The Software Engineering Body of Knowledge for Professional Engineering in Canada

by *R.D. Dony - University of Guelph, Guelph, ON*
P. Botman - True North Systems Consulting, Vanc., BC
W.E. Briggs - Univ. of New Brunswick, Fredricton, NB
R. Haggart - Olorin Enterptise Inc., Ottawa, ON
P.A. Taylor - McMaster University, Hamilton, ON

Abstract

In response to the need to define the academic requirements for licensing professional engineers in Canada, the Canadian Engineering Qualifications Board formed a committee whose mandate was to define the core and supplemental bodies of knowledge for Software Engineering Some of the issues and the outcome of the software engineering committee's deliberations are presented in this paper. After examining a number of inputs including national and provincial examination syllabi and curricula of accredited software engineering programs in Canada, a number of core topic areas were defined. These are: Discrete Mathematics, Data Structures and Algorithms, Software Development (includes Software Engineering Process, Requirements, Design, Construction, Testing, Maintenance, and Configuration Management), System Reliability and Safety, Digital Systems, Computer Architecture, Operating Systems, File and Database, and Systems and Control. These topics generally match well with the existing examination syllabi and curricula. The supplemental areas are less well-defined and include depth in core subjects, depth in areas of specialization, and breadth in application domains.

Sommaire

Dans le but de définir les exigences académiques nécessaires à l'attribution des permis d'ingénieur professionnel au Canada, le Bureau canadien d'accréditation des programmes d'ingénierie a formé un comité dont le mandat était de définir les connaissances de base et complémentaires en génie logiciel. Certains des points soulevés lors des délibérations du comité et les conclusions obtenues par ce même comité sont présentés dans cet article. Après avoir examiné un certain nombre d'éléments, notamment les programmes de formation canadiens accrédités en génie logiciel, les sujets suivants ont été établis comme étant des sujets de base: mathématiques discrètes, structures de données et algorithmes, développement de logiciel (incluant processus de génie logiciel, exigences, conception, construction, tests, maintenance et gestion de configuration), fiabilité et sécurité de systèmes, systèmes numériques, architecture d'ordinateur, systèmes d'exploitation, fichiers et bases de données, et systèmes et commandes. En général, ces sujets s'accordent bien avec les programmes d'examens et les curriculums existants. Les domaines supplémentaires ne sont pas aussi bien définis et inclus des approfondissements sur les sujets de base, détaillent des domaines spécialisés, et traitent de différents domaines d'applications.

1.0 Introduction

Provincial and territorial associations of professional engineers are responsible for the regulation of the practice of engineering in Canada. Each association has been established under an Act of its provincial or territorial legislature and serves as the licensing authority for engineers practicing within its jurisdiction. The Canadian Council of Professional Engineers (CCPE) is the national federation of these associations and provides a coordinating function among them. One of the means of providing this coordination is the generation of guidelines. Such guidelines are an expression of general guiding principles which have a broad basis of consensus, while recognizing and supporting the autonomy of each constituent association to administer its engineering act.

Two of the four working boards of the CCPE are the Canadian Engineering Qualifications Board (CEQB) and the Canadian Engineering Accreditation Board (CEAB)). The CEAB was established to accredit undergraduate engineering programs which provide engineers with the academic requirements necessary for registration as a professional engineer in Canada. The CEQB's primary role is to develop national guidelines on professional engineering qualifications, standards of practice, and ethical professional conduct. It is also responsible for the CCPE Examination Syllabus that describes an examination program to assess the academic qualifications of individuals who have not graduated from an engineering program that has been accredited by the CEAB.

In September 2000, the CEQB and the CEAB, discussed the idea of developing a body of knowledge for each engineering discipline. The body of knowledge for each discipline would consist of a Core Body of Knowledge (CBOK), a Supplementary Body of Knowledge (SBOK), and other basic science, mathematics and complementary knowledge specified by the CEAB and CEQB for all engineering disciplines. The CBOK for a discipline comprises all the material and areas that must be studied by each person in that discipline. Additional specialization within a discipline which is not part of the CBOK is the SBOK. A committee, whose members are the authors of the draft report [1] which this paper summarizes, was formed in February 2001 with the mandate to define the CBOK and SBOK for Software Engineering. At the same time, a parallel committee was formed to define the Chemical Engineering BOK. Some of the issues and the outcome of the software engineering committee's deliberations are presented here.

1.1 Definition of Software Engineering

The following definition of software engineering is adapted from the definition of professional engineering in the CCPE Guideline on the Professional Engineering Practice in Canada [2].

Professional software engineering involves any act of planning, designing, composing, evaluating, advising, reporting, directing or supervising, or managing software-intensive products or processes that requires the application of engineering principles, and that concerns the safeguarding of life, health, property, economic interests, the public welfare or the environment.

Software engineers are concerned with the analysis, design, programming, testing, system integration, commissioning, support and retirement of software systems, particularly those systems which are critical to public health, safety, and the environment. Software engineers apply engineering methods and discipline to produce reliable systems which are of known quality, and fit for intended use.

1.2 Scope

The report does deal with knowledge areas common to other engineering disciplines. Such topics would be analogous to the “Basic Studies” in the CEQB Syllabi, including such topics as differential equations, statics and dynamics, basic electromagnetism and even thermodynam-

Acknowledgement

This article is copyright of 2002 IEEE. Reprinted, with permission, from the IEEE Canadian Conference on Computer and Electrical Engineering, CCECE 2002, Winnipeg, May, 2002.

Editor’s Note: Due to the importance of this article on our industry, this article is reprinted in full. Your feedback is welcome.

ics or fluid mechanics for example. Such foundations in a broad range of basic sciences, both applied and engineering, allows a practitioner from any one discipline in engineering to be, at the very minimum, at least conversant with colleagues in other disciplines. As it is expected that many software engineers would apply their profession in the context of other engineering fields, this foundation is of particular importance. Of course, many of foundation areas will be common to one or two other disciplines, for example computer or electrical engineering.

The report is primarily concerned with the academic knowledge mastered by a software engineer. In pragmatic terms, this report enumerates those subjects which might be used as a basis for the evaluation of the background of a candidate seeking admission to the profession.

It is emphasized that a software engineer, in addition to such academic knowledge, is expected to have practical knowledge and experience in the various sub-disciplines of software engineering. This document does not attempt to address the nature and significance of these forms of software engineering knowledge. However the enumeration of topics within this document might facilitate future discussions of practical and experiential knowledge in specific sub-disciplines.

1.3 Use of Report

As a subcommittee of the CEQB, the focus of this work is directed at assisting the admissions process for professional engineering. In Canada, there are three criteria for admissions: minimum academic background, suitable work experience, and passing a professional practice examination. The committee's work focused mainly on the academic requirements for licensure by defining the minimum knowledge areas that a software engineer should possess. To that end, two of the more concrete applications of a body of knowledge for any engineering discipline are the CEQB syllabi, and the CEAB curriculum.

The CEQB examinations can be considered a rather direct embodiment of a BOK as there should be a straightforward mapping between knowledge areas and examination topics. However, technically, the CEAB criteria does not contain any reference to a BOK for a discipline, just general categories of subject material. Implicit in the process is the assumption that the Program Visitor has an idea of what the BOK should be for a program and judges the course offerings accordingly. If the BOK were to be made explicit and codified, the mapping between knowledge areas and course topics, again, should be straight-forward. Therefore it would be helpful if the portions of a discipline's BOK that map nicely to examination curriculum topics were arranged to facilitate this mapping.

Software engineers are concerned with the analysis, design, programming, testing, system integration, commissioning, support and retirement of software systems, particularly those systems which are critical to public health, safety, and the environment.

2.0 Previous Work

Before the committee's work, a number of organizations had began work in this area. The CEQB Syllabus for Software Engineering, item [3], was initially proposed as part of the 1998 syllabus. In response to concerns about the quickly evolving nature of the discipline, some associations/ordre developed modifications to the national syllabus [4,5].

During 1998-1999, a committee of Professional Engineers Ontario (PEO), the Engineering Disciplines Task Group (EDTG), developed a core body of knowledge guideline for use in evaluating CEAB graduates who have switched into the field of Software Engineering subsequent to graduating [6]. To reconcile the differences between these requirements and the original CEQB syllabus, PEO devised a new syllabus [4]. William's paper [7] discusses the rationale.

The report by *l'Ordre des ingénieurs du Québec (OIQ)* [8] is a comprehensive document that reports on the findings of the ad hoc group on software engineering. It discusses the definitions and features of software engineering, aspects of training, and presents a number of recommendations for the *ordre*. For the purposes of this report Table 3 on pages 15-16, summarizing topics for an education in Software Engineering and grouped into the CEAB academic units (AU) categories, is of primary interest.

Also considered for this work were the curricula of the first "Software Engineering" programs at Canadian universities that have been granted accreditation by the CEAB in 2001 McMaster University [9], University of Ottawa [10], and University of Western Ontario[11]. All three programs are offered through existing engineering faculties that currently have existing accredited engineering programs. As the first graduating classes for these programs was in 2001, they were visited by CEAB teams in the Fall of 2000 and were accredited by the CEAB in June, 2001.

The IEEE SWEBOK document[12] is the latest version of a project through the IEEE Computer Society, and managed through the *Université du Québec à Montréal*. The project has been sponsored financially by a number of companies and organizations including the CCPE. Its scope does not include the issue of the regulation of the practice of professional engineering. Its focus is on ten knowledge areas specific to the software development and maintenance process: requirements, design, construction, testing, maintenance, configuration, management, engineering process, tools, and quality. While the report goes into some depth into each of these topics, it does not deal with the larger issue of the supporting knowledge areas required for a regulated profession. For this report, the SWEBOK document was a useful input for the "Software Development" 3.3 area as developed below.

The final three items that were considered [6,13,14] are concerned with the evaluation of engineering experience in the software engineering field.

3.0 Core Body Of Knowledge (CBOK) For Software Engineering

3.1 Discrete Mathematics

For any in-depth treatment of software for both analysis and design, an additional set of mathematical tools are required in addition to the classical linear algebra and calculus topics. This area is termed "discrete mathematics" or "discrete structures." These topics are a common component of many Computer Science programs as they provide the mathematical foundations for the areas of data structures and algorithms.

Topic areas include: functions, relations, and sets, trees, graphs, logic, Boolean algebra, combinational methods, state models, proof techniques, basics of counting.

3.2 Data Structures and Algorithms

Both data structures and algorithms form the fundamentals of computer science and software development. For software engineering, the emphasis is on knowledge of the various basic structures and algorithms and their characteristics. Again, these topics are a common component of Computer Science programs. Knowledge of discrete mathematics is required.

Topic areas include: Queues, stacks, lists, heaps, trees, graphs, data abstraction, sorting, searching, parsing, pattern matching, divide and conquer, greedy methods, algorithm complexity, selection criteria.

3.3 Software Development

This broadly defined area forms the "core" of the unique discipline of Software Engineering. It is concerned with the application of the above theoretical foundations to produce specification-correct software in a real-world development environment. The area includes the software lifecycle of requirements, design, construction, testing, and maintenance. Also included is configuration management and software engineering process. Both the more formal academic concepts as well as the more practical, experience-oriented areas constitute knowledge in the area.

3.3.1. Software Engineering Process:

The Software Engineering Process is a systematic approach that starts at the conceptual phase (original idea) and concludes with an operationally reliable and maintainable piece of software. It includes the management functions of software development, quality, testing, and configuration management. The Software Engineering Process supports

a similar process at the Systems Engineering level and includes software engineering methods and tools.

Topic areas include: software engineering process concepts, process infrastructure, process measurement, process definition, qualitative process analysis, process implementation and change.

3.3.2. Requirements:

This topic covers the activities associated with software requirements in a manner that verifiably supports the subsequent phases: design, construction, test, operation, maintenance, etc. It includes the analysis of system level or other software architectures into which this design must fit.

Topic areas include: elicitation, analysis, requirements and specifications, functional and nonfunctional requirements, prototyping, formal specification techniques, validation.

3.3.3. Design:

This topic covers the activities associated with software design. Key elements are designing for testability, maintainability and quality.

Topic areas include: software architecture and structure, object-oriented analysis and design, component-level design, distributed models, design for reuse, quality analysis and evaluation, notations, validation.

3.3.4. Construction:

Topics include: reduction in complexity, anticipation of diversity, linguistic methods, formal methods, visual methods, validation, language evaluation and selection.

3.3.5 Testing:

Topic areas include: concepts and definitions, levels (requirements, design, construction, integration, operational), techniques, measures.

3.3.6 Maintenance:

This topic covers the activities associated with software maintenance. Maintenance is required to ensure that the software continues to meet its functional and quality requirements in a changing operational setting.

Topic areas include: documentation, requirements and specifications, functional and nonfunctional requirements, prototyping, formal specification techniques, structured design, object-oriented analysis and design, component-level design, distributed models, design for reuse, software maintenance, tools and environments.

3.3.7 Configuration Management:

Topic areas include: management of the software configuration process, configuration identification, configuration control, status accounting, auditing, software release management and delivery.

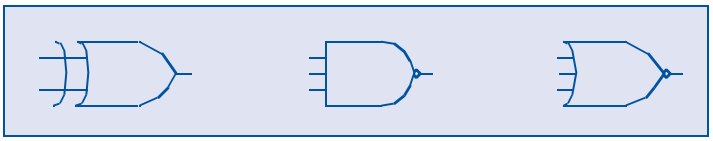
3.4 System Reliability and Safety

This area addresses the issue of reliability and safety from an entire system perspective. While quality and reliability concerns are of importance in all stages of the software development life cycle, it is important to emphasize the reliability and safety of the entire system, be it a physical realization or otherwise. For the professional engineer, this topic directly impacts the obligation to ensure the safety and well-being of the public through the design of verifiable, reliable software where the effects of software failure on the system may have safety implications. While many of the issues here are covered above, it has been enumerated separately to emphasize the obligation that the professional engineer has to produce safe and reliable systems.

Topics areas include: Formal methods and specifications, pre and post conditions and assertions, formal verification, uncertain and changing requirements, verification and validation, test plan creation and test case generation, black-box and white-box testing techniques, real-time and resource constraints, redundancy and fault tolerance, failure modes, probabilistic methods of analysis, quality measurements performance modelling.

3.5 Digital Systems

Knowledge of digital systems for a software engineer is required to ensure that computer is not merely a “black box” for running software. This area is a required component of both electrical and computer engineering (e.g. 98-Comp-A2). It builds on the introductory logic topics of discrete mathematics.



Topic areas include: design of combinational and sequential circuits, implementation using gates, logical system components, programmable logic devices and gate arrays, high-level description languages, small system design, data flow, signals, timing, basic microprocessor organization and interfacing.

3.6 Computer Architecture

This area further builds on the knowledge of digital systems as described above and, again, is common with computer engineering (e.g. CEQB syllabus examination 98-Comp-A3) but not at the same depth (e.g., just to the logic abstraction level, not the device level). The treatment of this topic from a computer science perspective typically does not go into the depth in the hardware organization that a prerequisite knowledge of digital systems allows.

Topic areas include: computer structure and processor architecture, CPU and memory organization, buses, computer interfacing, parallel and serial I/O, storage devices, instruction sets, addressing modes, registers, interrupts and I/O, special purpose processors, embedded systems.

3.7 Operating Systems

As most software runs under the supervision and resource control of operating systems, understanding the characteristics and limitations of operating systems and their effects on the execution of programs is necessary. The knowledge of these relationships is particularly important in control and real-time systems. Some areas of this topic are common with computer engineering (e.g. CEQB syllabus examination 98-Comp-A5) as well as computer science. Typically the former deals with the use and characteristics of operating systems while the latter may focus more on their design. Some concepts from computer organization (e.g. resources such as memory) are required.

Topic areas include: interprocess communication, synchronization, scheduling, resource allocation, memory management, multi-tasking and multi-processing, performance and measurement, real-time support requirements, deadlock, features of modern operating systems.

3.8 File and Database

Most software systems have some reliance on file and database systems. Further, the processing of large volumes of data may be required for safety critical systems. This area tends to be part of a core computer science program.

Topic areas include: data models, entity-relationships, mass storage devices, file structures, data base types (relational, hierarchical, network), file organization (sequential, indexed, direct access, hashing), query methods and languages, security and integrity, transaction processing.

3.9 Systems and Control

This subject area includes the fundamentals of stability analysis of feedback control systems. Since the need for regulation comes directly from the need to protect public safety, stability analysis is critical in the design of any software control system that interfaces to the physical world. This is a core subject in most electrical and computer engineering related disciplines.

Topic areas include: Models, transfer functions and system response, Root locus analysis and design, feedback and stability, Bode diagrams, Nyquist criterion, frequency domain design, state variable representation, PID control, digital control, Z transform, computer control interfacing and algorithms.

4.0 Mapping To CEAB Curricula And Examination Syllabi

Table 1 shows a set of general subject/knowledge areas common to many or all of the syllabi and curricula. For the three examination syl-

labi, labelled "CEQB" for the original CEAB 1998 syllabus, "PEO" for the Ontario revision, and "BC" for the British Columbia revision, the examination numbers are given that are concerned with the general areas listed on the left. The course numbers of the curriculum of the three accredited programs, McMaster University (MAC), University of Ottawa (UoO), and University of Western Ontario (UWO) are likewise categorized. The OIQ entries just specify whether or not the topic is included in its Table 3. As the SWEBOK document deals with only a subset of these topics in the software development area, albeit at a significant level of detail, it was not included in the table. References are made to specific examinations or courses that deal with the broad topic area.

At the broadest level, there appears to be a significant agreement between what the various sources consider as required knowledge areas for software engineering. As the field is evolving, it is expected that some differences do exist. The CEQB syllabus appears not to require discrete mathematics and elements of software testing and reliability. Both PEO and the McMaster curriculum require systems and control as do other software engineering programs seeking accreditation such as the University of Waterloo.

5.0 Supplementary Body Of Knowledge (SBOK) For Software Engineering

In addition to the core knowledge, it is assumed that an engineer will have additional knowledge. This additional knowledge makes up the SBOK. All of the SBOK topic areas require mastery of the topics in the CBOK. Areas may be deleted or added as technology and engineering techniques change over time.

The body of knowledge is augmented by the SBOK in a number of ways.

5.1 Depth in Core Subjects

Engineers may develop some of the knowledge areas in the CBOK to a greater depth. This would be analogous to advanced upper-year courses that further extend material established in earlier core courses.

5.2 Depth in Areas of Specialization

These areas are topics in software engineering that are in addition to the CBOK. Here, the practitioner applies advanced knowledge and skills in software engineering areas beyond the minimum core.

Examples include:

- Software Product Line Engineering,
- Software Process Assessment and Improvement,
- Software Estimation and Planning Tools,
- Concurrent Software Design,
- Real Time Systems Design,
- Human Computer Interface Design,
- Advanced Test System Design

5.3 Breadth in Application Domains

Software engineers may apply their skills in application domains outside of their specific discipline. In fact, this interface with other engineering disciplines is of particular importance to the professional software engineer and thus would require the software engineer to have a good working knowledge of the application domain. Since such areas would typically be related to other engineering disciplines, the practitioner would require a broader range of knowledge of the specific application areas.

Examples include:

- Industrial Process Control,
- Telecommunications (switching) Software,
- Avionics Software,
- Networking (protocol) and Distributed Systems software,
- Medical Components and Systems.

6.0 Summary And Conclusions

The recognition of Software Engineering as a distinct discipline within a licensed, professional engineering framework is a relatively recent development. However, the foundations of the discipline do have a much longer history. As a result, there is a general consensus as to what constitutes a core body of knowledge in the field. The topics include not only those concerned with the immediate development of software, and the software lifecycle, but topics such as digital systems, computer architecture and control systems. This is reflects an engineering philosophy, which requires that a software engineer be knowledgeable in all related aspects of systems and environments in which the software operates. Coupled with the implied basic sciences and mathematics, this breadth gives professional software engineers an academic foundation which ensures that the software-based products and processes they develop meet their professional obligations to uphold the public safety.

The various inputs to the committee's work all reflect general agreement as to the most important and central topics. Even the IEEE SWEBOK project, while not specifically addressing the issue of the requirements for licensing professional engineers in Canada, helped in specifying the software development topics. Of course, as the discipline evolves, the core body of knowledge must adapt accordingly. As this body of knowledge is maintained, the various inputs listed, and materials from their respective organizations, should be monitored for changes as well.

The supplemental body of knowledge is, on the other hand, not currently well defined. In general, a software engineer should have additional knowledge beyond the core topics - knowledge that includes

Table 1: Overview of Basic Topics in Software Engineering

General Area	CEQB[3]	PEO[4]	BC[5]	OIQ[8]	MAC[9]	UoO[10]	UWO[11]
Discrete Mathematics	-	BS17	-	Yes	2E03	MAT2343	SE251
Data Structures and Algorithms	A1, A4	A4	BS(A1)	Yes	2C04	CSI2114, CSI3105	CS027, CS210, CS340
Software Development	A5	A4	A3	Yes	2A03, 2B04	SEG2100, SEG2101, SEG3100	SE351, SE452, SE453
Testing, Reliability	-	A7	-	Yes	2F03	SEG4111	SE453
Digital Systems	A2	A2	BS(A2)	Yes	2D04	ELG1100	ECE339
Computer Architecture	A2	A3	BS(A2)	Yes	3G03, 3F03	ELG2181, CEG3391	ECE375
Operating Systems	A3	A5	A1	Yes	3B04	CSI 3310	SE201, CS305
File and Database	A6	-	A4	Yes	3H03	CSI 3317	-
Systems and Control	-	A1	-	-	3L03, 4A03	-	-

additional depth in one or more core topics, depth in one or more specialization areas, and/or breadth in application domains. There are already many software engineers with special expertise, or who are practicing in certain application domains. However this was not addressed at this time. This area will need to be developed in subsequent drafts, and input is sought from all readers of this initial draft. As with the core topics, the supplemental topics will also evolve as the discipline evolves, so review and maintenance of the document will apply here as well.

The management of software development including the entire lifecycle has not been explicitly included in this report. It is understood that project management techniques common across all engineering disciplines can be applied in software engineering. It is also understood that there may be situations unique to software engineering that requires alternative approaches.

7.0 Acknowledgments

The authors wish to thank the staff of the Canadian Council of Professional Engineers for their invaluable support in the committee's work, in particular David Lapp, Marie Carter and Begonia Lojk. We would also like to thank the Chair of the Body of Knowledge Pilot Project, Phil Sunderland, for his support and patience with the work.

8.0 References

[1]. "Core and Supplementary Bodies of Knowledge for Software Engineering." Canadian Council of Professional Engineers, Ottawa, ON, February 2002.

[2]. "Guideline on the Professional Engineering Practice in Canada." Canadian Council of Professional Engineers, Ottawa, ON, 2001.

[3]. "CCPE 1998 Software Engineering Syllabus." Canadian Council of Professional Engineers, Ottawa, ON, 1998.

[4]. "PEO Revised Software Engineering Syllabus." Professional Engineers Ontario, Toronto, ON, 1999.

[5]. "APEGBC Revised Software Engineering Syllabus." Association of Professional Engineers and Geoscientists of British Columbia, Burnaby BC, 1999.

[6]. "PEO EDTG Experience Requirements for Cross-Discipline Applicants Practicing in the Software Engineering Field." Professional Engineers Ontario, Toronto, ON, March 25 1999.

[7]. N. Williams, "Professional Engineers Ontario's Approach to Licensing Software Engineering Practitioners," in Proc. 14th Conference on Software Engineering Education and Training, (Charlotte NC), pp. 77-78, February 2001.

[8]. "Software Engineering Definitive Report." l'Ordre des ingénieurs du Québec, Montréal, QC, November 17 2000.

[9]. "McMaster University Software Engineering Curriculum." Department of Computing and Software, McMaster University, Hamilton ON, (CEAB Accredited 2001).

[10]. "University of Ottawa Software Engineering Curriculum." School of Information Technology and Engineering, University of Ottawa, Ottawa ON, (CEAB Accredited 2001).

[11]. "University of Western Ontario Software Engineering Curriculum." Department of Electrical and Computer Engineering, University of Western Ontario, London ON, (CEAB Accredited 2001).

[12]. "IEEE SWEBOK Stone Man Version (0.7)." IEEE, April 2000.

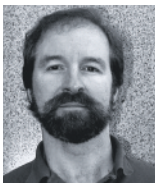
[13]. "CEQB Interview and Assessment Guide for the Evaluation of Software Engineering Experience." Canadian Council of Professional Engineers, Ottawa, ON, March 12 2001.

[14]. "CEQB Guide for the Presentation and Evaluation of Software Engineering Experience." Canadian Council of Professional Engineers, Ottawa, ON, June 13 2001.

About the authors

Bob Dony received his B.A.Sc. and M.A.Sc. in Systems Design Engineering, University of Waterloo in 1986 and 1988 respectively, and his Ph.D. in Electrical and Computer Engineering, McMaster University in 1995. He has been with the School of Engineering, University of Guelph, as an Assistant Professor in the Engineering Systems and Computing program since 1997. He is a member of the Canadian Engineering Qualifications Board (CEQB), and Chair of the Software Engineering subcommittee of the CEQB Body of Knowledge Pilot Project. He is a registered Professional Engineer in Ontario and is a member of the Academic Requirements Committee, Evolution of Engineering Admissions Task Force, and Emerging Disciplines Task Group of Professional Engineers Ontario (PEO). He also served as the Technical Committee Co-chair for the 2001 IEEE Canadian Conference on Electrical and Computer Engineering and is a member of the Canadian Conference on Computer and Software Engineering Education steering committee.

Pieter Botman is a Professional Engineer registered in the Province of British Columbia. With over 20 years of software engineering and management experience, he assists software organizations in software process assessment/improvement, software project management, software quality management, and product management. Mr. Botman is a member of the IEEE Computer Society, the ACM, and ASQ's Software Quality division. He can be reached at p.botman@ieee.org



Bill Briggs took his B.Sc. (Hon. Chem.) and Engineering Certificate from Mt.Allison University, went on to the M.Sc.E. program (Electrical) at the University of New Brunswick. He then worked for NB Power, first in the Nuclear Operations Group, and then in Distribution. He spent five years working as an R&D scientist at Fiberglas Canada Inc. For the last decade he has been a self-employed consultant working in a variety of hi-tech areas. He recently returned to the academic world as a Senior Instructor in the Department of Electrical and Computer Engineering at the University of New Brunswick.

Ross Haggart graduated from Carleton University in 1980 with a Bachelor's Degree in Electrical Engineering specializing in Computer Systems. He spent the first half of his career designing, developing and deploying computer systems for military and industrial applications. Subsequently, he worked in the Business Systems consulting industry primarily in Program and Project Management activities. Mr. Haggart is currently the Vice-President of Engineering for Cerkits Corporation, a Canadian Corporation that develops Security Robots for Military and Civilian environments. He is a member of the Professional Engineers of Ontario and serves on the Experience Requirements Committee (ERC) for the PEO and the Engineering Entrance Admissions Task Force.

Paul Taylor obtained his B.Sc. and Ph.D. from the University of Wales.

He is presently the Chair and Professor in the Department of Computing and Software, at McMaster University, Hamilton.

His research is directed towards the application of computer control to chemical processes. It covers three areas of expertise: control theory, process dynamics and modelling and real-time computer applications.