A Review of the Professionalization of the Software Industry: Has it Made Software Engineering a Real Profession?

Heng Ngee Mok

School of Information Systems
Singapore Management University
mok@ieee.org

Abstract

Every industry strives to be called a "profession", and software engineering is no exception. This paper attempts to define "profession" from three different perspectives and provides a chronological narration of the professionalization efforts of major IT bodies such as the IEEE Computer Society, Association of Computing Machinery and British Computer Society to promote software engineering from "occupation" to "profession". The outcome of this professionalization process is then examined against the three vastly different definitions of "profession" to qualitatively gauge the success of the professionalization process.

Keywords: Software engineering profession, software professional, professionalism, professionalization of the software industry, licensing of software engineers

Introduction

Being called a "professional" has a certain positive ring to it, and being a member of a profession comes with several privileges: social prestige, a higher pay packet, a monopoly of practice, and on a personal level, a sense of self worth and self-actualization. Traditionally, medicine and law have always been categorized as professions, and today, almost every self-respecting occupation is vying for the same label. This paper traces the global professionalization efforts of the software industry. This is followed by a discussion of whether these labors have been effective in transforming software engineering into a real profession, and software practitioners[1] into real professionals.

This is a significant question because the number of software practitioners has increased drastically in the past thirty years and represents a sizable portion of the knowledge-based workforce in modern states. If these professionalization attempts prove futile, there may be no further need to invest more resources into these efforts. On the other hand, if the professionalization of this nascent discipline is showing signs of success, perhaps we should – especially if it brings evident benefits to society – further encourage this transformation.

[1] The term "IT" is almost all-encompassing of activities related to computing. IT work can generally be categorized into "software" (production of code which does the job) and "infrastructure" (maintenance of the hardware which enables code to execute). This paper focuses on the software practitioners who work in the former and significantly larger category, although what is discussed may apply generally to infrastructure practitioners as well. Software practitioners include project managers, business analysts, software architects and designers, database designers, user interface designers, programmers and testers. General users of software (such as data entry personnel, or a clerk who uses a word processor as part of his job) are not considered software practitioners.

What is a Profession and Who are the Professionals?

The first task is to understand what a profession is, and who the professionals are. But finding a universally agreed upon definition of “profession” is itself a virtually impossible undertaking. While all Western societies will quickly point to medicine and law as exemplary examples of professions, most societies will not have a unified view of other disciplines: the clergy, dentistry, architecture, accountancy, academia and civil engineering (most of these are categorized as professions in many societies). Many others wallowing in the penumbra include diverse fields such as teaching, pharmacy, law enforcement, marketing, financial planning, realty, and even fund raising.

Defining Profession

The dictionary defines “profession” as “an occupation, especially one that requires advanced learning” (Hawkins, 1994). Unfortunately using the term as a synonym for “occupation” is superficial and overly simplistic, and this approach should be immediately discarded. Cogan (1953) defined the profession as “a vacation whose practice is founded upon an understanding of the theoretical structure of some department of learning or science, and upon the abilities accompanying such understanding”. Savan (1989) called professions “groups which apply special knowledge in the service of a client”. I consider both definitions deficient because almost all modern service-based occupations involve some theoretical expert knowledge, and that qualifies so many occupations as professions that the latter term loses its distinctive meaning. A profession has to be more than just a congregation of expert service providers.

Freidson (1970)’s attempt to define the term focuses on the control which a profession has on members and the market; he believes that professional autonomy is a useful index by which to rate the relative professional status of various occupations. His authoritative work states:

> *“…the most strategic distinction lies in legitimate, organized autonomy – that a profession is distinct from other occupations in that it has been given the right to control its own work.”*

In other words, according to Freidson, the amount of autonomy given to members of that occupation is the most significant factor which determines if it fits into the category of “profession”.

Weilie (2004) took a different approach when arguing whether dentistry can be considered a profession. He traces the literal meaning of “profession” to “public avowal” and – although the term itself does not state clearly what has been professed – he claims that there is a tacit declaration to protect and foster the benefit of the public. He continues to argue that a common characteristic of all professions is the promise of altruism which the profession has made to the society. The association of a profession with moral authority and public good is further underscored when Barnett (1997) argues that professionals have a duty – not merely the right – to speak out about issues concerning their expertise to enable lay citizens to make informed decisions. Following Welie, I adopt the second definition of a profession in this paper as such:

"A profession … is a collective of expert service providers who have jointly and publicly committed to always give priority to the existential needs and interest of the public they serve above their own, and who in turn are trusted by the public to do so."

When an occupation is recognized as a profession, a social contract is signed between members of the profession and the general public: in exchange for trust, respect, social status, a practice monopoly and the right of self-government, members of the profession provide their expert service to fulfill some critical need of the community in the best interests of the lay population.

A third and common way to look at how a profession is defined is by studying the characteristics of recognized professions and examining if the occupations under assessment possess the same traits. This approach – also known as the traits or inventory approach – was adopted by Goode, and Etzioni (as mentioned in MacDonald, 1995) extended it to classify occupations into "professions" and "semi-professions" *vis-a-vis* the absence or presence of certain attributes.

Unfortunately, the traits approach to professionalization has a glaring drawback. A billionaire with surplus cash purchases a Bugatti and parks it prominently on his driveway. A billionaire wannabe who wants the world to believe that he is a billionaire studies the billionaire's home, identifies the possession of a Bugatti as a trait, and takes on a major loan to purchase the same car. To passers-by who use the traits approach to identify billionaires, and has "possession of a flashy car" as one of the criteria, this aspirant is a billionaire by that definition, regardless of whether he really has a billion dollars.

Likewise, Esland (1980) thinks that this traits approach has more to do with supporting the professions' own conception of themselves, than with identifying some essential qualities of professional status. Like all other assessment frameworks, the validity and relative importance of each attribute have been criticized (Hall, 1968 and Ritzer, 1973). Roth (1974) detects a shift which "focuses not on the process of professionalization, but on its products", describing this scorecard orientation as being "contaminated with the ideology and hopes of professional groups rather than an independent assessment of what they achieve".

Defining Professionals

So, what about "professionals"[2]? Put simply, professionals are members of a recognized profession (regardless of whether they behave professionally on an individual basis or not). But a deeper look at this apparently obvious definition reveals fundamental difficulties: medicine is a profession, but are all medical practitioners automatically professionals? I would say not: doctors who have their practicing licenses suspended by the relevant Medical Council, and are diagnosing patients and prescribing drugs along some dubious back-alley do not seem to fit the bill for "medical professional".

[2] When discussing about professionals in this context, a distinction has to be made between "occupational professionalism" and "individual professionalism" (Ritzer, quoted in Freidson, 1973). Individual professionalism addresses the attitudes, personal values and utilization of skills and discretion expressed by an individual in the conduct of his work and is independent of the societal position of his particular occupation (Orlikowski and Baroudi, 1989). We are more interested in occupational professionalism i.e. the status of the occupation and whether or not it is legally or socially recognized as professional work. As discussed in Orlikowski, these two types of professionalism may exist independently in the sense that a doctor may behave unprofessionally (even though medicine is a profession), and a janitor may professionally return a wallet he finds.

According to Weilie, society decides which occupations qualify as professions. I think that it is the State which decides which practitioners within the profession qualify as professionals. This is accomplished via state licensing or certification either by a statutory board, or an independent council (such as the Law Society and Medical Council) which has been bestowed legal powers to do so. Within a recognized profession, state licensing or certification draws the line between the trusted professionals and the mere practitioners[3].

Defining Professionalization

"Professionalization" is aptly defined by Kultgen (1988) as "the organization of an occupation into the form assumed by paradigm professions, such as medicine, the law and architecture". In simpler terms, the professionalization of an industry is the process to promote an occupation to a profession. Larson (1977) highlights the benefits of professionalization by providing another definition: "professionalization is an attempt to translate one order of scarce resources – special knowledge and skills – into another – social and economic rewards (for the professionals)". She believes that the outcome of higher social mobility and market control by the profession are not direct reflections of skill, expertise or ethical standards, but rather the outcome of what she calls the "professional project".

Larson's usage of the term "project" implies a clear beginning and end, and a methodological course of deliberate actions planned and implemented by the perpetuators, and that is exactly what the software industry is doing in order to achieve the rewards which accompany profession status. The next section describes how the software industry is undergoing this remarkable – albeit artificial – makeover.

The Professional Project of the Software Industry

The US-based IEEE Computer Society and the Association for Computing Machinery (ACM) - two of the largest IT "professional" societies in the world - are in the best position to quarterback a global professionalization effort. They came together in 1993 to form a Steering Committee for the Establishment of Software Engineering as a Profession to act as a "permanent entity to foster the evolution of software engineering as a professional computing discipline" (Duggins and Thomas, 2002). Subsequently superseded by the Software Engineering Coordinating Committee (SWECC) in 1998, this committee based their initial efforts primarily on Ford and Gibbs's (1996) research. Ford and Gibbs studied several well-established professions and modelled a profession as consisting of the following eight components:

- Initial professional education,
- Accreditation,
- Skills development,
- Certification,

[3] It has to be noted that state licensing itself does not create professions: many activities including driving, beer brewing, gun handling, and even barbering in some US states are regulated. The State cannot create professions - even by generating licenses with explicit monikers such as "Professional Engineer". Society withholds this monopoly, and state licensing merely complements.

- Licensing,
- Code of ethics,
- Professional development and
- Professional society.

They came up with a simple 4-level evolutionary scale (from 0 for "non-existent" to 3 for "maturing") to quantify the maturity of each component and applied it to the field of software engineering. The report concluded that "the current state of the infrastructure of the software engineering profession shows it to be quite immature: only 'professional development' and 'professional society' have begun moving past the ad-hoc (level 1) stage".

In typical engineering fashion, the SWECC started tackling the professionalization challenge by initiating three projects to move the remaining components from "non-existent" to "maturing":

1) *Code of Ethics and Professional Practice for Software Engineering (SWCEPP).* A committee was set up to invent an equivalent of the Hippocratic Oath for the software industry. In 1998, the SWCEPP project was completed and after a year-long review, version 5.2 of the code was finally adopted by both societies (Gotterbarn, Miller and Rogerson, 1999).

2) *Software Engineering Education Project (SWEEP).* SWEEP aims to produce a set of accreditation guidelines for universities offering software engineering degrees. The SWEEP committee analysed the desired student outcomes for a software engineering graduate as compared to those for computer science and computer engineering graduates and published 19 guidelines[4] to help educators implement a software engineering program (Lethbridge et al, 2006). This committee was subsequently superseded by the Software Engineering 2004 (SE2004) Steering Committee.

3) *Software Engineering Body of Knowledge (SWEBOK).* Since "every profession is based on a body of knowledge and recommended practices", SWEBOK is an important project to formalize a set of knowledge every practicing software engineer should know (Tripp, 2003). In 2004, the findings were published in "Guide to the Software Engineering Body of Knowledge" to "provide a consensually validated characterization of the bounds of the software engineering discipline" (Abran, 2004). Documenting the state of software engineering practice was not easy considering how divided software practitioners are on which methodologies and processes are "correct", and the rapid rate at which methodologies, tools and programming paradigms become obsolete. Hence, it is not surprising that the Guide was not unanimously greeted with approval and several academic and industrial experts still will not concur that all the knowledge areas mentioned in the Guide are necessary for a software engineer to carry out his job effectively. The SWEBOK Guide is next scheduled for a much-anticipated refresh in 2010 (Bourque, 2009).

In tandem with the codification of the Body of Knowledge, another important initiative is the Certified Software Development Professional (CSDP) certification program. In 1999, the

[4] The guidelines are published in this document: "Software Engineering 2004 Volume: Recommendations for Undergraduate Software Engineering Curricula" (SE2004)

IEEE Computer Society started work on establishing the CSDP certification examination aimed at mid-level software engineers. One of the five published objectives of the CSDP is to "raise standards of the (software engineering) profession for the public at large" and it is believed that "the CSDP credential … promotes software engineering as a profession" (Tripps, 2002). Acknowledging that not all software practitioners will qualify as CSDPs, a new certification called the Certified Software Development Associate (CSDA) for entry-level software engineers was launched in 2008. It is difficult to assess the success of the CSDP and CSDA programs but there is a general lack of push factors for practicing software engineers to take the certification examination. I also think that it is difficult to set it apart from the other more mature IT software-related certification programmes offered by IT vendors and non-profit organizations.

International IT Professional Practice Programme

Across the Atlantic, the British Computer Society (BCS) focused not only on software practitioners, but cast the net wider to include the whole IT community with its "Professionalism in IT programme" which debuted in 2005. In a white paper published by the BCS, Hughes and Thompson (2007) mentioned three "essential elements" which make a practitioner a professional:

- Competence
- Integrity, responsibility and accountability
- Recognition that professionals have a public obligation

The first two elements are already in the Ford/Gibbs model, and it is a pleasant surprise to see the third. In the same white paper, Hughes and Thompson laid out an ambitious blueprint to build the "international IT profession" in three successive stages:

1) *Definitions and Requirements*. This phrase sees the creation of a common body of knowledge and definitions to ensure consistency.

2) *National Standards*. This stage will witness the creation of standards for individual professional institutions to have clear guidelines for the development of a skills framework, ethical and behavioural standards, competency framework, disciplinary procedures, certification standards and processes.

3) *International Benchmarks*. Cross boundary arrangements are to be made for the accreditation of professional institutions and the international certification of professional practitioners.

The BCS is campaigning its own accreditation programme which it started in 2004: the Chartered IT Professional (CITP)[5] which aspires to be the international qualification found in the third stage of BCS's professionalization roadmap. This 3-phase programme is still in its

[5] The CITP programme was modelled after the British Chartered Engineer qualification. To join the ranks of the 17,000 CITPs today, an IT practitioner has to submit evidence of being in a role which demonstrates "significant influence and responsibility, full accountability, a challenging range of complex work activities and well developed business skills" for at least twelve months. Since 2009, additional requirements have been imposed on CITP candidates: they are required to pass a "Breadth of Knowledge" multiple-choice-question examination which assesses proficiency and pass an interview conducted by two assessors.

infancy, and it will be interesting to see how this grand plan gets implemented in the next few years.

The Licensed Software Engineer?

"Licensing"[6] has been listed as one of the components in the Ford/Gibbs's model, and I have suggested that state licensing is the benchmark differentiator between the professionals and the practitioners within a profession. In the US, professional licensure for engineering disciplines is decentralized, and independently regulated by each state's jurisdiction. A milestone was achieved in 1998 when the Texas Board of Professional Engineers decided to recognize software engineering as a distinct discipline of engineering and to require that software engineers be licensed as Professional Engineers to practise in Texas (Charles, 1998 and Dorchester, 1999). Inevitably, this move sparked off a huge debate about the pros and cons of licensing software engineers. Considerable resistance to Texas's decision arose especially from companies which employed software engineers who would be required to be licensed (Boykin, 2007).

The proponents of licensing software engineers argue that it grants the industry "professional" status, is an inevitable step in the professionalization process, and protects the public by ensuring that a minimum acceptable standard is achieved by those licensed. The detractors argue that these advantages may apply to most other mature industries, which software engineering is not. Knight and Leveson (2002) explain why Texas's proposal to regulate software engineers was impractical:

- The Fundamentals of Engineering examination required for licensing as a Professional Engineer is inappropriate for software engineers. (A practicing software engineer really does not need to know Ohm's Law.)
- The time required to update the examinations does not match the rapid rate of technology change in computer science.
- The breadth of people involved in the production of software would make licensing of all of them impractical and not particularly helpful.

Another stumbling block was the absence of a "reasonable number" of EAC/ABET-accredited[7] programs offering an undergraduate degree in software engineering at that time (Thornton, 2009).

To study the (potentially large) implications that licensing software engineers will have on Texas – as well as other states which will use Texas as a model for similar regulation – the ACM established the Task Force on Licensing of Software Engineers Working on Safety–Critical Software to analyse the implications. Findings recommended that "ACM take a stand against government efforts to require the licensing of software engineers as impractical, ineffective with respect to protecting public safety, and potentially detrimental with respect to

[6] A distinction has to be drawn between licensing and certification: the latter is "usually not a legal or state-controlled process" (Knight and Leveson, 2002), while licensing is usually controlled by the government.

[7] ABET is the Accreditation Board for Engineering and Technology. ABET is a non-governmental organization responsible for the specialized accreditation of educational programs in applied science, computing, engineering and technology in the USA. EAC is the Engineering Accreditation Commission, and is specifically responsible for engineering programs.

economic and other societal and technological factors", prompting the ACM Council to pass the following motion in 1999:

> *"ACM is opposed to the licensing of software engineers at this time because ACM believes it is premature and would not be effective at addressing the problems of software quality and reliability…"*

ACM reiterated its position that the organization was not against software engineering being viewed as a profession. In fact ACM "believes it is important to foster the emergence of a true IT profession, not just software engineering". It is just that "a field does not need licensing to be a profession" (White and Simons, 2002).

At the same time, another ACM task force (the Task Force on Assessment of the Software Engineering Body of Knowledge Efforts) criticized the way the SWEBOK project was progressing as "(failing) to recognize the gap between actual practice and what appears in textbooks". Predicting that the SWEBOK effort is "highly likely to fail", the task force recommended that ACM "refrain from pursuing activities like SWEBOK that have a significant chance of reducing the public's understanding of, confidence in, and assurance about key properties of software" (Notkin, Golik and Shaw, 2000). In a dramatic turn-about, the ACM Council withdrew support for the SWEBOK project and resigned from the SWECC, thus ending a short era of collaboration. The two largest professional societies for IT practitioners seem to be diverging on how the professionalization process should proceed.

The push for licensing did not lose steam: in 2007, lobbyists for software engineer licensing including representatives from the Texas Board of Professional Engineers, IEEE-USA, IEEE Computer Society and the National Society of Professional Engineers (NSPE) came together to establish the Software Engineering Licensure Consortium (SELC). The SELC coordinated collection of licensure board letters of support from 10 states in the US and convinced the Board of Directors of the National Council of Examiners for Engineering and Surveying (NCEES)[8] to approve the development of a Principles and Practice of Engineering (PE) examination for the discipline of software engineering with IEEE-USA as the lead sponsor. This new PE examination is expected to debut in 2012 (Society Recognizes, 2009) and will be independent from the existing electrical and computer engineering PE examinations with only an estimated 20% syllabus overlap (Thornton, 2009). IEEE-USA's intention is that only software engineers offering services directly to the public will need to be licensed, and only software which affects the health, safety and welfare of the public will require oversight by a licensed software engineer (Society Recognizes, 2009).

The authorization of the development of a software engineering PE examination under NCEES's auspices is definitely a breakthrough, but it is still up to the various US states to decide if they need software engineers to be licensed after the examination is ready. Today, the software industry remains unregulated in most countries, and the implications of this on the professionalization process, as well as the resulting perceived professional status of the software industry remain unclear.

[8] NCEES is the organization that develops, administers and scores the examinations used for engineering and surveying licensure in the USA. The Council's members are the engineering and surveying licensure boards from all states in the USA (NCEES, 2010 March).

The Aftermath: Is Software Engineering a Profession?

The professionalization efforts of the IT societies have largely shaped the current perceived "professional" status of the software industry. To determine if these efforts have come to fruition, I will examine the software occupation today against the three definitions of "profession" listed earlier.

Is Software Engineering a Profession by Freidson's Definition?

Based on Freidson's definition, Orlikowski and Baroudi (1989) derived four criteria to argue that information systems workers are not professionals. I shall use the same criteria to examine if software practitioners have achieved the same level of autonomy today:

- *Technical autonomy.* Professions exhibit power to set the parameters of their own work via peer review or evaluation, rather than depend on an external controlling entity to determine if an action is correct or wrong, acceptable or not. Software practitioners typically do not employ a system of peer review, and unlike lawyers, doctors or civil engineers, are not personally or legally liable for defects in their services or software products in most countries. Despite the invention of the Code of Ethics, there is no software council that convenes disciplinary hearings on mal-practising software engineers. Nor is there any system of censure or debarment for defaulters. The Code of Ethics wasn't constructed to be obligatory or legally binding, and it lacks state and community backing to be so.

- *Control over education and training process.* Although the SE2004 guidelines for planning software-related courses in universities are in place, conformance to these guidelines is not mandatory or globally widespread. Bourque (2009) claims that SWEBOK 2004 has served as "a major input" to the design of curricula of undergraduate and graduate courses, corporate training and certification programs but it is difficult to verify that SE2004 has influenced the course structure of existing software engineering programs. In fact, a recent study of 28 Master's degree programmes in software engineering offered largely by US universities concludes that "few programs cover all SWEBOK areas well", and "many programs skip some SWEBOK areas completely" (Pyster et al, 2009).

  The SE2004 committee does not validate conformance to the recommendations, and SE2004 remains what it was designed to be: mere guidelines. Today, practising software practitioners still have vastly different educational backgrounds ranging from nil (totally trained on-the-job) to Ph.D. holders majoring in software engineering. Even with the SWEBOK Guide formalized, the IEEE Computer Society clearly still does not have control over its formal knowledge base. The CSDP certification and CITP qualification do not seem to have gained immense popularity amongst current practitioners[9], and acceptance as employment prerequisites amongst employers as well.

- *Competition and regulation from other occupations.* The popular portrayal of software systems is that of a business enabler: software systems do not exist in isolation but are constructed out of specific business needs. The consideration of software practitioners do

[9] In October 2008, the IEEE Computer Society claims that there are "nearly 1000" CSDPs in the world (Society Recognizes, 2009). As of August 2008, 17,000 of the 67,000 BCS members are holding CITP status.

not form the strategic priorities of most companies (unless they are employees of a software house), but are actually influenced by the key business units which drive the company's financial bottom-line.

- *Control over other occupations and clientele.* Doctors and lawyers exert enormous influence over nurses and paralegals, and their professional opinions are not usually challenged by clients. On the other hand, the work of software practitioners is largely quarterbacked by business sponsors (the clientele) who commission the software project. Although the business sponsors and end users will usually not dictate the internal design and architecture of the software product, they "control" the project via explicit requirements laid out at the onset. There is no evidence that software practitioners manipulate the business sponsors or exert control over any other category of occupations.

In summary, it does seem that despite the professionalization efforts, software practitioners still do not exhibit the necessary levels of professional autonomy or control over their clients, other occupations or the prerequisite educational or training requirements to practise as suggested by Freidson.

Is Software Engineering a Profession by Weilie's Definition?

Unlike Freidson, Weilie emphasized on a profession's altruistic commitment to give priority to the existential needs of the public. This condition is extremely difficult to fulfill given the nature of the software industry. When we break an arm, we consult a doctor to mend it; the basic medical care provided by doctors is an existential need. In a modern democratic society governed by the Rule of Law, the needs for basic human rights are fundamental and we consult a lawyer when such essential rights are threatened. For the case of software, the first question which comes to mind is: can software be considered an existential need? Software products and services are very important elements in the business ecosystem and are increasingly viewed as key differentiating factors for profit-generation. Nonetheless – cyborgs aside – it is unlikely that products of software practitioners will become "existential" needs for life sustenance in the foreseeable future.

Let me stretch the argument a bit: assume that software is so prevalent in our society that it gets elevated to "existential need" status because life as we know it stops functioning without software. Even if that was true, there is no indication that the software industry as a whole has ever committed to give priority to such needs in an altruistic manner, and the public is unlikely to trust the software industry to draw from a core of morality when they create software programs.

Like banking, the software industry stemmed from a purely commercial origin (some say military), and the general public does not associate altruism with the nature of software. It is a perception issue: doctors lessen pain, firemen save lives, soldiers fight for the country, but the software industry produces code for profit. In the absence of an obvious link between altruism and software production, the social contract between the laity and the software industry is unlikely to materialize. Based on Weilie's definition of a profession, the software industry's professionalization scheme seems quite hopeless, especially so since it was never designed to inject altruism into practitioners in the first place.

Is Software Engineering a Profession by the Traits Approach?

The professionalization efforts are largely guided by the Ford/Gibbs's model of professions, which is formulated based on the traits approach. So it is most sensible to benchmark the success of the efforts against this model. A comprehensive evaluation of the current state of the software industry based on the eight components is beyond the scope of this paper, but it is obvious there are improvements to different extents in each component:

- *Initial Professional Education:* SE2004 guidelines were finalized for undergraduate courses leading to degrees in software engineering.
- *Accreditation:* ABET is the major accreditation body for undergraduate and graduate programs for software engineering in the USA, and by 2009, there were 17 EAC/ABET-accredited programs in the USA (Thornton, 2009). The IEEE Computer Society owns the CSDP certification program. In the UK, the BCS is the formal accreditation body for the CITP qualification.
- *Skills Development:* SWEBOK 2004 was certainly a milestone, though it was not met with universal applause. Efforts at skills development can be based on the framework listed in the SWEBOK guide.
- *Certification:* SWEBOK 2004 forms a bedrock upon which certification examinations can be set. Though the success of the CSDP certification program is difficult to ascertain, at least a certification program exists.
- *Licensing:* Currently software engineers are licensed as professional engineers in Texas, some Canadian provinces and Australia (IEEE Computer Society, 2008 Sept.). NCEES is developing a national-wide examination for software engineering which is expected to be ready in 2012, but it is still up to various states to decide if software engineers working within their jurisdiction need to be licensed. In the UK, the CITP chartered status is more appropriately categorized as a "license" for IT practitioners instead of a certification because its issuance is backed by the government, but there is still no legal requirement for practising software engineers to be CITPs.
- *Code of Ethics:* There is now a Code of Ethics and Professional Practice for Software Engineers, even though it may not be well known (even amongst software practitioners) or bears any real significance to the industry. More effort is required to garner recognition and acceptance of this pledge.
- *Professional Development:* Like the "skills development" component, SWEBOK 2004 helps to provide common ground for the development of professional development programmes. Despite that, professional development programmes span a wide spectrum and lacks a centralized accreditation body and standardized quality assurance.
- *Professional Society:* The major global IT/software societies such as the IEEE Computer Society, ACM and BCS have voluntarily assumed leadership roles in several "professional" initiatives. IEEE Computer Society and ACM do collaborate on major standardization projects, but have disagreed vehemently on key policies (such as the direction which SWEBOK and licensing are going). Unfortunately, the diverse political agendas driving each professional society, the lack of local support for such societies,

limited cross-organizational collaborations and inter-societal rivalry are obstacles to international standardization and coordinated effort.

It is not surprising that the professionalization efforts by the professional societies over the last decade have significantly improved software engineering's standing in every component of the Ford/Gibbs's model since the action plan was architected based on it. The model does not propose a minimum score for each component to be achieved before software engineering becomes a profession, but if the professionalization efforts continue at the same pace, the discipline should be assured that most of the components would have progressed to the "specific" (stage 3) or "maturing" (stage 4) phases within another decade.

Conclusion

This article started with a discussion about the definitions of "profession" from three different sources: Freidson, Weilie and the traits approach. This is followed by a summary of the professionalization efforts for software engineering undertaken by the professional societies in the past two decades, which include the creation of a Code of Ethics for software engineers, the codification of a Body of Knowledge, the creation of standards for software engineering education, development of a software developer professional certification and progress toward the licensing of software engineers as a distinct category of Professional Engineers.

In the third part of this article, I measured the current status of the software engineering discipline against the three definitions of "profession". I believe that the software industry still falls significantly short of Freidson's prerequisite of technical autonomy, and fails to satisfy the altruistic requirements of Weilie. But from the trait's approach perspective, the professionalization process has made momentous progress.

Acknowledgements

The author would like to acknowledge Kelvin Pang, Noi San Koh and the anonymous reviewers who have provided useful comments.

References

Abran, A., Moore, J.W. et al. Ed. (2004). *Guide to the Software Engineering Body of Knowledge (2004 Version)*, p. xvii. IEEE Computer Society

Barnett, R. (1997). *Higher Education: A Critical Business*, chap. 10. Buckingham: Open University Press

Bourque, P. (2009). SWEBOK Refresh and Continuous Update: A Call for Feedback and Participation. *Proceedings of the 2009 22nd Conference on Software Engineering Education and Training*, pp. 288-289. IEEE Computer Society

Boykin, D. (2007). Is it Time to License Software Engineers? *PE Magazine*, Dec. 2007, pp. 26-29. National Society of Professional Engineers

Charles, J. (1998). A License to Code. *IEEE Software*, Sep/Oct 1998, 15, 5, pp. 119-121. IEEE Computer Society

Cogan, M.L. (1953). Toward a Definition of Profession. *Harvard Education Review*, 23 (11), pp. 33-50. Harvard Education Publishing Group

Dorchester, D. (1999). Why License Software Engineers. *IEEE Software*, Mar/Apr 1999, 16, 2, pp. 101-102. IEEE Computer Society

Duggins, S.L. and Thomas, B.B. (2002). A Historical Investigation of Graduate *Software Engineering Curriculum. Proceedings of the 15th Conference on Software Engineering Education and Training* (CSEET'02), pp. 78-87. IEEE Press

Esland, G. (1980). Professions and Professionalism. *The Politics of Work and Occupations*, pp. 213-250. Toronto: University of Toronto Press

Fernández-Sanz, L. and García- García, M-J. (2003). Software Engineering Professionalism. *Upgrade*, IV, 4, August 2003, pp. 42-46. Council of European Professional Informatics Societies

Ford, G. and Gibbs, N.E. (1996). *A Mature Profession of Software Engineering*. (Tech. Report CMU/SEI-96-TR-004). Pittsburgh: Software Engineering Institute (Carnegie Mellon University)

Freidson, E (1970). *Profession of Medicine: A Study of the Sociology of Applied Knowledge*. p. 71. New York: Dodd, Mead & Co.

Gotterbarn, Miller and Rogerson (1999). Computer Society and ACM Approve Software Engineering Code of Ethics. *IEEE Computer*, Oct., pp.84-88. IEEE Computer Society

Hall, R. (1968). *Professionalization and Bureaucratization*. American Sociological Review, 33, 1, Feb., pp. 92-104. American Sociological Association

Hawkins, J.M. Ed. (1994). *Oxford English Mini Dictionary* (Rev. 3rd Ed.). Oxford: Claredon Press

Hughes, C. and Thompson, C., (2007). *The International IT Professional Programme.* British Computer Society

IEEE Computer Society (2008, Sep.). *IEEE CS Certification Programs*. Retrieved Sep, 1, 2008, from http://www2.computer.org/portal/web/csda/faq

IEEE Computer Society (2009). Society Supports Software Engineering PE Examination. *IEEE Computer*, Nov 2009, pp. 87-88. IEEE Computer Society

Knight, J.C. and Leveson, N.G. (2002) Should Software Engineers Be Licensed? *Communications of the ACM*. Vol. 45, Issue 11, Nov. pp. 87-90. Association of Computing Machinery

Kultgen, J. (1988), *Ethics and Professionalism*, p. 100. Pennsylvania: University of Pennsylvania Press

Larson, M.S. (1977). *The Rise of Professionalism: A Sociological Analysis*. California: University of California Press

Lethbridge, T.C, LeBlanc, R.J.Jr, Sobel. A.E.K.S, Hilburn, T.B., Diaz-Herrera, J.L. (2006). SE2004: Recommendations for Undergraduate Software Engineering Curricula, *IEEE Software*, 23, 6, pp. 19-25. IEEE Computer Society

Long, L.N. (2008). The Critical Need for Software Engineering Education. *CrossTalk: The Journal of Defense Software Engineering*, Jan 2008, pp. 6-10. Software Technology Support Center

MacDonald, K.M. (1995). *The Sociology of the Professions*, Chap. 1. London: Sage.

NCEES (2010, Mar). *About NCEES*. Retrieved Mar, 1, 2010, from http://www.ncees.org/About_NCEES.php

Notkin, D., Gorlick, M. and Shaw, M. (2000). *An Assessment of Software Engineering Body of Knowledge Efforts (A Report to the ACM Council)*. Association of Computing Machinery

Orlikowski, W. & Baroudi, J. (1989). The Information Systems profession: Myth or Reality? *Office: Technology and People*, 4(1), pp. 13-31. Cambridge, Mass: Center for Information Systems Research, Alfred P. Sloan School of Management

Pyster, A., Turner, R., Henry, D.,Lasfer, K. And Bernstein, L. (2009). Master's Degrees in Software Engineering: An Analysis of 28 University Programs. *IEEE Software*. Sep/Oct 2009, pp. 94-101. IEEE Computer Society

Ritzer, G. Professionalism and the Individual. In Freidson, E., Ed. (1973). *The Professions and their Prospects* (2nd Ed.), pp.59-74. Sage Publications

Roth, J. A. (1974). Professionalism: The Sociologist's Decoy. *Sociology of Work and Occupations*, 1,1 (Feb.), pp. 6-23. London: Sage Publications

Savan, B. (1989). Beyond Professional Ethics: Issues and Agendas. *Journal of Business Ethics*, 8, pp. 179-185. Netherlands: Springer

Society Recognizes (2008), Society Recognizes Newly Certified Software Developers, *IEEE Computer*, Oct. 2008, p. 83. IEEE Computer Society

Thornton, M. (2009). Software Engineering PE Examination Development Approved. *IEEE USA Today's Engineer Online*. Retrieved March, 1, 2010, from http://www.todaysengineer.org/2009/Sep/Software-PE.asp

Tripp, L. L. (2002). Benefits of Certification. *IEEE Computer*, Jun., pp. 31-33. IEEE Computer Society

Tripp, L. L. (2003). *Guide to Software Engineering Body of Knowledge (2004 Version)*, p. viii. IEEE Computer Society

Welie, J.V.M. (2004). Is Dentistry a Profession? Part 1: Professional Defined. *Journal of the Canadian Dental Association*, Sep. 2004, 70, 8, pp. 529-532. Canadian Dental Association

White, J. and Simons, B. (2002). ACM's Position on the Licensing of Software Engineers. *Communcations of the ACM*. Vol. 45, No. 11 Nov., p. 91. Association of Computing Machinery

Acronyms

ABET	Accreditation Board for Engineering and Technology (USA)
ACM	Association of Computing Machinery
BCS	British Computer Society
CITP	Chartered IT Professional (UK)
CSDA	Certified Software Development Associate
CSDP	Certified Software Development Professional
EAC	Engineering Accreditation Commission (USA)
IEEE	Institution of Electrical and Electronic Engineers
IT	Information Technology
NCEES	National Council of Examiners for Engineering and Surveying (USA)
NSPE	National Society of Professional Engineers (USA)
SE2004	Software Engineering 2004 Volume: Recommendations for Undergraduate Software Engineering Curricula
SELC	Software Engineering Licensure Consortium
SWCEPP	Code of Ethics and Professional Practice for Software Engineering
SWEBOK	Software Engineering Body of Knowledge
SWECC	Software Engineering Coordinating Committee
SWEEP	Software Engineering Education Project

Author's Biography

H.N. Mok is a senior instructor at the School of Information Systems, Singapore Management University where he teaches software architecture and development.