Putting the Engineering (back?) into Software Engineering

Dr. David A. Cook
Stephen F. Austin State University
cookda@sfasu.edu

Dr. Eugene W.P. Bingue
U. S. Navy
dr.bingue@gmail.com
Who I am

- Retired AF Officer (23 years) where I qualified for the Air Force Specialty Code for Software Engineer. I taught software engineering at the USAF Academy and the Air Force Institute of Technology. I also taught at Keesler AFB (technical training) for 6 years
- Former Consultant for the Software Technology Support Center (12 years)
- Professor of Computer Science, Stephen F. Austin State University
- Columnist for Crosstalk, the Journal of Defense Software Engineering (I write the Backtalk column for every issue)
- ABET Program Evaluator since 1998, and a Commissioner and/or Team Chair since 2007
Who is paying for this trip?

SFA is in Nacogdoches, TX. It is NOT in Austin, nor is it associated with any school of higher learning in Austin.

In fact, it is generally acknowledged by those of us who graduated from Texas A&M that there ARE no schools of higher education in Austin. 😂
What this talk is about

• What “typical” software engineers are ready for out of school

• What they are lacking

• How to include engineering skills into software-educated developers

• How to improve software skills in engineering-background developers
PREMISE

• We don’t really know what a software engineer is or does – except that they “engineer software”
• This is NOT a criticism – software engineering is HARD – and requires a lot of skill that separate it from computer science, information technology, or even “traditional engineering”
The Easy(?) part - Non-technical skills

• Communication skills
  • Interpersonal communications
  • Written communications

• Leadership

• Conflict resolution

• How to set an example, and “walk the walk, not just talk the talk”
Non-technical skills continued

• How to help?

• In a perfect world, all software engineers would have taken
  • Technical Writing
  • A public speaking or oral communications class

• In reality – not always required by many academic institutions
Oral and Written Communications

• NOBODY wants to admit that they can’t write well

• More people are afraid of speaking in public that of dying
  • .....which suggests that folks would rather be the guest of honor at a funeral
    that have to give the eulogy

• Overcome by
  • Practice, practice, practice
  • Mentoring with effective feedback
  • “Cold” reading, “Viewing” of presentations
Find self-help

• One example is “Tongue and Quill” – an AF publication.
  • Covers how to write and speak
  • Yes – very military based, but LOTS of nuggets of information to glean
  • Contains simple yet effective advice

<table>
<thead>
<tr>
<th>Plain Language Concepts (From AFI 33-360, Table 6.3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Be Clear</td>
</tr>
<tr>
<td>• Use plain language whenever possible; avoid jargon</td>
</tr>
<tr>
<td>• Avoid overuse of acronyms (if used, make certain they are established</td>
</tr>
<tr>
<td>[written out] upon first use)</td>
</tr>
<tr>
<td>• Use the active voice</td>
</tr>
<tr>
<td>• Format your documents so that it’s easy to read and understand</td>
</tr>
<tr>
<td>• Use table and figures if that’s the best way to show information</td>
</tr>
<tr>
<td>Be Concise</td>
</tr>
<tr>
<td>• Remove unnecessary words</td>
</tr>
<tr>
<td>• Write sentences with 20 words or fewer and that contain a single</td>
</tr>
<tr>
<td>thought, action, etc.</td>
</tr>
<tr>
<td>• Use seven sentences or fewer per paragraph</td>
</tr>
<tr>
<td>Be Specific</td>
</tr>
<tr>
<td>• Include only information that the reader must know</td>
</tr>
<tr>
<td>• Use words with precise meaning</td>
</tr>
<tr>
<td>• Include details that are directly relevant to the main point</td>
</tr>
</tbody>
</table>
The HARD part – technical knowledge

• First – let’s examine the requirements between the different “types” of degrees

  • Software Engineer
  • Computer Science
  • Information Technology
  • Computer Information Systems
Definition of ENGINEERING
The activities or function of an engineer

• a: the application of science and mathematics by which the properties of matter and the sources of energy in nature are made useful to people
• b: the design and manufacture of complex products <software engineering>

Definition of SCIENCE
The state of knowing: knowledge as distinguished from ignorance or misunderstanding

• a: knowledge or a system of knowledge covering general truths or the operation of general laws especially as obtained and tested through scientific method
• b: such knowledge or such a system of knowledge concerned with the physical world and its phenomena: natural science
How do we determine what engineers know?

• We need to look at the accreditation agency for all three

• ABET (formerly Accreditation Board for Engineering Technology)

• ABET accreditation sets the global standard for programs in applied science, computing, engineering, and engineering technology.
ABET (from their web site)

• We are a not-for-profit, non-governmental accrediting agency for programs in applied science, computing, engineering, and engineering technology and we are recognized as an accreditor by the Council for Higher Education Accreditation.

• ABET accreditation provides assurance that a college or university program meets the quality standards of the profession for which that program prepares graduates.

• We accredit programs, not institutions. We provide specialized accreditation for post-secondary programs within degree-granting institutions already recognized by national or regional institutional accreditation agencies or national education authorities worldwide.

• Our accreditation is voluntary, and to date, more than 3,400 programs at nearly 700 colleges and universities in 28 countries have received ABET accreditation. Approximately 85,000 students graduate from ABET-accredited programs each year, and millions of graduates have received degrees from ABET-accredited programs since 1932.
Differences between programs

- Software Engineering falls under the EAC (Engineering Accreditation Commission), which sets general and program-specific requirements.

- Computer Science falls under the CAC (Computer Accreditation Commission), which sets general and program-specific requirements.

- Software Engineering:
  - Lead Society: CSAB (Computer Science Accreditation Board)
  - Cooperating Society: Institute of Electrical and Electronics Engineers

- Computer Science / IT / CIS:
  - Lead Society: CSAB (Computer Science Accreditation Board)
General Criteria - Engineering

**GENERAL CRITERION 3. STUDENT OUTCOMES**

The program must have documented student outcomes that prepare graduates to attain the program educational objectives.

Student outcomes are outcomes (a) through (k) plus any additional outcomes that may be articulated by the program.

(a) an ability to apply knowledge of mathematics, science, and engineering.

(b) an ability to design and conduct experiments, as well as to analyze and interpret data.

(c) an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.

(d) an ability to function on multidisciplinary teams.

(e) an ability to identify, formulate, and solve engineering problems.

(f) an understanding of professional and ethical responsibility.

(g) an ability to communicate effectively.

(h) the broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context.

(i) a recognition of the need for, and an ability to engage in life-long learning.

(j) a knowledge of contemporary issues.

(k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.
General Criteria - Engineering

GENERAL CRITERION 5. CURRICULUM

The curriculum requirements specify subject areas appropriate to engineering but do not prescribe specific courses. The faculty must ensure that the program curriculum devotes adequate attention and time to each component, consistent with the outcomes and objectives of the program and institution. The professional component must include:

(a) one year of a combination of college level mathematics and basic sciences (some with experimental experience) appropriate to the discipline. Basic sciences are defined as biological, chemical, and physical sciences.

(b) one and one-half years of engineering topics, consisting of engineering sciences and engineering design appropriate to the student’s field of study. The engineering sciences have their roots in mathematics and basic sciences but carry knowledge further toward creative application. These studies provide a bridge between mathematics and basic sciences on the one hand and engineering practice on the other. Engineering design is the process of devising a system, component, or process to meet desired needs. It is a decision-making process (often iterative), in which the basic sciences, mathematics, and the engineering sciences are applied to convert resources optimally to meet these stated needs.

(c) a general education component that complements the technical content of the curriculum and is consistent with the program and institution objectives.

Students must be prepared for engineering practice through a curriculum culminating in a major design experience based on the knowledge and skills acquired in earlier course work and incorporating appropriate engineering standards and multiple realistic constraints.

One year is the lesser of 32 semester hours (or equivalent) or one-fourth of the total credits required for graduation.
Specific Criteria - Software Engineering

SOFTWARE AND SIMILARLY NAMED ENGINEERING PROGRAMS

Lead Society: CSAB
Cooperating Society: Institute of Electrical and Electronics Engineers

These program criteria apply to engineering programs that include “software” or similar modifiers in their titles.

1. Curriculum
The curriculum must provide both breadth and depth across the range of engineering and computer science topics implied by the title and objectives of the program.

The curriculum must prepare graduates to analyze, design, verify, validate, implement, apply, and maintain software systems; to appropriately apply discrete mathematics, probability and statistics, and relevant topics in computer science and supporting disciplines to complex software systems; to work in one or more significant application domains; and to manage the development of software systems.
General Criteria – CS, CIS and IT

**GENERAL CRITERION 3. STUDENT OUTCOMES**

The program must have documented student outcomes that prepare graduates to attain the program educational objectives. There must be a documented and effective process for the periodic review and revision of these student outcomes.

The program must enable students to attain, by the time of graduation:

(a) An ability to apply knowledge of computing and mathematics appropriate to the program’s student outcomes and to the discipline

(b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

(c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

(d) An ability to function effectively on teams to accomplish a common goal

(e) An understanding of professional, ethical, legal, security and social issues and responsibilities

(f) An ability to communicate effectively with a range of audiences

(g) An ability to analyze the local and global impact of computing on individuals, organizations, and society

(h) Recognition of the need for and an ability to engage in continuing professional development

(i) An ability to use current techniques, skills, and tools necessary for computing practice.
GENERAL CRITERION 5. CURRICULUM

The program’s requirements must be consistent with its program educational objectives and designed in such a way that each of the student outcomes can be attained. The curriculum must combine technical and professional requirements with general education requirements and electives to prepare students for a professional career and further study in the computing discipline associated with the program, and for functioning in modern society.

The technical and professional requirements must include at least one year of up-to-date coverage of fundamental and advanced topics in the computing discipline associated with the program. In addition, the program must include mathematics appropriate to the discipline beyond the pre-calculus level. For each course in the major required of all students, its content, expected performance criteria, and place in the overall program of study must be published.
Specific Criteria - Computer Science

**Student Outcomes**
The program must enable students to attain, by the time of graduation:

(j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices. [CS]

(k) An ability to apply design and development principles in the construction of software systems of varying complexity. [CS]

**Curriculum**
Students must have the following amounts of course work or equivalent educational experience:

a. Computer science: One and one-third years that must include:

1. Coverage of the fundamentals of algorithms, data structures, software design, concepts of programming languages and computer organization and architecture. [CS]

2. An exposure to a variety of programming languages and systems. [CS]

3. Proficiency in at least one higher-level language. [CS]

4. Advanced course work that builds on the fundamental course work to provide depth. [CS]

b. One year of science and mathematics:

1. Mathematics: At least one half year that must include discrete mathematics. The additional mathematics might consist of courses in areas such as calculus, linear algebra, numerical methods, probability, statistics, number theory, geometry, or symbolic logic. [CS]

2. Science: A science component that develops an understanding of the scientific method and provides students with an opportunity to experience this mode of inquiry in courses for science or engineering majors that provide some exposure to laboratory work. [CS]
Specific Criteria - Information Technology

**Student Outcomes**
The program must enable students to attain, by the time of graduation:

(j) An ability to use and apply current technical concepts and practices in the core information technologies of human computer interaction, information management, programming, networking, and web systems and technologies. [IT]

(k) An ability to identify and analyze user needs and take them into account in the selection, creation, evaluation, and administration of computer-based systems. [IT]

(l) An ability to effectively integrate IT-based solutions into the user environment. [IT]

(m) An understanding of best practices and standards and their application. [IT]

(n) An ability to assist in the creation of an effective project plan. [IT]

**Curriculum**
Students must have course work or an equivalent educational experience that includes:

a. Coverage of the fundamentals of

1. the core information technologies of human computer interaction, information management, programming, networking, web systems and technologies. [IT]

2. information assurance and security. [IT]

3. system administration and maintenance. [IT]

4. system integration and system architecture. [IT]

b. Advanced course work that builds on the fundamental course work to provide depth. [IT]
Specific Criteria - Information Systems

Student Outcomes
The program must enable students to attain, by the time of graduation:

(j) An understanding of and an ability to support the use, delivery, and management of information systems within an Information Systems environment. [IS]

Curriculum
Students must have course work or an equivalent educational experience that includes:

a. Information Systems: One year that must include:

1. coverage of the fundamentals of a application development, data management, networking and data communications, security of information systems, systems analysis and design and the role of Information Systems in organizations. [IS]

2. advanced course work that builds on the fundamental course work to provide depth. [IS]

b. Information Systems Environment: One-half year of course work that must include a cohesive set of topics that provide an understanding of an environment in which the information systems will be applied professionally. [IS]

c. Quantitative analysis or methods, including statistics. [IS]
The bottom line is.....

• Those coming from the “Engineering” side of the house know how to be engineers, but might have less background in computer science
  • Limitations of course hours (120 max to graduate, with typically 60+ being state-required University and College requirements
  • Limitations of prerequisites for SE vs. CS majors

• Those coming from the “Computer Science” side of the house know how to be computer scientists, but maybe have 3 hours in “software engineering”
  • NOT taught by engineering dept. – self-taught by the C dept.
More issues

• CS/CIS/IT might have a single team project
• Capstone course or project NOT required
• Team skills are severely lacking (one project meets ABET requirements)
• One course (and maybe not even a full course!) on software engineering and its concepts

• CS programs not required to work with Engineering for the required software engineering course
• In short – they might have software engineering experience, but they are NOT engineers (.... and it really hurts me to say this)
What can we do to help?

• For the engineers
  • More language experience (additional languages)
  • A computer-based course in software engineering

• For CS
  • Experience in “group work”
  • More experience with large-scale engineering
  • More of “Big Picture” architecture and design
  • Ability to work for “adequacy” rather than “optimum”
  • Experience on interaction with non-software parts of engineering
In a perfect world

• CS majors would have to take a series of software engineering courses (that are at least developed in coordination with engineering) that includes a focus on non-software requirements and design

• CS would include a capstone project, that takes a project through from inception to customer transition
  • Group project, with members from various disciplines

• The lack of the above can be mitigated by on-the-job training and patience
In a more perfect world

• We should learn from Civil, Electronic, and Chemical Engineering disciplines.
  • We don’t build bridges in Civil Engineering 101
  • We don’t build Semiconductors in Electronic Engineering 101
  • We DON’T let Chemical Engineering 101 near

• But we have let Software Engineers build systems before they have any true knowledge of either the system or (software) engineering.
A few other suggestions to improve productivity

• Consider the Personal Software Process (PSP) to improve software productivity (and software quality) for those who have limited experience coding – or for those who have quality issues
  • Very helpful for those whose quality issues involve repeated testing

• Consider extending the PSP with the Team Software Process (TSP) to increase group productivity and help with schedule planning
  • Very helpful for those who have trouble meeting deadlines
THANK YOU!!!

If you have questions..... comments..... or want to express a differing opinion.....

David A. Cook
Professor, Dept. of Computer Science
Stephen F. Austin State University

cookda@sfasu.edu