course and earned a grade of R. Since an R is an elected grade, and students do not have to give a reason for resignation, all R grades were omitted from this analysis. The Spearman rank-order correlation test was used, since we have strictly ordinal data in this analysis.

The first group examined consisted of those students who took the AP Computer Science A exam and then proceeded to take CSE 115. The analysis produced a significant correlation between the student's AP exam score and their overall grade in CSE115, $r_s(49) = .42$, $p < .01$.[‡]

In the second analysis, students who took the AP Computer Science AB exam and then proceeded to take CSE 115 were examined. This analysis showed that there was not a significant correlation between the students' AP exam score and their overall grade in CSE 115, $r_s = .21$, $n = 27$, $p > .05$.

4.2.3 Discussion

The results for the AP Computer Science A exam show a correlation with CS1 grade, indicating for example, that a high grade on the AP exam will indicate a high grade in CS1. However, this type of correlation is not useful due to the fact that the AP

[‡] For statistical reporting purposes recall the following information: $r_s(F) = m$ reports the information about the correlation coefficient. r_s is the abbreviation indicating that we are using the Spearman rank-order correlation test. F represents the degree of freedom, which is one less than the total number of data points in the sample. m is the actual correlation coefficient. Positive numbers represent a positive correlation, while negative numbers represent an inverse correlation. Recall that for any statistical analysis, *p* values that are less than .05 are considered statistically significant results at the 95% confidence level. Results that have *p* values less than .01 are considered statistically significant at the 99% confidence level. For the results that are not statistically significant, the correlation coefficient is still reported in the same manner as above, but instead of reporting the degree of freedom, we simply report the size of the sample data, represented by the letter *n*.

exam is administered at the end of an academic year of high school study. It has not been proven reliable or valid except for use as a predictor for success in CS1 course, or CS1-CS2 sequence, in the case of the AB exam. We are looking for an assessment of CS1 knowledge to be administered after the completion of CS1 in a higher education setting.

Another shortcoming of the AP exam is that the questions on the exam are given using the language C++ and testing student's knowledge of the object-oriented paradigm. The goal for our assessment is one that is language-independent and paradigm-independent. The AP exam fails to serve our needs in both of those points.

The results of the AP Computer Science AB exam seem troubling in that they show a lack of correlation for CS1 grade. This is likely due to the difference in emphasis of the AB exam, which focuses on material both in a CS1 curriculum as well as a CS2 curriculum as opposed to strictly a CS1 curriculum. We must recall that a non-significant result simply shows us that it is not possible to say definitively that a good grade on the AP Computer Science AB exam would lead to a good grade in CS1, or vice versa. Prediction of poor performance is also not possible. In some cases, there may be a correlation, but there is not enough significance in the trend to have a statistically significant result. It is also important to note, however, that this exam also has the shortcomings of the A exam in that the exam questions are specific to C++ and to object-oriented programming.

Overall, the results of the analysis show that one of the AP exams does have correlation with student course grades in our institution's version of CS1. However, it is not a viable solution for the problem this dissertation seeks to address, due to the shortcomings of the exam.

5. Proposed Work

The proposed research for this dissertation will occur in several different phases. The phases include the process of actually generating the assessment instrument as well as analysis and testing of it. The end product will be an assessment instrument for CS1 that has been analyzed for reliability and validity, and that will assess the achievement of students who have completed any of the forms of the programming-first CS1 courses as described in CC2001.

For some of the phases of the proposed work, human subjects are involved. Therefore, the approval of the Institutional Review Board (IRB) will be sought prior to commencement of the investigations.

5.1 Phase I: Topics, Questions & Grading for Programming-First Approaches

The goal of this analysis phase is to find the intersection of all of the topics for the programming-first approaches. This analysis will produce a list of topics that are common to all of the programming-first approaches outlined in CC2001. The topics will include skills as well as requisite knowledge defined in the course outlines for CS1. It is possible that during this analysis of CC2001, it will be found that the intersection of the topics covered by all three programming-first approaches will be small or empty. If this is the case, analysis will concentrate on the programming-first approaches for the entirety of the first year, CS1 and CS2. Another possibility that may also be explored at this time will be to look at the topical coverage of the CS1 courses according to the pre-defined Knowledge Units in CC2001.

After this analysis, creation of exam will begin with the creation of the individual questions for the exam. The questions will be language-independent, but may require students to create solutions in a language that they are familiar or comfortable with. This phase of the work can be seen as the largest and most critical phase. Many decisions will have to be made about what types of information will be tested and exactly what type of testing will occur. The test may be a mixture of performance testing for skills that students will need to have in terms of programming, and objective testing, which will help us determine their knowledge about the subject material of CS1. After these types of decisions are made, the writing of the actual items can begin. Writing good test items will be the key to the success of this phase, and it is possible that the items themselves will need to be evaluated by someone with known proficiency in test item writing.

At this time, a grading rubric will also be established for each question. The rubric will outline the correct answer and an explanation for how partial credit should be awarded if the entire answer is not correct.

5.2 Phase II: Survey of CS1 Educators

For this phase, a web-based survey will be constructed. The subjects of this phase of the research will be members of *listservs* and newsgroups whose readership is involved and interested in the development of educational advances for the Computer Science curriculum. One such *listserv* is SIGCSE.MEMBERS. An announcement will be sent to the lists and newsgroups asking for voluntary participation in assessing the relevance as one part of determining the validity of an assessment instrument for CS1.

For this phase, a web-based survey will be constructed. The first set of questions on this survey will be used to gain information about what type of CS1-CS2 sequence is

given at the participant's institution, as well as the participant's involvement in the introductory courses. These answers will allow us to later analyze which responses came from instructors whose institutions have a programming-first CS1. The subsequent questions on this survey will pertain exclusively to the content of the questions created in Phase I. Participants will be asked to indicate whether they believe a given topic is in fact the topic that is being tested in each question, and whether the question would be a fair assessment of knowledge of that topic. The participants will then give opinions on the questions overall and provide any additional comments or feedback in a free response area.

Participants will also be asked to evaluate the grading rubric for each question. They will be asked to rank it as fair or unfair, and once again comments and suggestions can be entered in a free response area.

Participants taking the survey will not be given a time limit to answer the questions on the survey. That is, when they begin analyzing the questions created in Phase I, they will be allowed as much time as necessary to complete their responses. It will not be a timed web survey. However, after a specified date, the survey will be removed from the web and will no longer be available for new participants to complete. This date will be included in the announcement of the survey being sent to the various *listservs* and newsgroups.

5.3 Phase III: Analysis of Survey

In this phase, the results of the survey administered in Phase II will be analyzed. Since the survey will be designed to give the appropriate feedback about the relevance of the assessment instrument, which will eventually help in determining the validity of the

instrument, we will look to these results to show relevance. Recall that since reliability is also an essential part of validity, all this phase can really help us determine is some measure of face-validity and content-validity. We still need to gather data on the reliability before making an overall judgment of the validity of the instrument. The results of the assessment of the grading rubric will also be analyzed. Provided that the results from the community polled provide us with the feedback that this exam does indeed have this relevance to the topic area that we are trying to assess, the instrument will move on to actual field testing.

However, if the community at large does not feel the instrument is suitable for this task, the metric will need to be redesigned and tested again based on the commentary and feedback of the participants in Phase II.

5.4 Phase IV: Field Testing

The subjects in this phase of the research will be those students who are enrolled in CSE 115 and/or CSE 113 at the University of Buffalo, SUNY. Only students who are at least 18 years of age at the time of the study will be allowed to participate. Student participation is completely voluntary.

The instrument that was previously created and evaluated in Phases I through III will now actually be administered to students. Since it is an assessment of overall achievement in the CS1 course, administration will happen at the end of the semester. Students will not be allowed any aids during the administration of the exam. The students will have a time limit of three hours, which is the standard length of time devoted for a final exam at our university.

The grading rubric that was created and evaluated in Phases I through III will be used at this time to grade the student's exams. The analysis of the students' responses to the questions will be analyzed by at least two independent scorers. The scorers will each be responsible for grading all of the questions on every exam. These scorers will not be aware of how the other has scored the student for a particular question and will be given instruction and training on how to follow and use the grading rubric when grading each question.

The type and extent of training will be dependent on what types of questions have been developed for this instrument. If questions become entirely multiple choice, the training process and grading will become quite mechanical. However, if the test has been designed with questions that students will give a free-response or essay-type answer, the training of the graders must include careful explanation of the grading rubric and possibly sessions where sample questions are graded and the raters are given feedback about how accurately they are following the rubrics.

5.5 Phase V: Analysis of Grades, Reliability, and Validity of Test

After the ratings have completed their grading, the scores that each rater has given each student for each question will need to be analyzed to see if there is any statistically significant difference between the raters' scoring for that student. If statistically significant differences arise, the cause will need to be determined and this could result in a new grading of the exam with different raters. Provided that the rater's scores do not have these differences, the test itself will undergo analysis for reliability.

To test for reliability, we will use a measurement of internal consistency, such as the odds-evens method or the split-half method. In both of these methods, the exam is

split into two halves, and each half is compared to the other to give the overall reliability of the metric (Marshall and Hales 1972; Ravid 1994; Kaplan and Saccuzzo 2001). Once the test has been shown reliable, we can make a determination about the validity of the test, using the reliability data as well as the relevance data collected previously.

6. Contributions and Significance of Proposed Work

The first part of this work will establish the commonality among the three programming-first approaches to teaching CS1 presented in CC2001. The thorough investigation of these three seemingly disparate approaches will bring to light the basic commonalities amongst them. In this way, researchers and instructors alike can understand the basic skill set that students who leave CS1 should have no matter which approach was used to teach the course.

The second and perhaps more important contribution will be the assessment instrument that will be developed. The first validated and reliable means of assessing a student's understanding of the material in CS1 will prove useful to individual course instructors. It will also provide a useful benchmark for studies that focus on the relative success of different approaches to teaching CS1. This instrument will be available to test if a particular teaching technique, pedagogical advance, or lab environment really improves students' performance in CS1. If instructors use the instrument as a means of assessing their students' performance in a CS1 course, the results could indicate poor performance of a particular instructor or teaching technique. A poor result may cause the institution to reassess their current methodologies or curriculum in the CS1 course. Also, this instrument can then be used in conjunction with the work already completed by Ventura (2003) to further analyze the results presented in that work.

7. References

1. ACM/IEEE-CS Joint Curriculum Task Force Curricula (1991). Computing curricula 1991, IEEE Computer Society & Association for Computing Machinery. **2003**.

2. Alphonce, C. G. and P. R. Ventura (2002). Object orientation in CS1-CS2 by design. 7th annual conference on Innovation and Technology in Computer Science Education, Aarhus, Denmark.

3. Alphonce, C. G. and P. R. Ventura (2003). Using Graphics to Support the Teaching of Fundamental Object Oriented Principles. OOPSLA 2003 Educator's Symposium, Anaheim, California.

4. AP (2003). AP Course Descriptions, The College Board. **2003**.

5. Astrachan, O. and D. Reed (1995). AAA and CS1: The applied apprenticeship approach to CS1. 26th SIGCSE technical symposium on Computer science education, Nashville, Tennessee.

6. College Board (2003). Course Description - Computer Science. **2003**.

7. Committee on Computer Science Curriculum (1968). "Curriculum 68: Recommendations for the undergraduate program in computer science." Communications of the ACM **11**(3): 151-197.

8. Committee on Computer Science Curriculum (1978). "Curriculum 78: Recommendations for the undergraduate program in computer science." Communications of the ACM **22**(3): 147 - 166.

9. Cooper, S., W. Dann, et al. (2003). Teaching objects-first in introductory computer science. 34th SIGCSE technical symposium on Computer Science Education, Reno, Nevada.

10. Culwin, F. (1999). Object imperatives! 30th SIGCSE technical symposium on Computer Science Education, New Orleans, Louisiana.

11. Decker, A. (2003). "A tale of two paradigms." Journal of Computing Sciences in Colleges **19**(2): 238-246.

12. ETS (2003). ETS Major Field Tests, Educational Testing Service. **2003**.

13. Evans, G. E. and M. G. Simkin (1989). "What best predicts computer proficiency?" Communications of the ACM **32**(11): 1322 - 1327.

14. Evans, M. D. (1996). "A new emphasis & pedagogy for a CS1 course." SIGCSE Bulletin **28**(3): 12 - 16.

15. GRE (2003). GRE Subject Tests, Educational Testing Service. **2003**.

16. GRE Score Use (2003). Guide to the Use of Scores, Educational Testing Service. **2003**.

17. Guzdial, M. and E. Soloway (2002). "Teaching the Nintendo generation to program." Communications of the ACM **45**(4): 17 - 21.

18. Hagan, D. and S. Markham (2000). Does it help to have some programming experience before beginning a computing degree program? 5th annual SIGCSE/SIGCUE conference on Innovation and technology in computer science education.

19. Hunt, F., J. Kmoch, et al. (2002). How to develop and grade an exam for 20,000 students (or maybe just 200 or 20). 33rd SIGCSE technical symposium on Computer Science Education, Covington, Kentucky.

20. Joint Task Force on Computing Curricula (2001). Computing curricula 2001 computer science, IEEE Computer Society & Association for Computing Machinery. **2003**.

21. Kaplan, R. M. and D. P. Saccuzzo (2001). Psychological Testing: Principles, Applications and Issues. Belmont, California, Wadsworth/Thomson Learning.

22. Kurtz, B. L. (1980). Inivestigating the relationship between the development of abstract reasoning and performance in an introductory programming class. 11th SIGCSE technical symposium on Computer Science Education, Kansas City, Missouri.

23. Leeper, R. R. and J. L. Silver (1982). Predicting success in a first programming course. 13th SIGCSE technical symposium on computer science education, Indianapolis, Indiana.

24. Lister, R. and J. Leaney (2003). Introductory Programming, Criterion-Referencing, and Bloom. 34th SIGCSE technical symposium on Computer Science Education, Reno, Nevada.

25. Major Field Test (2002). Reliability Coefficients and Standard Error of Measurements, ETS. **2003**.

26. Major Field Test (2003). Major Field Tests: Description of Test Reports, ETS. **2003**.

27. Major Field Test Content (2003). Major Field Tests Computer Science Description, ETS. **2003**.

28. Marshall, J. C. and L. W. Hales (1972). Essentials of Testing. Reading, Massachusetts, Addison-Wesley Publishing Co.

29. Mazlack, L. J. (1980). "Identifying potential to acquire programming skill." Communications of the ACM **23**(1): 14 - 17.

30. McCracken, M., V. Almstrum, et al. (2001). "A multi-national, multi-institutional study of assessment of programming skills of the first-year CS students." SIGCSE Bulletin **33**(4): 1 - 16.

31. Nagappan, N., L. Williams, et al. (2003). Improving the CS1 experience with pair programming. 34th SIGCSE technical symposium on Computer Science Education, Reno, Nevada.

32. Neebel, D. J. and B. T. Litka (2002). Objective based assessment in a first programming course. 32nd ASEE/IEEE Frontiers in Education Conference, Boston, Massachusetts.

33. Owens, B. B., R. D. Cupper, et al. (1994). New models for the CS1 course: What are they and are they leading to the same place? 25th SIGCSE symposium on Computer science education, Phoenix, Arizona.

34. Parker, P. E., P. D. Fleming, et al. (2001). Differentiating assessment from evaluation as continuous improvement tools. 31st ASEE/IEEE Frontiers in Education Conference, Reno, Nevada.

35. Pattis, R. (1993). The "Procedures Early" approach in CS1: A heresy. 24th SIGCSE technical symposium on Computer science education, Indianapolis, Indiana.

36. Proulx, V. K., R. Rasala, et al. (1996). Foundations of computer science: What are they and how do we teach them? 1st conference on Integrating technology into computer science education, Barcelona, Spain.

37. Ravid, R. (1994). Practical Statistics for Educators. Lanham, University Press of America.

38. Reges, S. (2000). Conservatively radical Java in CS1. 31st SIGCSE technical symposium on Computer Science Education, Austin, Texas.

39. SAT (2003). SAT Program Handbook, The College Board. **2003**.

40. Stein, L. A. (1996). Interactive programming: revolutionizing introductory computer science, ACM Computing Surveys. **2003**.

41. Ventura, P. R. (2003). On the origins of programmers: Identifying predictors of success for an objects-first CS1. Computer Science & Engineering. Buffalo, University at Buffalo, SUNY.

42. Whitfield, D. (2003). "From university wide outcomes to course embedded assessment of CS1." Journal of Computing in Small Colleges **18**(5): 210 - 220.

43. Wilson, B. C. and S. Shrock (2001). Contributing to success in an introductory computer science course: A study of twelve factors. 32nd SIGCSE technical symposium on Computer Science Education, Charlotte, North Carolina.