

## Finding a Job

**B**ob Glass's Loyal Opposition articles are always worth reading, but his July/August column ("A Big Problem in Academic Software Engineering and a Potential Outside-the-Box Solution") is unusually relevant. To give some context: My original degree is in computer science (which didn't prepare me for software development in the real world). Over the past 11 years, I've worked in various IT roles with a power utility and financial services companies. I'm currently completing a part-time MBA, considering doing a PhD, and probably teaching an MIS course next January whose content I'll help define.

During the course of my MBA (and out of personal interest), I did some research on systems development practices. I've been bemused by the gaps, with some notable exceptions, between information systems researchers, software engineering researchers, and computer science researchers. These gaps occur even in areas of overlapping interests (agile versus formal methods, reasons for project failure, and so on). Maybe some of this has to do with the fact that engineering, science, and commerce—the homes for software engineering, computer science, and management information systems—often belong to different faculties, with the usual accompanying academic baggage. Some interdisciplinary practice certainly wouldn't go astray.

In terms of your discussion of adjunct professors' effectiveness, it might have a lot to do with motivation. The adjuncts I've come across

are often genuinely interested in teaching and do it part time. Tenured professors are typically interested in their academic careers; teaching effectiveness has only a small bearing (wrongly, in my view) on career progression. This doesn't preclude good teaching by full-time staff, but it doesn't help it. Adjunct staff often have much more motivation for teaching effectively. The other category of excellent teachers I've come across are what I term the "wise owls," full-time faculty members (interestingly most with masters degrees) who have retired from industry, bringing 30 or more years of experience to bear in teaching.

*Dermot Casey*  
Project manager  
GE Consumer Finance  
Dermot.Casey@ge.com



There is another solution to the recruitment problem in software engineering departments, one that is ideally suited to the SE field. I now believe it should be mandated.

I am a senior engineer at Quantel, making high-end film-editing and special-effects systems. I have been working in industry for nine years. I just finished my PhD, taking more than six years, one day a week. The two systems I built as a student received over 20 person-years of effort, and I used these systems at work as test cases.

Six years ago, I asked my then-employer for a day release to study for a PhD. With funding from my employer, I applied to Reading University. In the middle of the research, I left that job and proposed my day release and research

We welcome your letters. Send them to software@computer.org. Include your full name, title, affiliation, and email address. Letters are edited for clarity and space.

requirements to all prospective employers. It was a good yardstick in interviews to see if a company was willing to commit to my four-day week.

I do not know of anyone else who has taken this approach to PhD work. It has made my career exhilarating. It has also let me observe how the other half lives from both sides of the fence.

I believe that the future of software engineering lies in both industry and academia. If you are not well versed in both, you can't say that you study software engineering. I believe that to claim you're a software engineer means you've been part of a team that has written large programs and shipped them to customers or users. Obviously, this could include open source and does not mandate a commercial setting. Without this sort of experience, many academics do not understand the problems I've been studying.

So, to those who hire, I recommend recruiting PhD grads with an industrial track record and mixing part-time post-doc "industrial" researchers with full-time academics.

*James Cain*  
Senior engineer  
Quantel Ltd.  
james.cain@quantel.com

After reading Bob Glass's article in the July/August issue, I have some questions.

How hard is it for older engineers to transition into academia (retiring from one career to start another, teaching)? I am thinking about pursuing my PhD after about 20 years in industry, and I recently got my masters at night school. Is getting a PhD a young man's game? What if I am more interested in teaching than in research? Would that disqualify me? How important is tenure? There is no such thing as tenure in industry.

Also, I teach as an adjunct at night now, and talking to one full-time professor, I got the impression that few peers are available to talk to during the day. What other cultural differences should I be aware of?

*Bob Schaefer*  
Adjunct instructor  
Daniel Webster College  
schaefer\_robert@dwc.edu



SINGAPORE  
MANAGEMENT  
UNIVERSITY

## School of Information Systems Openings for Faculty

Applications for tenure-track and practice-track are invited at all levels.

**The Singapore Management University (SMU) was officially incorporated in January 2000. It holds the unique position of being Singapore's first private university funded by the government of Singapore. SMU's mission is to generate leading-edge business and business-technology research with global impact, and to produce creative and entrepreneurial leaders for the knowledge-based economy.**

### SMU – Carnegie Mellon Partnership

SMU and Carnegie Mellon University (Pittsburgh, USA) have entered into a close partnership to jointly establish the SMU School of Information Systems (SIS). Carnegie Mellon faculty are actively participating in SIS faculty selection, mentoring and development, and in the design of the SIS undergraduate curriculum, research centre, and post-graduate and professional programmes.

### SIS Research themes include:

#### 1 e-business technology and management

- Business process integration
- Collaborative work
- Data management
- Internet distributed computing (including web services, grid, peer-to-peer)
- Mobile, embedded and pervasive computing

#### 2 Information security technology and management

- Available and secure computing services and systems
- Available and secure network services and systems
- Secure and legal access to devices and data
- Privacy and trust

#### 3 Architecture and software engineering

#### 4 Information systems management

- Economic and risk analysis of information systems
- Management of design and development projects
- Management of ongoing IT operations

SIS research and educational projects will demonstrate innovative IT applications and economic value propositions in the following industry sectors: financial services, supply chain & logistics services, manufacturing, health & medical services, and the public sector.

SMU is committed to innovative pedagogy. Candidates must be capable of designing and delivering one of the following undergraduate courses: Object oriented systems, Data management, Networking, Software engineering, Enterprise systems and integration, Security, Performance & quality of service, or Architectural analysis.

Tenure-track applicants must have a PhD from an internationally recognized university in the areas of Information Systems, Information Technology, Computer Science or related disciplines and an outstanding record of academic research and journal publishing that is commensurate with their desired rank. Tenure-track faculty must also demonstrate a strong interest in research oriented business applications in the targeted industry sectors.

Practice-track faculty applicants must also have a PhD in the related IT disciplines from an internationally recognized university, an outstanding record of participating in leading-edge applications that impact business practice, and a record of professionally relevant publications in applied magazines or conferences.

Qualified candidates should submit a cover letter, curriculum vitae, and three letters of recommendation and samples of published work. All candidates please submit electronically or hardcopy to:

**Dr Steven Miller, Dean, SIS**  
c/- Office of Faculty Administration  
Singapore Management University  
469 Bukit Timah Road  
Singapore 259756  
Telephone: +65 6822 0385  
Email: siscv@smu.edu.sg

Selected candidates will be asked to interview at Carnegie Mellon University

Singapore Management University  
469 Bukit Timah Road  
Singapore 259756

SMU. We Mean Business

[www.smu.edu.sg](http://www.smu.edu.sg)

**Bob Glass responds:**

Although finding a job in industry can be harder when you're older, I feel that transitioning to academe when you're older would not be. At least, that is, if your area of expertise is software engineering, since there are all too few industry software engineers who have a PhD and want to teach.

Teaching is the specialty of smaller universities; you would have an easy time getting into one of those, but not as easy getting into a larger one (they tend to specialize in research).

Tenure is a huge barrier—it's "up or out." That is, if you aren't granted tenure, you have to leave your institution. However, if you have a PhD and are good at what you do, that needn't worry you.

Cultural differences? Industry people *do*, and academic people *envision*. If you like doing, that could be a problem. Academics are better talkers than listeners, typically (because they're paid to talk, less so to listen). Academics love pursuing concepts; if you like engaging conversations, academe is a more fun place to be.

**Kudos**

I've enjoyed reading several recent articles in *IEEE Software*—especially "Verbing the Noun" by Dave Thomas and Andy Hunt and "The Difference between Marketecture and Tarchitecture" by Luke Hohmann, both in the July/August 2003 issue.

I usually read most of the articles in *Software* and find something to think about along the way—can't ask for more than that :-)! Keep up the good work.

Linda Rising  
Independent consultant  
risingl@acm.org

**Skipping the Bandwagon**

I have long enjoyed Bob Glass's columns. I just got around to reading his "Questioning the Software Engineering Unquestionables" in the May/June issue.

I found myself in such violent agreement on the "let's not just jump on any bandwagon" approach that I felt compelled to write you this note. I am the chief software architect (lofty title) for Lockheed Martin Air Traffic Management programs. The systems we build are safety critical. About a year ago, we had a band of young programmers (OK, I've just turned 50) suggest that we use XP practices. My first reaction was "What's broken with the current



process? If it is something, then let's concentrate on that rather than jumping to a completely new paradigm." After reading up on XP, my second reaction was "... sounds like it might work for small projects, but not for the kind of systems that we build." I'm glad to see somebody else had the same view. In the end, I more or less prevailed; we did examine our current processes to see if we could apply any aspects of XP. (It didn't hurt that management didn't want two people working each line of code.)

It seems to me that most new technologies are elevated to buzzword status without full examination, including "field trials." In general, whenever I evaluate some new process or case tool, I have to throw out any claim made by the vendor, see how it would fit into our existing structure, and estimate savings (or loss!) based on our current process measurements. Usually things don't come out so far ahead in the harsh light of day.

Your column is one of the few that examines things with an unbiased eye, and it is usually the first thing I turn to when I open up the latest copy of *IEEE Software* (even if it is after that copy has been on the shelf for a while). Thanks!

Jon Dehn  
Chief software architect  
Lockheed Martin  
jon.dehn@lmco.com

**Postwaterfall**

I enjoyed Warren Harrison's article on classical software engineering ("Is Software Engineering ... over the Hill?") in the May/June issue. I think the waterfall method becomes less relevant as the development team gets closer to the user. It's harder to do something linear and sequential if you're in regular contact with the user through email or instant messaging. When you're that close, you're receiving bug reports in near-real time and can respond with a fix in a matter of hours.

Previously, if you rushed and sent out something full of errors, it would reflect poorly on you and take an incredible amount of resources as you sent out fix after fix through the mail or in person. Thus, it made a great deal of sense to hold to a regimented waterfall model. That's not the case now. As people are more accustomed to instant gratification with product delivery and maintenance, they are more accepting of a less-than-perfect product—they know how quickly they can receive a fix from the developers.

So, I think the waterfall method is being replaced by Barry Boehm's spiral method (I think it was Boehm, at least). That way, you get something highly iterative (pleasing to the users) and systematic (good for the engineers). You're bound to loop around the spiral several times if you're operating this way, but it seems to fit more with what users expect.

Cody Powell  
Software development engineer  
American Innovations  
cody.powell@aimonitoring.com