Through a Glass, Darkly

Software – The Technology that “Don’t Get No Respect”

There’s an old saying, first said by a well-known comedian, that “.... don’t get no respect.” The dots in that ungrammatical expression could stand for whatever you believe doesn’t get as much respect as it deserves.

I would like to volunteer “software” for those dots. I think a case can be made that software-ignorant folks are disrespecting software in important ways. And that, in turn, leads to others jumping on the “software don’t get no respect” bandwagon.

Some cases in point:

1. An Australian bank’s ATM system went off the air on a busy weekend, causing their customers to be unable to access their money via the bank’s ATMs. Newspaper reports of the “disastrous outage” attributed the problem to software “coding errors.” The papers seemed to think that was a sufficient explanation; there was no analysis of what those “coding errors” might be, for example.
2. The state of Queensland in Australia suffered some devastating flooding over the past several months, and a governmental inquiry is being conducted into what the causes of the flooding were, and how similar problems might be avoided in the future. One story on the inquiry’s deliberations was headlined “Software Glitch Hit Dam Level Control. But there was a problem with the article supporting that headline. The only mention of software in the article was about a key hydraulic model which was “not available because of software problems.” There was, once again, no analysis of what the software problem was, or how it impacted the hydraulic model in question.
3. The Queensland Health governmental organization decided to develop a new payroll system, marrying SAP and a commercial system called Workbrain, obsoleting the old system. But they made a classic IT blunder – they failed to make parallel runs between the old and new systems, putting the new system on the air without adequate testing. It appears to be true that a key government official made the decision that parallel running would not be cost effective. But the end result, no matter who is at fault and how, is that – once again – software is blamed for the problem (which has become huge, with many people getting paid too little and some getting paid too much). Perhaps it is software management that should “get no respect,” but in any case this is yet another story of software failure.
4. The reservation system of an Australian airline failed several months ago, leaving tens of thousands of passengers stranded. The airline blamed the software, and in particular the software responsible for switching over to a backup system. But, when all the facts came to light, the problem was a single solid state disk failure. That is, in spite of the fact that software got the blame, the problem was in fact a hardware one!

There's one other story about software problems from the recent "down under" press, but this one is more of a humorous story than a "software don't get no respect" one:

5. A New Zealand retail store has a software control system for determining when to unlock the store's doors to allow the public to enter. Apparently the control system failed to understand how the Good Friday holiday worked in New Zealand, and opened the store's doors at the normal time – with no employees present! Several "customers" made off with free purchases before store management became aware of the problem and closed the doors.

In this case, in fact, software did cause the problem it was accused of causing, and in fact deserved the "no respect" it got!

Still, all of this adds up to a disturbing picture of software and its contributions to the world at large. Those of us in the software world, for the most part, see the contributions of software to today's world as enormous and in fact almost staggeringly successful ("software crisis" naysayers to the contrary). But that view does not tend to be shared by the public at large. And, of course, the newspapers are not helping, crying software "wolf" whether it's deserved or not, and failing to pursue what kind of software problem is at the heart of some of these failures.

Software, I would assert, deserves tons of respect. But that, unfortunately, doesn't mean that we will get it!

"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 former columnist for several periodicals including Communications of the ACM (the "Practical Programmer" column) and IEEE Software ("The Loyal Opposition"). He was for 15 years a Lecturer for the ACM, 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."