Submitted June 2011

Prepared By: Doug Jones
SAC: Computer Science
Outcomes Assessed: Communication, Community and Environmental Responsibility

Introduction

Computer Science is a transfer program designed to allow students to complete their lower-division undergraduate requirements for a four-year degree in Computer Science. Campus-based (Rock Creek and Sylvania) and Distance Learning courses focus on discrete mathematics, system architecture and design, and programming in C, C++, Java, and Assembly. Supplemental courses include Unix/Linux training, video game development, and a Gen Ed (Science) survey course of the discipline.

This report has two sections. The first section discusses changes to the program driven by last year's assessment of Critical Thinking. The second section discusses this year's assessment of Community and Environmental Responsibility, and the recommendations generated from this year's survey.

Changes Implemented From Previous Assessment

The Computer Science SAC (CSSAC) believes that the results of last year's assessment support the conclusion that student's critical thinking skills are sufficient for academic success in the Program, and to meet the goals of PCC's Core Outcomes. Nonetheless, the CSSAC believes that critical thinking skills are a "force multiplier" when meeting the challenges of the discipline, and improved critical thinking skills will confer disproportionate benefits to students upon transfer to their university of choice. The CSSAC implemented revisions to teaching practices to emphasize particular areas of critical thinking. It is important to note that the curriculum must conform to the requirements of our transfer partners, resulting in little flexibility. Revisions to teaching practices, such as technology, lab assignments, examinations, and classroom management, are the Program's most effective tools to achieve PCC's college-wide outcomes.

Critical thinking methodologies in Computer Science emphasize problem analysis and resolution techniques that are independent of the particular technological context of the problem. Assessment results indicate that the techniques employed by some students may be focused on particular technologies, and not helpful when a different technology is required in a course, or when a technology becomes obsolete.

The CSSAC has diversified the technology used in course sequences to provide additional incentive for students to avoid relying on technology-specific problem solving methodologies. Although some technology is common to many courses, students will be confronted with new technology throughout a sequence of courses, along with more complex problems.

Examples of such a technology sequence in this effort include:

- *Unity* software has been introduced in CS 233G, while *Game Maker* software is used in CS 133G

- *CeeBot* software is used in CS 160 to introduce programming concept, while *DevC++* is common in CS 161, and *Visual Studio* is common in CS 162.

- The *make* utility is introduced in CS 140 for source code management, while *Subversion* software is used in later courses

Current Assessment

The CSSAC assessed the *Communications* and *Community and Environmental Responsibility* outcomes for the 2011 report.

The Communications outcome was evaluated by analyzing existing homework submissions in one section of CS 201 and one section of CS 260. The results represent 40 students.

The Community and Environmental Responsibility outcome was evaluated by analyzing the responses to survey questions in one section of CS251, one section of CS 260, one section of 261, and one section of CS 201. The results represent approximately 45 students. The survey was conducted as a mandatory, ungraded assignment.

All of the students involved in this effort are 2nd year students who will transfer in the Fall 2011 term (September, 2011). Instructors used a common rubric to evaluate their student's data. The results were aggregated and discussed by the CSSAC at the April 26th, 2011 meeting.

Results of this analysis include:

- Students have generally good communications skills, but there are some important caveats:
    - Language skills are critical to communications. Students with poor English skills have poor communications skills
    - Conversational fluency in English is not a good indicator of communications skills for Computer Science. Effective communications requires mastery of a technical vocabulary.
    - Students are more skilled at verbal than written communications. While most students have satisfactory verbal skills, many students have inadequate writing skills.
- Students are aware of the environmental issues involved in the disposal of old computer equipment
- Most students support community e-cycling programs
    - International students may have a different "green ethic" and may make different judgments about recycling
- Most students do not believe program efficiency is a community or environmental responsibility issue
    - Students are generally unaware of the power requirements of different types of computers
- Most students are aware of basic malware avoidance procedures when using the Internet

The results of our analysis suggest several possible revisions to improve our Program's support for PCC's college-wide outcomes.

In no particular order, the first revision is to increase the requirement for group work. It appears that our students lack teamwork skills, and in particular are unable to identify and address teamwork problems.

Teamwork is extremely important in this discipline, as it is in most other fields, and our students need to be able to solve teamwork problems as well as technical problems. In addition to imparting collaborative skills as part of the course material, teamwork will receive increased emphasis on lab assignments and other assessments in all CS courses.

The second revision is to increase the writing and composition component of our courses. Students tend to have strong programming and technical skills but lack the written communications skills necessary to communicate the details of the discipline, and can be unable to describe problems in sufficient detail to permit analysis.

Programming courses will require more rigorous design and internal source code documentation. Non-programming courses (such as game design) will increase emphasis on properly constructed proposals and project plans.

A third revision is to provide additional support to help students learn the technical vocabulary used in Computer Science. Additional resources such as web-based dictionaries, translation guides for technical terms, and emphasizing technical vocabulary in tutoring will improve communications skills.