Software Engineering Education (SEEd)
Peter B. Henderson
Department of Computer Science & Software Engineering
Butler University
Indianapolis, Indiana 46208 USA
phenders@butler.edu

The March 2004 issue of SEN just arrived so I don’t have any feedback from my previous SEEd column to share with you. However, I did attend CSEE&T 2004 (Conference on Software Engineering Education & Training), the SIGSCSE Technical Symposium on Computer Science Education 2004, and the working group on software engineering education in Norfolk, VA the first week of March - will report any activities of interest regarding software engineering education in this column.

If you have not already done so, please read Joe Saur’s “Software Engineering Education (SEEd) My $0.02” letter to the editor on page 2 of the March 2004 SEN. Joe points out that many of the skills he uses on the job he didn’t learn in his CS education and that his view is that what he does is neither a science nor engineering. In response to the latter I would argue that most of the items on his job list, Project Management, documentation, Configuration Management, and Quality Assurance are engineering based.

Should these, and the other jobs Joe lists be the focus of an undergraduate software engineering program? There are many intertwining issues here a few of which are: a) What should the focus of an undergraduate experience be - education or training?, b) The academic world versus the real world, c) Engineering as an apprenticeship discipline. If you ask practicing engineers “How much of what you learned in college do you use in your job?” the answer would most likely be very little. In response to the question “How much of what you learned on the job do you use in your job?” the answer would probably be a lot. And the general response to the question “Did your undergraduate engineering education adequately prepare you for your job?” would be yes. I believe these are the responses we should be seeking from our software engineering graduates because engineering education has a long tradition of success that we should try to emulate.

Of course there are alternative views. I have heard it said that software engineering is more like industrial engineering that traditional engineering due to its focus on process rather than technical details. What may be more interesting is the potential for cross fertilization in which the better ideas from software engineering and traditional engineering migrate into each other.

Computing Curricula 2004 : The Overview Project

In addition to creating curricula volumes for computer science, information systems, software engineering and computer engineering there are now efforts to develop an overview volume and potentially a new information technology volume. The overview project has several objectives:

a) To develop an overview volume that tries to identify commonalities and differences between the computing disciplines and gives an overarching view.
b) To provide a framework for continued computing curriculum activities including the potential for new areas such as information technology.
c) To create a brief descriptive document which provides information about careers in the computing disciplines for prospective students and their parents, school guidance counselors and college admissions offices. This will be an extremely important document.

It is good to see these computing curricula efforts flourishing and maintaining momentum. In the past they would pop up every 10 to 12 years then fade away.

Computing Curricula Software Engineering

CCSE earned kudos from the computing curricula committee for its successful efforts to bring together an international constituency. For software engineering education, this is a “no brainer” since most of the undergraduate software engineering programs worldwide have developed outside the US over a longer period of time. Isn’t it great that so many software engineering educators worldwide can work together in the best interests of the discipline?

If you are interested check out the progress of the second draft of the CCSE Volume at http://sites.computer.org/ccse/ . The second review of this document will be completed on March 22, 2004 and the final version of the volume should be available before the end of the year.

To help colleges and universities that are considering developing undergraduate software engineering programs using the CCSE Volume workshops are being offered in conjunction with various computing conferences. A workshop entitled “Generating Undergraduate Software Engineering Programs Using the Computing Curricula Software Engineering Volume” was given at CSEE&T 2004. Similar workshops will be offered at ITiCSE 2004 in June and FIE 2004 in October. If you, or someone you know associated with an institution of higher education are interested in starting an undergraduate software engineering program, please email me and I will put you in contact with the appropriate parties.

Bulldozer Thread

Human psychology is neat! Why does an email thread from 5 years ago stick in my mind? I call this the “bulldozer thread” because it began with the following paraphrased comment at the ACM SIGCSE 1999 Symposium town meeting: “The role of programming in computer science was questioned through the observation that civil engineers do not find it necessary to master bulldozer driving, but software engineers and computer scientist must be able to code programs in addition to be able to design them.” This lead to a long discussion on the ACM SIGCSE...
listserver which I kept and read every so often. It also struck home since during my undergraduate engineering education I spent significant time with civil engineering students. They learned to use surveying tools, etc., but not bulldozers. Electrical engineering students didn’t learn how to solder. Of course, we all had slide rulers – am I dating myself?

This brings to mind the distinction between an architect and a builder - one primarily designs and the other primarily constructs. This could be a house, office center, mall, automobile, or a software system (for a related perspective read the editors filler “Software Engineering Degree versus Software Architect Degree” on page 16 of SEN March 2004). This naturally leads to the questions “Is a software engineer an architect/designer or a builder/coder?” “Should a software engineer know how to architect/design software systems?” “Should a software engineer know how to build/code software systems?” Probably both, but to what relative degree?

I predict that as software engineering matures, the educational emphasis on coding/programming will diminish; however, we must always strive to seek the proper balance between rigor, discipline, abstraction, design and practice. Education in traditional engineering disciplines emphasizes mathematical and scientific foundations upon which the fundamental principles of the discipline are build, problem solving, professionalism, ethics, etc. – preparation for engineering large, complex systems. Outsourcing of coding will also have a significant impact on the balance between design and programming for software engineers.

Conference on Software Engineering Education and Training 2004

CSEE&T 2004 (http://www.cs.virginia.edu/~cseet04/) was held in Norfolk, Virginia (USA), March 1-3, 2004 and included three keynote presentations: “Focusing Software Education on Engineering” by John Knight, University of Virginia; “Will the Real Software Engineer Please Stand Up?” by Frank Anger, National Science Foundation; and “Software Security Clue Distribution” by Gary McGraw, Chief Technology Officer, Cigital Labs. There where also workshops, papers presentations and panel sessions addressing a wide range of issues relating to software engineering education and the discipline. The conference proceedings are available from the IEEE Computer Society (Order Number PR2099).

Attending CSEE&T and the Working Group on Software Engineering Education, February 28-29 left some interesting impressions about the status of software engineering education worldwide and in the US. In summary:

- The growth of new software engineering programs seems to be slowing. This is probably in response to the general downturn in the technological economy and uncertainty about outsourcing.

- Software engineering education, being one of the new kids in the computing education arena, is still seeking respect, coordination and a preverbal home. The CCSE effort is an important step toward defining undergraduate SE education, earning respect and coordinating activities, but more is required. Accordingly, to assist in these endeavors an umbrella organization will be formed. One of its goals will be to provide a rallying point for software engineering education activities that currently seem to be spread amongst a variety of groups, initiatives, conferences, etc. (CCSE, FASE, SEPLA, WGSEE&T, CCSEE&T are some of the acronyms).

- A recent US National Institute of Standards and Technology (NIST) study claims software errors cost the US economy approximately $60 billion US dollars per year, or about 0.6% of the gross domestic product. Wonder what the figure is for the world?

See http://www.nist.gov/public_affairs/releases/n02-10.htm

These are some of the highlights. Of course there is much more, so I have asked keynote speaker John Knight to be a quest editor of a future SEEd column entitled “Focusing Software Education on Engineering.” If you have been following my column you understand why his title/presentation resonates with me. Here is one quote from John’s abstract “The application of rigor should be routine as in classical engineering disciplines.” Sound familiar? Please stay tuned!!

SEEd Home Page: Current and past SEEd columns can be found at the following URL http://blue.butler.edu/~phenders/SEEd

A personal comment – I hope sometime in the future the word “Training” will be dropped from the name of the CSEE&T conference. There is a distinction between “education” and “training.” I strongly believe that to develop high quality undergraduate software engineering programs the focus must be on education.

1 I like the phrase “classical engineering” better than “traditional engineering.”