COMPUTING EDUCATION IN SWITZERLAND

Regarding “The Future of the Computing Profession” (The Profession, July, pp. 88, 86-87), some parts of the system that Neville Holmes describes already exist in Switzerland.

In addition to the classic university education, there is a vocational training/professional education system with a long tradition (sometimes referred to as a dual-education system, similar to the German system; tinyurl.com/24fpxbo) in which about 60 percent of the Swiss population between ages 16 to 20 receives its education. After this three-to-four-year course of study, it’s possible to pursue further study at the university level, but for many professions, this provides a popular and solid education.

In 2000, a new course for the information system professional was introduced (tinyurl.com/2cfaco6). This is a very popular program, mostly offered by companies that take on the responsibility for the practical aspects of the education program. Students spend two days a week receiving a classic classroom education, and they spend the other three days in the workplace.

Thus far, there is no real conflict between computer science majors receiving a university education and professionals in a four-year vocational training program.

Fabian Meier
fmeier@gmail.com

SOFTWARE ASSURANCE

Regarding the very important topic of improving software security in Computer’s September issue (Samuel T. Redwine Jr., “Fitting Software Assurance into Higher Education,” pp.41-46; and Ann E.K. Sobel’s interview of Gary McGraw, “Software Security in the Real World,” pp.47-53), I suggest that one effective tactic worth considering would be to work at the root of a problem by having university computer science department require secure programming in all courses for CS majors in order for the departments to be accredited.

Phase-in could be straightforward as accreditation reviews come up by the usual schedules, with early adopters getting special attention from employers and students. Criteria could be as straightforward as requiring all student programs to pass some standard security verification (such as the present Flawfinder or similar programs) as part of the homework turn-in process, similar to well-established processes of checking for plagiarism. Refinement could be separate from, and thus not a load on, academic activities, as accreditation agencies and providers of security turn-in sites or software independently pursued their own improvement paths, which they could choose to make community-accessible. In addition, making such a requirement part of accreditation would give clear justification for department budgets to include software security.

Thomas Reynolds
trie3@computer.org
www.trafford.com/07-1219

Ann Sobel responds:
Thanks for your suggestion of targeting the root of the problem: why do students know so little about software assurance/security?

The topics covered in any computer science program are influenced by (at least) two entities—the IEEE-CS/ACM Curriculum Guidelines for Computing Programs and the ABET accreditation criteria for computer science programs. The CS2008 computer science curriculum guidelines respond directly to this issue in that they include the topic of software assurance/security throughout the recommended core and elective materials. Since most computer science programs pay careful attention to curriculum recommendations, it can be expected that these programs will pay more attention to software assurance. Furthermore, curriculum recommendations have influenced the program outcomes contained within the ABET accreditation criteria for computer science, although these criteria tend to change relatively slowly.

If software assurance is regarded as an essential topic, it will eventually be incorporated into the required computer science program outcomes. Once this happens, a program seeking accreditation must demonstrate that its graduates are enabled to achieve this outcome. This will cover more than 300 computing programs.

On the other hand, we need to temper the claim that software assurance/security must appear in all software-related courses. There is an enormous number of topics that academics believe must appear in computing programs, and not all of them can be included in the available program hours.

Your suggestion is obviously in line with my beliefs and will most likely be aligned with future versions of the IEEE-CS/ACM Curriculum Guidelines and eventually with accreditation program outcomes.

Although I found Computer’s September issue on software assurance very interesting, I submit that there may be better approaches. Certainly, there are at least useful adjuncts that would improve the assurance of proper software.

In the DOS days of the 1980s, I independently showed that it is feasible to build an absolutely scumware-proof PC. A beltway bandit came up with the identical solution
on a USAF contract in that same time-frame. However, I never saw that such a solution had been used by the government, which rolled out Fortezza/Missi and other Band-Aid approaches instead.

Currently, it’s still possible to architect a graphics-interface-oriented PC that is absolutely secure from outside attacks. But it’s essential to build in the security initially by using a proper systems architecture.

With outside attacks eliminated from consideration, we only need to solve the problem of preventing a software factory from delivering code that could unilaterally cause damage. And the operating system portion of the scumware-proof PC, along with the special hardware used, can eliminate those possibilities. There are other techniques that could virtually eliminate building problems into the applications.

Why doesn’t the government follow up on a proven solution instead of making Rube Goldberg attempts to tack on security after the fact? Are they afraid of the antivirus industry’s demise? Do they prefer to be able to hack into other countries’ PCs rather than protecting ours, even if it leaves our critical infrastructure at risk?

If the government doesn’t choose to embrace a proven secure solution, perhaps a handful of Fortune 100 companies would be willing to fund the development and take control over the IT environment while eliminating the cost and effort of constantly upgrading versions of Windows alternating with new PC hardware every year. This approach would pay for itself in short order by eliminating risks as well as better controlling costs imposed through an excessive rate of change; and by the commissions that would be made from selling the solution to other companies, and even other governments, that do care about security.

William Adams
williamadams@ieee.org

**NETWORK ACCESS CONTROL**

In “Whatever Happened to Network-Access-Control Technology?” (Technology News, Sept. 2010, pp. 13-16), David Geer states, “Customer confusion about NAC has contributed to lower-than-expected adoption.” Today, IT professionals must be in the forefront in the battle against hackers and thieves because security threats are coming from all angles.

The increasingly complex enterprise networks and ever-changing environments present a wide range of challenges for the IT professional. The network access layer is particularly vulnerable because it is behind the organization’s firewall and is part of what most enterprises consider the trusted network. The use of mobile devices such as laptops and smartphones combined with a host of other devices ranging from security cameras and POS systems to USB devices and printers causes network boundaries to disappear.

Misconfigured network access control can cause network vulnerabilities. Managing devices such as routers, firewalls, and gateways is a major challenge. Thus, tools that integrate network and application-level access control in a single framework from design, verification, and optimization play an important role in creating a successful NAC implementation.

On the other hand, a company’s employees need to use multiple applications, and therefore different authentication mechanisms with varying strength. Identity and access management are required to secure access to confidential resources. Adding security policies that are specific to the device, the user, and its location will enforce policy compliance, facilitate network access control, and help to close existing and potential future security gaps.

The most crucial step in managing NAC requires an understanding of what devices and users are connected to the network; whether they are authorized to use the network connection; where they are connected; if they meet the security standards; and what resources they require.

As the security risk is dynamically changing in networks due to new threats or users’ behavior, enabling proactive access control will be essential in future network defense.

Hong-Lok Li
thl@ams.ubc.ca

**EXPLOITING NEUROPLASTICITY**

Neville Holmes has produced another thought-provoking article in his September The Profession column (“Seven Digital Steps to Avoid Utter Hell,” pp. 92, 90-91). Neuroplasticity has a dark side, however. It goes by the name of brainwashing. History is replete with examples of movements that seemed obviously beneficial but that ultimately proved disastrous.

Digital technology is not necessary to convince large numbers of people that a particular assertion is true. Human beings prefer consensus and will go to great lengths to establish one. In an experiment conducted in the 1950s to see if peer pressure would affect how individuals viewed something, Solomon Asch showed that many people would ignore the evidence of their own eyes in order to maintain the consensus of the group.

As it turns out, the tools that Holmes proposes as a means of achieving consensus will probably do the opposite. These tools enable people to think for themselves. When you give individuals the ability to do their own thinking, they have a disconcerting tendency to do exactly that, and they will often reach conclu-
LETTERS

REMEMBERING WATTS HUMPHREY

The worldwide software engineering community lost one of its brightest lights when Watts Humphrey, who was called the father of software quality, died on October 28. I feel the loss not only professionally, but also personally because I worked with Watts for 23 years at the Carnegie Mellon Software Engineering Institute (SEI).

I met Watts when I joined the SEI in February 1987. Watts had started a few months before I did and was just beginning to staff the Software Process Program and establish the program’s technical agenda. With Watts leading the way, the members of this group produced the initial appraisal methods and evolved the five-level maturity framework, creating the Software CMM. By 1992, these concepts were becoming well established in the community, and Watts began to work on applying his theories to software engineering practice.

Watts first tested his theories on a process that he developed for managing his personal checking account. Next, he tested this on the personal software development process by writing more than 60 small programs in Pascal and C++. Then he began working with organizations to pilot this new personal process for software engineers.

Not long after, Watts published his first Personal Software Process (PSP) book, A Discipline for Software Engineering, and developed a course for software engineers. When I enrolled in the first PSP course offered at Carnegie Mellon University, it changed my career.

When you learn how to properly measure your own performance and analyze the result in order to improve, you get real, lasting, behavioral change that leads to performance gains and improvement. In the class, we went from underestimating our work by about 40 percent to being within a few percent under or over estimate on each assignment. By the end of the course, we had a 10 times reduction in the number of defects that escaped to the unit-testing phase. These results were unbelievable. If I hadn’t been there, I wouldn’t have thought this possible. The next challenge was how to achieve similar results on real projects.

After the course, I began working with Watts to transition Team Software Process (TSP) and PSP into software engineering practice. During the course of our work together, we became close friends. What will stick with me? First, the belief that with both the maturity model and PSP/TSP, Watts created a framework that is the right stuff for software engineering and probably most kinds of related work. It works. Second is the value of data. When you have personal data for a process, you have a fundamentally different kind of understanding of how that process works.

I will never forget the Watts-isms. Watts was a master at reducing the complex to the simple, and there are hundreds of these little gems. For example:

- **Watts on planning**: If you can’t make accurate plans, plan often.
- **Watts on producing quality work**: If you want a quality product out of test, you must put a quality product into test.
- **Watts on life**: In life, we all reach the same end, so we need to concentrate on the trip. Devote yourself to excellence, and you just might achieve it. That would be worth the trip.

Watts made a commitment to improve the profession of software engineering, and when he made a commitment, he meant it. He inspired many professionals to join him on this mission, where he was equally committed to us. His commitment is now ours.

—Jim Over, Senior Member, Technical Staff, Software Engineering Institute

on provenance and reproducibility as they relate to computational science and engineering. Thus, I was delighted to read his column contribution titled “e-Science and the Web” in Computer’s May 2010 issue (Web Technologies, pp. 90-93), where he echoed my discussion of the distinction between repeatability and reproducibility. Although his column includes URLs to various websites, it omitted a URL for my work. Interested readers can find the original discussion of this topic in my paper written in 1998 and available at www.toolsmiths.com/docs/CT199801.pdf.

Any discussion of provenance and reproducibility for computational science and engineering that does not also address citation and attribution leads to a contradiction in terms. It is not possible to maintain standards for scholarly peer-reviewed reproducible science without proper citation and attribution. Even in a one-page editorial, care should be taken to attribute ideas and contributions to their original authors. In a four-page column, confusion quickly arises as to whether the text represents a contribution of the column’s author or discussion of another author’s work.

Unfortunately, failure to cite and attribute properly has not been limited to editorials or columns but has become a growing problem impacting many peer-reviewed papers. Another worrisome trend has been the growing frequency of the appearance of statements about funding that is featured prominently in abstracts and introductions rather than in footnotes or acknowledgments. But funding of any amount is not the substance of the science or engineering that should be presented in a paper for scholarly review and publication.

Additional commentary with suggestions for practices that might alleviate some of these problems can be found in a blog article at www.portaldoors.org/Blog/PostID/5.aspx. Carl Taswell ctaswell@computer.org