

CURRICULUM MERGER QUESTIONS

The authors of “An ‘Offshore-Resistant’ Degree Program” (David A. Swayne, Qusay H. Mahmoud, and Wlodek Dobosiewicz, *The Profession*, Aug., pp. 104, 102-103) describe a merger of curricula between computer science and communications. The first paragraphs appealed tremendously, until I realized that by “communications” they mean the techie skills needed to build more infrastructure. Oddly, the other side of communications isn’t addressed in the coursework at all.

Offshore-proofing a software career means becoming a master of communicating between the engineering team and the boss, the customer, and other stakeholders. The person who bridges the technology/customer/business-unit gap will be in hot demand. Yet the only nontech class an engineer would encounter using the proposed curriculum is tech writing. The graduate of such a program will be a computer science drone who won’t know how to compose a letter to the welfare agency when his job goes to India.

Another glaring omission is software engineering. Only one class, titled Distributed Software Engineering, is even likely to address process issues. Anyone can code. Real software engineers know how to build world-class software. They won’t learn those skills in this too-tech-laden program.

Jack Ganssle
Baltimore, Md.
jack@ganssle.com

The authors respond:

In our description for *Computer*, we chose to concentrate on the more glaring curriculum problems in a typical university program. Mr. Ganssle lists some problems that we omitted in our description, but not in our design. Space constraints limited coverage, but we went well beyond “technical.”

Ethical Issues, Workplace Preparation and Technical Writing, and Technology and Society are courses aimed at three different levels in the program.



They require critical analysis, and they develop formal writing and presentation skills. Three other free electives are offered, and a period of study in Europe is an option.

All senior programming courses—Cryptography and Network Security and Internet Technologies being two examples—have significant and relevant software engineering components. Students produce four comprehensive reports, one for each four-month work term. These reports are a requirement for work-study programs in Canada. Our decision to delay the work-study program until after Semester 4 is part of the plan to increase the students’ chances for successful outcomes. Our first cohort was 100 percent employed.

The senior paper is a full-fledged honors thesis requiring the student to produce a project plan in Semester 7, with execution to be completed under supervision before graduation. The curriculum engages both institutions’ learning objectives. The degree programs must demonstrate a plan for students to attain mastery of these objectives by program completion.

A longer description of the Guelph-Humber skills and values objectives is available at www.guelphhumber.ca/academic_programs/academic_philosophy/index.shtml.

We appreciate this opportunity to expand the description of our program.

PATCH ON DEMAND SECURITY

In “‘Patch on Demand’ Saves Even More Time?” (*Security*, Aug., pp. 94-96), Angelos D. Keromytis proposed using a vaccination system that could automatically generate patches to pro-

tect an application’s source code against viruses, worms, Trojan horses, and other hacker attacks. Waiting to develop a security patch after an attack occurs could prove to be a fatal mistake. A patch must be available somewhere other than on the system itself to apply when it is needed.

Currently, although some products, including antivirus software, claim to offer “patch on demand” security, they simply perform a regular check on the system and update its definition files.

The author suggested that it might be possible to integrate vulnerability discovery and patch generation into a system so that it could “identify a previously unknown vulnerability by observing the actions of an attacker.” This is more like an artificial intelligence approach, which has better chance of dealing with “zero attacks.”

Two AI approaches can be helpful in implementing this process: the model-based technique and artificial neural nets.

Developers could use the model-based technique to construct behavior and attack pattern models to use in predicting the outcome of various solutions to hacker attacks. The challenge is to build the models correctly to capture hacker attack characteristics.

The more sophisticated artificial neural nets concept requires using software to develop experiments modeling the behavior of the human brain. Each example of an attack causes the neural net to change its structure and store the new information as a pattern of weights. After training, the computer could use knowledge nets and their corresponding weights to find certain actions. The question is how to properly identify the attack patterns stored in the knowledge base.

To prevent computers from becoming easy prey for hackers, patch on demand must be capable of generating security patches by itself and applying them instantly.

Hong-Lok Li
Vancouver, B.C.
lihl@ams.ubc.ca

DESIGNING FOR MISUSE

Bob Colwell's "The Art of the Possible" (At Random, Aug., pp. 8-10) was both wise and amusing, but I must take issue with one of his examples—that of the infamous hot cup of coffee in the lap, which he uses to impugn the intelligence of the user of the product.

Let's look at it from McDonalds' vantage point. Here is an item *designed* to be sold at drive-through windows to people in cars, who will obviously not have a stable place to set it down. McDonalds *purposely* makes its coffee 20 degrees hotter than anyone else does because doing so extracts more color from the grounds, so the company can use less coffee and save money.

There had been more than 700 complaints about less severe burns from the excessively hot coffee before the case that made the news, so McDonalds *knew* it had a problem. However, the company paid out less to the victims than it saved on coffee bean costs, so it chose not to change its practice.

This is exactly analogous to the Ford Pinto case, where management *knew* that their product was dangerous, but as long as the amount they paid out for injuries to people was less than what they saved by not fixing the problem, they made a "management decision" to let people be injured.

Products for mass consumption need to be designed with the knowledge that they will sometimes be misused, and the wise engineer will try for a design such that foreseeable misuse will not be disastrous, as with finger guards on power saws.

Sometimes the only way to change the minds of the MBAs is to make the injuries sufficiently expensive to the company that fixing the design defect becomes profitable.

Harvey S. Frey
Santa Monica, Calif.
hsfrey@earthlink.net

Bob Colwell responds:

There was clearly more to the McDonalds coffee incident than I

knew about, and I thank Harvey Frey and several other alert readers for bringing it to my attention. After reading through several accounts of this lawsuit on various Web sites, I must acknowledge that, whether I agree with the lawsuit's outcome or not, it is at least not the poster child for frivolous money-grubbing that I had assumed it was.

None of the Web sites I saw made any reference to the color of the coffee, nor to the high temperature being some kind of profit-generating tactic. Most of them referred to the better taste associated with higher temperatures.

It does seem to me that McDonalds should have been willing to cover the medical costs involved in this accident, and the plaintiff said she would have settled for that outcome had McDonalds agreed to it.

As for products needing "to be designed with the knowledge that they will sometimes be misused," well, I agree with Frey's proposal that wise engineers will do this. The problem is that it's such a short step from that observation to the degenerate viewpoint that if anything happens to me, including things due to my own inexperience, bad judgment, or bad luck, then somebody somewhere owes me a lot of money. That short step has within it the potential to ruin innovation to the point where businesses are no longer competitive. And I think that issue is two orders of magnitude more important than whether McDonalds' coffee is served too hot.

OPEN SOURCE CHALLENGES

Having been a software developer for decades, I found "A Critical Look at Open Source" by Brian Fitzgerald (IT Systems Perspective, July, pp. 92-94) especially interesting.

Although I'm too old to become an F/OSS expert, I strongly support F/OSS usage in the company I work for. I agree that Fitzgerald correctly identifies the challenges, including development talent, code quality, and code modularity.

But are these new risks for software development? Why would a conventional approach, that is, proprietary software, avoid those risks? The same faults have affected software development since its early beginnings.

Maurizio Vinci
Bresso, Italy
maurizio.vinci@computer.org

The author responds:

I too am a committed supporter of F/OSS, a fact that may not have come across in my article. However, I intentionally presented a critical view to stimulate debate on F/OSS and to identify potential problems in advance so they can be addressed to ensure that F/OSS flourishes.

As it happens, I agree with Maurizio Vinci on most of the points he makes. I think his conclusions actually bear out my suggestion that F/OSS actually is not a paradigm shift in software engineering. However, I do believe that the problems identified are more critical in the case of F/OSS in the absence of colocated software developers working for a company with all the traditional organizational sanctions this implies.

I would choose to disagree with Maurizio regarding his comment that he is too old to become a F/OSS developer—the F/OSS community needs all contributions, especially from the older and wiser constituency.

We welcome your letters. Send them to computer@computer.org. Letters are subject to editing for style, clarity, and length.

ERRATUM

In *Computer's* July issue, Figure 2 in "Policy-Based Dynamic Reconfiguration of Mobile-Code Applications" by Rebecca Montanari, Emil Lupu, and Cesare Stefanelli included several typographical errors. *Computer* regrets the error. The revised figure is posted in the Computer Society's Digital Library at <http://csdl.computer.org/comp/mags/co/2004/07/r7073abs.htm>.