Is Software Engineering an Engineering Professional Discipline?

Recently two articles have been published addressing issues regarding the discipline of software engineering. Software Engineering - Missing in Action: a Personal Perspective by David Parnas[1], and Will Software Engineering Ever Be Engineering? by Michael Davis[2]. To summarize, Parnas argues that software practitioners have failed in their goal of creating a new engineering discipline, and Davis argues that software engineering is not part of engineering because it is not “fundamentally about physical systems”.. We encourage you to read these two articles to gain a better perspective on these important issues.

What does all this have to do with software engineering education? As you might guess, a lot! In his first SEEEd column, March 2003, Henderson predicted that within 50 years software engineering undergraduate programs would supplant CS programs as the primary preparation for software practitioners[3]. A conservative prediction, but let’s wait another 42 years to see what happens. It is worth pointing out that the number of undergraduate software engineering programs has been steadily growing. At last count there were 40 BSSE programs in the United States alone, 21 of which are accredited by ABET. In 2003 there were about 20 BSSE programs in the U.S. and 7 were under review for ABET accreditation.

In a 2004 SEEEd column [4] Henderson received substantial flack from readers when he contrasted the goals of CS and SE programs: “…many CS graduates don’t have an engineering mindset, which includes an emphasis on what it means to be a professional engineer, …”. How dare you claim that computer scientists are not professionals!!! Subsequently this contrast was addressed very eloquently by Ewan Tempero in his article Software Engineering & Computer Sciences — Two Worlds [6].

There is still considerable misunderstanding and confusion regarding the differences and similarities between the disciplines of computer science and software engineering. Parnas clarifies the general distinction between engineering and science: “The engineering student must learn how to use science and mathematics to build things, while the science student must learn how to add new knowledge to previously known information.” However, computer science students are typically taught to ‘build things’ without sufficient advice on how to build them or how to evaluate their results. For an alternative discussion of the differences between computer science and software engineering see Computing Curricula 2005: The Overview Report, published by the ACM and the IEEE Computer Society as a guide in selecting an undergraduate computing program.

Another hot topic, in our view, is the role of mathematics, which is not explicitly addressed in Davis’s CACM article, but is in Parnas’s IEEE Computer article. Parnas claims that the role of maths in software engineering/development is not nearly as clear as it is in traditional engineering programs. Indeed, he notes that “… some doubt that it {mathematics} has any relevance at all {in software engineering/development}.” In our minds the distinction is that in traditional engineering disciplines foundational mathematics and science are integral components of many engineering courses. For example, one can’t model, or understand, the behavior of an electronic circuit without mathematics. However, many parts of software engineering have not yet attained this level of sophistication, partly because of the general abstract nature and the complexity of this young, emerging discipline.

Many of you probably know at least one traditional engineer. When you have a chance ask them what they think of the discipline of software engineering. Is it an engineering discipline? Is it a professional discipline? In the United States you might be presented with the following definition of engineering from ABET (Accreditation Board for Engineering and Technology)¹:

“The profession in which a knowledge of the mathematical and natural sciences gained by study, experience, and practice is applied with judgment to develop ways to utilize, economically, the materials and forces of nature for the benefit of mankind”

If you are perceptive you will quickly identify at least two key phrases which may trigger an argument that SE is not a true engineering discipline. These are: ‘natural sciences’ and ‘forces of nature.’ Davis: insists that all engineering is fundamentally about

¹ By the way, in the U.S. ABET accredits both CS and SE programs.
physical systems, and he complains that some electrical and computer engineering departments have stopped requiring statics, dynamics and thermodynamics in their curricula. He also complains about other engineering disciplines that have started to substitute biology for physics or chemistry.

Software engineering is not based upon natural science. It is based upon the "sciences of the artificial" [7]. But software engineering is not alone. As Simon points out, "It has been the task of engineering schools to teach about artificial things: how to make artifacts that have desired properties and how to design." So *all* engineering depends to some extent on the sciences of the artificial, especially the science of design. By the way, the reason that ECE departments are giving up statics, dynamics and thermodynamics is to allow time for their students to study more discrete math, computer science and software engineering!!

To address such issues, in [4] Henderson presented the following refined definition of Engineering (Note: underlining represents refinements, and accepting CS as a science):

“The profession in which a knowledge of the foundational mathematics and sciences gained by study, experience, and practice is applied with judgment to develop ways to utilize, economically, the materials and laws of nature for the benefit of mankind”

To date, he has not heard from ABET regarding plans to consider his refined definition. **SMILE**

References