Information Systems Management, 28:184–185, 2011

ISSN: 1058-0530 print / 1934-8703 online
DOI: 10.1080/10580530.2011.562402

Taylor & Francis
Taylor & Francis Group

THROUGH A GLASS, DARKLY

On the Aging of Software

Robert L. Glass

Recently, a colleague sent me a copy of the slides he uses to teach his Systems Analysis course, and I enjoyed thumbing through them. I didn't always agree with how he presented the material he chose, or even his choices of material, but still – it made for a fascinating exercise and great mental stimulation.

However, it was when I came, near the end of his slides, to his discussion about what I would have called Software Maintenance (*) that my blood started to boil. There, on his slide on which he was discussing what happens to software as it grows older, he wrote: **software ages like eggs, not wine.**

In one sense, I wasn't surprised that he said that. In fact, as software gets older, it does often get worse. So it would be unusual for anyone to see software getting better with age, like fine wine. And I suppose we have all seen examples of older software that did have a rancid smell about it, like rotten eggs.

And there was another sense in which his proclamation is not a surprise. There is a strong belief in parts of the software world in some sort of software crisis, one in which "software is always over budget, behind schedule, and unreliable." It's a "software don't get no respect" kind of position. And the statement that made by blood boil, above, was of that ilk. Not only do we do a bad job of building software in the first place, it seems to imply, but we don't do all that well when it comes to maintaining it, either. (As you may know, I strongly disagree with all such "software crisis" thinking; but that is a subject for another time).

There is more to this matter of software aging than at first meets the eye. First of all, software in and of itself simply does not change in any way with age. It doesn't come to smell like rotten eggs; it doesn't reach the pinnacle of fine wine. It just sits there, doing absolutely nothing. And the reason for that is instructive, as we will soon see below.

There is a saying that I much prefer to the one above about eggs and wine. In this saying, the reason why software in and of itself doesn't change with age is nicely summarized. That saying is, **hardware deteriorates in the absence of maintenance. Software deteriorates only in the presence of maintenance**.

At the superficial level, this statement says what I was implying above about software not changing by itself. Hardware may get old and rusty and breakable with age, we see in this saying, but nothing whatsoever happens to software – unless you set out to change it. This thing called "maintenance" (which is, of course, very different for hardware than for software, in that for software it is more about changing functionality than in correcting problems) is in fact the only way to make software deteriorate. A ham-handed maintainer can make a mess of a formerly-successful software product. Even a good maintainer, if he or she fails to understand the original "design envelope" of a software product, may wreak havoc as they modify it. It is true, in other words and on some occasions, that software can age like an egg.

But at a more profound level, our saying about software deterioration says something really vitally important. There is nothing in software, the product of those wonderful Sciences of the Artificial that Herbert Simon wrote about, that is physical, and thus there is nothing there that will decay. And that makes software quite unique. It is that uniqueness that I think we need to emphasize to our students of Systems Analysis more than what may happen to software as it ages.

And that brings me to one last saying about software. In a sense it says the same thing as the "deteriorates" saying, except it says it in a whole new context. This new (very old!) saying is, **old hardware becomes obsolete; Old software continues to go into production every night**.

Isn't that a much more profound thing to say about software than that it ages like eggs? I suppose that I should confess here that software maintenance is a pet topic of mine.

When I first learned to program, I spent a number of important hours maintaining the work of some pretty superb programmers, one fantastic learning experience. Later, I made maintenance a specialty topic of mine, encouraging academic computing programs to include courses on the subject (mostly unsuccessfully!), writing books about it (when no one else was doing that!), and studying the research others were doing on the subject (some of that research is wonderful, and some is absolutely atrocious. To understand which I see as which, see the section on Maintenance in my book Facts and Fallacies of Software Engineering, Addison-Wesley, 2003, where I spend as much time on this subject as I do on any other subject that book covers).

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

AUTHOR BIO

Robert L. Glass (E-mail: rlglass@acm.org) is President of Computing Trends, publisher 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 95 papers on computing subjects, Editor of The Software Practitioner newsletter, and Editor Emeritus of Elsevier's Journal of Systems and Software. He was a Lecturer for the ACM for 15 years, 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."