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

Again, I waited until the last minute – I work best under pressure. Hope so this time! OK, so what is the purpose of this column? Who reads it? Who is interested in software engineering education and why? Why do we even need a discipline of software engineering? Doesn’t computer science suffice to prepare software practitioners?

In the last SEN I wrote a response to a letter submitted by William Griswold regarding the ongoing CCSE (Computing Curricula for Software Engineering) efforts [1]. In particular, his view is that software engineering is a sub-discipline of computer science. This implied to me a belief that there is little distinction between students who graduate with computer science degrees versus those who graduate with software engineering degrees. In other words, why bother to introduce a new degree when a computer science education with a concentration in software engineering suffices?

So, the natural question is “What is the difference between computer science and software engineering education?” Or, perhaps rephrased with an eye towards the future “What should the difference be between computer science and software engineering education?” This has been a theme of previous SEEEd columns.

In my response, I tried to make a case for software engineering as a separate discipline using the argument that in general computer science graduates are educated differently from traditional engineering graduates in a number of ways. Specifically, many CS graduates don't have an engineering mindset, which includes an emphasis on what it means to be a professional engineer, don’t follow the rigorous, well defined apprenticeship career building model of engineering, and often lack the foundational mathematics and science backgrounds required of traditional engineering graduates.

Strong opinions - perhaps. At least one reader took me to task for my statements that seemed to imply that I was claiming that software practitioners are unprofessional and my comments would hurt the profession. I am sorry if others inferred this also because I do believe that most software practitioners, just like most practicing engineers, are professionals. However, I do believe there are different expectations in the corporate world when employing a licensed professional engineer versus a software engineer.

Almost anyone with some computer experience can call themselves a programmer, software developer and/or software engineer. This started over 40 years ago when companies such as IBM hired college graduates, mainly mathematics and music majors, to retrain as programmers since computer science/software engineering graduates were scarce. At that time, before the Mythical Man Month [2], this practice was also viewed as being cost effective (Is this similar to the growing practice of outsourcing software development overseas?) Careers in the computer field generally required less preparation time and less rigorous training (e.g., all educational programs weren’t accredited, no professional licensing, apprenticeship growth was not stressed) than those for the traditional engineering professions. This was beneficial to everyone - the workforce (relatively easy to acquire basic skills and find a job), the educational institutions (growth of computer oriented programs of study) and the software industry (need to fill positions).

Engineers are entrusted with the protection of society - lessons learned from numerous physical engineering failures have evolved into rigorous requirements for practicing engineers. Software failures are currently viewed by society as expected and mostly an inconvenience. These are not the standards a true engineering discipline aspires to. Today, software development is a bit like an unexplored territory, exciting and untamed. Software engineering is currently in its learning and maturing phase, and SE education is an important component of its growth. Yes, I believe that given time, effort, and discipline software engineering will become a true engineering discipline.

As the old saying goes, “The only constant is change.” But people fear change because of the unknowns. Today, outsourcing, licensing and software engineering education are some of the unknowns for the IT industry. In my first SEEEd column a year ago I made the prediction that “Within 50 years, software engineering will supplant computer science as the educational discipline for professional software developers.” So there are now 49 years left.

Will software engineering education, licensing, and more rigorous requirements for software practitioners help to improve the quality of software and help to temper societies view of the software industry? I would like to think so.

Free Advice: Would You Tell A Kid To Study Programming?
This is the title one of a series of articles that appeared in the November 17, 2003 issue of Information Week [3]. Here is one quote from this article “…the future of software engineering isn't in America, it's in lower-cost countries.” But the article is not all doom and gloom for US software practitioners. Indeed, it points out that good opportunities will always be available for interested and dedicated young people who are well-educated in the basics and engineering. I believe a shift of focus toward software engineering education will help to address this by raising the level of expectations of its programs, graduates and employers. Other articles in this issue are titled: “Computer Science Is More Than Programming,” “Future View: Software Jobs Will Be Mechanized In Long Run” and “The Programmer's Future.”

Notice the use of the terms ‘computer science,’ ‘programming,’ ‘programmer,’ and ‘software jobs’ in these four article titles. Indeed, the term ‘software engineer’ appears only once in the bodies of these four articles (see quote in the previous paragraph). The
name game! The people responsible for getting a software product out the door, on time and near cost, and for maintaining that product are software engineers – pure and simple (well, at least it should be).

**IT downturn and overseas outsourcing, reality checks**

The growth of IT careers in the 20th century was phenomenal, and is so still in many countries. Reality always has a way of providing balance. “The lower echelons of the skill levels are going to be washed away,” is one quote from the article “The Programmer’s Future.” Actually what I believe is happening is that opportunities in computers other than programming, computer science and software engineering are growing rapidly.

For those promoting software engineering and SE education this is an ideal opportunity to significantly increase the skill level of future software professionals and help to address the quality issues of the software industry.

**Is software engineering the silver bullet?**

No, but hopefully it is a step in the right direction for a maturing software industry.

"10 pounds in a 5 pound sack"

Thanks to Will Tracz for this quote in a recent email exchange we had discussing undergraduate software engineering education. Basically, how can SE educators possibly fit all the concepts, information and knowledge required into an undergraduate SE curriculum? The **CCSE** effort [1] made this clear to me since there were so many “things” SE educators felt were important and should be included in the curriculum. Some believed this could not be done (“10 pounds in …”), or that the discipline was not sufficiently mature to define a curriculum for an undergraduate SE program, or that only graduate programs would suffice to prepare software engineers.

I feel now is a good time to start this endeavor. The discipline has matured, knowledge areas have been defined in software engineering body of knowledge (**SWEBOK**) [4], and the software industry in many countries is facing tough times, both from competition elsewhere and from the public perception. High quality undergraduate software engineering education can help to address all of these issues. Remember this is the only the starting round of the **CCSE** and the volume will be updated to reflect a maturing discipline and improved teaching techniques.

My personal perspective is that such programs should primarily focus on developing technical competence. Undergraduate studies are the best time for students to learn, apply and reinforce foundational concepts and ideas to become technically competent software engineers. Other aspects such as the management of software projects, and all this entails, can be learned through apprenticeship, experience and graduate studies. However, it is very important that the seeds for management and life long learning be planted in undergraduate programs.

**CCSE (Computing Curricula Software Engineering)**

By the time you receive this the second draft of the **CCSE** volume should be available at [1], and I hope you will respond to requests to critique the volume. Sometime this year the final volume is anticipated. In the next installment of **SEE**d I plan to report on the continued growth of undergraduate SE programs throughout the world.

**SEE**d Home Page: Current and past **SEE**d columns can be found at the following URL: [http://blue.butler.edu/~phenders/SEEed](http://blue.butler.edu/~phenders/SEEed)

**Acknowledgements**

I would like to thank Will Tracz, SEN editor, for the great job he is doing and putting up with my last minute contributions. Even so, he rewarded me with a holiday fruit basket with our family thoroughly enjoyed. Keep up the great work Will.

**References**

[1] [http://sites.computer.org/ccse/](http://sites.computer.org/ccse/)


