

**Software Engineering in Canada,
the U.S. and the UK: Inter-professional
relations and the emergence
of a new profession**
WANE Working Paper #9

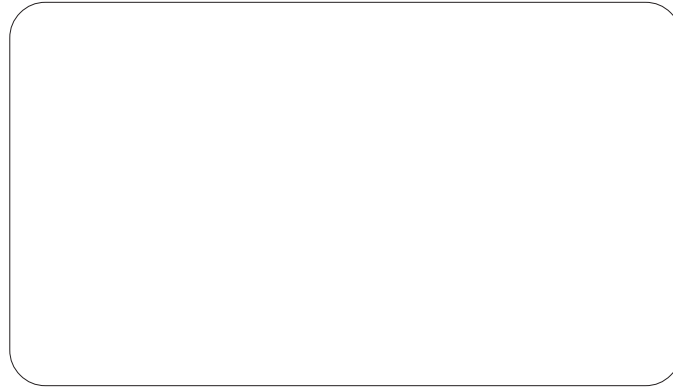
Tracey Adams, Ph.D.
Workforce Aging in the New Economy



www.wane.ca

Workforce Aging in the New Economy

A Comparative Study of Information Technology Employment



WORKFORCE AGING In The NEW ECONOMY (W.A.N.E.) explores the relationships among workforce aging, employment growth in information technology (IT) labour markets, and the transformation of employment relations in the new economy. This work involves a multi-disciplinary, cross-national comparison of IT employment and workforce aging in Canada, the United States, the European Union, and Australia.

Principal Investigator:
Julie McMullin, Ph.D.

Project Manager:
Terri Tomchick, MA

Workforce Aging in the New Economy

The University of Western Ontario
Social Sciences Centre, Room 3207
London, Ontario CANADA
N6A 5C2

t: 519-611-2111 x.81236

f: 519-661-3200

e: wane@uwo.ca

**Software Engineering in Canada,
the U.S. and the UK: Inter-professional
relations and the emergence
of a new profession**

WANE Working Paper #9

Tracey L. Adams¹
Workforce Aging in the New Economy
University of Western Ontario
London, ON CANADA

Tracey L. Adams, Ph.D.
Professor, Department of Sociology
University of Western Ontario

September, 2004

I would like to thank Rob Downie and Erin Demaiter for their research assistance, and Julie McMullin for her comments on an earlier version of this paper.

Introduction

One of the principal professional developments in computing during the past 20 years is the emergence of a software engineering profession. Although the term software engineering was coined over 30 years ago, it was not used regularly to refer to a distinct set of computing-related tasks until much more recently. In the rapidly changing field of information technology, software engineering, as a distinct occupation, is a relative newcomer. Nevertheless, software engineering is a rapidly professionalizing occupation. In a number of countries around the world, engineers and computer scientists have endeavoured to define the domain of software engineering, and delineate its knowledge base and scope of practice. Formal education programs in this area have become more prevalent. Moreover, in many countries credentials for software engineers are being established, and the licensing of software engineers has begun amidst much controversy.

Efforts to establish a software engineering profession have implications for both the nature of employment in the IT field, and the sociology of professions. Sociologically, study of the development of software engineering as an occupation offers insight into the field of inter-professional relations. In some nations, in some periods, computer scientists and engineers have worked together to create a new occupation that intersects their two disciplines. In other social contexts, inter-professional conflict has erupted over the professional regulation and education of software engineers. Exploring the emergence of software engineering sheds light on the contextual factors that encourage inter-professional

collaboration, and those that foster inter-professional conflict. Occupationally, the development of software engineering has provoked changes within the computing and engineering fields more broadly. The emergence and redefinition of this area is encouraging practitioners in each field to consider anew the nature of their work, their expertise, and their relationships with others in the field.

This paper explores the rise of software engineering in the United States, the United Kingdom and Canada, focussing particularly on the period between the early 1980s and the present. Several research questions are central:

- i) what is software engineering?
- ii) how is it related to computing and engineering occupations and professions?
- iii) why and how are software engineers seeking greater professional status?
- iv) why has software engineering emerged in a context of consensus and co-operation in some nations, like the U.K., and yet been the source of conflict in Canada?

All of these questions are inter-related. Answering them requires a look at the history of the professional activities, concerns, and interactions of the principal computing organizations in each of the three nations.

Sources and Methods

To explore the emergence of software engineering (SWE) in Canada, the UK and the US between the early 1980s and the present day, document analyses were conducted. The principal source of

information on trends in computing occupations and organizations, and the rise of SWE, are the publications of the most dominant IT professional organizations in each nation. In Canada, the principal IT organization across the period has been the Canadian Information Processing Society (CIPS). Its journal publications, position papers, and news releases provide a great deal of information on professional issues relating to IT employment in Canada. CIPS has a somewhat checkered publication history, producing four different journals successively between the 1960s and early 1990s. For more recent information, from CIPS, I had to rely on papers and news releases available from the organization's web-site (www.cips.ca).ⁱ Additional information on software engineering in Canada was found in the web publications of the Canadian Council of Professional Engineers (CCPE), the Professional Engineers of Ontario (PEO), and the Canadian Association of Computer Science (CACs). For information on professional activities in the United States, the primary source used was the *Communications of the ACM*, a journal published by the American-based Association for Computing Machinery (ACM), since the late 1950s. While this journal contains articles relevant to computing practice, historically, it has also been a primary medium through which ACM presidents and other leaders communicated to the membership, as well as a key forum for the discussion and debate of professional issues. For this present analysis articles published between 1980 and 2002 on professional issues in computing and software engineering were utilized. Other ACM publications related to software engineering also informed the analysis. Information about computing in the United Kingdom was obtained from the publications of the British Computer

Society, especially their *Computer Bulletin* (1980- 1991), and *Software Engineering Journal* (published jointly with the Institute of Electrical Engineers (IEE) between 1986 and 1996).ⁱⁱ For all three countries, additional publications including professional legislation, publications on software engineering ethics and knowledge, and organizational web-site reports further enhanced the analysis.

While they provide invaluable data on professional activities, these sources are limited. Most importantly, these publications are somewhat representative of the professional projects being pursued in all three nations, but they are not entirely inclusive. There were numerous IT-related organizations active in the 1970s, 1980s and 1990s, especially in Canada and the United States, whose publications were not examined. Moreover, the American Institute of Electrical and Electronic Engineers (IEEE) Computer society, and the British IEE have also been centrally involved in computing professional projects, and especially the emergence of software engineering, and their publications have not been examined as closely as those of their sister computing societies. Nevertheless, the major professional initiatives pursued by these organizations were recorded and discussed in the publications of the BCS, ACM, and CIPS. Limiting the investigation to articles published by these three organizations, yielded hundreds of articles on the topic. The purpose of this research was to grasp the principal issues shaping the emergence of software engineering in the three nations under consideration, and the sources used were sufficient for this research endeavour.

A Brief History of Computing and Software

To understand the origins and history of software engineering, one needs to situate it in the context in which it arose, and especially consider developments in the industry, discipline and profession of computing. The emergence of software engineering (SWE) is the product of a confluence of trends, which include the rapid expansion of the software industry from the late 1960s on, professional developments within computing, and efforts to establish computer science as a scientific academic discipline. In this section I explore professional developments in computing that predated 'software engineering,' but nonetheless shaped it. I also detail changes in software practice that encouraged the growth of SWE as an occupation and profession. In all three nations under consideration, it was the combination of professional ambitions, and the increasing prevalence and importance of software that contributed to the birth of software engineering.

The profession and discipline of computing

Computing as a field and discipline emerged with the invention of electronic digital computers in the years following World War II, and the establishment of a computing industry in the 1950s and early 1960s. The first computing organizations arose in the United States in the late 1940s; engineers and scientists constructing and utilizing computers were eager to interact with each other to discuss their work (Adams, 2004a). The Association for Computing Machinery (ACM) was formed in 1947, and computer groups within electric and radio engineering organizations were established around the same time.ⁱⁱⁱ By the late 1960s, these

organizations had expanded both their membership and their mandate – becoming more interested in advancing computing as a profession and a discipline (Orden, 1967, p. 145; Adams, 2004a).

'Professional' organizations in computing emerged later in the United Kingdom and in Canada, perhaps reflecting the slower development of a computing industry in these nations. The British Computer Society (BCS) was established in the late 1950s in a conscious effort to bring together computing academics, scientists and business users of computers. In the late 1960s the BCS decided to become a professional group, and in the ensuing years worked to establish credentials and membership criteria that would define BCS members as professional computing workers. In Canada, the Computer and Data Processing Society of Canada was formed in the late 1950s. By the late 1960s, this organization had changed its name to the Canadian Information Processing Society (CIPS), and like its counterpart organizations, was formulating plans to establish itself as a 'professional' organization by establishing computing credentials and restricted membership. These latter plans did not come to fruition, and CIPS remained less focussed on professional issues, until the 1980s. Numerous other organizations representing computing-related workers appeared in the 1960s and after in these countries, with varying levels of commitment to professional development in the field of computing (see Adams, 2004, and Ensmenger, 2001 for a further discussion of these organizations).

While computing organizations were, by and large, interested in enhancing professionalism, they differed in the extent of their commitment to this project, and the

way in which they pursued it. The BCS was the most explicitly “professional” association: in 1968, academic and experience requirements had to be met to obtain membership in the organization. Membership itself became a credential, with the ‘MBCS’ (member of the BCS) used to signify competence and knowledge in the computing field. The BCS established examinations in core areas of the computing discipline, as tests of competence, and later endeavoured to accredit university computer education programs, in order to determine which were sufficient to meet its high standards. Graduates from accredited institutions could have some of the BCS examinations waived. The BCS became committed to enhancing the professional standing of its members and raising the status of its profession. In 1984 it was granted a royal charter conferring on it a limited number of professional privileges and securing its place as the pre-eminent professional society of computing workers in the U.K., and arguably, amongst English-speaking countries more generally.

In contrast, while the ACM was committed to enhancing the professionalism of its members – and did restrict membership to those with a university degree – it was less focussed on credentials, than on defining computer science as a legitimate scientific discipline and university subject (Adams, 2004a). Through the 1960s, in particular, its work on developing a computer science curriculum was highly influential, not only to computer science education, but also to the definition of the discipline itself. Other American-based organizations, especially the Data Processing Management Association (DPMA), placed greater emphasis on establishing credentials and examinations for data processing and computer workers.

As noted, the Canadian organization CIPS had a membership that was fundamentally ambivalent about professional issues: a large minority of members were eager to establish qualifications and credentials for computer-related workers, yet the rest were doubtful. Many Canadians interested in credentials, pursued those established by the DPMA.^{iv} However, certification from the DPMA and its successor the ICCP has never succeeded in becoming a definitive indicator of competence and skill in the computing field.

There are many reasons why efforts to define computing as a ‘professional’ endeavour have had only limited success. These reasons include the great variety of workers in the computing and IT fields, and the variability of their education and training backgrounds, rapid change in the IT field, and practitioner ambivalence (Adams, 2004a). However, one of the principal barriers to professionalization in this field is the fact that there is no single “computing” occupation to professionalize; neither is there a clearly delineated set of skills or knowledge base. In essence, professionalization efforts have been geared towards raising the status of a field of endeavour – a cluster of occupations – rather than a single occupation with a defined scope of practice. This is in direct contrast the majority of professions which are clearly based on single occupations with a relatively clear scope of practice and body of knowledge (Adams, 2004a). There was no clear sense of who, precisely, was a computing professional in the publications of computing organizations in the 1960s and 1970s. They referred somewhat vaguely to a “computing profession” or “information processing profession” (for example, Sidlo, 1961; Carlson, 1970; Willmott, 1975; Givens, 1967). When a specific occupation was mentioned, it was most often the

“professional programmer” who was singled out, particularly in the UK (Barron, 1975, Orden, 1967). No specific knowledge base for these professionals was evident, although members of the DPMA, BCS, and ACM computer science curriculum committees did endeavour to delineate one. Despite their efforts, the computing body of knowledge remained vague enough that in 1969, CIPS described it only by asserting that “the computer is at the heart of it” as was the “method of using it [the computer], i.e. programming” (1969, p. 8).

By the 1970s, rapid occupational expansion in the computing and data processing field, and a corresponding influx of people who had little formal computer science education, ensured that professional organizations remained committed to both establishing professionalism and finding the means to distinguish the skilled from the less skilled. However, these were difficult tasks. The BCS was most successful in establishing its own professional credential as a mark of competence – although there were many in the computing field who were not BCS members. In Canada and the U.S., computer science degrees appeared to signify competence in the field, but few workers had taken such training, and many contested whether such education was actually valuable in computing employment (Adams, 2004a). The DPMA and ICCP certificates were only moderately successful in demarcating the skilled and unskilled (Ensmenger, 2001). Despite their limited success, professional leaders in each country remained committed to professionalism, and this commitment was renewed in the 1980s in the context of rising social concern over the effectiveness of software practice and production.

The Expansion of the Software Industry and the “Software Crisis”

While historians date the advent of the modern computer to the 1940s, and the rise of the computer industry to the 1950s, the software industry is said to be a product of the late 1960s (Johnson, 1998). In the 1950s and 1960s, dominant companies like IBM bundled software with the hardware that they sold. Accompanying both the software and the hardware was a strong customer service commitment. There was little market demand for additional software, and IBM was willing to develop software upon customer request (Johnson, 1998). Hence, although during the 1960s a few significant software firms arose, the industry remained a small one until 1969 when IBM made the decision to unbundle software. The industry expanded rapidly in the following decades.

Concerns about software quality arose along with the rapid expansion of the software industry, and the spread of computers to more areas of social life. These concerns were evident in the 1970s (Hoare, 1975; Denning, 1980), but increased in the 1980s, such that many spoke of a “software crisis” (Chapman, 1990; Shore, 1988). By and large, this was a crisis of quality. People were concerned that the software constructed failed more often than it should, and that it tended to be more costly than predicted (Chapman, 1990; Denning, 1980, 1990; Hoare, 1975; Shore, 1988). Customer satisfaction was said to be quite low. Concerns within the industry were partly driven by increasingly vocal complaints outside of it. Computer failures that provoked stoppages, and mistakes that affected service provision in a variety of industries, were invariably blamed on poor software (Chapman, 1990; Denning, 1990; Kocher, 1989). Hacking and computer

sabotage were similarly seen to be made possible through software inadequacies, such as 'trapdoors' and poor security (McIlroy, 1990).

To a public already ambivalent about the spread of computers, high profile cases of security breaches, computer fraud, software failure, and computer viruses, added fuel to the fire (Chapman, 1990; Kocher, 1989). The U.S. federal government published a report blaming software inadequacies for the failure of some high-profile, high-budget defence technologies (Chapman, 1990). Media reports of instances where faulty software design contributed to the malfunctioning of health care technology and therein patient death, heightened concern over the social consequences of faulty software (Collins et al., 1994). While some computer specialists held that such criticism against the software industry was often unfair – for example, arguing that the defence technologies were unfeasible and doomed to failure from the beginning, regardless of their software – they themselves frequently acknowledged that problems did exist (Chapman, 1990). The application of software to what are now commonly called “safety-critical” areas, including health care and defence technologies where a software malfunction can have dire consequences, has heightened concern over the state of the software industry. Overall, high profile cases of software failures, combined with the high cost of software maintenance and production, suggested to many that the field of software development was in need of radical change.

Professionalism and Software Practice

Through the 1980s and 1990s, writers proposed many different solutions to the

problems that appeared evident in software production. Their attention tended to focus on the education of computing workers. Various contributors called for more practical training, more software-related training, more math in the curricula, and so on. Associated with these proposed changes was a broader one: improvement in software practice was increasingly said to require the adoption of engineering principles and techniques (Denning, 1990; Thomas, 1987; Thornley, 1990; Wagner, 1990).

The argument was probably first advanced in the 1970s. For instance, in the BCS *Computer Bulletin*, Hoare (1975, p. 6) argued that software developers and programmers had a lot to learn from engineers who based their designs on “sound mathematical theories and computational techniques,” and generated products at a predicted cost and on-time, with “few design failures.” Through the 1980s and into the 1990s, such comments became more common-place. What was lacking in current software practice, it was asserted, was an attention to engineering design principles. This argument may have particularly resonated with those leaders in the ACM and BCS who had long maintained that computer science was, in essence an engineering profession (Denning, 1987; Hoare, 1981; Oettinger, 1967). While it is unlikely that all computing leaders agreed with this depiction, the view that software practice could benefit from engineering principles became commonplace.

It was with the increased emphasis on bringing engineering principles into software practice, that the term ‘software engineering’ became more common. The greater use of this term seems to reflect the confluence of two trends: the ongoing professional ambitions of computing leaders

and the 'software crisis.' As a term and practice method, software engineering both promises to improve software practice, and conveys a professionalism that terms like programmer and software developer do not.

The Emergence of Software Engineering:

The term 'software engineering' reportedly dates from a 1968 NATO conference in Germany. Those who organized the conference chose it deliberately to raise debate and spark controversy, as well as to propose that software development and engineering should be brought together (Thomas, 1987; Tomayko, 1998). However the term and the discipline did not become common for another decade at least. This section traces the emergence of software engineering in Canada, the U.S. and the U.K., from the 1980s on. The development of software engineering (SWE) varies across the three nations, although many of the key underlying issues and trends remain the same. The following analysis illustrates that perceptions of SWE have changed over time. Moreover, it highlights the role of inter-professional relations and professional development in shaping the emergence of SWE – as a process, a field, and an occupation.

Software Engineering in the U.S.

During the 1980s, there was little mention of software engineering in the pages of the *Communications of the ACM*. Indeed, before 1989, only four pieces were published with the terms 'software engineering' in their title: one editorial, one article and two letters to the editor. In the next decade, fully 27 articles were published explicitly on software engineering, while many others made reference to SWE.

Further demonstrating the growing interest in SWE, ACM began publishing a separate journal, *ACM Transactions on Software Engineering and Methodology* in 1992. Thus, it was not until the early 1990s that software engineering as both a set of tasks and a distinct occupation really became a key focus within the ACM. Nonetheless, trends in software and computer science education through the 1980s led to calls for a change in the nature of software practice and education that foreshadowed the later rise of SWE.

What is clear during the 1980s and 1990s in the ACM literature is that there was substantial professional and public concern surrounding software quality. The presence of a software 'crisis' was taken as one sign amongst many that computer science was a discipline in need of change (Chapman, 1990; Denning, 1984). Through the 1980s and early 1990s, three types of solutions were commonly proposed. First, many called for changes to the computer science curriculum. Much of the discussion in this literature centred around the need for more math in Computer Science and SWE programs (Berztiss, 1987; Findler, 1985; Gries, 1991; McCracken, 1987; Ralston, 1984; Steen, 1985). For many, more math training would provide students with fundamental skills needed to keep up with changes to technology and computing practice. Others held that the curriculum needed to be brought up-to-date, especially with respect to software practice (Frailey, 1988; Herbst, 1985; Parikh, 1985). During this era, curricula were revised to stress software engineering methods (Koffman et al., 1984, p. 998; 1985; McCracken, 1987).

A second and related concern, was to bring greater clarity to computer science and computer science education by specifying

the nature of the discipline and the curriculum that should support it, (and formalizing these standards through accreditation). In the early 1980s, computer science was described as an ambiguous discipline whose boundaries and focus were not clearly delineated (Denning, 1984; Tartar et al., 1985). In an effort to bring greater clarity to the discipline, and by extension computing practice, ACM established a Task Force on the Core of Computer Science. This task force published a report in 1989 that established paradigms for the discipline of computing, defined as including both “computer science and computer engineering, because ... no fundamental difference exists between the two fields in the core material” (Denning et al., 1989). Here computing was said to be a blend of mathematics, science and engineering. The paradigms inherent in computing included *theory*, that stemmed from mathematics, *abstraction* rooted in science and the scientific method, and *design*, “the bedrock of engineering” (Denning et al., 1989, p. 10). This definition of the discipline is consistent with earlier curricular work (especially that of the 1960s), that defined computer science as lying at the crossroads between engineering and science (especially math) (ACM C³S, 1968; Denning, 1987). Education was said to require a balance of all three of these elements, and a model curriculum was constructed in accordance (Denning et al., 1991). Computer science education was also said to require a greater emphasis on professional ethics and responsibility. The increased emphasis on engineering, and bringing engineering and science together within computer science, was consistent with the emergence of software engineering as an aspect of computer science, and software engineering was given a significant place in the model curriculum.^v

A third proposed solution that emerged later in the decade, involved the licensing and regulation of software practitioners (Chapman, 1990; Kocher, 1989; Neumann, 1991; Shore, 1988). For some, high profile cases of software failure and poor software design, indicated that regulation of software developers was necessary, to ensure software quality (Chapman, 1990). Some feared that if software practitioners did not mobilize to regulate themselves as professionals (through organization, codes of ethics and higher standards of qualification) then the government might decide to regulate them itself, in a restrictive fashion (Bourque et al., 1998; Kocher, 1989; Shore, 1988). While many in the computing field had long been sceptical about professional regulation, advocates contended that regulation would protect software workers from the pressure to compromise their standards in order to meet employer deadlines, and further that it would weed out the lesser-skilled, ineffectual and unethical (Chapman, 1990; Kocher, 1989; Neumann, 1991; Shore, 1988). Nevertheless, others remained sceptical, preferring an individual responsibility model of professionalism that emphasized ethical practice and professional responsibility (Heinlein, 1991; Weaver, 1989).^{vi}

During the 1990s, there was more of an emphasis on software engineering. Writers increasingly advocated the adoption of an ‘engineering’ focus amongst software developers and computer workers more generally (Denning, 1990; Nelson, 1987; Thornley, 1990; Wagner, 1990). ‘Software engineering’ came to be seen as a principal occupation within the computing field. Some individuals complained that while the use of the term was becoming more common, it was largely undeserved – engineering principles had not been applied

fully enough for software workers to use the term legitimately (Gordon, 1990; Lewis, 1989; Loeser, 1989). Others saw the use of the term as a somewhat pathetic attempt by programmers and other computer workers to increase their respectability and status (Rickert, 1989). However, despite the occasional dissenter, most commentators wrote to advocate a greater engineering focus within software practice, and embraced the term 'software engineering.'

Although the SWE term was widely used in the late 1980s and early 1990s, it was rarely defined. Quite often the term was used almost interchangeably with 'programming' (for instance, Wagner, 1990; Wilkes, 1991). While this is a very narrow use of the term, it is a significant one, given that in decades past many believed 'programming' to be the central, professional computing occupation. Software engineering began to replace programming as the computing 'profession' most mentioned in the literature. By the 1990s, programming's status as a professional activity was being called into question,^{vii} but the term 'software engineering' clearly connoted a set of tasks requiring skill, decision-making, and expertise. The SWE term was also used to refer to a broad but unclear set of computing tasks; that is, "as an umbrella term of ...breadth (and, at present, vagueness)" (McCracken, 1987, p. 4). Indeed, at times SWE appears to represent the bulk of computing practice, as when then-President of the ACM Bryan Kocher (1990) suggested in 1990 that the ACM change its name to something more professional sounding, like the Society for Software Engineers. The ambiguity inherent in the term, combined with its growing importance within the field, and frequent criticisms that it was ill-specified and lacked effective standards (Denning, 1990; Ramaley, 1990), prompted professional activity.

In 1993, the ACM joined with the IEEE Computer Society to form a "Joint Steering Committee for the Establishment of Software Engineering as a Profession." This committee focussed on three key initiatives: (1) establishing a code of ethics for software engineers; (2) establishing curricula and accreditation standards for software engineering education programs; (3) the ambitious project of defining a core software engineering body of knowledge, or SWEBOK. In 1998, the joint steering committee was superseded by a joint "Software Engineering Co-ordinating Committee" (SWECC) to "foster the evolution of software engineering as a professional computing discipline" (quoted in Notkin et al., 2000, p. 2). As the activities of these joint committees have been central to the emergence of SWE in the U.S., they will be reviewed in some detail.

A detailed code of ethics for software practice was produced by a joint working group of ACM and IEEE-CS members, and published in 1997. Codes of ethics have been traditionally seen as an important step along the path towards professionalism. The SWE code emphasized professional responsibilities to the public and profession, and also identified 'good practices' to guide work behaviour (Gotterbarn et al., 1997). The ACM and the IEEE-CS, also worked out arrangements for the accreditation of software engineering university programs. At first, accreditation of SWE programs was split by engineering and computer science accreditation boards. The Computer Science Accreditation Board (CSAB) had primary responsibility for accrediting software engineering programs in computer science departments, in co-operation with the engineering board (ABET). This latter body had primary responsibility for accrediting SWE programs in engineering settings,

albeit with the “co-operation” of the CSAB. More recently, the CSAB has become a “participating body of ABET”, and it plays a lead role in “the accreditation of programs in computer science, information systems, and software engineering.” (CSAB, 2004).

Most striking of the joint professional initiatives has been the mammoth, international project aimed at identifying a Software Engineering Body of Knowledge (SWEBOK). Committee members believe that reaching a consensus on a core body of knowledge is crucial to the development of SWE as a profession (Abran et al., 2001). Thus far, the project has created two SWEBOK documents: an initial ‘straw man’ version (Bourque et al., 1998) and a modified and more complete ‘stone man’ version (Abran et al., 2001). The final version was released in 2004 (Abran et al., 2004). The SWEBOK project attempts to identify precisely what SWE is, and what it entails, and it delineates the relationship of software engineering to computer science and engineering more broadly. It represents a remarkable, if only somewhat successful, attempt by two related professional organizations, to define a shared occupation and scope of practice, in a mutually satisfying manner. A closer look at the SWEBOK provides an example of professions co-operating to hold a professional jurisdiction in tandem, and it helps to identify those factors that encourage and discourage inter-professional co-operation.

As its basis, the SWEBOK draws on a definition of SWE produced by the IEEE-CS. Here software engineering is seen to consist of “the application [and investigation] of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to

software” (Bourque et al., 1988, p. 8). Software engineering is then, a method or approach to software practice, but one that can be applied to a variety of tasks in the software field. The SWEBOK also carefully demarcates SWE from computer science, arguing that computer science is the founding discipline of SWE but remains separate from it. That is, computer science is to software engineering, what chemistry is to chemical engineering: it provides a core basis of scientific principles which engineers then utilize and apply to construct useful and efficient products (Bourque et al., 1998, p. 11). More specifically, it is argued, “the fundamental goals of computer science and software engineering differ” (Bourque et al., 1998, p. 11). Computer science seeks to extend knowledge in computing, while SWE aims to apply “this knowledge to building software” (Bourque et al., 1998, p. 12). Thus, while software engineering has stemmed from computer science and is linked to it, it is rightfully an engineering discipline and profession (Abran et al., 2001; Bourque et al., 1998). The 2001 version of the SWEBOK elaborates on these earlier definitions and knowledge outlines to identify ten key knowledge areas^{viii} and additional subareas, and it draws on the expertise of 500 software professionals in 41 countries around the world.

This construction of computer science as the science or theory upon which software engineering practice is based, has provoked some controversy. As noted, professional leaders in computing have frequently maintained that computing is an engineering discipline: it draws on science to produce effective information technology products. In some sense, the definition of SWE is a culmination of earlier arguments. However, computer science has never been structured as a purely ‘theoretical’ discipline, and some

computer scientists, believing ‘practice’ (even engineering practice) is essential to their discipline, resent this classification.^{ix}

The production of the SWEBOK, the SWE code of ethics and SWE curricula, by a joint committee of engineers and computer scientists affiliated with the ACM and IEEE-CS provides a striking example of inter-professional co-operation. These two organizations have worked together to define a discipline and profession that is allied to both, and lies within the jurisdictions each claims. This co-operation seems to have been facilitated by the history of the two organizations, and their relationship with each other. Both organizations were born in the late 1940s with the invention of the very first electronic computers. While ostensibly, the IEEE-CS may be seen as an organization for ‘engineers’ interested in computing, and the ACM as a non-engineering organization, in fact the division between members is not tightly drawn. The IEEE-CS has non-engineering members, and the ACM has many members who are engineers. Both the ACM and IEEE-CS sought to represent the field of computing, more generally, and, recognizing this shared mandate, they have co-operated a great deal throughout their history. They have hosted joint conferences, co-founded other organizations (like AFIPS and the ICCP)^x, and they have worked together in the past on curricular and educational issues. There was even some discussion of merging the two organizations (Bechtolsheim, 1985; Grosch, 1987). Although they have remained separate, the two have a long history of joint ventures and co-operation. The view of some ACM leaders, that computing was an engineering discipline, likely also facilitated ties with the IEEE-CS. In essence, then, the collaboration aimed at establishing SWE as

a profession was merely an extension of the joint ventures undertaken by the ACM and IEEE-CS in the past. Software engineering was deemed of interest to both groups, and they decided to work together to define and professionalize it.

Increasingly, however, the interests of ACM and IEEE-CS with respect to SWE are diverging, and the extent of collaboration between the two groups has diminished. ACM has pulled out of SWECC – the joint committee aimed at creating a SWE profession. This action may be a harbinger of future professional conflict; at the very least it signals a divergence of interests between the two groups. By and large, ACM’s withdrawal from the software engineering project seems to have been spurred by disagreement over licensing. In June of 1998, the Texas Board of Engineers decided to license software engineers (Bagert, 2002). They sought help from the ACM and IEEE-CS “in defining performance criteria for software engineering licensing exams to be administered in Texas” (ACM 2000, p. 3). SWECC believed its SWEBOK project could meet the state’s needs and those of others interested in licensing. However, ACM leaders did not like this change in focus and many began to fear that “the primary purpose of SWECC and its SWEBOK project was to support the licensing efforts of software engineers” as professional engineers (ibid). ACM established a committee in 1999 to explore “whether licensing software engineers was ‘in the best interests of the field of computing and the public’” (Knight & Leveson, 2001, p. 1).

After much investigation this committee ruled strongly against licensing software engineers, especially under the current

system of engineering licensure, arguing that it would do little to ensure that only skilled people worked on safety-critical software. The ACM committee identified many concerns with licensing, but three, in particular, stand out (Knight & Leveson, 2001, 2002). First, state licensing of engineers tends to be voluntary, and uncommon for engineers employed by others – as most software engineers are; hence state licensing would likely not ensure that people producing safety-critical software were licensed and capable. Second, the professional engineering license does not distinguish between specialties. As a result, there is little, except professional responsibility, to prevent a licensed engineer with little software training, from working as a software engineer. Third, for licensing, states require specific education: most notably, engineers must graduate from an engineering-accredited institution. However, currently most software engineers have computer science backgrounds, and hence would be ineligible.^{xi} Moreover, most states require examinations for engineers, that test knowledge in engineering subjects that few computer-science trained people have had exposure too.^{xii} Because of their reservations and their belief that SWECC and SWEBOK were both too committed to licensing, and doing little to advance ACM's key interest in improving software engineering in safety-critical systems, ACM decided to withdraw from SWECC and not support the SWEBOK endeavour (ACM, 2000). At the same time they reaffirmed their commitment to “continue to work closely with the IEEE Computer Society on projects that further the evolution of software engineering as a *professional computing discipline*” (ACM 2000, p. 2). This latter phrasing may be significant given the IEEE-CS's commitment to making software engineering a “legitimate

engineering discipline” (Abran 2001, p. 1; Bourque et al., 1998, p. 2).

Thus, while the two societies seemed to have had converging interests in the 1990s, by the opening years of this decade, they appear to be diverging. ACM has withdrawn from IEEE-CS's efforts to professionalize and licence software engineers. Moreover, while during the 1990s, there was a sense in the literature that software engineering was the principal and perhaps most important computing occupation, this is no longer the case. ACM has restated its “deep commitment to the professionalization of all parts of the IT field, not just software engineering” (ACM, 2000). Towards this end, it has launched a new initiative – The Information Technology Profession Initiative – to encourage the development of an “IT profession.” Furthermore, the idea that engineering is the solution to the software crisis has recently been called into question. For instance, de Champeaux (2002, p. 102) maintains that “the concept of software *engineering* has not been helpful, and plausibly has deeply exacerbated the software crisis” (italics in original). All of these changes may portend greater inter-professional conflict in future years.

SWE in the UK

Efforts to define and construct the field of software engineering began in earnest sooner in the United Kingdom, than they did in the United States and Canada. While the discussion of SWE figures only occasionally in the pages of the *CIPS Review* and the *Communications of the ACM* during the 1980s, it is the subject of numerous articles in the *BCS Computer Bulletin*, and a separate BCS/IEE journal, *Software Engineering*. Similarly, SWE was an area

of professional activity in Britain by the mid-1980s, roughly ten years before it received similar attention amongst computer professionals and engineers in North America. A look at the BCS journals published in the 1980s and 1990s reveals a computing profession eager to define SWE, and to work with engineers to advance and professionalize the field. While the success of inter-professional co-operation in the United States has been somewhat limited, engineers and computing professionals have mostly co-operated in the U.K., and the links between the two professions have become even closer since the 1980s, as the BCS has become an 'engineering institution'.

Although SWE figures prominently in the publications of the BCS in the 1980s, it is clear that, as in North America, there was a great deal of ambiguity surrounding exactly what precisely software engineering was. As Thomas (1987, p. 3) notes, during the 1970s and 1980s the term software engineering was "misused, in an attempt to add a false veneer of professionalism to the worthy art of programming." Thus, at times, like in the U.S., software engineering appeared to refer almost exclusively to programming. That programming was unworthy of the designation of software engineering, because it lacked a strong engineering basis, was the focus of a number of articles (Hoare, 1981; Thomas, 1987). Nonetheless, the term was used to argue that programming and software production more generally should "be based on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering" (Hoare 1981; Naur & Randell, 1968 in Thomas 1987, 174). Increasingly, the term was used to refer to the processes involved in the production of software broadly, including the design, construction and maintenance

phases. The term was almost all-encompassing. Indeed, it was argued that, broadly speaking, "most degree courses in computing are, or should be, courses in software engineering," and the majority of computer science students will come to work "initially at least, on the production or maintenance of software" (Bott, 1989, p. 174). Software engineering was said to cover "the whole process of software development, starting when the requirements for a product are first identified and ending, many years later when the software is finally allowed to expire" (Brown, 1985).

More technical definitions emerged, especially in the 1990s. For instance, Andriole and Freeman (1993, p. 166) characterize SWE as "the systematic application of methods, tools, and knowledge to achieve stated technical, economic, and human objectives for a software-intensive system." Although such definitions are more specific, they still apply to virtually all aspects of software development. Building on these broad definitions, more and more courses on SWE were offered in university settings (Cheetham et al., 1988), and some programs decided "to make software engineering the core philosophy around which the undergraduate [computer science] degree would be developed" (Winder, Cole & Eastal 1987, p. 133). With the expansion of the definition of SWE to encompass most aspects of software production and maintenance, some in the field more carefully demarcated SWE from programming. Notably, Lehmann (1991, p. 245) argued that "the difference between the role of programmers and software engineers is fundamental, although sometimes, particularly in small organisations,

responsibility for both areas may be vested in single individuals.” Thus, as in the United States, at times ‘software engineering’ was used as a fancy term for ‘programmer’ by many, but by the early 1990s, the term was more commonly being applied to virtually all software-related tasks in computer science and computer practice.

A number of factors encouraged the emergence of SWE in the 1980s in the UK, including concern over software quality, and engineers growing interest in the field. Also important in accelerating its development was social concern over the competitiveness of British companies in the software sector. In the mid-1980s the Office Advisory Council for Applied Research and Development (ACARD) established a working group to investigate the status of the UK software industry. The working group published a report in June of 1986 that raised the alarm over the state of the industry and software engineering in the UK. The report illustrated that the UK software industry was “small and fragmented,” with a very small share of the world software market (Crinean & Baguley, 1986, p. 37; Newton, 1986). Moreover, “the major UK software companies [were] much smaller than their major overseas competitors,” and must, then, have a good strategy to compete internationally (Newton, 1986, p. 24). Ultimately, the report concluded, “if the UK industry is to maintain and improve its competitive position it must also invest heavily in software engineering” (Crinean & Baguley, 1986, p. 35).

The working group formulated recommendations for the sector, including public purchasing to support the industry, encouraging industry to provide more training for those involved in making and using software, greater use of software, and

better marketing (Newton, 1986). The software industry seems to have supported this plan, and indeed, it blended with their own initiative to encourage good software practice (Crinean & Baguley, 1986). The working report also recommended that the IEE (Institute for Electrical Engineers) continue its involvement in the field of software engineering and explore certification for software engineers. The BCS had already been working towards professional certification in SWE, and was eager to be part of any such initiative (Newton, 1986a, p. 25). There was a sense in the British literature that the software industry was not competitive, partly due to problems in product quality and productivity precipitated by “the largely manual and scientifically unsophisticated processes used” (Hoffnagle, 1991, IPC). Bringing engineering principles into software practice was seen as the solution to the problem (Hoffnagle, 1991; Thomas, 1987; Conway et al. 1989, p. 248).

The publication of the ACARD’s working report on the software industry gave further impetus to the joint endeavours of BCS and IEE already underway. The BCS had forged ties with the Institute for Electrical Engineers from early on in its history, for example organizing a joint meeting in 1959 (BCS Annual Report, 1960). The two organizations also worked together to organize Joint Computer Conferences in the 1960s (Goodwin, 1964). Moreover, some of the early BCS members and presidents were engineers.^{xiii} During the 1970s, the organizations merged their libraries, and since 1977, BCS library holdings were housed and maintained by the IEE (Lance, 1982). The two organizations worked more closely together during the 1980s as they both sought to professionalize the field of software engineering. At this time, the two

took steps to bring their separate professional projects closer together.

In 1982, the IEE decided to enlarge its scope by admitting software engineers as members. According to one contemporary, this decision was motivated by members' "professional involvement with software, and ... the recognition of the need to raise professional standards in software engineering" (Pyle, 1986, p. 66). The IEE then had to work on specifying criteria for membership and professional standards. This was not easy in a field characterized by ambiguity surrounding precisely what software engineering was, and no clear practice standards. However, the IEE did not attempt to establish these professional standards entirely on its own, but showed a willingness to work with the BCS in doing so. These endeavours were not unanimously supported. Some members of the IEE felt that the organization should not be involved in software engineering (Pyle, 1986, p. 68). Others seem to have felt that the IEE should not be co-operating with the BCS. However, the dominant perception seemed to be that "the IEE and BCS (being the two institutions currently most concerned with true software engineering) should make a start together" to establish professional standards in SWE (Pyle, 1986, p. 68). Although the two organizations may have different accreditation criteria and "somewhat different interests," they shared an interest in SWE and their co-operation was worthwhile (Conway et al., 1989; Pyle, 1986, p. 68). During the 1980s, the two groups worked together by jointly publishing the Software Engineering Journal, co-hosting annual conferences on SWE, and formulating software engineering standards.^{xiv}

Co-operation between the BCS and IEE was likely facilitated by certain similarities in the nature of the two organizations, and the professional groups they represented, as well as BCS's efforts to establish itself as an 'engineering institution.' While 'engineering' in many countries, including Canada, is a professional occupation with an established licensure system and a protected title, this is not the case in the UK. Rather the term 'engineer' is applied more loosely to "anyone whose work relates to engineering – particularly manufacture or maintenance." (EC, (UK) 2003). The broadness of this term has limited professional advance, as "statutory recognition of professions in the UK requires the particular functions being controlled by statute to be defined ...[and] it has not proved possible to isolate the functions that engineers undertake" (ibid). Nonetheless, engineers in the UK are governed by a Royal Charter, passed in 1981, and revised in 2002. This charter established categories of registration for engineers, and the designations of "Chartered Engineer," "Incorporated Engineer," and "Engineering Technician." People seeking to acquire these titles, must meet certain standards set by an umbrella organization recognized in the Charter, the Engineering Council (EC), and be a member of a 'licensed' engineering organization, such as the IEE.^{xv}

In many ways, the British Computer Society is very similar to IEE and other engineering organizations in the UK. Notably, it too represents a professional group that is not protected by licensing or a statute, but is incorporated by a Royal charter, passed in 1984, which recognizes registration of computing professionals, by the BCS, and the professional credential "MBCS," held by

those who qualify as full members in the British Computer Society. To obtain membership in the BCS, interested parties must meet certain criteria set by the organization, which, like those of the IEE, require specific levels of professional competence as demonstrated by education qualifications, examinations, and work experience. Although the IEE is a much older organization than the BCS – the former organization was established in 1871 – in terms of professional status, the two are similar. These similarities likely fostered collaboration. Moreover, their commonalities were enhanced in the early 1980s as the BCS sought status as a licensed member organization of the EC. After successful negotiations and discussions between the EC and BCS – reportedly the EC was “highly impressed with the Society’s professionalism” – the EC accepted the BCS as a “nominating body” in 1985 (Newton, 1986b, p. 23). From this date, the BCS could nominate members for registration as chartered and technician engineers, with the chartered level being “roughly equivalent” to professional membership in the BCS, and the technician level “roughly equivalent” to the associate membership level (Newton, 1986b, p. 23). The EC would then “vet” the nominations and authorize those they saw as acceptable. In 1989 the BCS became a full chartered engineering institution and in 1996 it became licensed by the Engineering Council (with greater independence and involvement in granting engineering certification to members). These important steps were predicated on the Engineering Council’s decision to broaden “its definition of ‘engineering’ to include ‘computing’” without any corresponding change in the BCS position (Newton, 1986b, p. 23). By the late 1980s, the majority of BCS members were eligible for registration as

chartered engineers (CEng) (Newton, 1986b; Oakley, 1988), and many took advantage and acquired the C Eng designation (Low, 1988).

Thus, while the BCS was not nearly as established as the IEE, nor as firmly part of the engineering profession, it was increasingly recognized as a body of professionals allied with engineering, and this recognition likely facilitated interaction between the organizations. Hence, SWE licensing and accreditation, which have been thorny professional issues in other countries like the U.S. and especially Canada, have been less controversial in the UK. Engineers are not licensed, but registered, and registered computer professionals (including software engineers) can gain recognition as chartered engineers. Although they are not identical, professional standards are similar enough in the two organizations to facilitate inter-professional co-operation. Engineers and computing professionals have managed to divide accreditation duties, despite differing accreditation standards: at first, engineers accredited SWE programs housed in engineering, and the BCS accredited computer science SWE programs (Conway et al., 1989). Now many programs are jointly accredited. The lack of formal licensure appears to allow for flexibility in the area of educating software engineers.^{xvi}

Overall, the status of the two organizations and the professions which they represent, combined with a history of co-operation and a similar engineering ‘outlook,’ appear to have enabled inter-professional co-operation in the establishment of a software engineering profession, which ultimately crosses two disciplines: computing and engineering. Collaboration was also encouraged by the fact that each group had

something to gain from each other. As Newton (1986a) argued, “the BCS has the software engineers, and the IEE has the political awareness” to make professional progress a reality. Similarly Conway et al. (1989, p. 246) emphasized that collaboration brought together an institution with over a century of engineering expertise, with one “active in developing the earliest software education.” Thus, members of each organization recognized that they could gain by working together. Their co-operation is remarkable for its rarity. As the literature on professions demonstrates, it is not often that two related professional groups have decided to work together and share a jurisdiction. Inter-professional conflict is a more common outcome (Abbott, 1988). Such conflict is clearly illustrated in the Canadian context.

SWE in Canada

While ACM and BCS journals are replete with references to a software crisis and software engineering, very little was published on these topics in CIPS’ journals. Nevertheless, at least one CIPS leader cautioned about “the potentially disastrous effects of badly developed software” and advocated the importance of CIPS taking a leadership role in protecting the public (Foyer in Turner, 1985, p. 6). In the early 1990s, a few articles noted that satisfaction with computer systems and services was decreasing and that the public was increasingly affected by “computer errors” (Hadford, 1991, p. 11; McAuley, 1991, p. 8). For these leaders, such problems called for continuing education and greater professionalism amongst CIPS members. Overall, though, there appear to have been fewer articles devoted to documenting a software crisis, than to discussing the state of the Canadian software industry. Holding

that “Canada is uniquely poised to develop a world-class software industry” and that it had a great deal of competence in the field (Davies, 1982, p. 4), these articles questioned whether the industry could be “effective at taking advantage of the opportunity” (Potter, 1982, p. 5). There are a number of reasons why a software crisis and ‘software engineering’ as an occupation may have garnered less attention in CIPS publications. First, CIPS only regularly published a journal until 1992. The lack of attention to these issues may, in part, reflect CIPS uneven and shorter publishing record. Second, through most of this period, CIPS was engaged in a professional campaign, endeavouring to establish professional credentials for IT workers; the journal’s focus, as a result, was placed elsewhere. Thirdly, CIPS membership predominantly consisted of Information Systems Managers (49%), and programmers and analysts (25%). Hence, the journal focussed more on the industry and business aspects of computing, than on education and software practice. Whatever the reason, CIPS journals were largely silent on software issues.

The lack of Canadian-based publications on SWE notwithstanding, Canadians have been interested in the development of SWE and have made contributions in this area. However, they have done so largely through their involvement in American-based organizations like the ACM and IEEE-CS. Indeed, there may be more Canadian members in the ACM and IEEE-CS than there are in CIPS. Canadian-based computing professionals have published on SWE in the Communications of the ACM (for example, McCalla 2002; Parnas, 2002), and they have been involved in the SWEBOK project.^{xvii} These contributions do not demonstrate a distinctly Canadian viewpoint on the subject. It seems likely

that the development of SWE as a field and occupation in Canada, during the 1980s and 1990s, resembled developments in the United States. Thus, while software engineering as a set of techniques and practices allied with programming has been around for decades, software engineering as a distinct occupation is a more recent phenomenon.

In the 1970s and 1980s, 'software engineering' seems to have represented a fairly loose term applied to the tasks involved in designing and constructing software. Courses in software engineering were offered in computer science departments from the mid 1970s on. More courses in the field were added during the 1980s and 1990s, as the term became more common and the area more prominent within the computing field.^{xviii} Traditionally, Canadian computer scientists, like those in other countries "treated 'software engineering' as an area of research, not as a profession" (Parnas, 2002, p. 97). It is only in the 1990s that software engineering as a distinct occupation seems to have emerged. For instance, Canadian occupational classifications in the early 1990s did not distinguish software engineering from 'computer engineering' more broadly.^{xix} While computer science departments taught courses in software engineering, the occupation was not classified with other computer occupations like programming, but rather with engineering occupations. In Canada, as in the United States, it is only since the late 1990s that software engineering has been identified and accepted as a distinct computing-related occupation. However, while its emergence in other countries has been the result of concerted effort and co-operation, in Canada it has been the source of a great deal of controversy.

Engineers in Canada are self-regulating professionals governed by provincial legislation that protects the title of engineer – no-one but a professional engineer can call him/herself a professional engineer – and enables professional engineers to establish strict criteria for entrance into the profession. Currently professional engineers in Ontario, for instance, must graduate from an accredited university engineering program,^{xx} complete 4 years of relevant employment experience, and pass a Professional Practice examination on engineering law and ethics (Parnas, 2002; PEO, 2003). Those who complete all of these requirements are licensed as professional engineers and granted a "P. Eng" (professional engineer) designation. Given this professional legislation, anyone calling themselves a 'software engineer' who is not a licensed engineer is, technically, in violation of the law.

As part of their own renewed professional project to increase the status and public relevance of engineering in Canada, Canadian engineers have consciously decided both to expand their involvement in the IT field, in particular by laying claim to the field of software engineering, and to oppose and prosecute those illegally using the title 'engineer' (CCPE, 1997). Notably, the national body for engineers in Canada, the Canadian Council of Professional Engineers (CCPE) has resolved to incorporate "emerging fields of practice" into the "governance structure of the profession, commencing with the fields of environment and information technology" (CCPE, 1997, p. 12). Moreover, the organization is committed to ensuring "that engineering in Canada is conducted by professional engineers" (Ibid). In the past 8 years, software engineering has been a key

area of focus for Canadian engineering leaders. Although software engineering has long been associated with computer science, the official engineering policy on SWE is that it is an engineering discipline, and that no-one who is not licensed as professional engineer can use the title “software engineer” (PEO, 1999a).^{xxi} Engineers argue that they have a lot to bring to the SWE field. Prior to their involvement, the field was “populated with all manner of self-taught practitioner,” whose lack of training led to bad software and a high rate of software failure (Parnas, 1998; Rowlands, 1999, pp. 30-32). Licensing software engineers will ensure that software practice is guided by those who are more educated, more responsible and more accountable, and hence will improve the state of the industry (Parnas, 1998; Rowlands, 1999).

Until the late 1990s there appears to have been little inter-professional conflict between computer professionals and engineers. However, the potential for conflict had been identified. In 1981, computer scientist Theodor Sterling suggested that it was important to clarify the boundaries between the two fields by exploring the extent to which “data processing work, especially systems analysis and senior programming, is engineering within the definition of the Engineering Act” (1981, p. 7). Sterling’s concern had less to do with inter-professional conflict, than with employment practices. A few years later, Ron Foyer held that CIPS should communicate with “the engineering profession to avoid a ‘turf territory’ fight over professionalism in information processing, which could be extremely destructive” (Foyer, 1985, p. 32). It is unclear whether talks between the professions ever occurred. However, since 1997, an intense inter-professional battle

between engineers and information technology professionals has been occurring over the education and practice of software engineers. Open conflict erupted with the CCPE’s decision to sue Memorial University when the school attempted to offer a “software engineering” degree in its computer science department; CCPE claimed infringement of copyright. Only accredited engineering faculties can offer courses in branches of engineering, they asserted. While this case was never completely resolved, it did precipitate a great deal of discussion both between and within CIPS and engineering organizations, over which discipline can rightfully claim SWE, and who can practice as software engineers.^{xxii}

The engineering position is clear and straightforward: the terms ‘engineer’ and ‘engineering’ are protected titles owned by the engineering profession. Non-engineers cannot establish engineering programs, nor can they train people to become engineers. Software engineering is now a recognized engineering specialty, and in recent years a number of people have become licensed software engineers. Accredited SWE programs are now in existence, and the numbers of licensed software engineers will continue to increase.

Opposing engineers in this inter-professional conflict are CIPS and representatives from the Canadian Association of Computer Science (CACS/AIC), an organization of university computer science departments, schools and faculties. CIPS’ stance is that ‘software engineering’ is not an engineering discipline but a term of historical origin, that has come to be applied to a branch of practice within computer science and computing more broadly (CIPS, 2002; Gabrini, 2000; Wordsworth, 2002). As

Wordsworth, (2002) argues, “software engineering originated in computer science and developed connections to engineering much later.” CIPS defends the rights of non-engineers to perform software engineering work, and argues that it is IT professionals, and not engineers who have “software systems expertise” and skills to perform the work safely (CIPS, 2002; Wordsworth, 2002). The CACS (2001) reiterates that “the vast majority of Canada’s academic software expertise resides” in computer science departments. Moreover, it points out that the effort to restrict software engineering practice to engineers threatens the ‘right to practice’ of many IT professionals. Of particular concern to the CACS (2001) is the fact that software engineering is “increasingly being identified with the entire area of applied Computer Science.” Some engineers claim that computer science is a theoretical discipline that informs software engineering practice (Parnas, 1998); from this perspective there is no ‘applied’ computer science, only engineering. Thus, the efforts of Engineers to lay claim to the area of software engineering threaten the boundaries of the discipline of computer science and the practice of many in the computing field.

Following its counterparts in the U.S., CIPS has proposed a compromise with the CCPE, and has advocated a joint accreditation committee, that would see engineers and computer scientists working together to accredit, or at least to set accreditation criteria for, SWE programs in both computer science and engineering programs (Bassett, 2002). A joint committee of CIPS and engineering professionals was struck to explore joint accreditation of SWE programs, but this committee failed to reach a satisfactory solution, especially from engineering’s point of view. Because an

engineering education is central to the licensing process in engineering, strict accreditation criteria are essential for maintaining professionalism in the field. Engineers have more rigid accreditation policies than do information technology professionals – computer science is said to be a more flexible discipline in terms of options and specialization (Parnas, 1998). Establishing accreditation criteria that would serve computer science, while preparing students adequately for an engineering license, appears to be virtually impossible (DeVita, 2000; Parnas, 1998). Moreover, engineering leaders feel strongly that engineers must be trained in engineering faculties to acquire a broad engineering knowledge base and to preserve the “common engineering culture” (DeVita, 2000). Further, some argue, computer science graduates are not always “qualified to develop critical software products” (Parnas, 2001, p. 27); what the field needs are “engineers who understand basic engineering principles, but who also have the necessary specialized knowledge to develop software intensive products” (ibid, p. 39).

Thus, while organizations in the UK and US have managed to establish joint criteria for accreditation of SWE programs, such co-operation seems highly unlikely in the Canadian setting. As things stand currently, Canadian computer science departments are at least scaling back on the use of the term software engineering in the programs and courses they offer. Software engineering programs, accredited by an engineering body, are springing up in faculties of engineering across the country (Van Ihinger, 2001). While some software leaders have argued that there is a place in the software field for non-engineering software specialists (Parnas, 2001; Parnas in

Vanhinger, 2001), computer scientists in Canada have been slow to claim this non-engineering software field as their own.

At the heart of the inter-professional dispute in Canada are two related issues, that appear to have been central – in different ways – in the evolution of software engineering in the U.S. and in the U.K. First, is the professional and organizational structure characteristic of engineering and computing. Second is the nature and significance of engineering licensing. It is precisely in both of these areas that Canada's situation differs substantially from that in the US and UK. Engineering in Canada has a higher professional status and is a more closed profession in Canada, than in the other two nations. In Canada, no-one who is not a licensed engineer can legally work as an engineer. This legislation puts engineering on a par with other dominant, self-regulating professions such as medicine, dentistry, and law, to name but a few. In contrast, computing and information technology occupations are “less professional” in Canada, than in the other nations; or at least less organized. In contrast to the BCS for example, CIPS membership is not restricted, and its credential for IT professionals – the ISP (information systems professional) – is a voluntary one, established only 15 years ago, and not possessed by a majority of CIPS members or IT professionals in Canada. University training in the computing field is not even mandatory to achieve an ISP.^{xxiii} CIPS is currently in the process of having its ISP designation recognized through provincial legislation, but currently only six provinces have legislation that prevents those without an ‘ISP’, from claiming that they have one. CIPS’ status as a professional organization is far below that of the BCS, and – in size and strength in particular – less than the ACM as

well. The status and organizational distance between engineers and IT workers in Canada large. And this social distance is a key factor encouraging inter-professional conflict in this context. The fact that the two organizations do not have a history of working together exacerbates the situation.

The organizational and status gap between the two fields may not, in and of itself, ensure inter-professional conflict. However, the potential for conflict is further enhanced when there is also a difference in the way in which the two groups are regulated. In the computing and IT fields, virtually anyone can practice, regardless of their specific education background and training. Education backgrounds of people in the field have tended to be quite variable. Even those IT workers committed to professionalism do not share a common education background – as noted earlier, an ISP can be granted based on varying combinations of experience and education. IT leaders are opposed to a greater formalization of the field and explicitly opposed to licensing (Van Ihinger, 2001, p. 9). In contrast, engineers are strong advocates of licensing, and embrace their system wherein only those with a very specific education background, work experience and the completion of a common exam can practice. Engineers cannot compromise their professional standards with looser criteria for practice. At the same time, tougher criteria for computing-related workers seems unlikely – computing leaders have long been reluctant to restrict the field to those with only a very specific form of training (see Adams, 2004a). The two disciplines have “deep philosophical differences in professional approach” (CACS, 2001, p. 5), and do not have enough common ground for co-operation (also Parnas, 2002).^{xxiv}

Further encouraging conflict is the fact that both groups of occupations are seeking to extend and enhance their professional status. Engineers are explicitly endeavouring to raise “the profile and perceived value of the engineering profession ” (CCPE, 1997; Ridout, 1997). At the same time, CIPS is devoted to extending credentials and professional legislation for IT workers in Canada. These professional projects have spurred inter-professional conflict over the jurisdiction of software engineering.^{xxv} The conflict has been exacerbated by the fact that engineers and IT professionals have different ideals of professionalism. Engineers are following a more traditional ‘licensing’ route, while IT workers are stronger advocates of an “individual professional” model that emphasizes ethics and skill development, but not credentials and licenses (McCalla, 2002).

While IT workers have shown a willingness to work with engineers towards accreditation of software engineering programs, they have been fundamentally opposed to licensing, and in their battles with engineers have contested that ‘engineering’ is a protected title and argued that software engineers should not be licensed (CACS, 2001; McCalla, 2002; Wordsworth, 2002). These actions have prompted engineer David Parnas (2002, p. 97) to comment that computer science leaders seem “unaware of the nature of self-governing professions,” and hence have disputed issues out of their control, instead of working “to make sure [that] licensing criteria were appropriate.” Engineers as self-regulating professionals, who wish to maintain, and indeed increase, their professional status, must pursue the path they are currently following: establishing education and licensing procedures for software engineers,^{xxvi} and prosecuting those

unlicensed workers who violate the law by claiming to be ‘engineers.’ However, their path is a difficult one, as there are currently many practitioners claiming to be software engineers without an engineering license, and over 25,000 more with ‘engineering’ credentials from organizations such as Microsoft (Rawlines, 2002).^{xxvii} Engineers cannot prosecute them all.^{xxviii} Moreover, their success in prosecuting violators has been mixed. Engineers in Quebec recently won a court battle against Microsoft for the latter’s illegal use of the term engineer in its certification program (Schick, 2004). However, the victory was at best a moral one, as the penalty was only a \$1,000 fine. A recent Alberta case against an Apple Canada-certified systems engineer was dismissed by the court because, it was claimed, the defendant “presented no injury to the public” by claiming to be a systems engineer, without possessing a license (Aschaiek, 2004, p. 15). Thus, the law is being broken so extensively, that it is unlikely that engineers can successfully uphold their position that only licensed engineers can call themselves software engineers.^{xxix} Nevertheless, their present course could ensure that in the future, only full engineers practice software engineering, or at least that true software “engineers” have a more privileged place in the labour market.

Discussion

The next few years should be important ones for the establishment of the software engineering profession, and the articulation of its relationship with computer science, especially in the United States and Canada. While the relationship between computing and engineering in the UK appears to have stabilized into one of peaceful co-operation and co-existence, inter-professional conflict

should continue in Canada. Whether this conflict intensifies in the United States as software engineers pursue their professional project, or whether it will decrease, is difficult to predict. It seems likely that the effort to professionalize software engineering will continue, especially in the face of the growing 'internationalization' of information technology labour, and the migration of software engineering jobs overseas. If efforts to regulate and professionalize software engineers have been prompted by concerns over software quality, these concerns (and professional activities) will likely increase with the growing use of foreign-based labour. How this trend might inform the direction of the SWE professional project, and in particular the development of SWE licensing, will be interesting areas of exploration for future research.

Another area for future exploration is the impact of SWE's professional project on computing more generally. As we have seen, ACM is in the process of seeking to advance the "IT" profession. Moreover, work is being conducted on outlining a C-BOK, or computing body of knowledge; this project may, in part, be a response to the SWEBOK effort. Computing professionals may work to define software engineering as representing only a fraction of software-related work. This seems especially pertinent in the Canadian setting, where engineers' ability to take over the training and licensing of future SWE workers, may lead computer science and IT professionals to lay claim to other occupations and tasks in the software field. The growing tendency of computer science departments to continue to teach tools for software practice, while distancing themselves somewhat from the term 'software engineering' might suggest

that this trend has already begun. Regardless of what trends emerge in the future, there is every reason to believe that these inter-professional developments will occur differently in the UK, U.S., and Canada, given the organizational, professional, and legislative differences that characterize the occupations in these three countries.

Exploring professional developments in software engineering in each of these 3 nations, highlights the importance of organizational structure and professional status to inter-professional conflict. While previous work, has highlighted that inter-professional conflict is often spurred by the professional projects of occupations in related jurisdictions (Adams, 2004b), this study indicates the additional importance of differences in professional status, professional regulation, and inter-organizational ties. It was in the U.K., where the professional status and legal status of engineering and the computing profession are quite similar, that inter-professional co-operation emerged most fully. The efforts of the BCS to have themselves considered an 'allied' engineering profession, also minimized the conflict. In the U.S., inter-professional co-operation between computing professionals and engineers has also been low, although it may be increasing. The history of co-operation between computing professionals and engineers, and the definition of computing as an engineering profession, facilitated collaboration in the creation of the SWE profession. The fact that most engineers, like computing workers, are not professionally licensed, likely also contributed to good relations. The one key area of disagreement that has emerged, however, centres precisely around this key difference in professional regulation:

engineers have access to licensing and support it, while computing professionals do not. Moreover, computing workers are wary of having SWE defined as a full engineering profession (requiring engineering education), preferring to see it as a “computing profession” requiring computing education. Inter-professional conflict in Canada, is largely the product of the status difference between a professionally regulated engineering profession, and a poorly organized, lower status, IT/computing field. Canadian engineers are seeking to incorporate the area of SWE and appear to have little to gain from co-operating with the lower-status computing profession. Their interests are divergent. Here inter-professional conflict is the highest.

This analysis suggests, then, that it is possible for two professionalizing occupations to claim a jurisdiction jointly. However, co-operation is only likely when those occupations possess a similar status, overlapping philosophies of practice (here an engineering focus), and perceive that their professional goals can be met through co-operation. When the status gap is large, philosophies differ, and there is nothing to gain by one profession from co-operation, conflict is more likely.

References

- Abbott, Andrew. (1988). *The system of professions*. Chicago: University of Chicago Press.
- Abran, Alain, J. Moore, P. Bourque and R. Dupuis, (Eds.). 2001. Guide to the Software Engineering Body of Knowledge, SWEBOK. IEEE Computer Society, www.swebok.org.
- Abran, Alain, Bourque, P. & Tripp, L.L. (2004). *Guide to the Software Engineering Body of Knowledge, Final Version*. IEEE Computer Society, www.swebok.org.
- Adams, Tracey L. (2004a). *Professionalization in computing related occupations: Canada, the U.S. and Britain*. WANE Working Paper, 45 pp.
- Adams, Tracey L. (2004b). Inter-professional conflict and professionalization: Dentistry and dental hygiene in Ontario. *Social Science and Medicine* 48: 2243-2252 (currently published on line, paper copy available in June).
- ACM. (2000). A summary of the ACM position on software engineering as a licensed profession. Available online at: www.acm.org/serving/se_policy/selep_main.html. Accessed on July 17, 2000.
- ACM C³S (ACM Curriculum Committee on Computer Science). (1968). Curriculum 68, recommendations for academic programs in computer science. *Communications of the ACM* 11 (3): 151-97.
- Andriole, S.J. & Freeman, P.A. (1993). Software systems engineering: The case for a new discipline. *Software Engineering Journal* 8 (3), 165-179.
- Aschaiek, Sharon. (2004). Alberta case thrown out: debate continues on software titles. *Engineering Dimensions*, (March/April), p. 15.
- Bagert, Donald J. (2002). Texas licensing of software engineers: All's quiet for now. *Communications of the ACM* 45 (11), 92-3.
- Barron, D.W. (1975). Professional programmers? *Computer Bulletin* 2 (4), 5.
- Bassett, Paul. (2002). Bassett proposal to restart SEAB discussion. Letter dated April 3, 2002, www.cs.usask.ca/spec_int/cacs/page13.htm.
- BCS [British Computing Society]. (1960). Annual Report, 1959-60. *The Computer Bulletin* 4 (9), pp. 37-40.
- Bechtolsheim, Stephan V. (1985). Letter: On the potential ACM/IEEE-CS Merger. *Communications of the ACM* 28 (3), 237-8.
- Bertziss, Alfs. (1987). A mathematically focussed curriculum for computer science. *Communications of the ACM* 30 (5), 356-65.
- Bott, Frank. (1989). Software engineering education. *Software Engineering Journal* 4 (4), 174-5.

- Bourque, P. et al. (1998). Guide to the software engineering body of knowledge. IEEE Computer Society. www.swebok.org.
- Brown, P.J. (1985). Software engineering ... and documentation. *Computer Bulletin, Series III*, 1 (3), 16-17.
- CACS (Canadian Association of Computer Science). (2001). CACS/AIC position on SEAB. Available online at: www.cs.usask.ca/spec_int/cacs/page14.htm, 8pp.
- Carlson, Walter M. (1970). What are your standards? *Communications of the ACM* 13 (2), 65-6.
- CCPE (Canadian Council of Professional Engineers). 1997. A Vision for the Engineering Profession in Canada, CCPE. www.ccpe.ca
- de Champeaux, Dennis. (2002). Software engineering considered harmful. *Communications of the ACM* 45 (11), 102-104.
- Chapman, Gary. (1990). Bugs in the program. *Communications of the ACM* 33 (3) 251-2.
- Cheetham, A. W., Woolliams, P.R., & Knowles, P.J. (1988). Software engineering: space invader of the 1980's? *Computer Bulletin, Series III*, 4(3), 24-5.
- CIPS (1969). Report by the Accreditation Committee to the Directors of the Information Processing Society of Canada. *IPSOC Quarterly Bulletin* 9 (3, Spring) supplement, pp 1 - 12.
- CIPS (2002a). Software Engineering: position paper. www.cips.ca/it/position/softeng/
- CIPS (2002b). What's in a name? Tech sector battles engineers on 'software engineering', www.cips.ca/news/media/news.asp?aID=1387
- CSAB (Computer Science Accreditation Board). (2004). www.csab.org (Accessed April 2004).
- Crinean, Kay and Lisa Baguley. (1986). The starts initiative. *BCS Computer Bulletin, Series III*, 2 (2), 35,37.
- Collins, W.R. et al., (1994). How good is good enough: An ethical analysis of software construction and its use. *Communications of the ACM* 37 (1), 81-91.
- Conway, D.E., Dunn, S.C. & Hooper, G.S. (1989). BCS and IEE accreditation of software engineering courses. *Software Engineering Journal* 4 (4), 245-8.
- Davies, Jack. (1982). Opportunities are there - if we don't "muff" them. *CIPS Review* 6 (July/Aug), p. 4.
- Denning, Peter J. (1984). Educational Ruminations *Communications of the ACM* 27 (10), 979-83.
- Denning, Peter J. (1987). Paradigms crossed. *Communications of the ACM* 30 (10), 808-9.
- Denning, Peter J. (1990). Human error and the search for blame. 33 (1), 6-7.

- Denning, Peter J. (1998). Computer science and software engineering: Filing for divorce? *Communications of the ACM* 40 (8), 128.
- Denning, Peter J. et al. (1989). Computing as a Discipline. *Communications of the ACM* 32 (1), 9-23.
- Denning, Peter J. et al. (1991). Computing curricula 1991. *Communications of the ACM* 34 (6), 69-84.
- DeVita, Peter. (2000). Memorandum: Recommendations of the (PEO) Software Engineering Panel www.peo.on.ca/news/Software_ccpesubmission.htm, 3 pp.
- EC (Engineering Council, UK). (2003). Registration. www.engc.org.uk/registration/engineerprotection.asp; www.engc.org.uk/registration/.
- Ensmenger, Nathan. (2001). The 'question of professionalism' in the computer fields. *IEEE Annals of the History of Computing* 23 (4), 56-74.
- Findler, N.V. (1985). Letter: On mathematics and computing education. *Communications of the ACM* 28 (10), 1026.
- Foyer, Ron. (1985). The danger: divisiveness, and how to prevent it. *CIPS Review*, 9 (May/June) pp. 32-3.
- Frailey, D.J. (1988). Letters: More on the Computer Science Decline. *Communications of the ACM* 31 (8), 946-7.
- Gabrini, Philippe. (2003). Software engineering: Where does it belong? www.cip.s.ca/it/position/software/, 7 pp.
- Gardner, Ellen. (2000). "Shifting the Tide: Regulation of software engineering practice, Engineering Dimensions, July/Aug, p. 27.
- Givens, W.I. (1967). A 'subprofessional' comments. *Communications of the ACM* 10 (7) 396.
- Gordon, R.M. (1990). Letters: Measure for measure. *Communications of the ACM* 33 (6) 268.
- Gries, David. (1991). Teaching calculation and discrimination: A more effective curriculum. *Communications of the ACM* 34 (3), 45.
- Grosch, Herb. (1987). Letters: After nineteen years. *Communications of the ACM* 30 (12) 996.
- Hadford, Gary. (1991). Does certification point the way? Canadian Information Processing, Sept/Oct., pp 11-13.
- Harris, Marilyn. (1991). Certification Update. CIPS National Newsletter, Sept/Oct., p 4.
- Hartmanis, Juris. (1994). Turing aware lecture: On occupational complexity and the nature of computer science. *Communications of the ACM* 37 (10), 37-42.

- Heinlein, E.B. (1991). ACM forum: Legal fine line. *Communications of the ACM* 34 (7), 15.
- Herbst, R.T. (1985). Letters: Follow-up on the industry-University dialogue. *Communications of the ACM* 28 (6), 565.
- Hoare, C.A.R. (1981). "Professionalism." *Computer Bulletin Series II* 29, pp. 2-4.
- Hoffnagle, Gene F. (1991). Epilogue: engineering the software development process. *Software Engineering Journal* 6 (5), IPC.
- Johnson, Luanne (James). (1998). A view from the 1960s: How the software industry began. *IEEE Annals of the History of Computing* 20 (1), 36-42.
- Knight, John, Leveson, N., et al., (2001). On licensing of software engineers working on safety-critical software: Final report of the ACM task force. Accessed August 2001 at: www.acm.org/serving/se_policy/selep_main.html
- Knight, John, Leveson, Nancy, et al. (2002). Should Software Engineers be Licensed?" *Communications of the ACM*, 45 (11), 87-90.
- Kocher, Bryan. (1988). You and ACM: Partners in progress. *Communications of the ACM* 31 (9), 1033, 1046.
- Kocher, Bryan. 1989. President's letter. *Communications of the ACM* 32 (6), 660-1.
- Kocher, Bryan. (1990). President's letter: Eschew obfuscation. *Communications of the ACM* 33 (6), 625.
- Koffman, E.B, Miller, P.L. & Wardle, C.E. (1984). "Recommended Curriculum for CS1, 1984." *Comm ACM* 27 (10), 998-1001.
- Koffman, E.B., Stemple, D. & Wardle, C.E. (1985). "Recommended Curriculum for CS2, 1984." *Comm ACM* 28 (8) 815-818.
- Lee, John. (2002). The engineers among us. www.itbusiness.ca, Dec. 13, 2002. Article also available at www.peo.on.ca/enforcement/ITbusiness/dec172002.html.
- Lehman, M.M. (1991). Software engineering, the software process and their support. *Software Engineering Journal* 6 (5), 243-4.
- Lewis, Philip M. (1989). Letters: Information systems is an engineering discipline. *Communications of the ACM* 32 (9), 1045-7.
- Loeser, Rudolph. (1989). Letters: Software engineering. *Communications of the ACM* 32 (8), 919.
- Low, Cameron. (1988). Guest editorial: Engineer, training, market ... *BCS Computer Bulletin, Series III*, 4 (3), 3.
- McAuley, Patrick. (1991). "Keep in touch with the pace of change." *Canadian Information Processing*, Sept/Oct., p. 8.
- McCalla, Gord. (2002). Software engineering requires individual professionalism. *Communications of the ACM* (11), 98-101.

- McCracken, D. (1987). Ruminations of computer science curricula. *Communications of the ACM* 30 (1), 3-5.
- McIlroy, M.D. (1990). Letters to the editor: Green light for bad software. *Communications of the ACM* 33 (5), 479.
- Nelson, David A. (1987). "Letters: Relevance vs. Rigor." *Comm ACM* 30 (7) 585-6.
- Neumann, Peter G. (1991). "Inside Risks: Certifying Professionals." *Comm ACM* 34 (2), 130.
- Newton, R.W. (1986a). "Software, a vital key to UK competitiveness," *BCS Computer Bulletin, Series III, 2* (3), 24, 25, 28.
- Notkin, D., Gorlick, M. & Shaw, M. (2000). An Assessment of Software Engineering Body of Knowledge Efforts. A Report to the ACM Council, May 2000 (www.acm.org).
- Oakley, B. (1988). Professional societies: The next ten years. *Computer Bulletin, Series III, 4* (2), p. 3.
- Oettinger, Anthony G. (1967). President's letter. *Communications of the ACM* 10.
- Orden, Alex. (1967). The emergence of a profession. *Communications of the ACM* 10 (3) 145-7
- Ould, M. & Thewliss, D. (1987). Making software engineering standards usable. *BCS Computer Bulletin, Series III, 3* (2), 28-30.
- Ould, M. & Thewliss, D. (1986b). The society and the engineering council. *BCS Computer Bulletin, Series III, 2* (3), 23.
- Parikh, Girish. (1985). Letters: Academia is over 30 years behind. *Communications of the ACM* 28 (3), 240.
- Parnas, David L. (1998). Software engineering programmes are not computer science programmes. *Annals of Software Engineering* 6, 19-37.
- Parnas, David L. (2001). Why software developers should be licensed. *Engineering Dimensions*, May/June, pp. 36-39.
- Parnas, David L. (2002). Licensing software engineers in Canada. *Communications of the ACM* 45 (11), 96-98.
- PEO (Professional Engineers of Ontario). (1999a). "PEO adopts official policy re: software engineering," www.peo.on.ca/news/softwareeng_policy.htm, posted May 1999.
- PEO (Professional Engineers of Ontario). (1999b). Media release: Engineering Association embraces software practitioners. Sept 7, 1999. www.peo.on.ca/news/Old_Press_Releases/1999/software_release.htm.
- PEO (Professional Engineers of Ontario). (1999c). Engineering disciplines task group. www.peo.on.ca/member/TaskComm/EDTF_rpt1999.htm.

- Potter, Mike. (1982). "Penetrating U.S. Market demands careful strategy. *CIPS Review* 6 (July/Aug), pp. 5-7.
- Pyle, I.C. (1986). Software engineers and the IEE, *Software Engineering Journal* 1(2), 66-8.
- Ralston, Anthony. (1984). The first course in Computer Science needs a mathematics corequisite. *Communications of the ACM* 27(10), 1002-1005.
- Rawlines, Karen. (2002). Engineers warn Microsoft BRANDING: Professionals say use of term 'engineer' is strictly regulated. New Brunswick Telegraph-Journal, August 8, 2002; also available through www.peo.on.ca/enforcement/NB_Aug_2002.html.
- Rickert, Neil W. (1989). Letters: Computing as a discipline. *Communications of the ACM* 32 (11) 1286-7.
- Ridout, Peter. (1997). President Peter Ridout's messages: Engineering Dimensions/The Link. www.peo.on.ca/publications/Past_President/pridout.html, 12 pps.
- Rowlands, Joyce. (1999). Software engineering: "Wild West" or maturing discipline? *Engineering Dimensions*, Sept/Oct, pp. 30-33.
- Schick, Shane. (2004). Quebec engineers win court battle against Microsoft. www.itbusiness.ca, April 8, 2004.
- Shore, John. (1988). Why I never met a programmer I could trust. *Communications of the ACM* 31 (4), 372-5.
- Sidlo, C.M. (1961). "The Making of a Profession." *Comm ACM*. 4 (9), pp. 366-7.
- Steen, Lynn Arthur. (1985). Mathematics and computing education: A common cause. *Communications of the ACM* 28 (7), 666-7.
- Sterling, Ted. (1981). Engineers threaten domination of DP jobs. *CIPS Review* 5 (Jan/Feb), pp. 6-8.
- Tartar, John et al. (1985). The 1984 snowbird report: Future issues in computer science. *Communications of the ACM* 28 (5), 490-3.
- Thomas, Martyn. (1987). Professionalism and software engineering, *Computer Bulletin, Series III*, 3 (4), p. 3.
- Thornley, David H. (1990). Letter to the editor. *Communications of the ACM* 33 (6), 629-30.
- Tomayko, James E. (1998). Forging a discipline: An outline history of software engineering education. *Annals of Software Engineering* 6, 3-18.
- Turner, John. (1985). Education and professionalism rank high on Ron Foyer's list of priorities. *CIPS Review* 9 (March/April) pp 5-6.
- Van Ihinger, Sharon. (2001). Debate continues on joint accreditation council: CEAB accredits first software programs.

- Engineering Dimensions*, July/Aug, pp. 8-9.
- Wagner, F. (1990). Letters... *Communications of the ACM* 33 (6), 628-9.
- Weaver, J.W. (1989). More on Regulation and Legislation. *Communications of the ACM* 32 (11), 1284.
- Wilkes, M.V. (1991). Software and the Programmer. *Communications of the ACM* 34 (5).
- Willmott, G.M.R. (1975). Each cobbler to his own last. *Computer Bulletin, Series II*, (5), 16.
- Winder, R, Cole, R. & Easteal, C. (1987). Software engineering in a first degree. *Software Engineering Journal* 2 (4), 133-9.
- Wordsworth, Charles, (2002). Letters to the Editor – Engineering moniker – the sequel. *Computing Canada*, Aug. 23, Vol. 28, No. 7. Also available through, www.peo.on.ca/enforcement/ComputingCanadaBusinessAug232002.html.

Appendix ONE: **List of Acronyms**

ABET	Accreditation Board for Engineering and Technology (US)
ACM	Association for Computing Machinery
BCS	British Computer Society
CACS	Canadian Association of Computer Science
CCPE	Canadian Council of Professional Engineers
CIPS	Canadian Information Processing Society
CSAB	Computer Science Accreditation Board (US)
DPMA	Data Processing Management Association (US)
EC	Engineering Council (UK)
ICCP	Institute for Certification of Computing Professionals
IEE	Institute of Electrical Engineers (UK)
IEEE	Institute of Electrical and Electronic Engineers (US)
PEO	Professional Engineers of Ontario (Can)
SWE	Software Engineering
SWEBOK	Software Engineering Body of Knowledge
SWECC	Software Engineering Co-ordinating Committee (US)

ⁱ While not ideal for conducting research on the nature of professional change within Canada, the sporadic nature of CIPS publications is itself evidence of the organization's and profession's status during much of its 45 -year history.

ⁱⁱ After this date, the journal was succeeded by the *IEEE Transactions on Software Engineering*. However, with the change, the journal's focus altered: it became more of a technical journal, and thus, no longer carried many articles related to professional issues.

ⁱⁱⁱ The American Institute of Electric Engineers and the Institute of Radio Engineers formed computer groups in the 1940s. In the early 1960s these two organizations merged to form the Institute of Electrical and Electronic Engineers (IEEE), and their computer groups merged to form one computer society, informally referred to as the IEEE-CS.

^{iv} The DPMA had a Canadian branch (DPMA Canada). In 1973, DPMA, ACM, IEEE-CS, CIPS, and several other societies joined together to form the Institute for Certification of Computing Professionals (ICCP). This organization took over the DPMA certification examinations, and certified many Canadians and Americans in the data processing and computer fields.

^v It is also noteworthy, that the given definition of computer science "omits the areas where computer scientists spend more of their time" especially programming (Rickert 1989, 1287). This may represent an attempt by computing leaders to distance themselves from a core area of practice, increasingly viewed as requiring less skill, and as only one small aspect of the much broader area of 'software engineering.'

^{vi} It should be noted that the ICCP has incorporated a SWE specialization into its CCP (Certified Computing Professional) credential program, with a similar aim of signifying competence and weeding out the ineffectual.

^{vii} For instance, a 1991 Canadian article asserted that "programming is not a professional level activity" because it required little decision-making (Harris 1991, 4).

^{viii} The ten knowledge areas consist of software requirements, design, construction, testing, maintenance, configuration management, engineering management, engineering process, engineering tools and methods, and quality (Abran et al., 2001).

^{ix} Such arguments have been more prominent in Canada (for instance, CSAC 2000), where the ability of computer science departments to teach software engineering is being threatened (discussed below). However, the idea that computer science requires engineering of a different sort, and is inherently practical is a theme running through the ACM literature as well (see for example Hartmanis, 1994, 41; Denning 1998).

^x AFIPS is the American Federation of Information Processing societies, and for several decades it was the organization that represented the United States in the International Federation for Information Processing.

^{xi} Here, the recent change in accreditation policies is significant. Computer science programs are now accredited through ABET, and hence, this problem is currently less relevant.

^{xii} Additional concerns include the following: engineering licensing is state-based, while software transcends borders and software engineers could have to be licensed in every state in which their software is used, at great personal expense; multiple choice exams used by licensing bodies could not capture SWE knowledge; examinations would not be flexible enough to keep up with technological change; it is impractical to license the many that work with software; and there is no comprehensive body of knowledge for software engineering safety critical systems (the SWEBOK does not focus on safety-critical software knowledge) (Knight and Leveson 2002).

^{xiii} Two of the most notable include Earl Mountbatten who was the BCS president in 1966-7, and president of the Institution of Electronic and Radio Engineers in 1946 (BCS 1966, 18) and B.R.V. de Ferranti, engineer, member of the IEE, and BCS president in 1969 (BCS 1968, 3).

^{xiv} A joint IEE/BCS working group on Software Engineering Standards was established in the mid-1980s to identify standards acceptable to both professional groups (Ould and Thewlis 1987).

^{xv} Registration criteria include specific academic qualifications, experience and training, and a professional examination. There are roughly 35 engineering institutions, each of which is related to a field of engineering. For instance, while electrical engineers would generally belong to the IEE, civil engineers might belong to the Institute of Civil Engineers. Membership is not mandatory for engineering practice, although it is essential for registration.

^{xvi} Beginning in April 2004, the BCS offered a new diploma module for its professional examination, 'Software Engineering 1'.

^{xvii} The two SWEBOK reports were produced by a team of researchers, most of whom are based at the Université du Québec à Montréal (UQAM).

^{xviii} For instance, the first course in SWE at the University of Western Ontario was offered in the 1973-4 academic year and after. The course was entitled "Systems programming and software engineering." By the mid-1990s, the computer science department was offering 6 courses in the area.

^{xix} National Occupational Classifications from the early 1990s, counted software engineers as computer engineers in its broad 'other engineers' category. Other software workers (like software analysts and programmers) were classified as computer systems analysts and programmers.

^{xx} Programs are accredited by the national Canadian Engineering Accreditation Board (CEAB).

^{xxi} Moreover, the PEO has resolved that "any software component of a product or system whose development is the practice of professional engineering, as defined under the Professional Engineers Act... shall be approved by a licensed professional engineer" (PEO 1999c). Thus, engineers are attempting to extend the role of engineers in the software development process.

^{xxii} In 1999, it was agreed that CIPS and the CCPE should co-operate to formulate a resolution to the problem of software engineering education, and that the CCPE would refrain from prosecuting other university programs for 5 years (CIPS 2002a). In the interim, computer science departments have reduced the number of courses and programs explicitly in SWE, while Engineering departments and faculties have begun to offer accredited SWE programs.

^{xxiii} Individuals with post-secondary computer-related education and experience are eligible for the ISP, as are those with little education, but extensive work experience (if they entered the field prior to 1976), or those who pass an examination and possess some work experience.

^{xxiv} This appears less true in the U.S. and especially in the U.K., where computing professionals can obtain engineering credentials. The lack of 'philosophical differences in the other countries has encouraged co-operation.

^{xxv} See Adams (2004b) for a more detailed discussion of how inter-professional conflict is often generated by conflicting professional projects, and the drive for professional status.

^{xxvi} Software practitioners can obtain a P.Eng licence in Ontario by graduating from a CEAB approved SWE program, having 4 years of employment experience and completing the professional practice examination. However, the PEO does enable non-engineers to qualify for a license without an engineering education, if they have suitable employment experience (more than 5 years in the software field) and a demonstrated knowledge base in specified areas of SWE practice (PEO 1999b). Similar licensing criteria have been established in British Columbia (Gardner 2000). By 2002, Ontario had licensed about 300 people as 'software engineers' (Parnas, 2002: 98)

^{xxvii} The 2001 Canadian Census counts 18,615 software engineers working in the country; only a tiny fraction of these workers claiming to be 'software engineers' would be licensed.

^{xxviii} It must be noted that the CCPE's policy is first to approach companies "that uphold unlawful designations in their own IT departments and individuals who illegally refer to themselves as engineers" to encourage compliance with the law (Lee 2002).

^{xxix} Engineers recognize their limitations here. Former PEO president Peter Ridout (1997: 3) believes it unlikely "that we'll ever be able to establish an exclusive scope of practice, to protect the public interest, in all aspects of such emerging areas as computer software .."