# Software Engineering: Where Does it Belong?

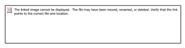
*written by Philippe Gabrini, I.S.P.*

**Software Engineering: the term**
Most people do not remember where they were and what they were doing in October 1968. They also were totally unaware that the term "software engineering" had just been coined by two Algol 60 men, Friedrich Bauer and Louis Bolliet, computer scientists involved in programming languages and compilation. At the time, most computer science people were not aware that they needed the help of software engineering to develop their software systems. And if we are now trying to find where software engineering belongs, it is really because of the legacy from 1968. Ideally, in 1968 our forebears would not have saddled us with the term software engineering. But the rationale behind the choice of that term was that since in civil engineering it was possible to design and build a solid and reliable bridge within a given time and a given budget, it should be possible to do the same thing with software products.

However, most people who are in software development and who have designed bridges (admittedly a very small number) fail to see any similarity between the building of bridges and the building of software. When it comes to developing software, we deal with abstract ideas and thus software is not as constrained by the boundaries of physics and the real world: the construction metaphor is not the best one "Rather than construction, software is more like gardening - it is more organic than concrete." The number one bestseller among computer science books is a book in three volumes titled The Art of Computer Programming by the best known computer scientist, Donald Knuth, suggesting that software development is more an art than a science.

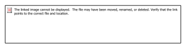
Software engineering is a term that has been adopted over the years, not as a new discipline of engineering but rather as a new discipline of computer science. The title of a very recent book seems to prove it: Software Engineering: an Engineering Approach, an extraordinary title suggesting that titles like Mechanical Engineering: an Engineering Approach or Civil Engineering: an Engineering Approach are possible. With other new terms like "Knowledge Engineering" and "Genetic Engineering" one can but wonder whether the same questions regarding the use of the word "engineering" will be asked over and over.

Using the construction metaphor, methods and process specialists have tended to describe software development as very linear, when it should rather be a spiral. Another way to try and decide where software engineering belongs would be to make an inventory of all books and papers about software engineering published in the past 30 years. Then the authors' background could be examined to see if they were computer science or engineering people. The scale of such a study is beyond our means, but most people would agree that the vast majority of such authors come from computer science, not from engineering. The Straw Man version of the Guide to the Software Engineering Body of Knowledge has used a similar but more limited approach for the definition of the knowledge areas based on 24 textbooks.



## Software Engineering: the definition

Since 1993, the IEEE Computer Society and the ACM have set up a joint Software Engineering Coordinating Committee "to establish the appropriate sets(s) of criteria and norms for professional practice of software engineering upon which industrial decisions, professional certification, and educational curricula can be based." One of the committee's projects is the Software Engineering Body of Knowledge, commonly known as SWEBOK. In order to produce the desired body of knowledge the committee has given a well-known software engineering research lab, the Software Engineering Management Research Lab of the Computer Science Department of UQÀM, the task to help define software engineering. This lab was chosen because of its past involvement in international standardization bodies, such as ISO and IEEE-CS software engineering standards. Such standards organizations develop and structure experts consensus on best practices in various domains, in this instance software engineering, out of a multiplicity of individual views or models on any single topic or technology, thus allowing easier technology transfers and implementations by the industry. It should be noted that it is a computer science department and not an engineering department that was given the task to help define software engineering.



## CIPS involvement

The Canadian Information Processing Society (CIPS) was founded ten years before the birth of software engineering, in 1958. Under its auspices, the University Accreditation Council was formed in 1979, preceding the American Computer Science Accreditation Board (CSAB) by a few years, and initiated the accreditation of computer science programs. Nowadays, the accreditation is the responsibility of the Computer Science Accreditation Council, still sponsored by CIPS. Envisioning a future fully licensed profession, CIPS has also been involved in the certification of computer professionals since 1989, through the Information Systems Professional of Canada designation (I.S.P.) Professional certification provides an assurance that the holder has attained the designated professional qualifications. Accreditation provides one measure to the Certification Council, which reviews the I.S.P. applications, which is that the graduate from an accredited program has met all the academic requirements that effectively prepare the graduate to function and excel in the systems environment found in modern business.

In 1998, the Computer Science Accreditation Council was asked by a number of computer science departments to consider making accreditation possible for a number of undergraduate software engineering programs that had already been developed and implemented. The Computer Science Accreditation Council has thus gone ahead and defined criteria for university undergraduate software engineering programs. These criteria have been recently adopted by CIPS. At the time, the SWEBOK project was barely underway and the council had to use whatever documents existed in order to achieve a meaningful definition. This operation made it clear that defining the software engineering discipline was not a trivial task, even though the council concentrated on software development. The problem was that the number of sources was not large and that finding a common denominator was difficult.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

**Software Engineering: accreditation criteria**
As mentioned earlier, some software engineering programs were already in existence in Canadian universities; the available information used by the Council was thus completed by the descriptions of various software engineering bachelor degree programs or program options from universities across Canada, be they in Science Faculties or in Engineering Faculties. The Council worked with these documents and tried to find a common denominator. What was reached was a consensus of the council members in the form of a list of ten software engineering components as follows:

a. **Requirements and specifications**
   **This is a very standard part of the software development process that is found in all software engineering books. Most course codes for this area are either computer science: CS, CMPT, INF or software engineering: SE, SEG, SENG. Note that software engineering course codes originate either from computer science or computer engineering departments.**
b. **Principles of software analysis, design and architecture**
   **This area is also found and extensively covered in all software engineering books. Most course codes are either computer science or software engineering.**
c. **Design for embedded, real-time or distributed systems.**
   **This is found in most software engineering books, but sometimes placed in the "Special Topics category." Course codes are a mixture of computer science, computer engineering or software engineering.**
d. **Design and evaluation of user interfaces**
   **This topic is found under different names such as "Human Computer Interface" or**

**"Human Factors in Software Engineering." Course codes are mostly computer science with a few software engineering.**

e. **Software construction**
   **Sometimes referred to as Coding, and admittedly the kernel of software development, this area is often neglected in software engineering books. Some people wish to reduce this to a simple translation phase from a perfect design, when the reality is quite different. All course codes are computer science.**
f. **Documentation, tools, components, etc.**
   **This subject is usually not covered independently but is dealt with over the various components of software engineering. All course codes are from computer science.**
g. **Management of the software process, including metrics and maintenance**
   **This is another classical part of software engineering well covered in software engineering books. Course codes are divided between computer science and software engineering.**
h. **System performance**
   **This specialized subject is considered only in a software engineering context. Course numbers are mostly software engineering.**
i. **Quality assurance, testing**
   **For this subject course codes are divided between computer science and software engineering.**
j. **Software safety**
   **Sometimes related to software reliability, all course codes for this area are computer science.**

Looking further at the courses in software engineering programs, it can be noted that all the software engineering course descriptions can be found in computer science departments. Some software engineering course descriptors were found to include Computers and Society, Media, Ergonomics, Network-Centric Computing, Distributed Systems and the Internet, which raises more questions.

From the above list one can see that software engineering, as defined in computer science, includes a number of related and sometimes overlapping subjects which may be required to provide general valuable software solutions. The complete accreditation criteria go beyond the software engineering kernel defined above and comprise other subjects in order to provide an appropriate balance. Besides software engineering, there are five other areas in computing sciences: algorithms and data structures, programming languages, systems software, computer elements and architectures, and theoretical foundations of computer science. The complete criteria also include courses in mathematics and statistics, and encourage students to explore areas outside computing, like engineering, business, humanities or social science.

## Software Engineering knowledge areas

For purposes of comparison, now that the Stone Man version of the SWEBOK is available, its knowledge areas are listed below:

- Software Configuration Management
- Software Construction
- Software Design
- Software Engineering Infrastructure
- Software Engineering Management
- Software Engineering Process
- Software Evolution and Maintenance
- Software Quality Analysis
- Software Requirements Analysis
- Software Testing

**The American solution**

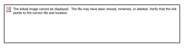
In order to understand the software engineering situation better, it is worthwhile to examine what is being done in the United States. The past year has seen the start of an integration of the accreditation activities for computer science and for engineering. The Accreditation Board for Engineering and Technology (ABET) is a federation of 28 professional engineering and technical societies. Representatives from these societies, who are practicing professionals from industry and academia, form the body of ABET through its Board of Directors and three working Commissions:

- Engineering Accreditation Commission (EAC)
- Technology Accreditation Commission (TAC)
- Related Accreditation Commission (RAC)

The Computing Sciences Accreditation Board (CSAB) was established to provide for accreditation of post-secondary baccalaureate programs that prepare students for entry into the computing sciences professions. The two member societies of CSAB are the Association for Computing Machinery, Inc. (ACM) and the Institute of Electrical and Electronics Engineers, Inc. -- Computer Society (IEEE-CS). The Computer Science Accreditation Commission (CSAC), CSAB's only commission, administers the accreditation process for programs in computer science.

Under the proposed integration, CSAC would become a fourth working commission of ABET: Computing Accreditation Commission (CAC) which would cover accreditation for computer science and possibly later information systems. CSAB will continue as an independent organization and it is anticipated that it will join ABET as one of the participating society members. In this role it will continue its involvement in accreditation for computing degree programs. Under the integration plan CSAB is expected to be the lead society for computer science and for software engineering and a cooperating society for computer engineering (in conjunction with IEEE). This means that CSAB will be in charge of defining the accreditation criteria for software engineering, while EAC will be in charge of the accreditation process. This type of joint responsibility is expected to simplify joint visits (computer science and engineering) and avoid duplication of efforts for the universities seeking accreditation.

There is a will to make the transition a smooth one, and integration is not expected until some time in 2001. In the meantime criteria for software engineering accreditation are still being defined, but there is every indication that these new criteria will not be based on the engineering criteria. For example, it is expected that the software engineering criteria will require more discrete mathematics, and will do away with specific physics, mechanics and strength of materials courses, as well as engineering problems, differential equations and control theory.



**Proceed with care**
CIPS members are all involved in software development and the question of software engineering is vital to them. But the question is not about the definition of software engineering, there are a number of ongoing activities within the computer science community to do just that, including the SWEBOK project. The question is not to determine the degree to which people use software engineering in their software development tasks. The question is more pragmatic: how can the Canadian universities produce technical people who will be productive in the software business and ensure the protection of the public? In some Faculties of Education there is a school of thought that claims that if you possess the pedagogy you can teach anything. Some other faculties take the opposite view: that in order to teach a subject you must master it (particularly true in science). One should be wary of not letting the software engineering question fall in the mould of the pedagogy school of thought, i.e. if one has engineering training (whatever that is), one can do anything in applied science. The dangers that such thinking brings forth are great and should not be underestimated.

One should not forget the fact that the knowledge areas of software engineering, as defined in the future software engineering body of knowledge, are almost entirely taught in computer science departments. The protection of the public is an important issue, that professional certification addresses, but it is not the prerogative of engineering. Unless we have too many resources in the computing field, which is clearly not the case, we must find a way to maximize the production of computing specialists as well as the quality of their education. This can only be done, in a pragmatic manner, by using the available competencies that can be found in computer science departments as well as in computer engineering departments. Any other solution would be counter-productive.

It is also vital to remember that computer science and especially software engineering are young disciplines that have not quite matured yet, are still evolving and should not be frozen in their current larval state. This explains why software engineering, as it exists now, is not the only way in which software may be built, although one will find that whenever good process is used, the steps involved will be similar. Business in general has very little use for pure "software engineers", to use a term that is very much in use, but has huge needs for people with broader technical backgrounds. These needs should be taken into consideration in order to make sure that the decisions to be reached do not restrict the software engineering area to a select group. Firstly, restriction always creates a bottleneck effect. Secondly, when restriction is involved there is a need for a very tight definition of the restricted subject, and this does not seem possible if the current all encompassing results of the software engineering definition efforts are considered. CIPS, through its Accreditation and Certification programs, has tried to define suitable combinations of education and experience to meet the real world needs of business - software engineering is just a component.

**Acknowledgement** CIPS would like to thank Philippe Gabrini, I.S.P. for his efforts as the principal contributor to this position paper. Philippe is a member of the Computer Science Accreditation Council (CSAC) sponsored by CIPS. Philippe Gabrini has been a professor at UQAM for more than 25 years and was a pioneer in the development of UQAM's computer science programs. He is the author of a dozen textbooks on programming languages and software construction and regularly teaches software construction courses. He acts currently as a knowledge area specialist in the Software Construction area of the Software Engineering Body of Knowledge (SWEBOK) project.

CIPS would also like to acknowledge the contributions of CSAC members who took the time to comment upon the draft version of this paper.

Search

## Information

- About CIPS
- Join Us
- Certification (I.S.P./ITCP)
- Program Accreditation
- Defining the IT Profession
- IT Resources
- CIPS Community

## CIPS Log-In

- Membership Log-In
- Job Posting/Intranet