Through a Glass, Darkly
Controversy Is Brewing:
Math is Overtaught, and Business Undertaught,
In Academic Computing Curricula

Robert L. Glass

There’s a controversy brewing that could well explode in the computing field in the next year or two. It is perhaps only somewhat relevant to Information Systems (IS), and yet what happens in the broader computing field almost always has repercussions for IS.

That controversy is about the wisdom of teaching mathematics to computing students. Academics in the computer science (CS) and software engineering (SE) fields, especially, have a deep-seated belief that math is an important scholastic background for those who develop software. That belief is not only institutionalized in CS and SE curricula, but it tends to be the foundation for many academic papers and discussions about where the computing field is, and where it should be. CS and SE people simply believe that math is a critical foundation for whatever happens in the computing student’s professional future.

Where did the controversy come from? Nearly a decade ago, an academic named Tim Lethbridge did a study [Lethbridge 1998] in which he sought to learn what academic subjects practicing software engineers had taken that they really didn’t need/use in practice, and which ones they wished they’d had that the academic world hadn’t offered them. In other words, he did a gap analysis of pedagogy vs. practice in the software engineering field. And, in fact, he not only did the study once, but he made some changes in what he had done and did it again, both times getting nicely consistent results.

His findings were fascinating. There were two major findings. We’ll discuss the first one here. The second one, perhaps more interesting to the IS field but less surprising in scope, we will discuss later.

That first finding was this:

“Mathematical topics were not very important to software engineers and appear to be taught more extensively than required.”

Traditional computing academics, as you might imagine, were appalled at this finding. One good academic CS/SE friend of mine, with whom I joke about our conflicting views of the role of mathematics in our field, wrote in a professional journal that the reason practitioners don’t use math is not because it is not useful to them, but because they have been insufficiently exposed to it. In other words, the Lethbridge findings were twisted to the extent that they became support for the pro-math position.

But that is definitely a twisting of the findings. In fact, at this point, I would like to editorialize a bit (this is, after all, an editorial column!) My academic background was
Mathematical. I earned a BS and an MS at one of the prime citadels of academic mathematics, the University of Wisconsin. I proudly carried my math sheepskin out into the practitioner world, only to gradually realize that almost nothing I had studied in academe was useful to me in the brand-new field of computer programming.

Actually, that’s an exaggeration, but only a slight one. I found myself, fresh out of college, in a job doing scientific/engineering programming. And scientific/engineering programming does indeed make rich use of mathematical knowledge, and in fact math is at the center of solutions to problems in that field.

But, in the job I was doing, the only math I needed was a math I had never taken – three-dimensional geometry! My job was doing “Master Dimensioning” of aircraft parts, figuring out how to manufacture and fit aircraft pieces together to a ten-thousandths of an inch tolerance. Highly technical. Highly mathematical. But having nothing to do with the math I had learned in academe.

Now that, of course, could well have been an anomaly. In other application domains, I would have to have acknowledged at that point in time, other forms of math might have been very important.

But my software practitioner career progressed in fascinating ways. I did systems programming, building compilers and other software tools. I did business systems programming, the bread and butter stuff that all companies do. And what I found, during that progression, was that there was absolutely no math involved in any of those problem-solving activities. There was no math in compiler-writing and tools development, except for trivial number conversion. There was no math, except for trivial arithmetic, in business systems programming. It became obvious over a short period of time that my mathematics educational background was (a) not very useful, and (b) gathering rust rapidly!

So long before Lethbridge’s study, based on the fairly broad sweep I personally have taken through the field of software practice, I was convinced that the need for mathematics in the field was vastly overblown. I continue to feel that way. I was not surprised by Lethbridge’s findings, and in fact I welcomed them as a vindication of my own position.

Now, given the age of the Lethbridge studies, what makes this topic something of interest to computing folk in 2007? Just this. Although it is uncommon in the computing field to replicate previous studies, someone has just recently done that with Lethbridge’s! In a recent study [Kitchenham, Budgen, Brereton, and Woodall 2005], the authors redid the earlier study, corrected a few research flaws they perceived in Lethbridge’s approach, and asked nearly the same set of questions of their target audiences that Lethbridge had asked of his.
What was the result of the later, quite recent, study? “Our study appears to confirm Lethbridge’s observations…” Once again, “mathematical topics were not very important to software engineers and appear to be taught more extensively than required.”

New chapter, same verse. According to practitioners who ought to know, math is overtought in the CS and SE curriculum.

As I said before, I’m sure we haven’t heard the last of this newly-stirred controversy. Traditional CS and SE academics are simply not going to give up on math without a fight. Stay tuned!

Now you may be wondering about that other finding, the one that is more relevant to IS academics, the one that won’t be very surprising? It was this:

“General business topics are quite important, but are not taught in proportion to their importance, in particular, management, giving presentations, leadership, and negotiating.”

That one, of course, is simply a reflection of one of the things that most distinguishes an IS education from a CS or SE one. A business focus, IS academics have known from the very beginnings of the field, is vital to business computing students. They offer those courses all the time. But, these studies found, CS and SE students are not getting what they need in those areas.

So I suspect there won’t be much follow-on to the second finding of the Lethbridge and Kitchenham studies, although there should be. But, on the other hand, there will be battles fought over that first one!

References:


“Through a Glass, Darkly,” is a Biblical expression for the unclear way in which we see the world around us.

Robert L. Glass is president of Computing Trends, publishers of The Software Practitioner newsletter, and an Honorary Professor of Software Engineering at Griffith University, Brisbane, Australia. He has been active in the field of computing and software for over 50 years, largely in industry (1954-1982 and 1988-2005), but also as an
academic (1982-1988 and 2005-present). He is the author of over 25 books and 90 papers on computing subjects, Editor of The Software Practitioner, Editor Emeritus of Elsevier’s Journal of Systems and Software, and a columnist for several periodicals including the two leading journals in his field, Communications of the ACM (the “Practical Programmer” column) and IEEE Software (“The Loyal Opposition”). He was for 15 years a Lecturer for the ACM (the leading computing professional society), and was named a Fellow of the ACM in 1998. He received an honorary Ph.D. from Linkoping University in Sweden in 1995. He describes himself by saying “my head is in the academic area of computing, but my heart is in its practice.”