

# “Certification is Discrimination”

James Bach and Luke Hohmann  
SmartPatents, Inc.

“So here’s the question for you: if you and your colleagues are about to have massive licensing and certification restrictions imposed upon your ability to call yourself a programmer, and upon the way you carry out your trade, what things would be most important to include or exclude for such certification regulations? If you don’t express an opinion, then you’ve abdicated and you’ll just have to live with whatever bone-headed rules the politicians come up with.”

— Ed Yourdon, private email to Luke Hohmann

## Introduction

This is a question of discrimination. Specifically, in what way should the powers that be discriminate in favor or against people who peddle software services? If Ed's premise is true, and the Y2K debacle will bring the specter of certification and licensing upon us, then our craft will be at the mercy of politicians, lawyers, lobbyists. They’ll do their deals under the hot light of media hysteria, and we’ll have to live with what they decide for us—just as the new software quality law, Article 2B, is being written with virtually no involvement of technical people (that’s why you probably haven’t heard of it). Still, since Ed asked, we do have some thoughts on the matter. Chiefly these:

- The most important thing is not what is certified or licensed, but *who* is subject to it, *who* grants it, and *who* controls that process.
- Different communities and application domains within the software industry have very different views about what should constitute certification or licensing for them.
- While technical people tend to think of certification in terms of competence, the issue will more likely be forced upon us as a way to promote accountability for work, rather than good work, *per se*.
- Many parts of our field are still too diverse, and changing too quickly to achieve consensus about what constitutes competence.
- When software people are licensed, they will take far less risk, software will cost much more, and technology development will slow to a crawl. Is that a better world?

## Certification != Licensing

What do licensing and certification really mean? According to *Webster’s Ninth New Collegiate Dictionary*, to certify is “to attest as being true or as represented or as meeting a standard.” Someone vouches that something is true. It is a testament of fact.

Certification is not the same as licensing. According to *Webster’s*, a license is “a permission to act”, or more specifically “a permission granted by competent authority to engage in a business or occupation or activity otherwise unlawful”. Although many licenses require some form of competency certification, it’s not an inherent part of licensing. It may be illegal to fish for bass without a fishing license, but no particular expertise is required to get a license.

A certification is a statement of fact. It provides a basis for making other decisions. A license is a granted privilege. Licensing provides control.

### **We are not opposed to certification or licensing... in principle.**

It's hard to be opposed to the basic idea of certification or licensing. Friendship, for example, involves a form of certification ("You're okay in my book..."). By publishing this article, Ed has "certified" that it meets the editorial requirements for *American Programmer*. We grant "licenses" in the form of privileges to our co-workers or spouses. By sending this article to Ed, we granted him a license to publish it. The basic utility of certification and licensing concepts on a personal level are beyond question.

The controversy begins when these ideas are pursued by companies, industries, or governments; when we have to decide as a group how to evaluate and regulate ourselves. That's when discrimination becomes a matter of grave concern and debate. Formal certification and licensing of programmers means creating what is essentially a new professional ruling class. Certain groups of people are bound to be disenfranchised by that process, and some of the "in-crowd" are bound to use their franchise to as a way to drive up the price of their services. The obstetric profession is a good example of this. After decades of treating all pregnancies as if they were serious illnesses, and all midwives as if they were criminals, research is finally showing that, for the majority of women, having a baby at home with the aid of a trained midwife is actually safer than going to the hospital (note that professional midwives pre-qualify their patients and work with medical backup nearby to handle serious complications). Since the majority of pregnant women are healthy enough to expect an uncomplicated labor, the obstetric profession would be in for big competition if more of their patients were aware of this research. Midwives make as little as \$10,000 a year, compared to the \$150,000-\$200,000 salaries of obstetricians.

In business, it's impossible to live without the raw concepts of certification and licensing. But realize that in practice those concepts require us to discriminate between who is *in* and who is *out*.

### **On what basis to license?**

The legal definition of a profession is well established. Cem Kaner discusses it in his paper *Computer Malpractice*, Software QA Vol. 3, No. 4. He notes that most courts have ruled that computing does not meet the requirements for legal treatment as a profession. Even if computing isn't legally considered a profession, though, it still could be subject to licensing, just as are the trades of plumbing and construction. So, if software people were licensed, on what basis might those licenses be handed out? We can think of a few: competency, accountability, or money.

*Competency:* We might give licenses to practice only to those people who meet a minimum standard of competence. How do we certify competence? A "competency certified" practitioner has passed some sort of test that verifies his or her ability to perform. Genuine competency certification is rare. More likely, certification means that the practitioner has merely survived some sort of obstacle course that is *presumed* to be related to competence. They took enough credit hours to earn a degree, or they passed tests that required the mere recitation of facts, rather than prove their ability to apply that knowledge to real problems (e.g. the American Society for Quality's Certified Software Quality Engineer program requires eight years of experience and passing a multiple choice test).

Obstacle-course certification is popular because it's much less expensive to verify, and it involves far less subjective judgment than does the certification of actual competency. Although obstacle-course certification is not the same as certified competency, we worry that those not intimately familiar with the domain will mistake such a certification for a guarantee of genuine competence or even professionalism. In that way, certification becomes mere tokenism that distracts us from the pursuit of excellence. There are professions, such as medicine and law, that work very hard to assure that their certification processes are related to competence. Is our field ready to work that hard?

Our biggest concern about competency certification is the glacial pace at which such certification standards evolve. Once codified, so many vested interests vie for control of the standard that innovation becomes virtually impossible, and real change is driven not by the technical community at large, but rather by bureaucrats and professional lobbyists.

*Accountability:* We could also grant licenses based upon a certification of accountability side-stepping many of the problems of competency-based certification. Accountability certification may involve nothing more than a demonstrated understanding of a programmer's responsibilities under the law (this should be simple, since there are so few to understand!), and perhaps professional liability insurance.

We can imagine, in this wild-eyed Y2K season, a law being passed that required programmers to be listed in a national registry, and to uphold a code of ethics, such as that of the IEEE or ACM. The ACM code is very demanding. It would certainly introduce interesting new dynamics into software projects if each programmer were legally obligated to adhere to it. One effect might be that programmers would lose any incentive for low-balling schedules, and project managers would get much more visible feedback about the realism of their scheduling suggestions ("You want to ship this *next week*? First I must advise you of the potential risks in accordance with section 2.5 of the ACM code."). Another effect might be that some programmers will be more passionate about team programming (in order to distribute their legal risk) or more passionately opposed to it (in order not to be held liable for someone else's mistakes). We'd fully expect malpractice insurance for programmers to do booming business, and software companies would have to pick up the tab (and pass it on to their customers). Independent programming consultants and small software companies would be hard hit by the new expenses.

Again, if we accept Ed's premise that certification and licensing will be forced upon us by an angry public, then accountability certification seems to us to address the matter more directly than does certification based on competency.

*Money:* Licenses could simply be purchased. If they were expensive enough, say \$10,000 a year, a lot of money would be generated that could go to further the development of our craft. Expensive licenses would encourage the development of software development guilds who would pay for the licenses on behalf of their members. These guilds would have tremendous influence over the kind of software that got developed and over programming practices. Expensive licenses would also fuel a thriving underground of unlicensed programmers who would telecommute, perhaps from offshore, or work for fly-by-night contracting firms. Similarly illegal labor already plays an important part in the agriculture and garment industries.

*Other:* We could grant a license based on some other attribute, such as gender or age. Sounds inconceivable, right? Well, it wasn't all that long ago that it was considered acceptable practice to discriminate based on ethnicity. Who is to say that in the mad dash for licensing that we won't pick an equally absurd attribute by which we license programmers.

### **It ain't what, it's who**

No matter how licenses are handed out, or what will comprise the administrative details of any related certification, the big question is *who matters*.

Who is "we"? There is no consensus on which set of people represent the "in" and the "out" crowd in the software industry. In Silicon Valley, the "in" crowd is certainly not the creators or followers of the SEI-CMM, while in Pittsburgh "Good Enough Software" are fighting words. This industry is composed of many different communities: academic research, embedded systems, MIS, medical systems, system software, telecommunications, utilities, games, multimedia, client-server enterprise systems, military, government systems. The list goes on. We don't yet have even a consensus about what communities even exist, much less a consensus on what standards apply to them. One of us (Bach) was a member of the ASQ CSQE team team, and was surprised to find that the standard was designed around a model of one unified field, without reflecting our diversity at all (Cem Kaner, another member of the team, discusses that process in his paper Software Negligence and Testing Coverage, Proceedings of STAR96, Orlando, Fl., May 1996, see <http://www.kaner.com>). But we know that technologies, quality standards, processes, market dynamics and cultures vary substantially between communities. Thus, it appears hopeless to us that a universally meaningful form of competency certification could be achieved. Each community will be on its own.

Who should be accountable? When you think certification, do you think only software developers should be certified? Why not the project managers, who demand the impossible and hold our jobs in ransom? Perhaps software customers should be certified, or else force to sign a blanket waiver before they are allowed to purchase software. Perhaps software itself should be certified, through some kind of national testing laboratory, or government SDA (Software and Drug Administration) inspection process. Good luck! We suspect that efforts to certify and license programmers that do not also address the issues of software management, documentation, component suppliers, SQA, SCM, technical support, and software quality itself will do little to address the fundamental issue of bad software.

### **How licensing and certification might affect you**

How might licensing affect you? Take a moment and examine your career. Have you ever changed jobs in such a way that you were suddenly working in an entirely new problem domain? If we pursue competency based certifications, will this affect your ability to change your job? As we presented above, failing to pursue competency based certifications could lead to a certification based on other factors. Are you willing to be discriminated against in your next job based on arbitrary reasons that may not directly relate to your ability to perform that job?

Then again, maybe it won't matter much. There are *already* laws under which you can be sued for negligence, even without any certification, licensing, or formal recognition as a

professional. Yet, in the history of the US, there is record of only five such lawsuits. Is Ed stirring up an empty hornets' nest with this question? Isn't the real issue, no matter how you consider this issue, *your* personal ethics and *your* own sense of professionalism?

### **Will we pay?**

Is the problem really that programmers aren't competent? Or, is it that the benefits of software technology are in such demand, and the lack of even moderately skilled programmers so acute, that most customers are willing to accept the risks associated with fast-paced software development?

The whole concept of certification and licensing may not be an issue at all once the rage and frustration of Y2K has died down. The extra costs will bring people back to Earth. If I am a certified software developer, I'm going to expect to be paid more money. If I am licensed, I'm going to take more time in creating software systems to ensure that I'm not sued for malpractice.

Will the marketplace accept these greater costs? Does the world really want us to slow down and stop cutting corners? For certain problem domains, yes, it will. But for many others, perhaps even most others, the answer is "No." The phenomenal development of Internet technology is a testament to the value of speculative, chaotic innovation. It has its place.

So even if the software industry is certifiable (*Webster's*: "to officially attest to the insanity of") the inmates are going to run the asylum for a while to come.

### **Acknowledgements**

The authors thank Cem Kaner for his review and critique.

### **Authors Biography**

James Bach is the Manager of Software Process Support at SmartPatents, Inc., the leading provider of analytical software tools for intellectual property management. James is also the editor of the IEEE Computer *Software Realities* department. Author of dozens of technical articles, he can be reached at [j.bach@computer.org](mailto:j.bach@computer.org).

Luke Hohmann is Vice President of Engineering at SmartPatents, Inc., the leading provider of analytical software tools for intellectual property management. Author of *Journey of the Software Professional: A Sociology of Software Development*, he can be reached through his web site at <http://members.aol.com/lhohmann>.