Robert L. Glass

# Revisiting the Industry/Academe Communication Chasm

*"The Greek notion of science held it above the vulgar pragmatics, leading to a pedantic tendency that tolerated intellectual laxity, sometimes with tragic consequences."* —Noah Kennedy, in *The Industrialization of Intelligence*

Three incidents happened recently that trouble me. Those three incidents were all attacks on the state of the practice of software by academics who apparently sincerely believed what they were saying.

One incident might not have gotten to me. It happens, from time to time, that academe seems to need to put down those who build software for a living. Two attacks I might have tolerated and ignored. But three was too much to bear. I fear the communication chasm between academe and industry, which I believe has existed for over 25 years (I gave a keynote on the subject to a Burroughs users group in 1971!), is getting worse, not better.

What were the three incidents? Let me describe them:

1. A computer science academic proposed to create a special issue for the *Journal of Systems and Software*. In the motivational part of the call for papers inherent in his proposal, he said this:

"Given the historical software crisis, most serious software practitioners in the software-building world are desperate for better ways of developing reliable systems . . . and would therefore adopt any tool and notation that works toward achieving reliability goals . . . ."

In that one sentence, I found three offensive things. First, there was a mention of a "software crisis." I thought the time had passed when computer scientists began every article with cries of crisis. There was good reason for that change, I thought. *Most software projects are largely successful, sometimes spectacularly so*. Sure, there are also failed projects, but they tend to come about only in massive projects that tax the state of the practice, and in optimistic projects using untried technology that taxes the state of the art.

Secondly, in the statement the word "desperate" was included. This was the most offensive cut of all. The software practitioners I know quietly and confidently go about their business of building software systems. They know and expect that systems are growing in size and complexity by a factor of 50 or more every 10 years. They accept and welcome the challenge of

ROBERT NEUBECKER

building systems that interweave with the very fabric of modern society. There is no sense of desperation in any of these people, nor should there be.

Thirdly, there was the phrase, "adopt any tool or notation." "Breakthrough" candidate tools and notations, accompanied by almost unbearable hype, are the norm in the software world. The capable and dedicated software practitioner has learned to ignore the claims of breakthrough, ignore the hype, and cut to the chase of the topic: Does the benefit of this new proposed technology exceed the cost? Does anyone know the answer to that question? More often than not, no one knows the answer. All too often, the benefit only marginally exceeds the cost (those studies that do exist tend to show the benefits of new technologies touted as "breakthroughs" lie in the 5% to 30% improvement range). "Adopt any tool or notation" out of some sense of "desperation"? Au contraire. Practitioners have learned to be wary, and reject new ideas (for better or for worse) much more often than they embrace them.

2. The second incident occurred in a paper submitted by an academic author to the same journal. In this submittal, the author made the statement: "Software tools are usually larger and more complex than application software." Acting as a reviewer, I was troubled by this statement. I have worked professionally in both the tools-building and the application-building world of software. In my experience, applications are almost invariably bigger than tools. But not knowing of any data on the subject, as a reviewer I asked the author to either support his position by citation, or remove it if he could not find support for it.

The author did neither. In his revision, the statement was changed to, "tools are larger and more complex than the majority of applications." Obviously, the author was completely convinced of the truth of the point he made, and was not interested in backing away from making it, even though he was not able to come up with the requested citation.

Two things trouble me about this incident. The first is this: academics, who are very interested in the construction of software tools, often engage in relatively complicated tools projects; but those same academics are not very interested in applications, and seldom engage in any application work larger than "toy" projects. Their built-in bias, based on their own personal experience but no experience beyond the hallowed and ivy-covered halls, thus supports the "truth" that tools projects are larger than applications. But this is insular and incorrect.

What also troubles me is the "cascading citation" problem. If I allowed this author to leave his particular "truth" in his work, then another author downstream could cite that position and thus provide relatively "irrefutable" evidence that tools are larger than applications.

**ACADEMICIANS DON'T UNDERSTAND THE EXPONENTIAL complexity effect in large problems; industrial folk don't understand the rudiments of what academicians have done for them. Laughable public statements are made by each side about the other's turf.**

The interesting thing about this particular dilemma is that I know of no data supporting either my position or the author's. I do know of some data that supports the notion that business application systems are growing by a factor of 50 every 10 years or so, and I know that the largest projects in the history of computing (1,000–2,000 programmers were involved) were in aerospace applications, but those are nothing more than interesting clues and anecdotal evidence. A worthwhile research project here would be a software census, one that would answer some of these unanswered questions.

3. The third incident involved an academic speaker making a presentation to a local software group. In the course of pre-presentation conversation, and

again during the presentation, he made the point that software practice used totally archaic techniques and was to be scorned. He went further, and named a particularly well-known software vendor as the most egregious of the offenders. Specifically, he included in his claims the statements that this particular vendor (a) did no testing, but relied on customer beta testing for the removal of errors, and (b) used assembler language for its projects.

In the case of this incident, as opposed to the previous two, we have stepped over into the realm where factual evidence is available. There are recent books and articles in the literature describing how this particular vendor does business, and there are plenty of expatriates from the vendor who have provided anecdotal but personal testimony on these subjects as well. These are the facts: This vendor instituted quality assurance as a practice over 10 years ago. Testing is comprehensive and thorough. Beta testing does detect errors not caught by internal testing, but that simply represents the state of the practice of software, one which no amount of "breakthrough" technology will fix. Further, this vendor uses third-generation languages on an ongoing basis, resorting to assembler only when interfacing or efficiency problems force it to.

Why am I so troubled by these attacks on practice? Because they are so unnecessary and so counter-productive. They are unnecessary because both academics and practitioners are typically bright and rational people who share similar goals—they want to help build successful software products using successful approaches. They are counterproductive because they obscure the similar goals and make enemies out of people who ought to be natural allies.

I mentioned earlier that this problem has concerned me for over 25 years. Just for curiosity, I dug back through my files and found my description of the communication chasm between industry and academe, published in 1981, in an old collection of essays of mine called *Software Soliloquies*. Here is how I described the problem and its symptoms back then:

1. Jargon barriers (Parlez-vous Computerese?) In the short (then-) 25-year history of computing as a field, the industrial and academic experts have managed to find a way to be unable to talk to each other. Can you imagine a *Journal of the ACM* article in *Datamation*? Can you imagine the opposite? If you succeeded, who would really understand what was being said?

2. Physical barriers (on a clear day, you can see to the edge of the campus (plant). I've been to academic computing conferences. I've been to industrial computing conferences. The sets of attendees are all too often mutually exclusive.

3. Unrealistic understandings (Around the real-time world in 80 days). The academic picture of the industrial world (and vice versa) is both skewed and disdainful. Academicians don't understand the exponential complexity effect in large problems; industrial folk don't understand the rudiments of what academicians have done for them. Laughable public statements are made by each side about the other's turf.

4. Working the wrong problems (What have you done for me (with me) lately?). Academic people tend to assume that student problems are typical programming problems, and that the real-world can be simulated in one (or two) semester projects. Industrial people tend to reinvent the same ad hoc wheel they invented last year, and not even remove any of the flat spots.

The fact is that each faction can gain a lot from the skilled professionals of the other. A bridge is needed between the two ghettoes.

Now isn't it sad that an essay based on a 25-year-old talk published 15 years ago is still valid today? Is no one in the field listening? I have friends and colleagues in the academic world of computing. I have friends and colleagues in the industrial world of computing. I like and believe in them all, on both sides of this terrible chasm. It is time to begin building the bridges that will help us surpass it.

I would like to believe that this column, which may appear to be an attack on academe, is actually a beginning of the end of such attacks. By identifying attacks from one side of the chasm toward the other for what they are, I hope I am helping with the necessary precursors to the bridge building that eventually *must* take place.

Would you help me? C

**Robert Glass** is the publisher of the *Software Practitioner* newsletter and editor of Elsevier's *Journal of Systems and Software*. He welcomes feedback: 1416 Sare Rd., Bloomington, IN 47401.