Software Engineering Professionalism: Is the End of Constraints and Conflicts in Sight? *

T. H. Tse
Department of Computer Science and Information Systems
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: tse@csis.hku.hk

1. Introduction

It is two years since I participated in the panel session entitled "Is the Harmonisation of Professional Standards for Software Engineers Feasible? Or Even Welcome?" in COMPSAC 2000. I highlighted a number of constraints and conflicts at that time [10]. When we revisit the issue today, is the end of the problems in sight?

2. Traits of Professionalism

The following have long been recognized as the essential traits of professionalism:

(*a*) A body of knowledge unique to the profession should be identified.

(*b*) The profession should show high levels of responsibility and accountability. A code of ethics and professional practice should be defined. Members of the profession should be motivated towards public service over and above personal gain.

(*c*) Members should have demonstrated competency in the profession, are committed to professional development, and have obtained a certification by a professional organization or a legally recognized license to practice it.

Are these applicable to software engineering?

2.1. Body of knowledge

Two years ago, I raised the concern about the confusion between the body of knowledge for software engineering and that for other disciplines such as computer science and conventional engineering. On one hand, there has been tremendous amount of progress in identifying a body of knowledge [4, 8]. The earlier confusion is no longer applicable in these proposals. On the other hand, this concept has not yet made its way to the universities. For example, the software engineering curricula reported in [2, 9] are still modelled after computer science and traditional engineering.

People in the industry have not yet recognized the need for an independent body of knowledge for software engineers either. For example, Beizer [1] proposed the following qualities that we should look for in an ideal software tester:

> Know programming, know the application, intelligence, hyper-sensitivity to little things, tolerance for chaos, people skills, tenacity, organized, skeptical, self-sufficient and tough, cunning, technology hungry, and honest.

I for one would be very reluctant to consult a medical doctor, lawyer, or accountant with these traits.

2.2. Responsibility and accountability

According to the IEEE Computer Society / ACM Joint Task Force on Software Engineering Ethics and Professional Practices [5],

> Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment.

Various people, however, use this as one of the reasons why software engineering should not be licensed. It is argued in [6], for instance, that

1) Our profession is not currently subject to malpractice litigation.
2) Becoming licensed will transform us into a profession whose members can be sued for malpractice. ...

* This research is supported in part by a grant of the Research Grants Council of Hong Kong and a CRCG grant of the University of Hong Kong.

5) Certification of software engineers might open certified members to malpractice suits. ...

7) Malpractice insurance premiums will become a significant tax on members of our profession.

Similar arguments have been put forward in [7].

In fact, we are witnessing a typical example of the conflict of interests. When the well-being of a profession contradicts with that of the general public, public interest should be given a higher priority. Thus, rather than being a nuisance, this is exactly the kind of protection that should be given to the public in large through licensing.

2.3. Certification and licensing

There has been a lot of debates for and against certification and licensing of software engineers. The strongest argument against licensing comes from the ACM Advisory Panel on Professional Licensing and Software Engineering. The key rationale for not licensing ourselves as professional engineers is that the latter has to take an examination on the Fundamentals of Engineering, which covers chemistry, computers, dynamics, electrical circuits, engineering economics, ethics, fluid mechanics, material science/structure of matter, mathematics, mechanics of materials, statics, and thermodynamics [7].

We should note, however, that the licensing of software engineers should not be restricted to professional engineers. This is yet another example of the confusion between the body of knowledge for software engineers and that for conventional engineers. Furthermore, the constraints of the movement towards software engineering professionalism in the United States does not necessarily entail the same kind of obstacle in Europe and Asia. For example, there is much less resistance about the licensing of software engineers from conventional engineers in Europe.

3. Conclusion

To conclude, even though we have seen some trends towards software engineering professionalism, the end of the long and winding road is not yet in sight. The strongest resistance comes from our own inertia, such as the lack of an independent body of knowledge, the confusions in accountability and responsibility, and the obstacles in certification and licensing. Internal conflicts do not help to make our profession more mature. Let us join hands in the movement towards a genuine software engineering profession.

I concur with Hawthorne [3]:

> Maybe the people doing the licensing [are] not perfect. Maybe the process [is] not perfect. But it [is] a necessary step.

References

[1] B. Beizer. Qualities of a good tester: a baker's dozen. *Quality Techniques Newsletter*, Software Research, Inc., March 2002. Available at "http://www.soft.com/News/QTN-Online/qtnmar02.html".

[2] Curriculum for the Software Engineering Program. Department of Computer Science, Concordia University, Montreal, Quebec, Canada, 2002. Available at "http://www.cs.concordia.ca/programs/ugrad/soen/curriculum.html".

[3] P. Hawthorne. The ACM and the licensing for software engineers. Position Papers, ACM Advisory Panel on Professional Licensing and Software Engineering, 1999. Available at "http://www.acm.org/serving/se_policy/papers.html#hawthorne".

[4] IEEE Computer Society / ACM Joint Task Force on Computing Curricula 2001 (CC2001). Software engineering education knowledge (SEEK) areas (draft). Available at "http://sites.computer.org/ccse/artifacts/KADescriptions.htm".

[5] IEEE Computer Society / ACM Joint Task Force on Software Engineering Ethics and Professional Practices. Software engineering code of ethics and professional practice (version 5.2). Short version available at "http://computer.org/tab/seprof/code.htm".

[6] C. Kaner. Software engineering as a profession after the withdrawal: one year later. *Forum for Advancing Software Engineering Education* (*FASE*), 11 (9), 2001. Available at "http://www.cs.ttu.edu/fase/v11n09.txt".

[7] J. Knight, N. Leveson, M. DeWalt, L. Elliot, C. Kaner, and H. Nissenbaum. Final report of an ACM task force on licensing of software engineers working on safety-critical software. 2001. Available at "http://www.acm.org/serving/se_policy/safety_critical.pdf".

[8] T. C. Lethbridge. What knowledge is important to a software professional? *IEEE Computer*, 33 (5): 44–50, 2000.

[9] J. B. Thompson and H. M. Edwards. Software Engineering in the UK 2001. *Forum for Advancing Software Engineering Education* (*FASE*), 11 (11), 2001. Available at "http://www.cs.ttu.edu/fase/v11n11.txt".

[10] T. H. Tse. Towards harmonized professional standards for software engineers: constraints, conflicts and concessions. In *Proceedings of the 24th Annual International Computer Software and Applications Conference* (*COMPSAC 2000*), pages 346–347. IEEE Computer Society Press, Los Alamitos, California, 2000.