A Joint CS/E&CE Undergraduate Option in Software Engineering

J.M. Atlee[*], P.P. Dasiewicz[†], R. Kazman[*], R.E. Seviora[†] and A. Singh[†]

[*]Department of Computer Science
[†]Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, Canada N2L 3G1

Abstract

This paper describes a software engineering option which has been developed and is being taught jointly by two departments in two faculties: Computer Science (Faculty of Mathematics) and Electrical & Computer Engineering (Faculty of Engineering). The attempt to create a joint option has resulted in certain strengths and weaknesses. The strengths derive from the different approaches to software engineering in the two departments. The weaknesses derive from the constraints of having to deal with two sets of departmental, faculty, and accreditation board constraints, which leaves the option less flexibility.

We describe the option, emphasizing three components: the course selection and, in particular the three new courses which were created specifically for the option; the CASE tools which accompany each of the new courses; and the project which spans all three of the new courses. The project is described in detail, emphasizing its bifurcated nature, with a real-time embedded system aspect for the computer engineers and an information system aspect for the computer scientists.

1. Introduction

1.1 Institutional Setting

The University of Waterloo is a medium size university with 16,000 full-time undergraduates, 2,000 graduate students, 800 faculty members and 2,200 staff. The university receives some 75% of its income from the provincial government; tuition fees provide most of the remainder. Many undergraduate programs operate on a co-op basis; students alternate between four month teaching and four month work terms.

Organizationally, the University is composed of seven Faculties. The Computer Science (CS) department is a part of the Faculty of Mathematics; the Electrical and Computer Engineering (E&CE) department belongs to the Faculty of Engineering. Even though the two departments are in different Faculties, they are housed in one building, the W.G.Davis Computer Center.

The CS department has over 40 faculty members and graduates 200+ Bachelors each year. It offers a 4-year Honours CS program and a number of options and joint programs (e.g., Honours Computer Science/Information Systems option, Joint Honours Actuarial Science and Computer Science, etc.). Its undergraduate course offering contains 30 course titles, not including service courses. The E&CE department has over 40 faculty members. It offers two undergraduate degree programs, Electrical Engineering and Computer Engineering. Its undergraduate course list has about 50 course titles. The department graduates close to 200 Bachelors annually, with roughly 2:1 ratio between

Electrical Engineers and Computer Engineers.

The CS programs are accredited by the Computer Science Accreditation Council (CSAC, an autonomous body of the Canadian Information Processing Society); E&CE programs by the Canadian Engineering Accreditation Board (CEAB, a board of the Canadian Council of Professional Engineers).

Although one department has Engineering and the other Science in its name, both have faculty members with scientific as well as engineering orientation. This is a source of strength: the history of educational institutions in these fields shows that this composition brings in complementary perspectives and creative tensions and reduces the likelihood of drifts into excessive pragmatism or irrelevance.

1.2 Rationale for the Option

In order to fullfil their roles and keep public support, educational institutions must respond to societal needs. In the case of software engineering, there has been growing and increasingly visible demand from industry and governments for graduates with stronger software engineering qualifications. Furthermore, the jobs available to graduates and co-op students in our two departments have led to similar but perhaps less strident demands for stronger software engineering exposure in the undergraduate curricula.

These pressures have been present for some time. Until recently, however, there was concern over whether a sufficient body of knowledge (principles, models, methods and techniques) existed to form a foundation for specialized undergraduate education in software engineering. Over the last several years, it has become evident that there is indeed an emerging body of such knowledge, even though it is still rapidly evolving and lacking in synthetic perspectives.

1.3 Departmental Response

The mere presence of such pressures and knowledge does not result in action. Other factors must be present. At Waterloo, these factors included the positive attitude of the two Departments and Faculties towards the idea of an SE option. What was crucial was the presence of sufficient number of faculty members with interests in software engineering and its subdisciplines. There was, furthermore, an accumulated body of experience with in-depth instruction in software engineering topics at the graduate and to some extent undergraduate levels. The idea of establishing an undergraduate option in Software Engineering was thus an evolutionary step, as opposed to a leap into a largely unexplored territory.

These positive factors were, to some extent, offset by the budgetary constraints under which the University operates. The teaching load at Waterloo is already higher than the North American average and the institutional mood is to cut resources rather than add them.

The initial discussions about the establishment of the option took place in early 1993 under the umbrella of the Waterloo Initiative in Software Technology [2]. Toward the end of 1993, a joint CS/E&CE option committee was set up. The committee produced its recommendations in early 1995. The SE option has since been approved by the two departments. It is presently in the approval process at the faculty level, with one approval already in place. The pilot version of the first new core course (a third year course on Software Requirement Analysis and Specification) will be given in Fall 1995. The first option students are expected to graduate in 1997. Their degrees will carry the designation “Option in Software Engineering”.

In the development of the option, the committee benefited from the wealth of information recorded in past proceedings of this conference. Particularly influential on committee’s thinking were the SEI report on Undergraduate Software Engineering Education [1] and the reports and accreditation guidelines of professional societies [3][4].

2. Option Goals and Overall Constraints

In the design of the option, the two principal goals were:

G1. to provide an option-level exposure to software engineering principles, models, methods, techniques and application classes;

G2. to ensure a satisfactory level of competence in the application of the principles, models methods and techniques to software development and maintenance.

The option curriculum had to satisfy a number of constraints:

C1. *the minimum increment constraint* - the number of new course titles had to be kept to a minimum. This constraint was primarily due to budgetary considerations which limited the number of additional teaching tasks the option would require.

C2. *the total course count constraint* (for base programs) - a base program student should be able to graduate with the option designation without having to exceed the number of courses needed for graduation in the base program (40 for each base program). This constraint was particularly tight for the Computer Engineering program; the Honors Computer Science program, as it stood, had more free electives available.

C3. *the accreditation constraint* - the base programs with the option had to be accreditable by both the Canadian Engineering Accreditation Board and the Computer Science Accreditation Council of Canada.

C4. *the acceptability constraint* - the option had to be acceptable to both Departments and both Faculties.

C5. *the A-accessibility constraint* (for non-base programs) - the option had to be accessible to A-average students in other engineering programs, such as Electrical Engineering. For those students, the C2 constraint did not apply. However, they had to be able to complete the option by taking at most one additional course per term above the normal load. (In Engineering, A-average students are allowed to take heavier course load than the standard five courses per term.)

The above constraints presented a severe limitation on option design. What made the option in the end possible was the fact that the base programs already covered a fair fraction of what we saw as the necessary minimum for the option designation.

To characterize the degree of existing coverage, we will use the framework presented in the SEI undergraduate software engineering curriculum [1]. The framework has four segments— Mathematics/Natural Sciences (at least 9 courses); Software Engineering Science/Software Engineering Design (14); Humanities and Social Studies (10), and Electives. The SE Science/SE Design segment is further divided into four blocks - Software Analysis (3 courses); Software Architecture (4); Computer Systems (3); and Software Processes (4).

In the key SE Sciences/SE Design segment, the two base programs already offered fairly high coverage for the first three blocks (generally in the 50-75% range, although the precise coverage differed somewhat between the two). The coverage of the Software Processes block was significantly lower, perhaps in the 25-33% range.

Although we were designing an option and not a full degree program, it was evident that the option design had to substantially increase the coverage of the Software Processes block.

3. Curricular Philosophy and Program Constraints

The option design had to accommodate the philosophies of education of the two departments and their specific curricular constraints. These are summarized below.

3.1 Computer Science Department

The Computer Science department has two historical foci which have together shaped the current content of its curriculum. The first is that the Computer Science department is part of the Faculty of Mathematics, and advertises its programs as being for students who wish a heavy concentration in some area of the mathematical sciences. The second is the department's long-standing involvement with educational and commercial software. For example, many educational institutions in the 1970s made use of the department's Watfor and Watfiv Fortran compilers.

The Mathematics faculty regulations require all CS students to be Mathematics majors, which means that they must take 1.5 years of Mathematics courses in a 4 year degree. Computer Science's departmental regulations add course requirements to the faculty pool. The principal effect of these regulations is that students are constrained to select 60% of their third and fourth year credits from CS major courses.

Finally, it has been our intention that this program be accreditable by the Computer Science Accreditation Council. The minimum accreditable program, according to the CSAC [3], requires:

- 1.5 years of study in computer science/computer engineering;
- 0.5 years of study in mathematics/statistics;
- 1.0 years of study in subjects other than computing and mathematics/statistics.

The combination of Mathematics faculty and Computer Science departmental regulations imposes a slightly different set of constraints:

- 1.3 years of study in mathematics/statistics;
- 1.3 years of study in computer science;
- 1.0 years of study in subjects other than computing and mathematics/statistics.

3.2 Electrical and Computer Engineering Department

The educational philosophy of the E&CE department is largely determined by its mission to prepare students for professional practice of engineering. Historically, this resulted in certain curricular content distributions and certain style of delivery of course content.

In content distribution, the minima to be satisfied under CEAB accreditation criteria were [4]:

- 0.5 years of Mathematics;
- 0.5 years of Basic (Natural) Sciences;
- 2.0 years of Engineering Sciences and Engineering Design (with at least 0.5 years of Engineering Sciences and 0.5 years of Engineering Design);
- 0.5 years of Complementary Studies (humanities, social sciences, arts, management, engineering economics, communication).

The criteria also require appropriate laboratory experience. It is important to note that the criteria do not define what specifically has to be covered in the individual segments. The current content of a segment is the result of historical evolution of the engineering discipline. In harmony with many other E&CE departments, the E&CE department philosophy is to include in the Engineering Science/Engineering Design segment:

a. two or more implementation technologies; and

b. two or more common application areas.

A major reason for [a] is the likelihood of technological change during the student's professional career. A student who has been exposed to more than one implementation technology will find it easier to adapt to another when it comes. The rationale for [b] is the broadening of the student's perspective on the engineering process. The students see how the requirement specification for a product are derived and tie in with the needs of the application.

For the base Computer Engineering program, block [a] includes digital hardware (4 compulsory courses) and software (5), with some coverage of the light-current analog technology (3). Block [b] contains Communications (1) and Automatic Control (1).

In terms of packaging, a technology specific course in the E&CE department tends to include both analytical and synthetic components (and also the mathematical foundations if not covered elsewhere). This results in better comprehension and better linkages, but carries the potential for the loss of general perspectives due to the specifics. Because of the 'preparation for practice' philosophy, courses expose students to relevant industrial standards. Where possible, commercial tools are used in projects and laboratories.

4. Option Curriculum

The combination of these sets of requirements left us with little flexibility. What has resulted is an option that, while not ideal, is palatable to both faculties and both departments—a significant accomplishment in our estimation.

In particular, we had wanted to design four new software engineering courses, covering all aspects of the software development life cycle. These courses were:

- Software Requirements Specification and Analysis
- Software Design and Architecture
- Software Quality Assurance, Testing, and Process
- Software Maintenance and Reverse Engineering.

We eventually reduced these four courses to just three. We did this because it was easier to fit three new courses into our curricula given the constraints listed in section 2. During option development, constraint C1 (*the minimum increment constraint*) was quantified as no more than four new courses (four courses is 10% of the total course count). The strongest reasons, however, to reduce our new course offerings to three came from two sources: C3 (*the accreditation constraint*), and because we felt that there were an inadequate amounts of pedagogical materials to support separate courses in Quality Assurance, Testing and Process (as one course) and Maintenance and Reverse Engineering (as the other). Thus, only three of these four courses were implemented as part of the undergraduate SE option. As a result, these two previously proposed courses were combined into a single course, Software Testing, Quality Assurance and Maintenance.

4.1 Curriculum Design

The central software engineering skills which we do want to teach in the combined CS/E&CE option are as follows:

- Software system design and the design of changes to such systems;
- Requirements analysis, specification, design, construction, verification, testing, maintenance and modification of programs, program components, and software systems;
- Algorithm design, complexity analysis, safety analysis, and software verification;

- Database design, database administration and maintenance;
- Design and construction of different types of computer systems such as human-computer interaction systems, real-time systems, embedded systems;
- Management of projects that accomplish the above tasks, including estimating and controlling their cost and duration, organizing teams, and monitoring quality;
- Selection and use of software tools and components;
- Appreciation of commercial, financial, legal, and ethical issues arising in software engineering projects.

Furthermore, two of the goals of the software engineering option are to foster non-technical skills which are crucial to successful software development—things like critical thinking, technical writing, presentation skills, etc. and to broaden a student's background, through recommending (and, in fact, demanding) as wide a range of non-technical courses as possible.

4.2 Option Outline

4.2.1 Technical Courses

Given this basis of core software engineering courses, the software engineering option curriculum consists of:

- 9 core technical courses
- 3 courses focussing on the software development life-cycle
- 2 courses from a short list of software intensive courses (compilers, operating systems, real-time systems, networks, etc.)

The core technical courses differ in the CS and E&CE versions of the option, but remain consistent with the cores of the respective departments, as shown in Table 1. Table 2 shows the mandatory

Table 1: Core Technical Courses

CS Dept.	E&CE Dept.
CS 241: Foundations of Sequential Programs	E&CE 203: Discrete Mathematics
CS 246: Software Abstraction and Specification	E&CE 251: Programming Languages and Translators
CS 340: Data Structures and Algorithms	E&CE 250: Algorithms and Data Structures
CS 342: Concurrent Programming	E&CE 324: Microprocessor Systems and Interfacing
CS 354: Software Systems	E&CE 354: Real-Time Operating Systems
CS 351: Digital Design and Architecture	E&CE 223: Digital Circuits and Systems
CS 360: Introduction to Theory of Computing	E&CE 380: Systems and Control

Table 1: Core Technical Courses

CS Dept.	E&CE Dept.
CS 370: Numerical Computations	E&CE 304: Numerical Methods
CS 448: Introduction to Database Management	E&CE 456: Database Systems

Table 2: Life-Cycle Courses

CS 445/E&CE 451: Software Requirements Modelling and Analysis
CS 446/E&CE 452: Software Design and Architecture
CS 447/E&CE 453: Software Testing, Quality Assurance, and Maintenance

Table 3: Software Intensive Courses

CS Dept.	E&CE Dept.
CS 486: Introduction to Artificial Intelligence	E&CE 457: Applied Artificial Intelligence
CS 488: Introduction to Computer Graphics	
CS 454: Distributed Systems	E&CE 428: Computer Communication Networks
CS 452: Real-Time Programming	E&CE 485: Computer Control Applications
CS: 457: Queueing Models: Analysis, Simulation and Applications	E&CE 429: Computer Structures
CS 444: Compiler Construction	
CS 466: Algorithm Design and Analysis	

software life-cycle courses. These courses have been developed jointly by the two departments. Table 3 shows various elective courses from which a minimum of two courses from either department must be taken. In Table 3, the courses in the same row form anti-requisites for each other as they deal with similar or significantly overlapping subject material.

4.2.2 Nontechnical Electives

In addition to this core, we are also requiring that students take one course in each of the following areas: Communications (both written and oral), Societal Issues (legal and ethical implications of computing), Business Issues, and Reasoning Methodologies (critical thinking and formal reasoning).

4.3 New SE Courses

One of the initial issues addressed while developing the curriculum was to determine the number of software engineering courses that would cover all the salient topics in the field at an adequate level for undergraduate education. As already mentioned, major guiding parameters on this issue were: the breadth and depth at which, we thought, the topics should be covered, the undergraduate-level pedagogical material that was available for the proposed courses, and the number of new courses that can be added to the existing schedules of the undergraduate curricula of the CS and the E&CE departments.

The underlying theme of the three new courses which form the core of the SE option was to develop a curriculum that not only covers all the phases of the software development cycle (requirements modelling, specification, design, testing, maintenance), but also puts an equal emphasis on fundamental principles of software engineering. The courses should make it possible to illustrate the application of principles and rigor in the practice of software engineering. At the same time, the courses also include discussions of commonly employed informal techniques and CASE tools.

Ordering of topics in the curriculum posed another challenge. Most of the topics covered in the three courses are so interrelated that it is impossible to find a sequence that would eliminate references to topics that would be covered later in the curriculum. Therefore, it was decided that the first course should also include an overview of the overall objectives, quality issues and processes in the software development. Also, the latter two courses should devote part of their time and effort on relating the material covered in the course with other phases of the software development cycle.

The following subsections present brief outlines of each of the three new courses. Each course also includes a substantial project component—a single project, broken into three phases, which connects the three courses. The project associated with these courses is described in section Section 4.4 .

It should be noted that although in this section we discuss only the three new courses dedicated to central issues in software engineering, the complete undergraduate curricula of the two departments contain courses covering topics such as data structures, algorithm analysis, programming paradigms, various computer systems (such as operating systems, user interfaces, file and database systems, real-time systems, computer networks, parallel and distributed systems, computer architecture), etc.

4.3.1 Software Requirements Modelling and Analysis

The course is intended to introduce students to the requirements definition part of software development. It discusses models, notations, and methodologies for software requirements identification, representation, validation and analysis. In addition, for the reasons outlined previously, the course also provides an overview of the quality issues, software development process and life-cycle models. The course covers following topics:

- Overview of software development process and life-cycle models
- Overview of the processes, products, and tools of requirements phase of software development
- Requirement elicitation
-.Informal and formal notations for behavioral requirements
- Specification of non-behavioral requirements
- Requirement validation
- Miscellaneous topics, such as reusability of specifications, relationship between requirements and design specifications, etc.
- Case studies

4.3.2 Software Design and Architecture

The course is intended to concentrate on software design activity. In addition, it executes a second pass through the various phases of the software development cycle. The topics covered are:

- Relationship of software design to software quality and other phases of the life cycle
- Models of software design process
- Informal and formal representation of software design and architecture
- Canonical design plans; case studies
- Design strategies and methods
- Design assessment
- Design quality assurance and verification

4.3.3 Software Testing, Quality Assurance, and Maintenance

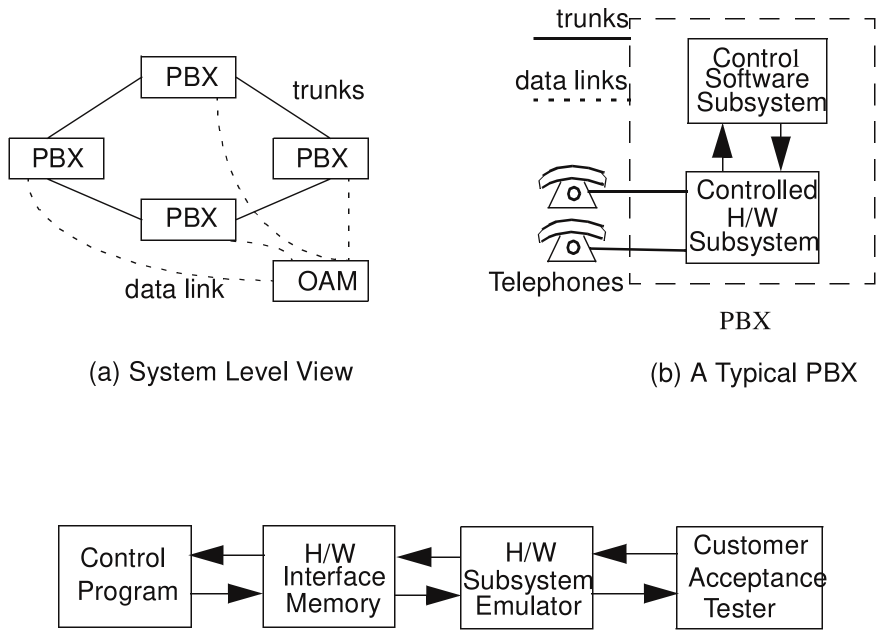
As the title implies, this course deals with testing, maintenance, and quality assurance related issues in software development. It also executes a third pass through the software development cycle. Issues related to project and cost management, reverse engineering and software reuse are covered in this course. The topics covered under this course are:

- Foundations of software testing
- Methodologies for systematic testing
- Testing of parallel, distributed, and real-time embedded systems
- Evaluation of Nonfunctional requirements
- Formal and informal methods for software verification
- Software debugging
- Quality assurance
- Software Maintenance
- Project Management
- Software reuse, reverse engineering

4.4 Project

The project, for the life cycle course sequence, is based on a project which has been successfully used in a 4th year software engineering course in the Department of Electrical and Computer Engineering. The project consists of the specification, design and implementation of the call processing and administrative software for a system of Private Branch Exchanges (PBXs) interconnected by trunks (see Figure 1(a)). Each PBX controls its own local calls, with all off-premise calls using the trunk lines. All PBXs are connected (via a serial data link) to the Operations, Administration and Maintenance (OAM) unit. The OAM unit provides a graphical user interface that allows operators to monitor and control functions of PBX administration (e.g., maintaining the PBX database, billing, etc.) and maintenance (e.g., requesting the execution of specific maintenance tests).

Figure 1(b) depicts a block diagram of a typical PBX consisting of the control software subsystem and the controlled hardware subsystem. The controlled hardware subsystem consists of a voice/signaling unit composed of: a Voice Switching Network (a time-space-time switch), two line interface shelves (30 line interfaces per shelf), a Service shelf (with digit receivers, call-in-progress tone generators), a Ringing Generator and a Tester. Each line interface includes a A/D and D/A converter that is connected to time-multiplexed links carrying 32 channels (32 8-bit samples) in a 125 microsecond frame. In addition, each line interface is configurable to controlled test and ring relays, on/off-hook loop detector, and voice channel selection control. Since the real PBX hardware is not available, a Hardware Subsystem Emulator is provided. The Hardware Emulator software emulates all the controlled hardware; it provides status and receives control from the control software through a shared memory interface.



(a) System Level View

(b) A Typical PBX

(c) Project Subsystems

Figure 1: Project Overview

The PBX software is divided into two major software subsystems, based on the interests and backgrounds of the two streams of students. The control software subsystem (Figure 1(c)) is responsible for call processing and automated maintenance. It monitors and controls the status of the voice/signaling hardware via the shared memory interface. For example, the digits being received by touchtone receivers are accessible by reading the appropriate memory locations. To establish a connection between telephones, an appropriate bit pattern would be written into a location in the switching network control memory. Periodically, the Automatic Fault Detection part of the maintenance software will run fault detection tests on individual hardware units, accessible via the shared memory. Because development of the control program emphasizes techniques used to develop real-time software for embedded systems, the E&CE students will work on this subsystem.

The OAM software (Figure 1(a)) is responsible for maintaining the PBX database used in call processing, executing manual maintenance tests, and billing. The software displays a graphical representation of the PBX's environment (e.g., type and number of line interface cards, trunks, lines' sets of allowable services, etc) that the operator can manipulate to request changes to the PBX database and the customer-information database. A similar graphical display is used to request maintenance tests, outside of the normal automated testing process, and to view their results. The OAM software also includes an HCI interface between the operator and the billing software, to maintain billing rates, including discounts for times-of-day and days-of-week, and to query a customer's bill. Because development of the OAM software emphasizes techniques used to develop information systems, the CS students will work on this subsystem.

Appropriate CASE tools are used to support the unique development requirements of both software systems. The real-time software system is modeled as a set of communicating finite state machines, while object oriented analysis and modelling techniques are used for the information subsystem.

The project consists of three primary phases. During the first course, the software requirement and specifications (SRS) are generated for each software subsystem with the primary deliverable being the SRS document. During the second course, a design document (based on the SRS) is produced and each subsystem is coded and validated using an automated customer acceptance tester. At this stage, the students would use whatever techniques that they are familiar with to perform unit and integration level testing of their implementation. Finally, in the third course, additional PBX and OAM functionality is introduced (e.g., new telephony features, such as Call Forwarding). This reflects the maintenance activities in the software life-cycle. At this stage, the formal testing methods (discussed in the course) are applied to the modified subsystems to ensure that the enhancements were implemented correctly.

5. Pragmatics of Option Delivery

Many delivery issues, such as the allocation of teaching resources and CASE tool procurement, are made more complex by the fact that the option is being developed jointly by two departments in different faculties. Decisions must conform to the philosophies, policies, and available resources of both departments.

5.1 Teaching Resources

Most of the option's technical courses carry either the Computer Science or Electrical and Computer Engineering course designation. The resources needed to deliver each of these courses are clearly the responsibility of the respective department offering the course. However, the option introduces new courses, (three of which were mentioned above; the fourth being a fourth year human-computer interaction course) that will be offered jointly by the CS and E&CE Departments to students from both departments.

Our current plan is that the two departments will contribute equally to the delivery of the four new courses. Each department will be responsible for developing and initially offering two of the courses, and for designating 'local' course coordinators for the other two courses. Teaching responsibilities for subsequent offerings will be divided evenly between the departments.

Given that Computer Science currently admits three times as many undergraduates as Computer Engineering, we anticipate that far more Computer Science students will register for the option than Computer Engineering students. If our estimation proves to be correct, the departments may decide to re-allocate teaching resources for option courses, based on the distribution of students in the option.

5.2 Teaching Assistants vs. Lab Technologists

The two departments differ in how graduate students and staff are used to support teaching efforts. The Computer Science Department grants teaching assistantships of 10 hours per week to virtually all incoming graduate students. Thus, there are always a large number of available teaching assistants to help mark course assignments, projects and exams. In the Electrical and Computer Engineering Department, a certain number of teaching assistant positions are allocated to a course. Because students volunteer to be teaching assistants, and because such assistantships are awarded on the basis of merit, there are fewer teaching assistants for E&CE courses.

Instead, the E&CE Department uses teaching resources to employ several full-time laboratory technologists. Laboratory technologists are responsible for maintaining and answering questions about hardware used in courses that have laboratory assignments. In addition, technologists are responsible for tutorial and consultant-support for software tools.

Because the CS Department does not employ the equivalent of laboratory technologists, a technologist from the E&CE department will be assigned to the each of the software engineering core courses, regardless of the course's instructor. Teaching assistants for each course will come from both departments.

5.3 CASE Tool Procurement

The use of CASE tools has become an integral part of the software development process. Furthermore, we feel that it is important to expose our students to the capabilities and limitations of commercial-grade CASE tools. Because we want to emphasize the *capabilities* of tools, we spent months deliberating the merits of various specification and design tools. In the end, we recommended the procurement of both SDT (a toolkit for the Specification and Description Language (SDL))[1] and Software through Pictures/Object Modelling Technique (StP/OMT).[2] Based on the information available, we decided that these were the most appropriate CASE tools for developing the option's projects[3].

SDT supports the design, analysis, validation, and conformance testing of designs written in the Specification and Description Language (SDL), standardized by the International Telecommunications Union. The SDL notation was developed specifically for designing telecommunication and networking software, which makes SDT a particularly appropriate CASE tool for the embedded-software project.

StP/OMT is a general-purpose analysis, specification and design tool. It supports several editors for specifying different views of a software system: the system's data objects and relationships between data objects, the system's modes of operation and the events that cause the system to change modes of operation, the system's functional requirements, the architecture of the system's components, etc. The specification of the OAM software requires all of these different modelling capabilities. In addition, StP/OMT maintains a common data dictionary, to help ensure consistent use of terms in all of the project's specification and design diagrams.

The major drawback of these tools is their expense. Although the educational discounts are very generous, it will cost us about $50,000 to purchase a sufficient number of licenses and to upgrade the memory and disk capacities of the undergraduate machines. During a time of financial cutbacks, it is politically difficult to justify spending tens of thousands of dollars on a single program. Fortunately, the Mathematics Faculty has an industrial grant for enhancing information-systems education. Money from this grant is being used to procure the CASE tools and to upgrade the machines. The Engineering Faculty will reimburse its share to the grant using future funding sources.

Lastly, because of our endeavors to choose the most appropriate CASE tools and because of our difficulties in finding funding for the tools, purchase orders for the CASE tools were not written until four weeks before the start of classes. We do not know how much time the Faculties' system administrators will have to install and test the tools before students are granted access.

1. SDT is a trademark of TeleLOGIC Malmo AB
2. Software through Pictures is a registered trademark of Interactive Development Environments, Inc.
3. At present, we have only considered CASE tools for the requirements and design phases.

6. Concluding Observations

The paper presents a joint CS/E&CE undergraduate option in software engineering. The option will be offered starting in Fall 1995. The first class of students are expected to graduate in 1997, subject to the completion of the approval process. The option was designed primarily for Computer Science and Computer Engineering students, but it could also be taken by A-average students from other engineering programs. The option curriculum consists of twelve compulsory software engineering courses plus three electives to be taken from a list of software intensive courses. It also includes a large software engineering project, which extends over three software engineering life-cycle courses.

The option design had to satisfy fairly severe constraints. It had to accommodate the differences between the two home departments, their home faculties and accreditation requirements. The satisfactory completion of the option design and its subsequent approval by the primarily concerned parties (the home departments) represented, per se, a major achievement.

Several sources of concern remain. A major one is the total number of compulsory courses in the base and option. Option students have very few elective slots left (outside of the three 'software intensive' electives, which can be chosen from a fairly long list). This is not a desirable situation. At present, we do not see an easy solution.

Over a longer term period, another major concern is whether we will manage to continue with a joint option, or whether the subsequent evolution will lead to two disjoint options, either de facto or perhaps even de jure. This occasionally happens when there is no dominant 'owner' of an option. We are hopeful that this will not be the case with the option presented. There are definite benefits to be gained by proceeding jointly. Educational history in North America clearly shows the advantages of approximation over separation in rapidly evolving disciplines of this kind.

Acknowledgments

Curricular changes do not take place in vacuum. The authors would like to acknowledge the assistance and contributions of their many colleagues to the curriculum presented above, in particular those of P.A. Buhr, Chair of the CS Curriculum Committee, W.M. Loucks, Associate Chair for Computer Engineering in E&CE, and the two departmental chairs, F.W.Tompa (CS) and S.K.Chaudhuri (E&CE).

7. References

[1] G.Ford, 1990 SEI Report on Undergraduate Software Engineering Education, Software Engineering Institute, Carnegie Mellon University Tech. Report #CMU/SEI-90-TR-3.

[2] P. Koch, Waterloo Initiative in Software Technology (WIST), University of Waterloo, 1993.

[3] Computer Science Accreditation Council: Objectives, Procedures and Criteria, Canadian Information Processing Society, 1994.

[4] Canadian Engineering Accreditation Board, 1994 Annual Report, Canadian Council of Professional Engineers, Ottawa, ON, 1995.