

# SHOULD SOFTWARE ENGINEERS BE LICENSED?

John C. Knight and Nancy G. Leveson

Software is being used increasingly in systems that affect public safety and where software errors might lead to unacceptable losses. This use has led to suggestions that software engineers working on safety-critical systems should be licensed as Professional Engineers (PEs) in order to protect the public interest.

To determine whether the ACM should support such efforts, a task force was formed to investigate the implications of licensing software engineers as PEs. We chaired that task force, members of which included Michael Dewalt (an expert in the certification of commercial aircraft software), Lynn Elliot (vice president of Guidant, which makes implantable medical devices), Cem Kaner (a lawyer and software testing expert), and Helen Nissenbaum (an expert in ethics and computers).

As part of our investigation, we met with the Executive Director and Deputy Executive Director of the National Society of Professional Engineers (NSPE), the head of the group that provides PE examinations, representatives from agencies that regulate safety-critical systems, supporters of licensing from the software engineering community, and others. From what we learned, the task force concluded that licensing within the framework of the existing PE mechanism would not be practical or effective in pro-

tecting the public and might even have serious negative consequences. Instead, a comprehensive approach is required that might incorporate a number of changes in culture, practice, and regulation.

Here, we present the reasons behind the conclusions of the task force and briefly present some alternatives and a way forward.

## What is Licensing as a Professional Engineer?

Licensing is a state activity required of those who provide certain types of services directly to the public, such as contractors, physicians, electricians, barbers, and child-care providers. A *professional engineer* is an individual who has been granted by a governmental authority the right to use that title and to offer professional engineering services to the public. Every state currently has an engineering licensing law, but the details vary from state to state. Enforcement also varies. For example, the state of Oklahoma requires that faculty teaching engineering design classes be licensed, but this part of the Oklahoma law has not been enforced.

The National Society of Professional Engineers (NSPE) provides a model law and lobbies legislatures to adopt licensing regulations. The model law requires a four-year degree from a university program accredited by the Engineering

Accreditation Committee (EAC) of the Accreditation Board for Engineering and Technology (ABET); an eight-hour examination on the fundamentals of engineering (FE) usually taken in the senior year of college; four years of acceptable experience; a second examination on principles and practice; and written recommendations from other professional engineers. Some states also require mandatory continuing education.

A PE in a field where licensing is required must be licensed in every state in which he or she practices. Most states charge licensing fees ranging from \$80 to \$200 per year. Some states allow licenses to be granted on the basis of possession of a valid license in another state whose requirements “meet or exceed” their own, but fees must still be paid in each state. Mandatory licensing is usually required for those offering engineering services directly to the public and for those involved in the design of facilities, roads, transportation, and construction where design documents must be submitted to state agencies for approval. It is important to note that, in the latter case, it is only the individual authorizing designs for submission who needs to be licensed despite the fact that arbitrarily large numbers of people might have been involved in the development of the design.

The requirements for licensing as a PE in order to practice the profession varies from state to state. Nearly all states provide an exception to required licensing for those who work in industry (that is, they are employed by a company) or those who are federal government employees. The number and type of engineers who are licensed also varies. Relatively few engineers are licensed as PEs, reflecting the fact that few directly serve the public or sign engineering documents required by the state. At the time of our investigation (2001), the field of engineering with the highest percentage of PEs was civil engineering (about 40%) with mechanical engineering having the second highest. The lowest percentage was found in electrical engineering (10%). At that time, the percentage of mechanical and electrical engineers going through the licensing process was decreasing while the percentage of civil engineers was holding steady.

It is important to note that the PE license in most states says “Professional Engineer” and does not distinguish among subdisciplines. Thus, even in a situation where an engineer is licensed, it is the responsibility of the individual to judge their competence to undertake the required engineering activities. PEs can, however, be disciplined by the state (fined or lose their license) if they practice beyond their level of competence. Licensed PEs are accountable for their activities and assume legal liability.

## Would Licensing Software Engineers be Practical?

After learning what was required to be licensed as a PE, our task force decided the process *as constituted when we examined it* would not be practical for software engineering. The reasons for this conclusion involve:

**The FE examination, required for licensing as a PE, is inappropriate for those receiving computer science degrees.** As part of the PE licensing process, every prospective PE must take an eight-hour FE examination composed of a morning general examination designed to match the ABET criteria for the first two years of an engineering degree, and a discipline-specific afternoon examination covering the material taught in the last two years of an engineering degree program. The morning general examination includes such topics as chemistry, dynamics, fluid mechanics, materials science, the structure of matter, electrical circuits, continuous mathematics, mechanics of materials, statics, and thermodynamics. Studying all these topics would leave little time for studying computer science and the topics more relevant to software engineering. The afternoon, discipline-specific examination can be taken in chemical, civil, industrial, electrical, or mechanical engineering or a general test can be taken that covers the topics in the morning examination in more detail. Because of a requirement that there be at least 100 ABET-accredited departments offering a degree in a subject to be included as a discipline-specific examination, software engineering is not eligible for inclusion at this time. To our knowledge, there are currently no ABET-accredited software engineering undergraduate degree programs in the U.S.

**Not all practicing software engineers graduate from departments that are eligible for ABET accreditation.** Some computer science departments are not in engineering schools, and even those departments that are in engineering schools rarely require classes in most of the areas of engineering tested by the FE examination. Many competent software engineers do not have degrees in computer science.

**Large software development efforts often span multiple states, and the resulting software might be sold in every state.** Requiring software engineers to be licensed in every state in which a company for which they work has branches and offices or the software they produce is used would be impractical. Some states have reciprocity agreements, but this is far from universal and software engineers would still be required to go through the approval process and pay the yearly licensing fees in each state.

**The time required to update the examinations does not match the rapid rate of technology change in computer science.** The typical three-year cycle to

update a licensing examination is not practical for many aspects of IT, where changes are rapid and continual. The examination would most likely be out of date most of the time unless the examination update cycle time could be reduced. Even more important, licenses in most states are valid for life. This raises the issue of whether a software engineer who received a degree in 1970, for example, and has not updated his or her knowledge and skills since then is still qualified to create safety-critical software. Some states require some type of continuing education for licensed engineers, but most do not and this requirement raises additional problems.

**It is unlikely that any reasonable test for software engineering skill in safety-critical systems could be put into a multiple-choice format.** Because of legal challenges from those test takers who did not get full credit for subjectively graded questions, NCEES (the group responsible for the PE examinations) has changed the examination to use only multiple-choice questions. There must be one answer that is correct, and the rest must be demonstrably incorrect. It is unlikely that any reasonable evaluation of software-engineering skill for safety-critical systems could be put into such a format.

**The breadth of people involved in the production of software would make licensing of all of them impractical and not particularly helpful.** As in the development of significant systems in fields such as civil engineering, many people with a variety of responsibilities are involved in the development of virtually any safety-critical software system. Of those involved, who would be included in the licensing requirement: Requirements writers? Designers? Coders? Software librarians? Quality assurance and test personnel? Project managers? Those who write system software or COTS software? Is a person who uses a spreadsheet, designs a Web site, or configures a SAP system practicing software engineering?

### **Would Licensing Software Engineers Protect the Public?**

Even if the practicality issues could be solved, we believe licensing software engineers would have no or little positive effect on the safety of the software being produced and could even have negative consequences. The reasons are as follows:

**Virtually everyone who designs or writes safety-critical software would be exempt from mandatory licensing requirements.** Consultants and teachers would be covered but not company or government employees who produce software or products with software in them. There are very few instances where safety-critical software is produced by an individual

who sells it directly to the public. Most software engineers would not fall under current PE licensing regulations and thus would not be required to be licensed nor find much benefit in voluntary licensing.

**There is no generally agreed upon comprehensive body of knowledge for software engineering of safety-critical systems.** Licensing examinations require a body of knowledge that is recorded and comprehensive. Different software systems and applications, however, require different techniques and knowledge. Although an IEEE committee has been trying to codify the material they consider essential for the successful practice of software engineering, they have explicitly excluded special needs and knowledge required for safety-critical software and systems, such as the skills and techniques necessary for designing and building real-time software. Relatively little scientific evaluation of software engineering techniques has been done, and it will be difficult to get consensus on what should be included and what should be excluded. As noted earlier, it is unlikely there would be a FE examination on software engineering, but even if there were it would not likely be effective in assuring competence.

**The PE examination is aimed at a pass rate determined by “minimal competence.”** Licensing that excludes large numbers of people from practicing a profession where there is a large and unsatisfied demand for practitioners will not be accepted. A minimal competency examination will, however, have little effect on the quality of safety-critical software. Past minimal competency examinations on computer-related topics, such as the DPMA, have been widely ignored and are essentially irrelevant.

**Construction or approval of software by a licensed PE is unlikely to assure safety.** The PE process licenses everyone as an “engineer”—it is up to the individual to determine whether they are qualified to practice a particular subdiscipline of engineering. Therefore, requiring that anyone building or approving safety-critical software be licensed as a PE is unlikely to solve any of the problems related to engineers practicing software engineering without sufficient expertise.

### **Voluntary Certification**

Although we focused our study on licensing, we note that many of the same drawbacks to licensing also hold for voluntary certification. The main difference between licensing and certification is that the latter is usually not a legal or state-controlled process, but most of the drawbacks and infeasible aspects hold for both. For example, any examination process would have to be objective; any questions requiring subjectivity in grading would be subject to the same legal challenges that forced the PE organization to eliminate such questions

and use a multiple-choice format.

One difference is because of the voluntary nature of certification, it can only be successful if it is accepted or supported by a majority of practitioners and employers. Past attempts to provide certification of software professionals have been largely ignored. It is doubtful that employers will find any minimal competency examination useful except perhaps for potential employees without formal education or experience and for relatively low- or entry-level jobs.

### The Implications of Licensing or Certification on Legal Issues

Licensing (or voluntary certification) may have legal and financial impacts on the practicing software engineer beyond those that might be expected. There is no tort of computer malpractice today. That is, courts have consistently rejected attempts to sue software engineers and to make them liable for software-related malpractice. In explaining their decisions, courts note that malpractice (professional negligence) involves harm caused by the failure of a person to perform within the standards of his or her profession. Software engineering is not a licensed profession, they say, and therefore software engineers are not subject to malpractice suits. If software engineers are licensed, they will be subject to malpractice lawsuits, and they will, most likely, need to carry malpractice insurance, which could be very expensive (for example, premiums in some professions are as high as \$50,000 per year even for an individual with a good claims history).

Becoming subject to malpractice suits has implications for the practice of software engineering beyond financial impacts. The process of determining that an act constitutes malpractice is only partially driven by engineers. In a typical lawsuit for malpractice, an injured or economically harmed person sues the engineer, often as part of a broader lawsuit. The person will bring evidence that the engineer acted in ways, or made decisions, that were not up to the professional standards of software engineering. This evidence will be evaluated by lawyers, liability insurance company staff and their lawyers, jurors, and judges. None of these people are engineers. If juries find that certain conduct is negligent, malpractice insurers will probably advise their insureds (engineers) against engaging in that conduct and may well provide incentives, in the form of lower insurance rates, for engineers who adopt recommended practices or who do not engage in counter-recommended practices. Over time, the determination of what constitutes good engineering practice may be driven more by the courts and insurance companies than by engineers. What happens if a practice required to limit malpractice litigation risk conflicts with what a

software engineer believes is necessary to provide safe and high-quality software?

Over the past 30 years, U.S. juries have been required to evaluate a wide range of engineering design issues in product liability suits. This caused enormous problems and led to major changes in the law (resulting in the adoption of a new Restatement of Products Liability that completely redefined the standards for holding a manufacturer accountable for a product's design defect). Unless the professional standards for software engineering are very clear, throwing determination of what constitutes good engineering practice to the courts may cause a great deal of harm. Licensing or certification opens the door to that possibility.

### Conclusion

The task force concluded unanimously that licensing software engineers who work on safety-critical systems as PEs would be neither practical nor effective in achieving the goal of protecting the public interest, and it could even have serious negative ramifications. The real issue, however, is not licensing per se but determining how best to protect the public without unduly affecting engineering progress, the economy, the engineering and computer professions, or individual rights.

Approaches other than licensing and certification might be more effective and need to be evaluated. These alternatives include government regulation and oversight; standards and codes; improvements in education including better textbooks and teaching materials, recommended curricula, and perhaps accreditation; oversight and pressures from the insurance industry; independent inspection and certification of software and products containing software; ethical standards and codes of practice; and various types of legal remedies.

A comprehensive approach with many mechanisms to improve software quality coupled with a cultural attitude that values quality and instills individual responsibility is needed to ensure acceptable safety of software-intensive engineered systems. Each industry must determine an appropriate mix of approaches that work together to solve their particular problems and fit within the cultural context of that industry. There are no simple and universal fixes that will solve the problem of ensuring public safety. **C**

---

**JOHN C. KNIGHT** (knight@cs.virginia.edu) is a professor of CS in the computer science department at the University of Virginia, Charlottesville.

**NANCY LEVESON** (ngl@safeware-eng.com) is a professor of aeronautics and engineering systems at MIT, Cambridge, MA, and co-founder of the Safety Engineering Corp., Seattle, WA. She is an ACM Fellow and recipient of the 1999 ACM Allen Newell Award.

---

© 2002 ACM 0002-0782/02/1100 \$5.00