CS-480—Senior Seminar
Summary of *Software Certification Debate*
Fall '02

Jim Rogers

Leonard L. Tripp, *Benefits of Certification* and Adam Kolawa, *Certification Will Do More Harm than Good*

Tripp, the chair of the IEEE-CS Professional Practices Committee, starts by quoting Ford and Gibbs's characterization of "A Mature Profession of Engineering" in terms of a trajectory that starts with professional education and accreditation and moves through skills development and certification to licensing, professional development, professional societies and a code of ethics. (Tripp explicitly adds professional standards.) The IEEE Computer Society's Certified Software Development Professional examination is intended to provide the certification step for Software Engineering.

He sees benefits in this program for software professionals (both individually and as a profession), for the industries they work in and for the public at large. The profession benefits from support of minimum competency standards and improved awareness and use of best practices. Industry benefits both from the use of these practices and from having a standardized benchmark by which to evaluate the skills of software professionals. The public at large benefits in the same way and from better public education about software engineering theory and practice. Finally the individual benefits from the opportunities certification and re-certification provide for professional development, from the ability to identify themselves as competent in software engineering and from raised standards throughout the profession.

Tripp emphasizes that CSDP is a professional certification program, not licensure. It is voluntary and not intended to create a barrier to entering the profession. It is an assessment tool and not a guarantee of competency. Moreover, he does not expect that all software professionals need to be certified. He does not, on the other hand, provide any assurances that CSDP will not be used in the ways not intended. And given that he starts his comments with a model of a training trajectory in which

certification is an intermediate step, it is not clear that this certification program is not, itself, an intermediate step along the way to a campaign for licensure.

Kolawa (the CEO of a software tools company and a physicist by training), on the other hand, voices concern about many of the things Tripp claims CSDP is not intended to be. His central claim is that bad software does not come from lack of skills or awareness of principles and best practices on the part of software professionals but, rather, from lack of tools that integrate those principles and practices into the production process. Thus CSDP is attacking the problem of (lack of) software reliability at the wrong point. He believes this sort of certification will not only fail to improve software quality but may degrade it if managers hire certified developers and then simply assume that they will produce high quality systems.

He is concerned that whether or not it is intended to be a barrier to entry into the profession, CSDP will make it hard for non-certified professionals to find employment and hard, and even potentially illegal, for managers to hire them. Moreover, he is concerned that the knowledge required for certification may become a limit rather than a lower bound—professionals with CSDP and their managers may feel little need for further professional development beyond the requirements for re-certification, and acceptance of innovative methodologies that have not been blessed by inclusion in the CSDP examination may be delayed or blocked.

He argues, further, that certification should be an unnecessary duplication of the level of assurance of skills that should be provided by a degree from an accredited CS program. On the other hand, he also claims that the best software professionals he has hired are not graduates of such program and these seem to be the very people he is afraid will be effectively barred by certification. While he, correctly, points out that verification that software professionals both possess adequate skills and are applying them effectively is the responsibility of their manager, he has no suggestions for how managers in general are to do this verification in the absence of certification.

Finally, he makes his own analogy with established engineering disciplines, claiming that Computer Science is not actually a science in the sense of Mathematics or Physics on the grounds that software practices are often dominated by fads which surface and resurface with disturbing rapidity. He points out that Math and Physics do not have certification and owe their strength to their reliance on the scientific method rather than the approval of professional societies to validate innovation.

He seems to have Computer Science and Software Engineering confused here.

It has often been pointed out that Software Engineering is not a true engineering discipline for exactly the reasons Kowala notes: it is not solidly founded in first principles of the underlying science but is still dominated by "rules-of-thumb". On the other hand, the immaturity of the associated engineering profession does not necessarily reflect the maturity of the underlying discipline. Although Computer Science is not a mature discipline in a great many ways, lack of development of Software Engineering is probably not one of the more critical ones. Unfortunately, if one refocuses his remarks towards Software Engineering rather than Computer Science, his intended conclusion no longer obtains: the engineering disciplines associated with the traditional sciences almost universally depend not only on certification but on actual licensure.