Adrienne Decker
Department of Computer Science & Engineering
University at Buffalo
adrienne@cse.buffalo.edu

How Students Measure Up: Creation of an Assessment Tool for CS1

Introduction

As 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. As the field evolved, a group was formed 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 (Curriculum, 1968) . Since that time, there have been several revisions made to reflect the changing times and trends in the field (Curriculum, 1978) (Curricula, 1991) .

The most recent of these has been Computing Curricula 2001, more commonly known as CC2001 (Curricula, 2001) . CC2001 has broken the curriculum into fourteen knowledge areas that permeate the entire discipline. Furthermore, the report divides the curriculum into three course levels, introductory, intermediate, and advanced. For each of these levels the report makes recommendations for approaches to the topics in each area. This section on approaches includes 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 aides that were most appropriate for courses. Of special interest was the introductory sequence of courses (CS1-CS2), due to the fact that these were the first courses that students were exposed to. CC2001 legitimizes six approaches to the introductory sequence, which include three programming-first approaches: Imperative-first, Objects-first, and Functional-first as well as 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 all of them.

CC2001, as with the previous curricula, does not provide faculty with instructions for how to implement the suggestions and the guidelines contained within. This leaves faculty to take their own approaches to the material, and invent assignments, lab exercises and other teaching aides for specific courses outlined in the curriculum. Whenever a new curricular device is conceived, the next natural step in the investigation is to see if the innovation actually helps student's understanding of the material. Investigations into some of these innovations has previously been measured by lab grade, overall course grade, resignation rate or exam grades (Cooper, Dann, & Pausch, 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, and validity, the relevance of the metric for the particular skill it is trying to assess are both essential whenever the results of these metrics are to be analyzed (Kaplan & Saccuzzo, 2001; Marshall & Hales, 1972; Ravid, 1994) . Also, within these types of studies, it is not often specified 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.

With all of the claims of innovation in CS1 curriculum, we need a way of assessing student's comprehension of the core CS1 material. The goal of this work is to create a reliable and validated assessment tool for CS1. The tool will be one 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 both the approach used for CS1 and should not rely on testing a student's syntactic ability with a particular language.

Theoretical Background & Previous Research in the Area

Before CC2001, there was a long debate over what is the most acceptable way to teach the introductory computer science curriculum. Marion (1999) gives a detailed description of what a CS1 course should be and what the goals are for the course. He does not try to influence the reader that his approach or presentation are the best, but rather that the goals are important. Fincher (1999) also offers up 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 sequence of courses. Owens, Cupper, Hirshfield, Potter, and Salter (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.

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 should appeal to this generation's sense of how programs should be (Guzdial & Soloway, 2002; Stein, 1996) . This somewhat echoes an approach given by Proulx, Rasala and Fell (1996) as well as Alphonce and Ventura (2003) who both advocate an approach to CS1 that utilizes graphics and graphical programming to motivate the core material in CS1. Reges (2000) also 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 language have also been explored. Amongst them are the Applied Apprenticeship Approach (Astrachan & Reed, 1995) and Pair Programming (Nagappan et al., 2003) . Each of these ideas has merits and for some approaches, anecdotal evidence would suggest their success, but since no validated assessment tools exist, they too are in question.

Another set of documented approaches do heavily rely on language, but more importantly paradigm, and issues that arise when teaching a particular paradigm. 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 for how to appropriately teach Object-Oriented programming, followed up by a strong course outline for an Objects-first CS1 advocated by Alphonce and 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 has been presented as to the real effect of these methodologies on learning the appropriate material for CS1.

The need for accurate assessment tools once again reveals itself when one looks at the literature on predictors of success for CS1 (Evans & Simkin, 1989; Hagan & Markham, 2000; Kurtz, 1980; Leeper & Silver, 1982; Mazlack, 1980; Wilson & 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, the CS1 courses may or may not have been clearly defined in the study. We can not be sure that the outcomes that were expected of these students were those in line with CC2001.

Even in recent work done on a course that embraces CC2001's recommendations for an Objects-first CS1 only uses 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 level of success in the students has been validated in some form.

Using well-defined educational objectives or outcomes assessment measures have been two techniques that are growing in popularity for creating new curricula in CS1 (Neebel & Litka, 2002; Parker, Fleming, Beyerlein, Apple, & Krumsieg, 2001) . These approaches take 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 was Bloom's Taxonomy of Educational Objectives (Lister & Leaney, 2003) . This approach uses Bloom's Taxonomy as a way to structure all aspects of the course and assign course grades to the students. 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 allude to the fact that with this approach, CS2 must be modified to embrace Bloom's Taxonomy as well, and the outcomes expected out of that course 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) . The outcomes that were expected from each student were clearly outlined and the course was designed to make sure these outcomes were presented. However, these outcomes seem somewhat generalized and vague. It is difficult to see whether or not they coincide with CC2001's recommendations for CS1, and once again their assessment methods were not proven valid or reliable, nor did they indicate if students were meeting the designated goal of success in the course.

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

One of the positives about this attempt at assessment is that it included problems that were well thought out and made an attempt to 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 that helped lead to uniform scoring. The students were not restricted to a particular language or programming environment, so the students completed the exercises in whichever way was most comfortable to them.

However, the study was flawed as recognized even by the participants. The problems given had an inherent mathematical flavor that would have disadvantages students with mathematical anxiety. They also admit in their analysis that one of the test questions "was undoubtly difficult for students who had never studied stacks or other basic data structures." (McCracken 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 to grow and make an assessment tool that could be more true to the current flavors of CS1 as described in CC2001.

Goals of the Research

The ultimate goal of the research is to create a validated and reliable metric for assessing student's level of knowledge at the completion of a programming first CS1. The test should be language and paradigm independent. This test will then be available to assess not only student progress, but also as a way to gauge particular pedagogical advances and their true value within the classroom.

The current hypotheses are:

- The current methods for testing pedagogical innovations are not adequate.
- A test can be written to assess a student's level of achievement with the CS1 curriculum.

Current Status

At the time of this writing, a proposal has been prepared and it is undergoing revisions from my committee. In progress is the analysis of CC2001 to determine what is the appropriate topical coverage for the tool.

Interim Conclusions

There has been one previous attempt at a means of assessing CS1 completed. This attempt, even the authors admit was flawed. However, the base idea and the methodology was strong. Using some of the techniques developed by this group, I believe an assessment tool can be created that can be proven reliable and valid for assessment of student knowledge at the end of CS1.

Target audience for the tool has been a challenge. Looking at the various sanctioned methodologies for CS1 given in CC2001, much care was taken to figure out which of them overlapped. The programming-first approaches all had much in common. This overlap was not seen between the non-programming first approaches and the programming-first approaches, or even amongst the non-programming first approaches. Therefore, the decision was made to create an assessment tool for only the programming-first approaches.

Validating the test will be accomplished using an expert review methodology. After the tool is prepared, a pool of experts in the area will be asked to assess the test's appropriateness for students and the clarity and difficulty of the questions on the exam.

The exam will be field tested as a final exam for a CS1 course. After the exam has been administered, reliability will be computing using one of the standard statistical methods, either odds-evens or split-halves. <<REFERENCE>>

Open Issues

Open issues for this research include

- Can there be an assessment tool that accurately assesses student's progress through a curriculum, given the differences across schools and curriculum?
- Can the non-programming first approaches be assessed using the same metric at the end of CS1? or even CS2?
- Can information gathered using this assessment tool help us determine if a particular pedagogical innovation is helping the students learn?

Current Stage in Program of Study

Defense of proposal in January and data collection to begin in the Spring semester.

What I Hope to Gain From Participation in Doctoral Consortium

I hope to gain input and feedback about my research ideas. I am also hoping for informed guidance on the approach I am taking towards my research and suggestions on how to proceed forward.

Bibliographic References

[1] Cantwell Wilson, B., & Shrock, S. (2001). Contributing to success in an introductory computer science course: A study of twelve factors. ACM SIGCSE Bulletin, Proceedings of the thirty second SIGCSE technical symposium on Computer Science Education, 33(1), 184-188.

[2] Ennis, R. H. and Millman, J. (1985). Cornell critical thinking test, level Z. Pacific Grove, CA: Midwest Publications.

[3]Evans, G. E., & Simkin, M. G. (1989). What best predicts computer proficiency? Communications of the ACM, 32(11), 1322-1327.

[4]Kurtz, B. L. (1980). Investigating the relationship between the development of abstract reasoning and performance in an introductory programming class. The papers of

the eleventh SIGCSE technical symposium on Computer science education : SIGCSE bulletin, 110 - 117.

[5] Leeper, R. R., & Silver J. L. (1982). Predicting success in a first year programming course. Paper presented at the Thirteenth SIGCSE Technical Symposium on Computer Science Education, Indianapolis, Indiana.

[6] Mazlack, L. J. (1980). Identifying potential to acquire programming skill. Communications of the ACM, 23(1), 14-17.

[7] Ventura, P. R. (2002). Objects-first CS1 not considered harmful: An empirical investigation of CS1. Paper presented at the SIGCSE Doctoral Consortium, Convington, Kentucky.