21st Century Global Software Development and Education Position Statement

James H. Cross II
Auburn University
cross@eng.auburn.edu

Abstract

This position statement outlines methods by which universities can determine industry needs and provides recommendations to reverse declining enrollments trends. Suggestions are offered for curricular topics important to software development in the future as well for software tools that will be necessary to make these topics less formidable for undergraduates.

1. Introduction

The U.S. Bureau of Labor Statistics has predicted software engineers to be one of the top ten fastest growing job occupations during the period 2002-2012 (see http://www.bls.gov/news.release/ecopro.t04.htm), yet the enrollments in computer science and similar programs are currently declining. The burst dot-com bubble and the extensive press regarding off-shore development of software have taken a toll on the available jobs for computing majors. While other countries have had undergraduate software engineering degree programs in place for years, these are just beginning to be implemented and accredited in the U.S. However, the effect these new programs will have on enrollments is unclear.

2. Meeting Industry Needs

Perhaps a more important issue is whether our degree programs are meeting industry’s needs. An industrial advisory board consisting of technical leaders and project managers can provide useful insight into current and future industry needs. In addition, carefully designed surveys for employers of current graduates can also be useful in determining the characteristics of an ideal graduate from the viewpoint of the company.

3. Attracting and Retaining SwE Majors

Efforts to attract students to the discipline may need to begin as early as middle school and continue through the first year of college. Many projects to address this issue are underway including efforts to broaden participation by minorities and women. However, the lack of retention of students that enter the program is somewhat more troubling. The first few “formative” courses can be critical to the success of students. At public institutions, experience indicates that as many as half of the students who begin in computing do not complete their degrees. While a portion of these are indeed in the wrong major, others just do not get “hooked” or simply cannot get past some hurdle in an early course. Twenty years ago, object-oriented programming and design, with its inheritance and polymorphism, were advanced topics. Now these topics are commonly taught in the first course and care must be taken not to lose our prospective software engineers as a result of technical overload. Fortunately, pedagogically sound tools and environments are available that help reduce the complexity of these topics (e.g., BlueJ and jGRASP).

With a solid foundation in the first courses, students are more likely to be successful in the core software engineering courses such as software construction, design and modeling with UML, quality assurance, and process. A capstone senior design project, which is found in most engineering curricula, should be one the most important activities in the curriculum. This team-oriented activity is intended to emulate a professional experience as it draws together much of the student’s previous coursework.

4. Topics and Tools for the Future

Many of the special topics courses in today’s curricula will become part of the advanced core in the future. There will likely be increased reliance on managed code for the Java and .Net platforms. Platforms for software projects will include small devices such as PDAs and cellular phones, as well as embedded systems. As advanced architectures with multiple CPUs become commonplace, programming techniques for multitithreading will need to receive more emphasis in upper level undergraduate courses. To make these topics less formidable for undergraduates, it will become increasingly important to make the use of appropriate software environments and tools an integral part of the courses. For example, special environments for cell phone software development and certification will be needed.