

SOFTWARE QUALITY IMPROVEMENT

While I was intrigued by the title and premise of “A Copper Bullet for Software Quality Improvement” (Michael Blaha, Feb. 2004, pp. 21-25), I have some issues with the article. Performing database evaluations on proposed commercial tools appears to be a viable method that probably should be included in any build-buy decisions.

However, I would like to have seen more meat in this article. For example, I’m interested in the algorithms for establishing the grades shown in Tables 2 and 3. I assume this was based on the impact and severity of the defects.

Another desirable piece of information that would affect the evaluation is the form in which the vendors supplied their database structure. I also would like to know exactly how long each of the evaluations took, compared to schema size and complexity.

I take issue with some of the implied definitions used in this article. As for reverse engineering, it usually begins with source code (database scripts, methods, procedures) and ends with a detailed design. The author of this article begins with a vendor-supplied “database structure” and ends with a grade. I would not call this reverse engineering or, at most, it is a minor part of reverse engineering.

The author describes software engineering as “the practice of thinking carefully before immersing yourself in the minutia of coding.” This definition trivializes the software engineering domain to the point of absurdity. It’s akin to saying that chemical engineering is thinking carefully before mixing some chemicals.

Finally, I find the title of this article misleading. Rather than addressing software quality improvement, it seems to describe improved COTS tool evaluation, which I would not elevate to a “copper bullet” status.

Hank Fly
Scottsdale, Ariz.
hank.fly@dhl.com



The author responds:

As Hank Fly acknowledges, the primary purpose of this article is to get readers to think about evaluating vendor databases to augment the benefits of software engineering. The poor quality I have seen in multimillion-dollar software packages is both amazing and outrageous. Why do customers tolerate this? Let’s at least try to make quality visible and factor it into purchasing decisions.

There are no “algorithms” for assigning the grades. Rather, I assigned the grades based on my observations according to the criteria in Table 1. Ideally, it would be better to assign the grades mechanically, but this would be difficult to do, given the wide variance in software quality, domain, and style. The grading scale in Figure 1 has worked well for me in practice. Maybe this article will stimulate the publication of data and approaches from others.

To get information on the time involved in making my evaluations, see the addendum on the Web site mentioned in the article (www.modelsoftcorp.com). Reference 4 in the addendum cites the source of the detailed data: M. Blaha, “A Retrospective on Industrial Database Reverse Engineering Projects”—Parts 1 and 2, *Proc. 8th Working Conf. Reverse Engineering*, IEEE CS Press, 2001, pp. 136-153.

The definition of reverse engineering that Fly offers is needlessly restrictive. As an alternative, I would refer readers to “Reverse Engineering and Design Recovery: A Taxonomy” (*IEEE Software*, Jan. 1990, pp. 13-17) by Elliot Chikofsky and James Cross. This sem-

inal paper provides a frequently cited definition of reverse engineering concepts and terminology that has become widely accepted.

Perhaps I could have provided a better definition of software engineering, but I certainly did not intend to demean the discipline. All too often, deep thinking is what is lacking in software development. To quote Fred Brooks (*The Mythical Man-Month*, Addison-Wesley, 1995, p. 42), “Conceptual integrity is THE most important consideration in system design” (the emphasis is from Brooks). I would equate deep thinking with conceptual integrity.

Michael Blaha
blaha@computer.org

NOT THE END OF SCIENCE?

Is John Horgan’s “The End of Science Revisited” (Jan. 2004, pp. 37-43) an elaborate spoof or is the title just provocatively misleading?

That science might be entering a phase of reduced acceleration and could, some day, come to a halt is an idea worthy of serious consideration. But I can find no evidence in Horgan’s article for either proposition. His thesis that “science itself tells us that there are limits to our knowledge” is unsupported. In 40 years as a scientist and engineer, I have never seen an argument to that effect, and Horgan certainly hasn’t provided one here.

What strange logic argues that a theoretical limit on the speed of travel and non-quantum communications leads to a conclusion that scientific discovery is reaching a boundary?

What Horgan does give us, allegedly in support of his thesis, is a noteworthy catalog of instances in which science and engineering have failed to fulfill the hopes of excitedly optimistic proponents in the timeframe they had estimated. But failure to deliver to an ambitious timetable, and even the recognition that some issues are more difficult to understand than had been predicted, does not mean the whole of science is coming to a stop.

Worse still, Horgan seems to misrepresent the very essence of science. His view of science as a list of facts tempts him to argue that at some point we will have all the facts. Science is not like that. It is rather an unstable collection of models, some cruder than others, and each subject to future refinement or replacement. The process of scientific questioning shows no sign of abating. Each new discovery typically throws up more questions than it has answered.

Not so very long ago there were those who argued that sustained flight by a heavier-than-air machine was totally impossible—that science and engineering had bumped up against a limit in that pursuit. John Horgan makes the same mistake, but on a grander scale.

*Raymond B. Muir
Melbourne, Australia
muir@i.net.au*

The author responds:

The most crushingly obvious consequence of limits on travel concerns space exploration. Our fastest spaceships, traveling 100,000 miles an hour, would take 30,000 years to reach our nearest stellar neighbor, Alpha Centauri.

In other words, barring some breakthrough that allows us to transcend Einstein's prohibition against faster-than-light travel, all our science-fiction fantasies about zipping at warp speed to other star systems and galaxies will always remain just that: fantasies. We will remain forever trapped here in our little solar system, trying to infer as much as we can about the infinite cosmos by sifting through the finite number of photons that fall our way. But we'll always have *Star Trek* reruns.

Describing science as a collection of "unstable" models is what I call a post-modern gambit. If science is incapable of achieving permanent truth, as post-modern philosophers claim, then of course science is unlikely to end.

But what separates science from philosophy and other less potent modes of knowledge is that science answers some questions beyond a reasonable

doubt. Do we lack sufficient evidence to believe in the periodic table, the genetic code, evolution by natural selection, or the recession of galaxies? What about the proposition that the earth is round and not flat?

*John Horgan
jhorgan@highlands.com*

SYSTEM ENGINEERING IN SOFTWARE CERTIFICATION

There has never been such an un-put-downable issue of *Computer* as the January 2004 edition. The entire issue contains knowledge of permanent value and is a textbook by itself. Kudos to the editorial team—keep up the tempo and spirit.

The article titled "A Tale of Three Disciplines ... and a Revolution" (Jesse H. Poore, pp. 30-36) deserves special appreciation. I have been associated with software engineering processes for mission-critical systems and with SE education for the past three decades. Hence, I found the articulation of circuit engineering and genetic engineering as worthy role models for software engineering quite appealing.

I offer to ride along with Jesse Poore's Sydney Carton bandwagon to salvage software engineering. However, I could not digest the parting shot that "Theoretically, software is the only component that can be perfect..." If this statement is meant as a motivating slogan, it is alright—but it does not pass scientific scrutiny.

*R. Narayanan
Trivandrum, India
r.narayanan@tcs.com*

Jesse Poore is on the right track with regard to certification. Each certification must include applicable systems engineering. I've spent most of my 27 years in the training simulation business. Starting my career in the early days of computing, applying systems engineering, and performing training task analysis have given me a unique view on this subject.

Since 1977, I've watched computing

science progress from ignored to nearly irrelevant. My early struggles with hardware engineers unaware of basic software engineering were replaced by struggles with software engineers delivering excessively elaborate, risky designs. It seems that clumsy, ancient Cobol has been replaced with countless cryptic object definitions characterized by Cobol-like function density. The cost/risk/schedule/performance problem donned a new mask and continued to haunt big software projects.

Systems engineering is the science of success, the rarest form of "common sense" expressed in a systematic path from an operations concept to an operational system. Unfortunately, it's above the tree line for most software engineers, and it's further shrouded in clouds by ineffective systems engineering texts.

Now let's look at the big picture. Task analysis, the art of drawing system-level performance requirements and test procedures from a mission scenario, has taught me that to implement a complex system, you absolutely must have an operations concept. What is the operations concept for software engineering? Filling this void will fix our system.

There is a pressing need to apply systems engineering methods to the problems Jesse Poore describes and to include these methods in the certification he recommends. We must define an operations concept for each software discipline and use systems engineering to solve these problems. This mitigation will make our wounded system work better.

*John F. Hubbell
Houston, Texas
jfhubbell@ieee.org*

We welcome your letters. Send them to computer@computer.org. Letters are subject to editing for style, clarity, and length.