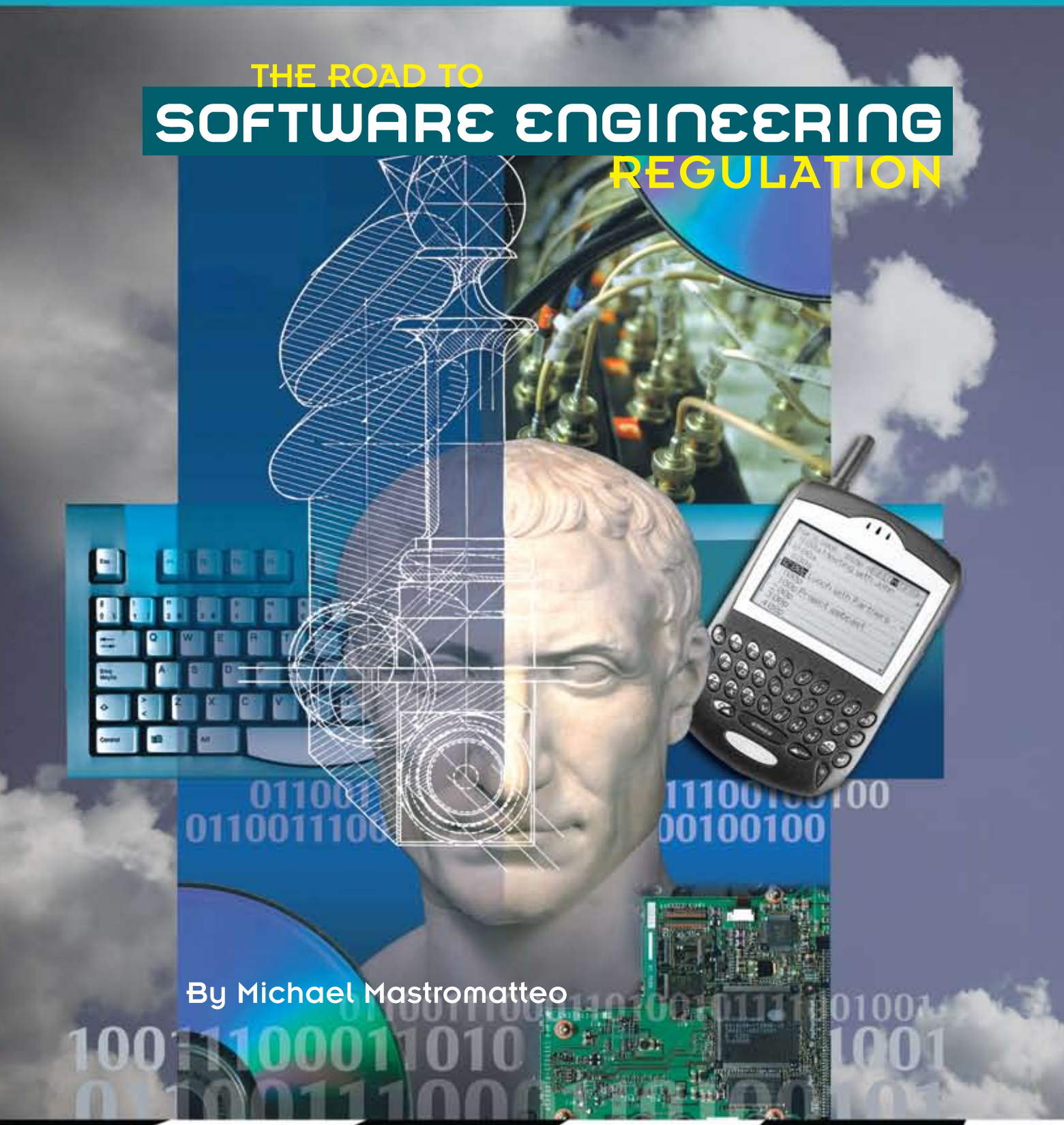
THE ROAD TO SOFTWARE ENGINEERING REGULATION



By Michael Mastromatteo

A just published guideline on using software-based engineering tools and another in the works on development of software for safety-critical engineering applications are PEO's latest efforts to bring regulatory clarity and understanding to this continually evolving discipline.

Although the term "software" entered the vocabulary nearly 50 years ago, its association with the term "engineering" has been a problem area for engineering regulators for only about the last dozen or so years. That's when, in the lead-up to Y2K, the potential for failure of much of the software on which modern life depends–and the resulting catastrophic consequences–began to be widely discussed, leading to a growth industry of unregulated individuals offering to fix perceived software flaws. At the same time, universities were introducing software "engineering" programs that were essentially computer science, while software industry giants were certifying those completing their technical training courses as "engineers."

Seeing difficulties ahead in enforcing the title provisions of the *Professional Engineers Act* as well as having concerns about the lack of regulation and professional accountability of those working in the software area, in May 1999, PEO council accepted a recommendation of the software engineering subgroup of its Engineering Disciplines Task Group (EDTG) and formally recognized that a specialty exists within engineering with an emphasis on software design, and that there were professional engineers practising engineering within that specialty.

Accordingly, PEO's position statement on software engineering maintained that the use of the title "software engineer" needed to be restricted to professional engineers, for the same reasons that the use of the title "civil engineer" is restricted to professional engineers–so the public can be assured of the qualifications and accountability of those using the title.

The next step in the recognition of software engineering as the practice of professional engineering in Ontario occurred in September 1999, when PEO announced it would begin licensing, as professional engineers, software practitioners whose work experience was mainly in the area of software design and development, but whose academic background was in something other than an accredited computer engineering or other information-technology-related engineering program, provided they met other licensing requirements. Indeed, developing the necessary syllabus to ensure such practitioners would meet the academic requirements for P.Eng. licensing was no small feat, given that the first true software engineering programs–at McMaster University, the University of Western Ontario and the University of Ottawa–would not be accredited for another two years.

The EDTG software engineering subgroup then turned its attention to defining the parameters of software engineering practice, and in December 1999, council approved two software-related practice statements. Practice Statement A held that any software

component of a product or system whose development is the practice of professional engineering, as defined under the *Professional Engineers Act*, must be approved by a licensed professional engineer.

Practice Statement B, referring primarily to computer-assisted design (CAD) and embedded software, stated that licensed professional engineers using software in the design process for a device or structure the design of which constitutes the practice of professional engineering must either use software approved by a licensed professional engineer or verify that the software used produced acceptable results.

IMPLEMENTING PEO'S POSITIONS

The EDTG delivered its final report to PEO council in 2002, in which it enunciated a process for PEO to use to recognize new areas of engineering, a process its software subgroup piloted in gaining recognition of software engineering.

How to put PEO's recognition of software engineering into practice then fell to council's External Groups Task Force–Software (EGTF–S) which, in 2004, presented its report to council. In recommending how PEO should proceed in developing its policies on software, the EGTF–S took the approach that the total software field can be subdivided into "domains of practice." By dividing the problem in this way, it said, the regulator could be more precise in determining which domains are engineering.

The task force also discussed the problem of the setting of standards in an emerging field that might not always involve the practice of professional engineering.

"In a fast-paced rate of change field like software practice, how standards are set and maintained is a significant question," the EGTF–S report says. "This complication sets the practice of software apart from traditional engineering disciplines and other professional practices with a more or less stable set of standards."

In elaborating on the difficulties inherent in attempting to impose a universal standard on a field as diverse as software, the EGTF–S report says: "We expect that most software practice standards will not dictate the details of how an activity or task will be accomplished, but rather state expectations of a defined activity in more general terms. In very few areas will there be absolute practice standards for software. It's almost never the case that there is one and only one right way to do things in software."

In September 2006, PEO council took up the recommendations of the EGTF–S report by directing that PEO consider a unique designation for a software specialty, as part of a broader study of disciplines and specialties.

Council also requested PEO's Enforcement Committee to begin stepping up its activities against unlicensed software engineers.

FAST FORWARD

In March, PEO's Professional Standards Committee (PSC) completed the first of two new guidelines related to software, for *Professional Engineers Using Software-Based Engineering Tools*, which was approved by council at its April 2011 meeting, and is now available from PEO's website at www.peo.on.ca/Guidelines/ProEng usingSoftwareEng Tools2011.pdf.

The guideline addresses questions as to the proper role for and responsibilities of engineers using "commercial or free source" software, regardless of whether it is written by practitioners or by others. It also makes recommendations to guide P.Engs who rely on software in providing some element of engineering services.

Among the issues addressed in the guideline are the engineer's duty to understand the use of software in an engineering context; training and support for the user; quality assurance for software output; and the need to verify the software's overall performance.

"It is often difficult to determine, just by using a program or by being given a description of its function, how the software deals with the engineering principles and technical information it incorporates," the guideline states. "Engineers should become familiar with the engineering principles, equations, models, algorithms and assumptions used in the software."

The new guideline replaces part of the content in PEO's older *Guideline on the Use of Computer Software Tools and the Development of Computer Software Affecting Public Safety and Welfare*, because the PSC came to the conclusion that separate guidelines are needed for these distinct activities.

A new PSC subcommittee has now just begun work on a guideline for *Professional Engineers Developing Software for Safety Critical Engineering Applications*, which will update and replace the remaining content of the older publication. This upcoming work will focus on software that is "the output of the engineering design process." A first draft is not expected for some time.

The subcommittee is basing its work on a definition and policy regarding software engineering developed by PEO's Enforcement Committee, and approved by council.

For the Enforcement Committee's purposes, software development is the practice of professional engineering:

- where the software is used in a product that already falls within the practice of engineering (e.g. elevator controls, nuclear reactor controls, medical equipment such as gamma-ray cameras, etc.);
- where the use of the software poses a risk to life, health, property or the public welfare; and
- where the design or analysis requires the application of engineering principles within the program (e.g. does engineering calculations), meets a requirement of engineering practice (e.g. a fail-safe system), or requires the application of the principles of engineering in its development.

Bernard Ennis, P.Eng., PEO's director of policy and professional affairs, says the PSC subcommittee will keep this definition in mind in drafting the guideline, focusing "primarily on when the software is used in a safety critical engineering application, such as control of a machine, system or plant where failure of the controls could have dangerous consequences."

Nick Pfeiffer, PhD, P.Eng., a member of the subcommittee, says the Enforcement Committee has done yeoman's work in clarifying some of the practice parameters for software practitioners.

Head of Pfeiffer Technologies Inc., a manufacturer of electronic controls for niche applications, Pfeiffer describes himself as an engineering generalist who has acquired expertise in software applications and embedded software for engine control systems.

"The Enforcement Committee definition of professional engineering explicitly defines when software development involves the practice of professional engineering," Pfeiffer says. "Software engineering is currently regulated in only three jurisdictions in North America–Texas, Alberta and British Columbia. Other jurisdictions may or may not have a clearly defined concept of when software development involves the practice of professional engineering."

So, clearly, with so few jurisdictions regulating software engineering in some manner, there remains a need for the profession to come to grips with the creation and use of a technology that has become increasingly pervasive and, in a sense, invisible.

As Lance Kinney, PE, executive director, Texas Board of Professional Engineers, explained in a recent opinion piece: "Traffic control systems, 'smart' buildings, and other systems that integrate software with the built environment pose a major risk that we cannot afford to take now or in the future. Software engineers need to be licensed to demonstrate competency, to take responsibility, and to protect the health, safety and welfare of the public."

SOFTWARE ENGINEERING EDUCATION MATURES

Today, there are about 50 PEO members who hold an undergraduate degree in software engineering. "However, we have additional P.Engs with diverse educational backgrounds who have been determined to meet PEO's software engineering syllabus. When these are included, I believe we have almost 200," says Michael Price, P.Eng., FEC, PEO's deputy registrar, licensing and finance.

In fact, there are now 13 software engineering programs accredited by the Canadian Engineering Accreditation Board (CEAB) throughout Canada (see box on p. 38 for the full list and dates of first accreditation).

The software engineering program at the University of Western Ontario is one of three marking the 10th anniversary this year of their CEAB accreditation. (The others are at McMaster and the University of Ottawa.) Enrolment in the program at Western has increased substantially over the last three years, with former program director Luiz Fernando Capretz, PhD, P.Eng., expecting the pattern to continue. He cites "excellent career prospects" for software engineers, with the advent of social networks, smart phones and computer games, as the top reasons for the renewed student interest in pursuing software engineering.

Capretz, who as of June 30 moved to the position of assistant dean (IT and e-learning) at Western's faculty of engineering, says that while there is room for individual creativity in software development work, the engineering rigour adds value to the preparation of software engineers. "The nature of an engineering degree is application of theory to solve problems," he says. "This idea is passed to students from year one. There is still an element of art and imagination in software construction, like in architecture. But the design aspect of software requires innovation and creativity like any other engineering branch."

LEGISLATION STILL LACKING

Yet despite a renewed interest in software engineering among students, David Parnas, PhD, P.Eng., former director of the software engineering program at McMaster University and a contributor to PEO's EDTG subgroup on software in the late '90s, believes there has been little regulatory progress in the software area.

"Probably the most important reason for this is the lack of clear legislation," Parnas told *Engineering Dimensions*. "We lack legislation that clearly states when a software product must be approved by a licensed engineer qualified to judge software–we also have to state what standards a product that requires such approval must meet."

Similarly, Michael Bennett, PhD, P.Eng., associate dean, faculty of engineering and applied science, University Ontario Institute of Technology (UOIT), believes that while the software field has matured, there is still room for enhanced regulatory oversight.

"The good news is that most software professors are now licensed–a vast improvement over the last 10 years," Bennett told *Engineering Dimensions*. "The emergence of software engineers as engineers is growing apace, but I do not see the imposition of regulatory controls happening yet in the marketplace. I do hope that the next wave of acceptance will see this."

Bennett, who helped establish the original software engineering program at Western, and repeated the experience years later at UOIT, suggests that it is industry and individual software practitioners who are clarifying the engineering-software link. "Software engineering graduates across the province are leading the charge in getting software recognized as an engineering discipline," he suggests. "For example, they demand that they work under the control of a professional engineer wherever possible. They also complain to PEO if they see the word 'engineer' misused."

Nonetheless, he sees a stepped-up role for regulators and the provincial government, especially when it comes to safety critical software issues.

"We need to demand that where safety is an issue, professional engineers sign off on the work," Bennett says, "exactly as in other branches of engineering. Software and communications are ubiquitous in the 21st century and we need to have safeguards in place to ensure the construction of quality software and the certification of outsourced software if it is to be used in safety-critical operations in Canada."

To this end, Bennett applauds the new PEO guideline on using software-based engineering tools, stressing that engineers must be fully cognizant of the tools they are using.

Peter DeVita, P.Eng., FEC, who chaired the EDTG software subgroup to whose work Parnas contributed, has long argued that PEO should do more with the regulation-making authority with which it has been entrusted, and calls for an aggressive approach to the provincial government in the software issue.

"We must drive new legislation as we, PEO, are the only ones in a position to understand what we are talking about with respect to a new discipline in engineering," he says.

DeVita also sees a role for PEO's Legislation Committee in driving the regulation of software engineering, and says that with the Enforcement, Professional Standards and Legislation committees leading the way, it will be possible for PEO to implement new actions that ensure that a PEO licence to practise engineering "has real meaning and real teeth."

The PSC's Nick Pfeiffer, on the other hand, recommends against too heavy-handed an approach to regulation. "I believe that software engineering is an immature discipline and that overly stringent regulation will stifle innovation and progress," he says. "However, I hope the guideline that the subcommittee is developing will prove useful and prompt many PEO members to obtain additional specialized knowledge and training. In my opinion, PEO would be well-served by having software engineering as a defined specialty, similar to structural engineering [in BC]–this is the approach taken by British Columbia and Texas and is being pursued by several other states." Σ

ACCREDITED SOFTWARE ENGINEERING PROGRAMS

(WITH YEAR OF FIRST ACCREDITATION)

University of Calgary, 2002

Carleton University, 2003

Concordia University, 2002

Lakehead University, 2002

McGill University, 2007

McMaster University, 2001

University of New Brunswick, 2006

University of Ontario Institute of Technology, 2009

University of Ottawa, 2001

University of Regina, 2007

University of Victoria, 2007

University of Waterloo, 2006

University of Western Ontario, 2001