Software Engineering is Illegal

Boris Beizer

Since the publication of this essay in 1995, Texas has been the first to recognize Software Engineering as a legal profession. The University of Texas was quick to follow with being the first to offer Software Engineering degrees by that name – although many universities offer such degrees, de-facto, none in the U.S. had offered them de-jure. We have 49 states and five territories still to go – so this essay is still germane. [1]

1 What's The Fuss?

The 30 May 1994 issue of ComputerWorld carried a provocative story. One of our colleagues in Tennessee, George Phelps by name, ran afoul of a 75 year old statute that specifies who has the right to use the title "Engineer." Because of this law, he had to change his title from "Director of Engineering" to "Director of Technology." His is a CENSORED Engineering company and CENSORED Engineering is not legal in Tennessee. Nor, by the way, is it legal in all 50 states and five territories. If you call yourself a "Software Engineer" or promote your company as doing "Software Engineering" you may be liable for fines of up to $10,000, be forced to change your company name, your title, your stationery, your advertisements, publications, etc.

The purpose of the Tennessee (and the 54 other statutes) is ostensibly to "... provide a measure of assurance that an engineer ... will in no way jeopardize the safety, health, or well-being of the general public." Just who is an engineer and who determines that? The legal qualifications for using the title "Engineer" in the United States are:

1. Graduate from an accredited engineering school, where accreditation is granted solely by the Accreditation Board for Engineering and Technology (ABET).
2. Pass the Fundamentals of Engineering Examination, created by the National Council of Examiners for Engineering and Surveying (NCEE) but administered by the various states.
3. Four years of work experience in a recognized engineering subprofession, where recognition, although legislated in each state, is effectively done only by ABET and NCEE.
4. Pass the Principles and Practices of Engineering examination, also created by NCEE but administered by the states, in one of the recognized subprofessions.

   There are also unofficial, requirements for joining the ranks of sanctioned professional engineers.
5. Wear a pocket protector filled with Dietzgen #2 to #4 lead pencils.
6. Own a K&E Log-Log Duplex Decitrig Slide Rule (if you are old enough to know what that is!)..
7. Be a white, Anglo-Saxon, male who wears a hard hat.

The first four of these might seem to be reasonable standards for protecting the public. After all, if anybody could hang out a shingle and call themselves an engineer, who knows what harm might come of it. These standards seem reasonable until you look into them and learn:

[1] This article is taken from a collection of Dr. Boris Beizer's essays "Software Quality Reflections" and is reprinted with permission of the author.

1. There are no accredited degrees in Software Engineering and unlikely to be any in our lifetimes – at least until the current generation of superannuated creditors dies off.

2. The examinations have more relevance to a Mesopotamian water works engineer of three-thousand years ago than to software. The examinations have few questions on electronics no questions on computer hardware, and no questions on software whatsoever.

3. According to most state laws, you must have a Professional Engineering (PE) license to legally teach in an engineering school, otherwise the school or department won't be accredited. How's that for self-perpetuation generation after generation?

4. Our profession is not recognized and very unlikely to ever be recognized by the self-appointed, self-perpetuated, self- interested, official recognizers (ABET, NSPE, and NCEE).

5. Based on the singular lack of success that other "new" engineering disciplines have had in getting recognized within the framework of so-called professional engineering (Electronics Engineering being the most notorious 7), we don't stand a chance.

2 The Legal Engineering Professions.

I'll have more to say about these examinations below. For now, let's see what kind of "Engineer" is legal in the United States.

1. You can be in one of the recognized major engineering professions, including: Aeronautical, Agricultural, Chemical, Civil, Control Systems, Electrical, Environmental, Fire Protection. Industrial, Manufacturing, Mechanical, Metallurgical, Mining, Nuclear, Petroleum or Structural – but you can't be a Software Engineer, because that major profession, the profession with more practitioners than the others combined – doesn't exist according to ABET and NCEE.

2. You can practice one of the recognized engineering specialties such as: Gas Engineering, Tool and Manufacturing, Automotive, Lighting, Safety, Die-Casting, Plastics, Corrosion, Refrigeration, Pharmaceutical, Lubrication, Photographic, and Sanitation – never ridicule our Sanitation Engineering colleagues and pay daily homage (as we all must) to its spiritual founder, George Crapper – civilization would be down the toilet without them. But you can't be a Software Engineer!

3. You can practice one of the lesser known subspecialties. The NCEE told me that designation of a subspecialty depends on the number of practitioners; after all, one "... cannot expect to certify every tiny-subprofession that has only a few hundred practitioners." And, surely, all of the following subspecialties must have many more practitioners than our forbidden specialty: Arctic Condition Engineering, Cable-TV, Cryogenic, Earthquake, Heating, Insulated Cable, Lubrication, Noise Control, Optical, Plumbing, and Wood. But you can't be a Software Engineer because obviously there aren't enough of us and you can't be a Software Engineer!!

4. There are societies for engineering sub-subspecialties, all of which are legal: Senior Wire Rope (I couldn't find an Ordinary Wire Rope or a Junior Wire Rope?), Annular Bearings, Ball Bearings, Ball Manufacturing, Carbide, Cylindrical Hydraulic (as distinct from just-plain Hydraulics?), Practical Refrigeration (as distinct from Impractical Refrigeration?), Rehabilitation, Tractor, and Wind – the last one intrigues me because as a sometime competitive sailor, I would love to know how to engineer wind. That's the way the wind blows on sub-subspecialities; but no matter how hard you huff and puff, you won't be allowed to be a Software Engineer, because there aren't enough of us. You can't be a Software Engineer!!!

5. There are other occupations that have traditionally carried the name "Engineer" and are therefore legal and exempt from these laws: Flight Engineer, Flight Electronics-, any serving

soldier from private to general in the U.S. Army Corps of Engineers, Motion Picture-, Radio-, Sound-, TV-, Railway-, Ship's-, Locomotive- (also known as Mobile Engineer, which comes in both steam and non-steam flavors) and last and least the Stationary and Steam Engineer who fires up the boiler in the basement. The janitor in your office building can call himself an Engineer (if you have a steam boiler, that is), but you can't because you can't be a Software Engineer!!!

6. An Engineer can be: concerned, illuminating, abrasive, military, rough & tumble, or even explosive – and we all know engineers that fit those descriptors; each of the previous have their own engineering society – but they better not be concerned or illuminating about software because YOU CAN'T BE A SOFTWARE ENGINEER!!.

7. If you satisfy the technical criteria for engineering, you can add another hyphen and go on to be a: Cuban-American-Engineer, Danish-American-, Native-American-, Hispanic-, Korean-, Polish-, Swedish-, Turkish-, or Ukrainian-American engineer. You can be a minority engineer, a black engineer, and even women can be engineers today. You can be a gay and lesbian engineer, or a Christian, Muslim, or Seventh Day Adventist engineer – all these groups also have their own special-interest societies, but heavens forfend that you hyphenate "software" with "engineering." It's not legal in all 50 states and five territories because YOU CAN'T BE A SOFTWARE-ENGINEER!!!.

Before you start worrying that the PE Gestapo is going to come down on you like they did poor Phelps in Tennessee, let's note that as lawbreakers in all 55 states and territories we are in good company. The biggest lawbreaker is the Federal Government, especially DoD, who annually lets thousands of contracts with "Software Engineering" in the title or in the specifications 9 . The FAA and almost every other branch of the government that buys technical software similarly mention – no, they don't merely mention software engineering, they insist that we practice it. The annual Software Engineering Conference, now in it's 17th year, as well as dozens of other annual conferences and symposia on Software Engineering are also lawbreakers. The Software Engineering Institute is breaking these ludicrous laws. Purdue University is a lawbreaker. Every publisher with Software Engineering titles is a lawbreaker. But the next biggest lawbreakers after DoD are the IEEE and ACM, both of whom sponsor conferences, publish journals, and have huge hunks of memberships who call themselves "Software Engineers." If I were really worried about the PE thought police, I would insist that the IEEE mail the SE transactions in a plain brown wrapper.

3 The Examinations.

I looked at these examinations: Id rather have root canal without anesthesia done by Steve Martins sadistic dentist in The Little Shop of Horrors than take one of these. The first one, Fundamentals of Engineering, the one you take before you practice, is bad enough; but the second exam that you take after youve practiced a while, makes the first look like a give-away.

The first exam is an 8-hour, open book exam in which you answer 140 compulsory questions in the morning and 70 in the afternoon.10 Most of the morning questions are hard-hat engineering questions including such juicy favorites as: the rate at which a slab of beef will give up heat, those horrible physics problems that involve two monkeys, a pulley and a string; and one of your routine four-body cis-lunar orbital calculations for a slingshot-effect ride past Mercury to land you in the vicinity of Pluto by June 2007 11 .

There are two morning categories you'll recognize: finance and mathematics. The earlier exams had a section on computers and software, but they since expurgated these subjects. The finance questions are easy – they're all built-in my calculator. The math is okay if you remember your calculus 101. But you couldn't count on passing the software questions when they did have them. About half involved number representation conversions (also on my calculator). The tough ones were the programming questions. You'll have to dig up very old, very obsolete BASIC and

FORTRAN manuals to learn the syntactic and semantic peculiarities of those languages as of two decades ago. And even that wouldn't help because according to the answer book, sometimes AND is equal to "OR" and ">=" can be used interchangeably with ">". The afternoon is more of the same, but only worse.

How did I do, you ask? Based on my self-administered attempt at several exams, I would have failed gloriously. I aced the math and finance, got two "wrong" on the software, picked up fifteen questions because as an avid sailor and navigator I've learned about vectors, aerodynamics, and hydrodynamics. Out of 210, had I gone the whole route, I might have managed 50 or so; possibly more because it is an open book exam. That was the old exam from 1986 and earlier. The latest study guides and presumably therefore, the latest exams, have no questions on computers and software whatsoever so my score would have dropped proportionately. Not only doesn't software exist for these so-called engineers, but neither do computers!

How about the Principles and Practices exam that you take after you've worked in the field for several years? The closest you could manage would be to opt for the Electrical Engineering exam and try to hit problems in digital systems, computer systems, and communications. Frankly, folks. it's hopeless. You're about as likely to pass as one of our hard-hat colleagues is likely to pass an exam on software quality assurance or object-oriented development.

It's a stacked deck; so deeply and thoroughly stacked that unless you're willing to re-educate yourself in engineering archeology and spend several years cramming, you can't possibly make it.

The examinations fascinated me, especially the fact that to the extent that the samples I saw dealt with software at all, it dealt only with toy problems in archaic Basic. I called the NCEE and asked why Software Engineering was not a recognized engineering discipline and why there were no questions on software. The reason, according to Roger A. Hadley, of the National Council of Examiners for Engineering and Surveying was:

"There is no such thing as "Software Engineering". That's just writing computer programs. Any high school kid or mathematician can do it. It is not a recognized engineering discipline because it isn't engineering. And no engineering school in the United States provides a degree in it – whatever did you call it? "Software Engineering"?

Yes, my fellow "high-school kids," anybody can do it!

We're concerned with quality, so we should ask what constitutes a passing grade for the exam. It varies from state to state, but generally, 70scored, that actually means that you only have to get half the questions right. How's that for a quality standard? How's that for protecting the public good? Those so-called engineers only have to be right half the time. The instructions in a popular cram book goes on to say that if you don't know, don't leave the answer blank. It's a multiple choice exam and no points are taken off for wrong answers. Random guesses will get you 20grade. When in trouble or in doubt, guess!! You'll only kill someone 80engineering ethics that these exams teach to young aspirants. Could we, the illegal Software Engineers, afford to guess? Could we sell software that was only 50use our software if that was our standard?

4 A Real-World Message.

There's a real-world message to be contrasted with the mythical message perpetuated by the so-called professional engineers. The examinations tell the story. The exams require a very broad base of knowledge and techniques across the entire range of engineering, excluding of course, Electronics, Computers, and Software Engineering. You need a smattering of electrical circuits, fluid mechanics, thermodynamics, statics, dynamics, chemistry, strength of materials, economics, etc. That was fine in the last century, and even up to the 30's and 40's when it was possible for a hard- working, hard-studying engineer to encompass that body of knowledge: there just wasn't that much of it to learn. For a 19th century engineer, it was proper and more important, it was possible, to get a smattering of knowledge across the entire field.

That once noble aspiration is of course, ridiculous today. I've devoted 40 years to the Software Engineering profession and there are huge areas within Software Engineering of which I am almost completely ignorant beyond the level of a technological layman. To have effective broad knowledge

within the software field alone would require reading, digesting, and internalizing every issue of say, ACM Computing Surveys. I don't have the time for that, or the need. At best, I read the summaries in that journal and the one-out-of ten surveys that most apply to my sub-sub-sub specialty, Testing and Quality Assurance. And I spend a lot more time in self-educational pursuits than most working Software Engineers because my business is technology transfer. How about the rest of you? We have to run like hell to keep up with an explosively evolving technology. We must prioritize our time and allocate our self-education, or else there's no time to do Software Engineering as contrasted with studying it.

It's the same in school. Employers don't want renaissance men. They want people adept in C++, or Windows, or ATM, or what have you. Just try to sell your superficial knowledge of strength of materials or fluid dynamics (which you learned at the expense of deeper software knowledge). You know the answer. If they need an expert in thermodynamics, they'll hire a real one to work with you and not someone with an operationally useless smattering of key words.

The PE's ideal of the mythological engineering renaissance man is a century out of date. It flies in the face of technological realities. It serves only one purpose to maintain an entrenched oligarchy intent only on perpetuating their self-interest at the public's expense. How has the "Professional Engineering" oligarchy maintained the myth? By simply saying that anything that doesnt fit their myth, such as Software Engineering, is not engineering.

5 Software Engineering Versus Engineering Software.

My research led me to examine all recent periodicals and journal articles that had both "software" and "engineering" in the title or abstract. There were tens of thousands such articles in the past decade and even thousands in just the last two years. The "Software Engineering" articles appeared in all the journals you and I read. But they were few in number compared to the "Engineering Software" articles in the traditional engineering literature. I checked that by eliminating from the search all computer, software, and related journals; looking only at the traditional mechanical, civil, chemical, etc. periodicals. Guess what the hot topic is, folks? "Engineering Software," of course. More articles on Engineering Software than almost any other engineering topic. Engineering software written by all those illegal Software Engineers. So you see, ours is a problem of permutations. You can write "Engineering Software" as long as you don't do "Software Engineering." The one permutation is legal, the other is not.

Doesn't it give you the willies to realize that all that software on which our hard-hat brethren depend, the software that calculates the right size for structural steel in our buildings, that's used to design the controls for the aircraft on which we fly, that calculates the radiation dosages to give cancer patients, that keeps us from having a weekly Chernobyl accident – doesn't it give you the willies to realize that all that "engineering software" could have been written by a bunch of high-school kids? It must have been written by high school kids because we professionals don't exist and because YOU CAN'T BE A SOFTWARE ENGINEER even if you write the engineering software used by those "real" engineers.

6 Why Should We Care?

Why should we care what a self-perpetuating ossified oligarchy decrees what is, by their aboriginal patriarchal standards, to be or not to be called "Engineer?" We know what we are, what we do, how we do it, and we'll keep on doing it and calling it "Software Engineering" despite all the witless laws on the books. Those laws are as irrelevant as the statutes in some states that, for example, requires every automobile to be preceded by a man with a lantern who clangs a bell at regular intervals. Ignore the moronic laws and concentrate on doing an ever-better job.

"Let it be." you say, "It's just semantics." You wouldn't dream of specifying the concrete for an elevated freeway or the codes for earthquake protection. You wouldn't dream of signing-off on the San Francisco Freeway even though many of you might know a hell-of-a- lot more about

concrete structures than they know about structured software. If they want to be irrelevant in the post-industrial world and not recognize our kind of engineering, it's their loss. We don't tell them how to link chains, they don't tell us how to link objects. That would be nice, but it doesn't work that way. These men, in their almost infinite arrogance, do presume to regulate how we do software. There is real power for the PE in those laws and real danger to the public. And sooner or later you will run afoul of these archaic statutes:

1. Legally, in most states, only a PE can give expert testimony in court on engineering projects. A smart lawyer might get me disqualified from testifying on the adequacy of software testing methods, for example. I'm sure we have a few PE's among our ranks, but very few expert in Software Engineering ever advertise that fact.

2. Legally, in most states, only a PE can call themselves an "Engineering Consultant" and only a PE can advertise as such.

3. In many civil projects-, especially when funded by state or municipal agencies, where there is a software content, you will have to pay a useless "consultant" who doesn't understand one word of what he's approving, to legally sign-off on your project. And there's no end of PE's willing to prostitute themselves for a fee. It's a waste of public money as far as software is concerned.

4. We should care because whenever a software-ignorant civil or mechanical engineer hires a bunch of high-school kids to do software for one of their projects: and when, as expected, the bottom falls out, it is we, the real Software Engineers, who take the flack. It doesn't matter if the software was written by amateurs or by engineers who know nothing about our profession and methods. We, the professionals, are blamed for every fiasco that involves software.

5. The main reason we should care, however, is because the public is harmed by these stupid laws. We should care because in Civil Engineering project after project with a high software content, signed-off by so-called "professional engineers," we see the public good damaged by hopelessly inadequate software miscrafted by amateurs no better than high-school kids instead of by professionals, to professional standards, and under professional quality assurance and quality control methods. Look at Peter Neumann's column in ACM-SIGSOFT and ask of the fiascos described therein how many were caused by unprofessional software developers working under the guidance and control of a so-called "professional engineer." I'll bet on the Denver Airport baggage handling system software for a start.

7 What to Do?

Ever since Barry's letter was published in ACM SIGSOFT (that notoriously illegal journal), there's been an initiative afoot to rectify this situation. But frankly, as laudable and as important as that initiative is, it isn't going to work. It isn't going to work because it is based on the false premise that the issue has to do with technology and that there are rational criteria for getting Software Engineering recognized and legalized. The key components of the initiative are: (1) define a body of knowledge, (2) define recommended practices, (3) establish a code of ethics, (4) establish certification/accreditation guidelines, and (5) define a curriculum. I'm for all of these because our profession would be improved by them, but...

But the initiative won't succeed. It won't succeed because it's based on an attempt to work within the system! "The System," being of course, the entire self-interested, exclusionary, PE hierarchy. It will fail because it doesn't recognize that the key issues are political and economic rather than technical. If we, by magic, created a consensus on all five of the above and presented it to that ossified oligarchy, it wouldn't matter. They would simply keep on insisting that we do not exist and that what we do can be done by high-school kids.

It's a problem of technological politics that demands political solutions. Here are some ways to do that:

1. Start the political initiative in California, where we have the most practitioners and votes. Follow up in other states with similar attributes such as Washington (Seattle area), Texas, Massachusetts, Oregon, Florida, Virginia, and Maryland. Don't bother with congressmen. Write to your state legislator and convey to him or her:
   (a) How many votes Software Engineers have in their district compared to the number of votes by so-called Professional Engineers. We outnumber them by 10:1.
   (b) How many jobs these laws might be costing and the advantage it gives to foreign suppliers not bound by such laws.
   (c) Any instance you know of in which a traditional engineering project in your state with a crucial software content was signed off by a so-called Professional Engineer to the detriment of the public's interest. Do some creative whistle blowing.
   (d) Ask why you can't get a license to practice your profession. Ask why it is with N thousand voting practitioners in his or her district and only K dozen voting PE practitioners, only the PE's can get a license to practice engineering.
2. A class action suit against ABET, NSPE, and NCEE based on their possible violation of the Sherman Anti-Trust Act.
3. Let's get Bill Gates and Dilbert interested.
4. Form our own PAC.
5. Software Engineering is the single largest group within ACM and IEEE. Push these notoriously apolitical entities into action. Flex your muscles.
6. Join forces with the next largest group of engineers, also effectively excluded, the Electronics Engineers. They were teed-off with the PE's forty years ago and they still haven't been totally legitimatized. The barrier for them is that first irrelevant exam because they have a chance of passing the Principles and Practices exam if they can get by the so-called fundamentals.
7. When you next make a contribution to your alumnus association, demand to know when your school intends to have an accredited program in Software Engineering under that name. Don't accept any obfuscatory academic-administrative BS and excuses for an answer. Let them know that your annual donation will be going to the first university that has a legitimate department teaching Software Engineering under that name.
8. Let the politicians, the lawyers, the hard-hat engineers, all know that:

   YOU ARE A SOFTWARE ENGINEER!!!

8 Prof. Parnas' Response to Part 1 of Beizer's Article.

I am surprised that Dr. Beizer would want his 1995 essay published in 2003. A great deal has happened in this area since that essay was written. Among other things there are now several accredited engineering programs specializing in software development in Canada and the US and the Engineering profession has made it clear that it is open to emerging disciplines including Software Engineering.

The situation is quite simple. In many jurisdictions, Engineering is, by law, a self regulating profession just as Medicine is. The regulatory bodies for both professions are duty bound to make sure that the public is able to easily distinguish between qualified and unqualified practitioners. The easiest way to do that is by means of a protected title. The title is given to those who

prove (through exams and experience) that they are qualified and removed from those who do not practice properly. Just as I would not want to have to do a reference check when I need a medical doctor, we should not have to study academic records and activity records when selecting someone to build a bridge or other structure. When I see that someone is an M.D., I know that he/she has had a basic medical education, appropriate years of apprenticeship, has passed certain exams, and will have her/his license removed should he/she be found to have been negligent. I should be able to identify qualified people easily in any profession that builds critical products.

The title is enforced primarily in situations where its use could cause confusion. If someone is offering Engineering services to the public or may be perceived to be doing so, they must be entitled to use the title. The title is often not enforced when there is no possibility of confusion. For example, nobody will assume that the driver of a train is qualified to offer engineering design services to the public. The use of the term "Sanitary Engineer" is more likely to induce a bad joke ("the rest are dirty engineers") than any misunderstanding. However, the use of the title "Software Engineer" is likely to cause a non expert to believe that the person has proven qualifications and could be a cause of confusion. There is justification for enforcing the title in such cases. I believe that the Engineering regulatory authorities have been negligent when they allowed the title "Software Engineer" to be used without restriction.

The present situation is far from ideal. The exams and other restrictions were developed for very different professions. There are two approaches to approving it, but neither one is advanced by the scathing cynicism expressed in Beizer's article. One approach would be for such groups as ACM to work with the Engineering regulatory authorities to improve the relevance of the tests. I have been disappointed by their refusal to do so. The ACM seems to be taking the position that we have to know how to produce perfect software before we can begin to distinguish between reasonably competent software developers and others who present themselves as competent. If we had taken that position about roads, I might be designing mountain bridges without any qualifications at all.

The second approach would be to choose a different title and set up a meaningful qualification procedure for professional software developers. Doctors of Medicine have not been able to keep other types of professionals from offering healing services. However such other approaches as Chiropractry have had to use a different title so that nobody will believe that they are medical doctors. There is no reason why professional software developers have to use the term "Engineer". Surely we have enough imagination to find a term that could not cause confusion. However, we must find the appropriate set of standards. Here I have been disappointed too. Although there are efforts to identify a core body of knowledge that should be possessed by every professional software developer, the proposals are not very good and have failed to identify the "core" body of knowledge that should be possessed by every software developer. Instead they are including folklore and technology as if they were fundamental principles.

Instead of writing scathing articles because the Engineers won't let us use their title without proving our competence, we should be either helping the legally established regulatory authorities to do a better job or starting a parallel system, with equally high standards, of our own.

Dave Parnas

Prof. David Lorge Parnas, P.Eng. (Ontario)
Director of the Software Quality Research Laboratory
SFI Fellow, Professor of Software Engineering
Department of Computer Science and Information Systems
Faculty of Informatics and Electronics
University of Limerick
Limerick, Ireland

9 Dr. Beizer's Response to Parnas' Comments

"I am surprised that Dr. Beizer would want his 1995 essay published in 2003. A great deal has happened in this area since that essay was written. Among other things there are now several accredited engineering programs specializing in software development in Canada and the US..."

As far as I know, only Texas (as mentioned in the essay) has accredited software engineering programs under that name. If there are other states that permit software engineering by that name, I may have missed them and will be glad to acknowledge their entry into the 20th century. As for Canada, I don't know the situation regarding the legality of software engineering there – but then, my essay was aimed at rectifying this ludicrous state of affairs the US – not in Canada. As for being germane today, very little has changed since 1995. The title "Software Engineer" is still not a legal title in all but a few (one?) US state and the various engineering accreditation boards have yet to come up with examinations and protocols that would accept even a minority of currently practicing, competent, software engineers under that title.

I'm just a mite puzzled about the use of that title by Parnas. He doesn't hesitate to call himself a "professional engineer" and to use that title in his signature block. Nor does he seem to be shy about calling himself a "Professor of Software Engineering." Does he mean that those of us who do not have a PE license should seek to call ourselves something else? I am puzzled because he seems to suggest in his rebuttal that we should drop our claim to the "engineer" title – but he will continue to use it – because he is a software engineer – or at least teaches it. Or is this title to be permitted only for Canadian software engineers? Yet, he urges us Americans to seek accreditation under some other "imaginative" title. Or is he urging us to emigrate to Canada so that we can legally practice our profession, as does he, under that name?

Parnas calls Software Engineering an "emerging discipline." By my reckoning, it is over fifty years old. We've been calling it "Software engineering", by that name, among ourselves and in the literature for almost forty of those fifty years. So what's this "emerging" nonsense? What's this "emerging" thing when we routinely deal with levels of complexity and create products whose work-hour content makes most traditional engineering projects seem like kid stuff?

Parnas summarizes the reasons that some kind of licensure for software is desirable. I and others, have been trying for over forty years to get some kind of meaningful, germane, recognized professional certification in place. And we have been thwarted at every step by entrenched engineering professional organizations – among others. But I agree with much of what he has to say:

"The regulatory bodies for both professions are duty bound to make sure that the public is able to easily distinguish between qualified and unqualified practitioners."

"The title is given to those who prove (through exams and experience) that they are qualified and removed from those who do not practice properly"

"The present situation is far from ideal. The exams and other restrictions were developed for very different professions."

"One approach would be for such groups as ACM to work with the Engineering regulatory authorities to improve the relevance of the tests. I have been disappointed by their refusal to do so."

Now we get to the points of fundamental disagreement.

"Instead of writing scathing articles because the Engineers won't let us use their title without proving our competence, we should be either helping the legally established regulatory authorities to do a better job or starting a parallel system, with equally high standards, of our own."

Parnas suggests that we try to construct a parallel system under a different name. We should also try to help the regulatory authorities to do a better job, etc. But we should not resort to sarcasm – especially "scathing sarcasm." Where has he been the past forty or so years? Apparently not in the same software universe in which I have been. My sarcasm was in direct response to four decades of inaction and insignificant progress toward any kind of meaningful licensure – despite constant efforts by many of us, trying to act nicely and within the establishment, to get meaningful recognition. The sarcasm was – and is – intended to shame the US establishment into a recognition of Software Engineering comparable to that which exists in other countries. I believe that my

sarcasm has been in a small way, instrumental in nudging American licensure establishments (e.g., ABET) toward a more realistic recognition of the facts of Software Engineering, as they are. Yes, there has been some (minuscule) progress (e.g., Texas) – but it is still glacial. And the public is still ill-served because we do not yet have legally meaningful professional standards and regulations in place.

Some people in our profession do not consider themselves to be "Engineers." But I and many others came out of more traditional engineering disciplines and have a philosophical point of view that is embedded in those older disciplines. We speak of software engineering, of course, but also of system and software architecture – in the original meaning of the word "arkitekton" meaning, "master builder." Why should we rescind our just claim to the titles of engineers and architects? Why should we deny the philosophical and ethical roots that many of us have in the earlier engineering professions? If forty or fifty years of practice hasn't "proven our competence" to those ossified authorities, what makes you think that another forty years of playing nice will accomplish anything?

I am a software engineer and proud of it. I don't want to be called an "applied mathematician," "digital accountant," "information specialist," "applied information theorist," "algorithmizer," "data organizer and regularizer," "licensed hacker," "data crunchmeister" or any other imaginative alternative descriptive term that we have not adopted in forty years.

Parnas and I appear to be after the same things. Professional recognition, meaningful licensure, and proper standards for our practitioners. Whilst he continues to apply the methods, which in my estimation have been tried (by me, among many others) and failed for over four decades, I will continue to use humor and sarcasm – to give the licensure establishment the swift kick in the butt which they richly deserve and which seems to be the only thing that gets them off the dime.

Boris Beizer

Software Engineer
(But not a licensed professional engineer, in either the US or Canada).