



Presents
An IT Metrics and Productivity Journal Special Edition

Focus on Robert L. Glass
President of Computing Trends
A CAI State of the Practice Interview
November, 2005

Biography of Robert L. Glass:

Robert L. Glass is President of Computing Trends, publishers of *The Software Practitioner*. He has been active in the field of computing and software for over 45 years, largely in industry (1954–1982 and 1988–present), but also as an academic (1982–1988). He is the author of over 20 books including *Software Folklore*, *Computing Catastrophes*, *Computer Shakeout*, *Software 2020*, *Software Runaways*, *Computing Calamities*, and *Facts and Fallacies of Software Engineering*. He is Editor Emeritus of Elsevier's *Journal of Systems and Software*, and a 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."

Our interview between Robert Glass and Michael Milutis, the IT Metrics and Productivity Institute's Executive Director, was conducted in October of 2005.

◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆

CAI: Could you tell us a little bit about yourself and your early career and how you got to be where you are today?

GLASS: I was a socially inept, mentally bright young man who couldn't decide what to major in during college. I finally chose Math because I seemed to be good at it. However, when I got to graduate school, the whole computing thing was starting to unfold on the scene, and I quickly realized that this was where I wanted my future to be. That was in the 1950s. Consequently, I became what you would call a software pioneer, and my career grew and developed at about the same pace as the computing field itself.

The companies I first worked for were in aerospace, and so I was fortunate enough to

become a part of the amazing growth in that field as well. I can't imagine a career more exciting than being a pioneer in two fields simultaneously.

CAI: One frequently sees you referred to in the industry as "the premier curmudgeon of software." That's an interesting reputation. How exactly did you earn it?

GLASS: Early in my aerospace career, I worked for a company that placed more importance on the role of the team, and less on the individual. I found that frustrating, since my career up until that time had been based on my own growth and capabilities as an individual. As a result, I began seeking a "sanity outlet" where my individual capabilities would be respected. To that end, I started writing stories about project failure for Computer World under an assumed name, Miles Benson. Eventually I discarded the false name and re-published that same material under my own name. I always have been kind of a loner, not a team player, and I am more interested in being open and honest than in playing political games.

CAI: You write a lot about software maintenance and are considered by many authorities to be one of the few real gurus in this niche. Why is maintenance such an important topic and why, in your opinion, should IT and software organizations be making an effort to get this right?

GLASS: I believe there is a 60/60 rule for software maintenance. Roughly 60% of all software work is maintenance, and roughly 60% of that 60% is enhancement.

Enhancement is about problem-solving, in spite of the fact that many people seem to think maintenance is largely about fixing errors. Most of the problems we solve via software are done as maintenance enhancements, not as new development. In fact, only 17% of maintenance is about repairing flaws. It is important to think of maintenance not as a problem we wish we could make go away, but as the most important solution we offer our customers.

Given these numbers, how can software maintenance not be considered important? More importantly, why would organizations not be making an effort to get this right?

Too many people think that software maintenance, like maintenance in most other fields, is about repairing damage. The significance of the fact that enhancement dominates maintenance tasks is that (a) it makes software maintenance differ from almost all other kinds of maintenance, and (b) it means that the primary task in

maintenance is about solutions, not problems.

CAI: In your book *Facts and Fallacies of Software Engineering*, you write that "Better software engineering development leads to more maintenance, not less." Could you explain what you meant?

GLASS: The better the software engineering techniques, the quicker and the more successful the software maintenance activity will be. There is always a backlog of unsolved maintenance problems in the software field, and more of that backlog can be worked off if each individual task is done more efficiently. That means that the better we do, the more we do.

CAI: Are there any other popular misperceptions that persist around the subject of software maintenance?

GLASS: Academics tend to believe that in teaching development they teach everything a programmer needs to know about maintenance. That's why there are few, if any, academic courses in maintenance. The fact of the matter is that the maintenance life cycle is pretty much the same as the development life cycle except for one critical thing: maintenance has a phase early in the process about "understanding the existing software product." This is vastly different from, and far more complicated than, any of the tasks of development. And research data shows that "understanding the existing software product" is the most dominant phase of maintenance, both in time and in cost.

These academic misconceptions about maintenance are costing us dearly, because no one is really being educated as to what maintenance is all about.

CAI: We still see most of the best thinking and publishing in our field-about metrics, about estimation, and about processes- being done in the area of new development as opposed to maintenance. Why is this the case given the fact that 40-80% of total software costs are directed towards maintenance? Is there anything we could be doing to change the mindset around this?

GLASS: There was an old "Little Moron" joke in the 1940s in which the little moron seemed to be looking for something under a street light. He was asked, "Where did you lose the thing you're looking for?" and he replied "Over there." "Then why," he was asked, "are you looking for it here?" "Because," he replied, "the light's better here."

The light is better for software development than for software maintenance. It's more fun to do, the academics teach it, and it's a much cleaner and crisper route to feeling good about solving customer problems. Maintenance is very complex work, and it takes a special kind of person to enjoy it and be good at it.

How could we change the current status of maintenance? I'm afraid that the answer is "We can't," but along the way I would suggest that we try paying bonuses to people who do maintenance. I would also suggest that we get more academics to teach code reading before they teach code writing.

Questions? Suggestions? Comments? Please contact the IT Metrics and Productivity Journal Editor at **michael_milutis@compaid.com**