Through a glass, Darkly

Three Obscure Things in Our Field That I Actually SUPPORT!!!!

Robert L. Glass

I'm a contrarian. What I often do in this column and my other writings is oppose something or other in the computing/information systems field. But in doing that I run the risk of being labeled an "againer." (A friend of mine was called an "againer" because whatever subject was brought up, he was against it!) So in this column I would like to do a "now for something completely different" thing.

Here, I would like to express support for some things that, while not in the midst of the field's mainstream (one might even call them "obscure"), still appear to be potential major contributors to its progress. I would like to name three such things:

1. The Impact project, and its attempt to trace the impact of computing / software engineering research on software engineering practice.
2. The recent creation of a list of the "25 most dangerous programming errors," and the further creation of a method for trying to eradicate them.
3. A (by now somewhat ancient) book on the "assessment and control of software risks."

In each case, I want to stress what I see as the positive contribution of these things, but at the same time I can't risk adding a "but..." to the end of each as I suggest ways in which they might be improved.

Impact project

There is a tenuous relationship, some would say even a chasm, between computing research and its practice. A recent study of the opinions of those attending a paired academic and practitioner conference made this quite clear (the IEEE Software editor's column "A Tale of Two Conferences," by Hakan Erdogmus, published in Jan., 2009) (for example, 51% of academics said that Software Engineering research leads software practice, whereas more practitioners (39%) saw practice actually leading research than the other way around (13%)).

The Impact project, funded by the U.S. National Science Foundation and several other professional societies, was established to try to determine in what situations research has had an impact on software progress. Recently, an article in Software Engineering Notes (Nov., 2008), "Impact of Research on Practice in the Field of Inspections, Reviews, and Walkthroughs" by a collection of well-known software engineering authors, did a nice, in-depth job, of showing how such a project could be conducted, and produced a

fascinating set of results. Similar studies are published aperiodically in the ACM journal Transactions on Software Engineering and Methodologies, including those on formal/automatable processes (in configuration management), formal product modeling practices (via programming languages), and semi-formal process practices (such as the reviews study that forms the basis for these comments, described above).

To conduct this kind of research, it is necessary to (a) determine the state of software practice in the field in question, (b) identify the research studies that may provide a basis for that state of the practice, and (c) seek links between the two. That quest is not easy. Note that, although the Impact project has been under way for a decade or more, it has so far only been able to study a few such practitioner practices, and it is not always clear how well the research studies underpin the state of the practice. And that's the "but..." I spoke of earlier in this column – I laud the Impact project for its goals and its achievements, BUT I hope to see more from it in the years to come.

The Most Dangerous Programming Errors

You may have heard of the Impact project already, since it has received occasional attention in the years it has been in existence. I suspect you have not heard of this project:

An organization called SANS set out to identify the 25 most dangerous programming errors, establishing a group of nearly 50 software experts from 35 "high profile" organizations (mostly from industry, with only a smattering from academe). The article defining what they found ("Top 25 Most Dangerous Programming Errors," published in SANS 2009 in January, and written by Bob Martin), said that the experts had "heated discussions" but fairly quickly reached agreement on the list.

Using the list, the group envisioned, one could

- . certify software as to whether it contained those errors
- . let programmers know how good their products were
- . provide material for college courses in reliable software
- . let managers know how well their programmers were doing.

The list of those 25 errors is (a) too long to present here, and (b) requires considerable explanation of some of the errors, and I suggest that you seek out the original article if you would like more detail.

So where's the "but..." in this topic? It turns out those "Most Dangerous Programming Errors" are solely from the field of software security. That is, the list contains the set of errors most likely to result in insecure software. BUT what I would like to see is a list of the most dangerous programming errors INDEPENDENT of the field of software security. Imagine using such a list to accomplish the same goals bulleted above for the security errors.

Assessment and Control of Software Risks

The field of medicine focuses on diagnosis and cure. Wouldn't it be great, said Capers Jones as he began to conceive of the book whose title is the name of this section, if the software field had something similar? And that's what he created in this book, back in 1994 (it was published by Yourdon Press).

In the book, he identifies the "10 most serious software risk factors," and discusses what can be done about them. He also does a brilliant job of identifying the different application domains in which those risks may occur, and orders the risks according to the domains (they differ enormously across domains, says Jones).

The book may be old in computing terms (1994 probably seems ancient to a fuzzy-faced programmer!), but there is little reason to doubt the continuing validity of those risks Jones identifies. The book was not a best-seller in its day, and I suspect has been little heard of since the turn of the millennium, but you would do well to dust off an ancient copy from your favorite library and have a look.

So, where, besides the age of the book, is the "but..."? You won't agree with all of Jones's risks – I certainly don't – and he grinds several of his favorite (or non-favorite) axes in creating his list, so the BUT is you have to take this work with a few grains of salt.

Robert L. Glass (rlglass@acm.org) 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 95 papers on computing subjects, Editor of The Software Practitioner newsletter, Editor Emeritus of Elsevier's Journal of Systems and Software, and is a former columnist and columnist for several periodicals including Communications of the ACM ("The Practical Programmer") and IEEE Software's "The Loyal Opposition." 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."