EFFECTIVE ASSESSMENT FOR EARLY COURSES IN COMPUTER SCIENCE: INSTRUMENTS OTHER THAN OUT-OF-CLASS PROGRAMMING ASSIGNMENTS

Lydia Fritz, Grand Canyon University

ABSTRACT

This is an experience paper that describes methods of student assessment in introductory- and intermediate-level computing courses. The paper explains the need for alternate methods in the evaluation of out-of-class programming assignments and enumerates several options that have been incorporated into freshman- and sophomore-level courses. I show how these techniques provide a more reliable assessment of student mastery of course objectives. In addition, I describe benefits in terms of increased student intellectual engagement and a deeper mastery of essential foundational material.

Keywords: assessment design, learning objectives, video presentation, poster presentation, CS1 assessment, student presentations

INTRODUCTION

Course objectives in computer science are often assessed with programming assignments. Some smaller assignments are completed in proctored settings such as the classroom or lab, but more substantial work requires the student to develop solutions without the constant presence of an instructor or lab assistant. Such assignments are known to be challenging exercises, particularly for the beginner; syntax must be correct, software must be properly utilized, and solutions must be logically sound. A single mistake can cause an entire assignment to fail. This is especially frustrating for students at the introductory level who have yet to develop the necessary skills to identify the source of errors and make necessary corrections. There is no guarantee that a working program submission is not plagiarized and therefore does not necessarily indicate that a student has mastered the learning objectives associated with the assignment (Ngo, 2016). Conversely, an incorrect submission does not mean the student failed to master the assignment objectives as program failure might have any number of root causes. This paper describes alternate methods of assessment that provide a more insightful measure of student mastery of learning objectives and outlines the added benefits of alleviating student stress and frustration while increasing learning, confidence, and interest in early programming classes.

NEED FOR ALTERNATE ASSESSMENT TECHNIQUES

At this university, many computing courses are project-based. This means that even in introductory courses, student evaluation is based on submitted programming assignments. A significant portion of class time is spent actively programming with the instructor present to assist. However, not all assignments are designed to be completed in the classroom. Important skills are gained in the out-of-class programming endeavor; the student becomes self-reliant and learns to plan solutions and resolve problems on their own (Walker, 2004). Students are encouraged to utilize a variety of resources such as textbooks, reference manuals, online documentation, and video tutorials as part of the problem-solving process. Unfortunately, this sometimes leads to an abuse of the very resources students are expected to utilize. Students have the means and ability to obtain solutions to programming assignments from a myriad of sources, including copying the

work of other students, paying online services for solutions, and downloading code that is freely available in online code sharing repositories (Ngo, 2016). The pressure to develop a working solution by a deadline, coupled with the abundance of information available online, creates a tempting environment. Even the well-intentioned student may find themselves assembling resources to create a solution they don't fully understand, leaving course objectives unmet (Abraham & Milligan, 2008). Other capable students might become frustrated and give up on completing the assignment altogether.

Alternate Assessment Instruments

Successful mastery of learning objectives cannot be determined by the evaluation of a programming assignment. It is essential that students learn to program in early computing courses; however, a more insightful assessment instrument is needed to determine if learning objectives are met. Furthermore, if the student is motivated by this measure so that her focus is on learning the concept, principle, idea, or technique, and not the completion of the assignment, then there will be less likelihood that the student will engage in plagiarism or otherwise circumvent the intended learning (Abraham & Milligan, 2008).

Several new assessment techniques have been introduced in both freshman- and sophomore-level computer science classes, including the following:

1. Short video production in which the student demonstrates and explains their solution. This assessment technique is similar to an in-class demonstration without costing classroom instruction hours. The student demonstrates and explains their work, justifies design choices, and elaborates on difficulties. Often, a prompt is included, directing the student to comment about some particular aspect of their work, such as efficiency, error-handling, or alternate solutions. Time limitations are also provided to keep discussions concise and on target.
2. Research-style poster presentation of a particular concept. This assessment technique requires the student to prepare a poster elaborating on a particular concept, often tied directly to a learning objective. The poster allows the student to demonstrate mastery of a concept without having to produce a program as an artifact.
3. Research-style poster presentation of a "large" project reflecting upon the major development phases. Many computing courses at GCU include a significant project that spans the entire semester. These types of projects are developed in well-defined stages. This assessment instrument allows the student to revisit each stage, comment on its role in the overall project life-cycle, and reflect on what worked well and what could have been improved during project development.
4. Essay response exams. This assessment works well in courses that employ written testing. For this technique—in advance of the exam—students are given a slate of approximately 10 questions that are eligible for inclusion on a free-response exam. Questions are open-ended and require explanation (or evaluation) of some paradigm, technique, structure, or potential solution. Materials, such as "cheat sheets," are not allowed on test day, but students are free to think about and prepare answers to questions in advance of the exam, often leading to a more thorough and more targeted preparation. A small subset of the original slate of questions is present on the exam, allowing adequate time for thorough responses, which are often augmented by drawings, charts, and the like.

Observations and Benefits.

Each assessment instrument described above is developed to directly target and measure learning objectives. All four instruments contain specific prompts designed to get the student to demonstrate their knowledge of a particular course objective, either through a verbal presentation or written response. In all cases, the instructor is able to measure the student's mastery of the objective directly.

There are additional benefits as well. All techniques require the student to prepare an explanation of material and to organize subject matter. It has been shown that this type of

preparation and organization results in increased learning over students who are studying material only for themselves (Bargh & Schul, 1980). These assessment techniques also provide much needed practice with communication skills. Computer science students must be able to communicate technical information to a variety of audiences, and therefore must be engaged in written and oral presentation activities (Beaubouef, Zhang, Alkadi, & Yang, 2011). The video and poster exercises require an oral presentation of material. Video presentations provide the reluctant or shy student an opportunity to practice and deliver presentations in a safe setting. Both the poster and the video deliverables can alleviate the stress incurred when a programming assignment does not succeed by allowing the student to present their work and outline challenges. Reducing student anxiety and increasing comfort in the course has been shown to be a leading predictor of success in the CS curriculum (Wilson, 2002). Finally, all assessment techniques described here allow the instructor to provide useful feedback that addresses specific points in the student's arguments.

Summary

The integration of assessments that require the student to communicate details of their work are a healthy addition to the introductory computing curriculum. Such assessments provide a reliable means of measuring student mastery of course objectives. Plagiarism of work is less relevant, as students must offer explanation for all deliverables. I have observed that students react positively to presentation assignments and enjoy talking about their successes and having the opportunity to share challenges encountered. Looking forward, research is needed to establish the benefit of other positive effects of the described assessment methods, including:

1. Improved ability to communicate within the discipline.
2. Increased confidence in professional interactions.
3. Increased intellectual engagement in future academic pursuits.
4. The effectiveness of targeted feedback on student.

References

Abraham, S., & Milligan, G. (2008). Software plagiarism in undergraduate programming classes. Information Systems Education Conference. Phoenix, Arizona.

Bargh, J. A., & Schul, Y. (1980). On the cognitive benefits of teaching. Journal of Educational Psychology, 72(5), 539-604.

Beaubouef, T., Zhang, W., Alkadi, G., & Yang, K. (2011). Beyond the computer science curriculum: Empowering students for success. Journal of Computing Sciences in Colleges, 26(4), 21-27.

Ngo, M. N. (2016). Eliminating plagiarism in programming courses through assessment design.

International Journal of Information and Education Technology, 6(11).873.

Walker, G. N. (2004). Experimentation in the computer programming lab. Inroads--The SIGCSE Bulletin, 36(4). 69-72.

Wilson, B. C. (2002). A study of factors promoting success in computer science including gender differences. Computer Science Education, 12(1-2), 141-164.