IEEE Copyright Notice

Published in: ***Proceedings of the 9th Conference on Software Engineering Education (CSEE'96),*** April 1996

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

Cite as:

J. M. Atlee, P. P. Dasiewicz, R. Kazman, R. E. Seviora, and A. Singh. 1996. A Joint CS/E&CE Undergraduate Option in Software Engineering. In Proceedings of the 9th Conference on Software Engineering Education (CSEE '96). IEEE Computer Society, Washington, DC, USA, 16-28.

BibTex:

```
@inproceedings{Atlee:1996:JCU:525263.793934,
 author = {Atlee, J. M. and Dasiewicz, P. P. and Kazman, R. and Seviora, R. E. and Singh, A.},
 title = {A Joint CS/E\&CE Undergraduate Option in Software Engineering},
 booktitle = {Proceedings of the 9th Conference on Software Engineering Education},
 series = {CSEE '96},
 year = {1996},
 pages = {16--28}
}
```

DOI: http://dx.doi.org/10.1109/CSEE.1996.491359

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

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

*Department of Computer Science
†Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ont., Canada N2L 3G1

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 bases; students alternate between four month teaching and four month work terms.

Organizationally, the University is composed of seven Faculties. The Computer Science department is a part of the Faculty of Mathematics; the Electrical and Computer Engineering 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 EEs and CompEs.

The CS programs are accredited by the Computer Science Accreditation Council (an autonomous body of the Canadian Information Processing Society); E&CE programs by the Canadian Engineering Accreditation Board (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; 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 the two departments have led to a similar but perhaps less strident demands for stronger software engineering exposure in the curricula.

These pressures have been present for some time. Until recently, however, there was a concern whether a sufficient body of knowledge (principles, models, methods and techniques) existed to form a foundation for more intensive 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 rather than add.

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]. Towards 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 3B course on Software Requirement Analysis and Specification) will be given in Fall 95. 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 CS had more room.

C3. *the accreditation constraint* - the base programs with the option had to be accreditable by both CEAB and CSAC.

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-level students in other engineering programs, in particular to Electrical Engineering students. The total course count constraint did not apply; instead, the N+1 constraint did. A student from a non-base program should be able to complete the option if he/she took one additional course per term. (In Engineering, an A-level student can take, with permission of the Department, one course per term over and above the normal load).

During option development, constraint C1 was quantified as no more than four new courses (four courses is 10% of the total course count). Because of C3, only three of these were made compulsory for option students. The implications of the other constraints are discussed below.

---RES: should the above paragraph be moved to Sect 4 (Option Curriculum)?

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 (15); 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 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

---RES: Rick, here are the constraints relevant to CS. Could you add something on the departmental philosophy (cf. below for E&CE)

o Mathematics faculty regulations: all CS students are Mathematics majors, which means that they must take 1.5 years of Mathematics courses in a 4 year degree;

o departmental regulations: both departments added their own course requirements to the faculty pool;

o Computer Science Accreditation Council (CSAC) requirements: the CSAC is an autonomous accreditation body established by the Canadian Information Processing Society. We wanted this option to be fully accreditable by the CSAC;

CSAC requires:

o 1.5 years of study in computer science/computer engineering;

o 0.5 years of study in mathematics/statistics;

o 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:

o 1.3 years of study in mathematics/statistics;

o 1.3 years of study in computer science;

o 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[3]:

- 1/2 year of Mathematics;
- 1/2 year of Basic (Natural) Sciences;
- 2 years of Engineering Sciences and Engineering Design
  (with at least 1/2 year of Engineering Sciences and 1/2 year of Engineering Design);
- 1/2 year 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 indivigual segments. The current content ofa segment is the result of historical evolution of the engineering discipline. In harmony with many other EE/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(5 courses) and software(5), with some coverage of the light-current analog technology(2). 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 all three 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:

o Software Requirements Specification and Analysis

o Software Design and Architecture

o Software Quality Assurance, Testing, and Process

o Software Maintenance and Reverse Engineering.

However, 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 above,

---RES: refer to constraint C2 above (Sect 2)??

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). As a result, these two previously proposed courses have been 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/ECE option are as follows:

o System design and the design of changes to systems;

---RES: system is a very broad spectrum term; do we talk here about Software systems?

o Requirements analysis, specification, design, construction, verification, testing, maintenance and modification of programs, program components, and software systems;

o Algorithm design, complexity analysis, safety analysis, and software verification;

o Database design, database administration and maintenance;

o Design and construction of human-computer interaction systems;

o Management of projects that accomplish the above tasks, including estimating and controlling their cost and duration, organizing teams, and monitoring quality;

o Selection and use of software tools and components;

o Appreciation of commercial, financial, legal, and ethical issues arising in software engineering projects.

Furthermore, two of the goals of the software engineering option are:

o to foster non-computer science and non-technical skills which are crucial to successful software development--things like critical thinking, technical writing, presentation skills, etc.

o to broaden a student's background, through recommending (and, in fact, demanding) a broad range of non-technical courses

---RES: the first item (to foster) go to Goals and Limitations (Sect 2) - please rephrase neutrally, without reference to CS. The second item (to broaden) has a problem - broad range may apply perhaps to CS, with its 10 non-CS, non-CE courses, for CompE the limit is 5, which is not a 'broad range'.

4.2 Option Outline

4.2.1 Technical Courses

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

o 8 core technical courses

o 3 courses focussing on the software development life cycle

o 3 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 ECE versions of the option, but remain consistent with the cores of the respective departments.

4.2.2 Nontechnical

In addition 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 to be addressed while developing the curriculum was to decide the number of software engineering courses that would cover all the salient topics in the area. Major guiding parameters on this issue were: the breadth and depth at which the topics would be covered, the undergraduate level pedagogical material that was available for each of the proposed courses, and the number of new courses that can be put into already existing schedules of the undergraduate curricula of the CS and the E&CE departments. We toyed with the idea of three or four new courses. In the end, keeping above constraints in view, it was

decided to propose three new courses.

The three new courses form the core of the SE-option. The underlying theme in this endeavour 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 in 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.

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 are to 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 three courses developed for the SE-option are:

1.Software Requirements Specification and Analysis

2.Software Design and Architecture

3.Software Testing, Quality Assurance, and Maintenance

4.

Following subsections present a brief outline of the courses. Each course also includes a substantial project. The projects associated with these courses are described in 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 processes 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
- Overview of notations and representations
-.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
- 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 with an emphasis on the relationship of software design with other activities in software development.

- 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
- 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 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 Private Branch Exchange (PBX). Figure X(a) depicts a system of PBXs interconnected by trunks. 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, Administrative and Maintenance (OAM) unit. The OAM unit provides a graphical user interface to perform the functions of administration (maintaining the PBX database, metering, etc.) and maintenance (request for manual maintenance functions, line in/out of service indication, etc.).

Figure X(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 (time-space-time switch), two line interface shelves (30 line interfaces/shelf), a Service shelf (with digit receivers, call in progress tone generators), a Ringing Generator and a Tester. Each line interfaces A/D and D/A converter is connected to time-multiplexed links carrying 32 channels (32 8-bit samples) in a 125 microsecond frame. Each line interface is configurable in addition to controlled test and ring relays, on/off-hook loop detector and voice channel selection control.

Figure X(c) depicts the major subsystems from the project perspective. The Control Program is the real-time call control software specified, designed and implemented by the students. Since the real PBX hardware is not available, a Hardware Subsystem Emulator is provided. The Hardware Emulator software emulates all the controlled hardware and provides status and receives control from the control software through a shared memory interface. The Customer Acceptance Tester (CAT) is a software subsystem used to automate the validation of the students control software. The CAT eliminates the extensive manual testing involved to ensure that the control software meets the customer requirements and performance specifications.

Since we do not have the actual PBX hardware in the departmental laboratories, a network of UNIX-based workstations is used for both the development and emulation environment. The UNIX workstations are interconnected by a campus wide Ethernet-based LAN.

The project is divided into two major software subsystems; primarily driven by the interest and backgrounds of the two streams of students. The first is that of real-time reactive software control, Control Program (Figure X(c)) for the ECE students. The second is information systems software, the OAM (Figure X(a)) for the CS students. Appropriate CASE tools are used to support the unique development requirements of both software systems. The real-time software system is modeled as communicating finite state machines while object oriented analysis and modeling techniques are used for the information subsystem.

The project consists of three primary phases. First, during SE-1, the software requirement and specifications are generated for each software subsystem with the primary deliverable being the

SRS document. Secondly, during SE-2, a design document (based on the SRS) is produced and each subsystem is coded and validated using the automated customer acceptance tester. At this stage, the students would use what ever techniques that they are familiar with to perform unit and integration level testing of their implementation. Thirdly, in SE-3, additional PBX and OAM functionality is introduced (i.e. requirement of handling trunks, additional features, advanced administrative and system maintenance facilities). This reflects the maintenance activities in the software lifecycle. 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.

[From JA; to be merged by PD]

Administration functions include

* HCI interface between operator and switching software, to maintain switch's image of the enchange's environment: subscriber lines, type-of-line information, trunks, subscribers' set of allowable services, etc. For now, subscriber services will be limited to originating-calls-only, terminating-calls-only, originating-and- terminating calls, and local calls (within exchange) only. One possibility for extending the system during maintenance would be to introduce new features, such as call forwarding.

* HCI interface between operator and switching software, to maintain translation (and routing?) tables.

* Maintenance of customer database: name, address, phone number, # lines, subscribed services, etc.

* Acquisition of billing data from switch

Billing functions include

* Generation of customers' bills.

- HCI interface between operator and billing software, to maintain billing rates, discount times-of-day and days-of-week, query customer's bill, etc.

Maintenance functions include

* HCI interface between operator and switching software to execute certain maintenance tests, outside of the normal, automated testing process: test single parameter of 1 subscriber line, run all tests on 1 subscriber line, run series of automated tests NOW.

- HCI interface between operator and switching software, to maintain automatic testing: update interval of testing, update which tests are to be run automatically.

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 and policies of both departments.

5.1 Teaching Resources

Most of the Option's technical courses are either explicitly computer science or explicitly electrical and computer engineering courses. The resources needed to deliver each of these courses are clearly the responsibility of the respective department offering the course. However, the option introduces four new courses, mentioned above, 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.

Eventually, however, the departments may decide to re-allocate teaching resources for Option courses, based on the distribution of students in the option.

---RES: Jo, prefix this sentence with something like 'Given that the CS department admits three times as many students then the Comuter Engineering program'

We anticipate that twice as many computer science students as computer engineering students will register for the Option. If our estimates are correct, it will be difficult to justify dividing the departments' teaching responsibilities evenly.

5.2 Teaching Assistants vs. Lab Technicians

The two departments differ in how graduate students and staff are used to support teaching efforts. The Computer Science Department grants half-time teaching assistantships to most 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,

---RES: Jo, a more accurate phrasing would be: "a certain number of teaching assistant positions are allocated to a course. These positions are awarded on the basis of expertise and merit."

teaching assistantships supplement a student's research assistantship. 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

technicians. Laboratory technicians are responsible for maintaining and answering questions about hardware used in courses that have laboratory assignments. In addition, technicians are responsible for tutorials and consultant-support for software in programming courses.

Because the CS Department does not employ the equivalent of laboratory technicians, a technician 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, however, that are assigned to an Option course will come from the same department as the course instructor. This arrangement will help to ensure a high rapport between instructors and assistants.

---RES: Jo - this may result in potential 'us vs. them' problems - e.g. the CS students may feel that the E&CE profs and TAs give preference to their home students. It would be better if we had a mix of TA's.

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

---RES: we should say what the abbreviation means

and Software through Pictures (StP)/Object Modeling Technique (OMT); based on the information available, we decided that these were the most appropriate CASE tools for developing the Option projects[1].

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 administrator project requires all of these different modeling 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

1. At present, we have only considered CASE tools for the requirements and design phases.

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, one of the Faculties has an industrial grant for enhancing information-systems education. Money from this grant are being used to procure the CASE tools and to upgrade the machines; the other Faculty will reimburse the grant using future University Academic Development Funds.

Lastly, because of our endeavors to choose the most appropriate CASE tools and because of our difficulties 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.

6. Concluding Observations

--RES: should we throw in the following observation? Software is an implementation technology; so is steel. There are no 'steel engineering' programs at universities; instead Civil and Mechanical use this technology, but for differrent applications. Will the same happen to Software Engineering programs?

Acknowledgment

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 Inititative 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, Ont., 1995.