The University of Ottawa's Software Engineering Program: Curriculum Design Issues for a New Subdiscipline

T.C. Lethbridge, R.L.. Probert, J. Raymond, D. Gibbons, D. Ionescu,
L. Orozco-Barbosa, S. Szpakowicz
School of Information Technology and Engineering
University of Ottawa, K1N 6N5

Abstract

The University of Ottawa has recently established a new undergraduate program in software engineering. In this paper we review how we resolved several issues that were raised during the development of the program. These key issues are: a) How should the program be made distinct from computer science and computer engineering programs? b) What should be the core topics in the program? c) What do we need to do to meet our objective of having the program accredited by the Canadian Engineering Accreditation Board?

1. Introduction

Undergraduate programs entitled 'Software Engineering' are appearing in many universities. In Europe and Australia, there have been programs for up to ten years (e.g. Cowling 1998, University of Melbourne 1998, Ford 1996b p. 49). More recently, North American universities have started to offer such programs (e.g. McMaster University 1998, Naveda 1997). This paper discusses the program recently approved at the University of Ottawa (University of Ottawa 1997) which will be the first bilingual program in Canada, and may be the first B.A.Sc. program overall in Canada.

The high level of demand for the graduates of such programs is widely known. However, the need for programs specifically called 'Software Engineering', in contrast to 'Computer Science' or 'Computer Engineering' has only been recognized recently. Key reasons for creating degree programs in software engineering as *Engineering* programs are: 1) Work governed by Engineering Acts (affecting public security, safety etc.) is increasingly dominated by software issues; 2) Existing engineering programs do not have enough time to cover the material that a software professional needs to know; and 3) It is becoming widely recognized that many engineering principles should be applied to software development processes.

Section 2 of this paper discusses the development and content of the University of Ottawa's program; section 3 discusses issues that required considerable debate.

2. The University of Ottawa Software Engineering Program

For several years, the University of Ottawa considered establishing a software engineering *stream* or *option*, that would have been taken as part of either the Computer Science or the Computer Engineering program (Lethbridge et al 1997). Events overtook this initiative, however: In 1997 the Electrical and Computer Engineering Department and the Department of Computer Science merged together to form the School of Information Technology and Engineering (SITE). One of the mandates of SITE has been to establish a software engineering program to complement the existing programs.

Deciding what material should be taught in the program involved considerable research and debate. Input was sought from members of SITE's Industrial Advisory Board which is composed of executives of a variety of Canadian high-technology and information technology (IT) companies. Several academics with senior software engineering posts were also consulted, and the committee examined the content of other software engineering programs or proposals, as well as a model published by the Software Engineering Institute (Ford 1996a). Finally, input about educational needs was sought from 187 industrial software practitioners and managers by way of a survey (Lethbridge 1998a, 1998b).

Most of our sources of input provided very similar recommendations – exceptions are discussed in section 3. Two important decisions that arose during the committee's deliberations were that the program should emphasize telecommunications software and entrepreneurship. These emphases will serve to distinguish our program from those at other universities.

The program, an outline of which is shown in Table 1, was approved by University Senate in Fall of 1997. Fifty students will be admitted into the software engineering program at the first year level in Fall 1998. At the same time, forty students will be selected from computer science or computer engineering to transfer into software engineering at the second year level. The first students should thus graduate in 2001, at which time accreditation of the program will be sought from the Canadian Engineering Accreditation Board.

3. Issues Discussed During the Program's Development

This section presents three issues about which much of the committee's discussions revolved: Differentiating the programs in SITE, deciding on the specific topics for the program, and achieving accreditation.

Differentiating Software Engineering from Computer Science and Computer Engineering

A frequently asked question is, "What is the difference between software engineering and computer science?" At the recent IEEE-sponsored 11th Conference on Software Engineering Education and Training, participants argued for four contrasting points of view: a) Software engineering is a clear subset of computer science; b) Software engineering and computer science are effectively synonymous; c) Computer science is a subset of software engineering; and d) the two disciplines are distinct – with software engineering being a branch of engineering, unlike computer science.

Parnas (1998) argues for the latter: He points out that engineers need a different kind of education from scientists; one that focuses on applying a core body of knowledge to produce reliable products, as opposed to learning how to extend our knowledge of what is true.

Those who argue in favour of point of view a), b) or c), say that almost all graduates of computer science programs actually perform an engineering-like activity after they enter the workforce. In the UK, in fact, the engineering society now considers computer science programs on an equal footing with engineering programs when it comes to issuing Chartered Engineer licenses (British Computer Society, 1998).

At the university of Ottawa, we have traditionally taught core software engineering courses as part of the computer science program, since there was nowhere else to teach them. With the introduction of software engineering we are attempting to make a clear distinction between the disciplines.

Important differences are: a) Software engineers will take a common body of courses with other engineers (italics in figure 1). b) Software engineers will take a significant number of required courses (bold in figure 1) focusing on methods for producing large-scale high-quality software. Computer science students and others will only be able to take at most two of these. c) Computer science students will be able to take more theory courses and courses in specialized areas such as graphics and artificial intelligence.

We firmly believe that the market will welcome this differentiation in the skills and knowledge of the two types of graduate. The expectation is that while graduates from both programs will perform many similar tasks (e.g. software design and programming) computer science graduates will tend to work more with more specialized computing technology such as graphics, simulation, compiler design, artificial intelligence etc. Software engineering graduates, on the other hand, will tend to work more on tasks such as high-level system requirements and architecture as well as the development of software engineering processes to ensure safety, reliability, quality and productivity.

	First year	Second year	Third year	Fourth year	Any year
Mathematics	*Calculus 1 & 2* *Linear algebra*	*Probability & statistics* Logic and discrete math 1 & 2			
Basic Science	*Chemistry* *Engineering mechanics* *Physics*				Three other courses
Software Engineering Topics (Engineering Science and Engineering Design)	*Engineering computing* Software design 1	Data structures File management **Software design 2 and 3 – includes software architecture**	Database management systems Algorithm analysis and design Operating systems **Large system development** **Software analysis & design** **User interface analysis and design** **Telecom software principles**	**Security in Computing** **Project management** **Software quality engineering** **Software evolution and reengineering** **Project course**	Three electives
Other Engineering	Introduction to electrical and computer engineering	Digital computer organization	Microprocessor systems Telecom systems & services	Real-time systems Computer communications	
Complementary studies	*Technical writing*	Business management *Engineering Economics*	SE Professional practice Small business management	*Technology & society*	One other course

Table 1: An outline of the University of Ottawa Software Engineering Program. For details, see University of Ottawa (1997). Courses in italics are taken by all engineering students. Courses in bold represent the core– courses that have been created specially for the new program, or adapted from what were previously computer science electives. Course names have been abbreviated.

It was also considered important to distinguish software engineering from computer engineering. We had little difficulty doing this: Computer engineering students at the University of

Ottawa take significantly more electrical and computer engineering courses, and correspondingly fewer software courses than will software engineering students. The expectation is that computer engineering graduates will work more with hardware and with the interface between software and hardware, than will software engineering students.

A final criterion for differentiating the programs was to ensure that, in all cases, students who attempt two degrees require at least one more year of study to obtain their second degree.

Deciding on Topics to be Taught in the Program

After we completed our research, it was relatively easy to come to an agreement about most of the topics in the software engineering program as shown in Figure 1. For some material, students will take courses that remain titled 'Computer Science', and have always been part of computer science programs. This material includes:

- Data structures,
- File systems and database management, and
- Operating systems.

In other cases we will use courses, without modification, that are classified as 'Computer Engineering' or 'Electrical Engineering', including

- Introduction to electrical and computer engineering,
- Digital computer organization (including assembly language programming),
- Microprocessor systems, and
- Real-time systems.

A third type of material about which there was little debate consists of material drawn from courses that were previously offered on an elective basis as computer science courses. This material has now been packaged and expanded into new, compulsory software engineering courses, but remains also available as electives to computer scientists:

- User interface design,
- Analysis and design techniques,
- Software quality engineering, and
- Software evolution and reengineering.

Yet another type of material had never been previously taught at the level of depth in which we were interested. We thus introduced entirely new courses covering:

- Software design principle and architecture (two courses),
- Large scale software development,
- Case tools (integrated into several courses), and
- Computer security.

There remained, however, several types of material about which considerable debate was required, including:

• How should algorithms and performance be covered? The options were to use a classic computer science course covering algorithm analysis and design, or to introduce a new software engineering course that would focus more on performance analysis. The eventual decision was to adopt the computer science course and integrate performance analysis in other courses.

• Should there be a course in programming languages? Such a course exists in most computer science courses and gives students exposure to a variety of languages. We decided, however, to have no separate course and instead to introduce students to a variety of languages, as well as how to choose a language, in courses throughout the program.

• Should there be a focus on telecommunications, or should this be one of several possible specializations? Our original idea was that students would choose from several different three-course 'streams' focusing on particular kinds of software: 1) Scientific and engineering software; 2) Telecommunications software; and 3) Business software. There was a request from the faculty of engineering, however, to boost the core engineering content of the program. Since also the strongest demand from local industry is for software engineers with telecommunications knowledge, we decided to require a three-course specialization in telecommunications in our program, and omit the other proposed streams. The telecom stream consists of two courses in electrical engineering and one course in telecommunications software engineering; students can take a second telecommunications software course as one of their technical electives. We expect that this specialization will differentiate our program from other software engineering programs.

Accreditation: Have We Covered Everything we Need to, And Should Criteria be Changed?

The criteria governing accreditation of an engineering program are found in a Canadian Council of Professional Engineers document (1996). We designed our program carefully, ensuring that it meets academic requirements for such elements as mathematics, basic science, engineering science, engineering design, complementary studies and exposure to other branches of engineering.

There was some discussion in the committee about two issues that arose from accreditation requirements: The first of these was which introductory programming course should software engineering students take? One option was the introductory programming course taught in first term of first year to all engineering students – this course emphasizes applications and programming languages of interest to other branches of engineering. The second option was to have software engineering students take the corresponding introductory programming course taught to computer science students. We eventually decided on the first option, in the interest of a common engineering first term. This necessitated the introduction of a new second-term course in computing so that at

the end of first year the students would have the necessary prerequisite material they need for second-year computing courses.

The second issue, again resolved in favour of uniformity of engineering programs, was whether to require software engineers to take the same basic science (particularly chemistry and mechanics) as other engineers. There was some debate about whether to require material that research shows will only be of direct use to a small subset of software engineers. We intend to discuss this issue over time with the CCPE, since we would like to see more flexibility in this requirement.

Following university approval of the program, we informally asked the Canadian Engineering Accreditation Board to look at it: Their most significant issue was the fact that we deliberately left out numerical methods and differential equations – topics which are currently required of all other engineering programs.

The rationale for the omission is as follows: Our research indicates that the mathematics required of all engineers, in particular advanced calculus and differential equations, is not of use to the vast majority of software engineers[1]: Instead, software engineers need significant coverage of discrete mathematics. This position is affirmed by the Joint Steering Committee of the IEEE Computer Society and ACM for the Establishment of Software Engineering as a Profession (Engel 1998), who have proposed accreditation criteria for software engineering that emphasize discrete mathematics plus probability and statistics. A similar emphasis is found in the latest (April 1998) software engineering accreditation criteria proposals from ABET – the United States engineering accreditation agency (ABET 1998).

We will adjust the University of Ottawa program if necessary to achieve accreditation. However we believe that since the criteria were designed before software engineering programs were envisioned, it might be more appropriate to consider minor criteria changes that would recognize the establishment of software engineering as a distinct subdiscipline.

In addition to seeking accreditation by CEAB, we will also seek accreditation by CIPS, the body that currently accredits computer science programs. The rationale for this is that in addition to being engineers, our graduates will also need to be computing professionals.

4. Conclusions

A report of Professional Engineers Ontario (Frise 1997) states: “If PEO doesn’t bring these people [software developers] in now, it will be done by the next royal commission after an

[1] For example, Lethbridge (1998a and 1998b) showed that software practitioners forget most of their calculus, and have made little use of it in their work.

accident." By starting to issue engineering degrees in the software field, the University of Ottawa is helping to ensure that it serves this societal need.

References

ABET, "Software Engineering Program Criteria", Accreditation Board for Engineering and Technology, www.abet.org, 1998.

British Computer Society, Web Page, April 1998, http://www.bcs.org.uk/aboutbcs/overview.htm

Canadian Council of Professional Engineers, "Canadian Engineering Accreditation Board: 1996 Accreditation Criteria and Procedures".

Cowling, A.J., A Multi-Dimensional Model of the Software Engineering Curriculum", Proc. 11th Conference on Software Engineering Education and Training, Atlanta, 1998, pp. 44-55.

Engel, G. L., & LeBlanc, R.J., "Draft Accreditation Criteria for Software Engineering", IEEE Computer, 31, 4, April 1998, pp. 73-77.

Ford, G., "The SEI Undergraduate Curriculum in Software Engineering", Software Engineering Institute, 1996a.

Ford, G. & Gibbs, N., "A Mature Profession of Software Engineering", Software Engineering Institute Technical Report CMU/SEI-96-TR-04, 1996b.

Frise, P. et al., "Final Report: PEO Task Group on Emerging Engineering and Multidisciplinary Groups", Professional Engineers Ontario, 1997.

Lethbridge, T., "A Survey of the Relevance of Computer Science and Software Engineering Education", Proc. 11th Conference on Software Engineering Education and Training, Atlanta, 1998a, pp. 56-66.

Lethbridge, T., "The Relevance of Software Education: A Survey and Some Recommendations", *Annals of Software Engineering*,, 1998b to appear.

Lethbridge, T., Ionescu, D., Mili, A. & Gibbons, D., "An Undergraduate Option in Software Engineering: Analysis and Rationale", Proc. 10th Conference on Software Engineering Education and Training, Virginia Beach, 1997, pp. 120-129.

University of Ottawa, Faculty of Engineering Calendar, April 1998 (www.uottawa.ca). An on-line version of a slightly earlier version containing rationale is in "Proposal for a Software Engineering Program", School of Information Technology and Engineering, http://www.site.uottawa.ca/~tcl/sepc/CompleteSEProgram.pdf, 1997.

McMaster University, "Software Engineering at McMaster University: Report of the Ad Hoc Curriculum Committee", March 1998.

Naveda, J.F. "Crafting a Baccalaureate Program in Software Engineering", Proc. 10th Conference on Software Engineering Education and Training, Virginia Beach, 1997, pp. 74-79.

Parnas, D., "Software Engineering Programmes are not Computer Science Programs", draft paper, Department of Computing and Software, McMaster University, 1998.

University of Melbourne, School of Electrical Engineering and Computer Science, http://www.unimelb.edu.au/HB/areas/NELEENG.html, 1998.