UK Computing: Too Much CS?

There's controversy brewing in the world of computing education.

The United Kingdom is instituting a new computing curriculum. And the controversy is about what subjects that should include.

As often happens in our field, the (British) computer science folks have grabbed the reins of this project, and the result is a curriculum biased in favour of CS-type things – coding and hardware design, for example. Other computing educators have complained, noting that there's lots more to the computing field than CS:

- digital literacy (the basic skills of the computing field)
- information technology (how to use technology to solve problems)
- computer science (how computers work, including coding)

Apparently, the current curriculum was highly influenced by the British Computer Society (BCS), which is of course a CS-type organization. Other educators have said things like "The BCS does not represent the IT industry. It represents computer science. Coding and programming is only one very small part of the IT industry."

It will be interesting to see how this controversy plays out. And, of course, this same controversy manifests itself in places like university curricula and organizations, where often CS dominates and in fact on occasion replaces Information Systems and Information Technology coursework.

Information source –

"New Computing Curriculum Still Does Not Meet Industry Needs," ComputerWorld UK, Sept. 20, 2013; Ann Nguyen

THE SOFTWARE PRACTITIONER

NOVEMBER – DECEMBER 2013 **The newsletter by and for software professionals.** **VOLUME 23, NO. 6**

Does it or doesn't it?

Size vs. Productivity: are large software projects less productive?

A frequently debated issue in the software engineering field is the relationship between project size and project productivity. In general, most people think that larger projects are more complicated than smaller ones, and therefore productivity on such projects will be lower. However, until now there has been little data available to support any such conclusion.

The good news is that there is now a report summarizing data from three organizations well-known for software project data collection. The bad news is that the data, and therefore the conclusions reached, differ!

The three organizations in question are

- the International Software Benchmarking Standards Group, ISBSG
- the Capers Jones consultancy (identified on the report as Namcook Analytics)
- Reifer Consultants LLC

In summary, both the Capers Jones and Reifer Consultants data imply the conclusion that larger projects exhibit smaller productivity. But the ISBSG data tends to imply the opposite.

There are several reasons behind the discrepancy. Although the intent of the study was to examine large projects, the ISBSG data was from smaller projects than the other two. In addition, two of the companies measured size in "function points," whereas the third (Reifer) measures size in "source lines of code" (which are then converted by an (arguable) factor into function points). (Function points are a measure best known as a way of gathering data from information systems type projects, whereas the Reifer data is most often gathered from military software projects, for which function points are not necessarily designed). And finally, some of the data was gathered for software development only type activities, whereas others of the data were gathered for software development support activities. Given all of that, it is disappointing but perhaps not surprising that no definitive conclusions could be drawn from the data.

In addition, the use of reusable components was treated differently by the companies involved. The Reifer data showed, for example, that for those projects where reuse was heavily involved, productivity improved, although perhaps still not up to the level of that for smaller projects.

In conclusion, it is possible to say here that (a) progress is beginning to be made in gathering data across data collection companies, but (b) comparing such data remains fraught with peril at this point in time.

Research Study Leads to Software Estimation Rules of Thumb

Here are some conclusions reached in a paper reporting on a "family of empirical studies ..." on software estimation [Jorgensen 2013]:

Make estimation comparisons to similar projects, and use work hours. The paper noted the stubborn tendency of software estimators to avoid comparisons with dissimilar projects, and noted at the end that "there was good reason for this reluctance." It also described the problems of using percentages as opposed to raw work hours in doing comparisons.

Attend to unique properties of the reference project (avoid the tendency to overlook dissimilarities).

Attend to estimation sequences (progress through estimating similar-sized tasks).

Avoid using small user stories as references (that tends to lead to whole-project under-estimation).

Attend to request formats (some formats tend to bias estimation responses).

Use combinations of independent estimates.

Reference:

Jorgensen 2013 – "Relative Estimation of Software Development Effort," IEEE Software, March 2013; Magne Jorgensen

Also in This Issue

"Our stock is up fifty bitcoins!"

Grand Theft Otto 6: so realistic it charges you with manslaughter.

"I was standing my ground!"

THE SOFTWARE PRACTITIONER

ISSN = 1083 - 6861
Distribution by Ebsco Publishing has been authorized

PUBLISHER:
Computing Trends,
18 View Street,
Paddington QLD 4064, Australia
61-7-33-11-12-13
email: rlglass@acm.org

EDITOR:
Robert L. Glass

ASSOCIATE EDITOR:
David N. Glass
Bill Medland

ART EDITOR:
P. Edward Presson

GRAPHICS
Graphics II, Port Matilda, PA

EDITORIAL ADVISORY BOARD:
David D. Lang
Consultant (simulation)

Steven C. McConnell
Construx Software Builders
(micros)

Donald J. Reifer
President, Reifer Consultants, Inc.
(management/large projects)

Unless otherwise stated, articles in The Software Practitioner are written by Robert L. Glass, cartoons are created by John Leatherman.

CALL BOARD

The Software Practitioner, a newsletter by and for software professionals, needs your help. This call board is our way of telling you what help we need:

Call for Papers: We especially like lessons learned, approaches tried, experiments conducted, surveys analyzed, unusual applications, controversy, humor. If it's something you'd like to read, we'd probably like to publish it. We pay for accepted articles in either subscription time (two years per published article) or advertising space (1/2 page per article), your choice.

Call for Subscribers: We need you. We hope you need us! Subscribe now, and make sure you get every issue of the Software Practitioner. The cost is REALLY low - $39/year, $29 for renewals, and $99 for institutions.

Call for Advertisers: Our readers are the people who make recommendations to the decision makers. People who want reality, not hype. If you'd like to reach that audience, we'd love to talk to you. Our rates? $99/page, $54/half-page, $29 quarter-page.

WANTED:

One or more articles for SP about the recent access failures of the Affordable Care Act system.
Contact rlglass@acm.org.

RESPONSE FORM

Please…

☐ Enter my subscription
  ☐ Individual, $39/yr.
  ☐ Institutional, $99/yr.

☐ Send me information for advertisers

☐ Consider the enclosed article for publication in SP

☐ Consider me as a reviewer on (topics)

Name _____

Address _____

_____

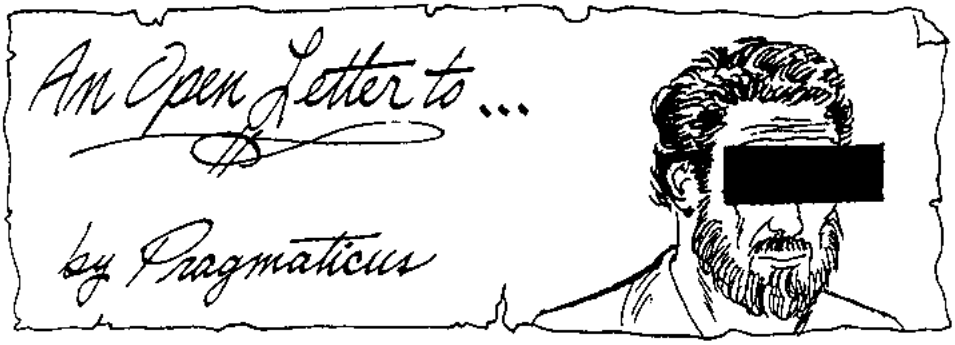
_____

E-mail _____

Shareware Subscriptions!!!

If you are an SP subscriber, get a friend to subscribe and receive a $5 rebate. Just have them put <u>your</u> name below, and send in their check with this form.

_____



An Open Letter to Whistleblowers
(and especially to Julian Assange and Edward Snowden)

I kind of like the generic idea of whistleblowers. To someone like me, an iconoclast of long standing, I like the idea of someone ratting on all those stuffed shirts who make rules that they expect the rest of us to obey, or who cheat and lie to get away with things that the rest of us wouldn't dream of doing. It's the good guys (the whistleblowers) vs. the bad guys (the stuffed shirts/liars/cheaters) all over again.

So then why is it, whistleblowers of the world, that when I'm confronted with actual, specific, living and breathing whistleblowers, I find I don't like them? Or, since this is an open letter to whistleblowers, I don't like YOU?

What's a whistleblower, you readers may be asking? Actually, readers, that's a very good question, because hiding behind the definition of the word is one reason I find myself liking the generic concept of whistleblowing, and disliking its perpetrators. Well, to quote one source of definitions of the word, in the book [Rost and Glass 2011], they offer this definition:

Whistleblowing is the act of exposing a wrongdoing in the hope of bringing it to a halt.

Now, who could possibly find fault with that? It's a noble effort, this whistleblowing thing, and I applaud all of you whistleblowers out there.

Oh, except for two of you guys: Julian Assange and Edward Snowden

Now, you readers may be wondering what it is that I dislike about these guys. Well you might ask. Many people, all over the world, find them just as admirable as I find whistleblowers in general. But here's the thing. The wrongdoing that these whistleblowers are choosing to identify I find myself doubting is really wrongdoing.

Now what they're doing, or in fact have done (they're out of action, both of them, for awhile) is expose certain pieces of secret information from American files to which they or some of their informers had access. Lots, in fact, of such secret information. Much of that secret information that they have chosen to release is embarrassing or worse to the American government. They show strategies that the Americans have engaged in, and the who/what/where of activities of the US, and all of what they have exposed has been classified as secret by a government that really didn't want that information broadcast to the wider world.

It's interesting who is choosing up sides on these matters. Those who are not fond of the US, and there are many of them across the world, are delighted by these exposures. Those who like the US but not certain of its actions are also delighted by them. Those who just like seeing the greatest country in the world (is that a fair statement? It is certainly arguable, but I think it's true) humbled, are delighted.

And then there are those who, like me, really don't like you Assanges and Snowdens of the world and what you are doing. I personally, earlier in my career, held a Secret clearance, and that was sort of like a sacred trust to me, one I would never have thought of violating. There's something almost immoral about violating such a trust, perhaps even treasonable. Those are strong words, of course but it is my viewpoint, and there are others who agree with me.

And then there's the information you are disclosing. Some of it is relatively innocent. There's a lot of international gossip in those secret files you've been releasing, and putting it out for the world to see is embarrassing, but not much more. There's some stuff about strategy and tactics also, in those secret files, and if you've ever played poker you know that you don't want certain information about your own holdings disclosed to the other players at the table (we all have secrets at times, and we don't want someone blabbing them about). And there's also some secret information whose release puts lives at risk, like the identification of agents and spies, and it is here that releasing those kinds of secret information verges on treason. And then there's something lurking behind all of this – what makes a whistleblower so certain that the information being released is, to quote our definition above, about a "wrongdoing"? It takes a certain kind of ego for a whistleblower to decide that his judgement is superior to those who originally classified the material.

But, truth to tell, this only tap-dances around the fringes of why I don't like you, Assange and Snowden. There's something strongly odiferous about the personal behaviour of each of you that makes me want to detach you from the category "whistleblower," which I generically like, and put you in my own personal "bad guy" category.

First of all, there's the matter of ethics. You say, Julian Assange, that you believe in openness and honesty. And yet your own personal behaviour, as documented in lots of different places, is one of holding your own cards close to your vest, exposing as little about yourself as you can get away with. You're a hypocrite, Julian Assange, and perhaps the worst example of your kind in the world today.

Oh, and then there's the little matter of that rape charge back in Sweden. It's ironic that many of the people who support you are in the political camp of those who strongly oppose rape. Now I realize that you see this as a trumped up charge, one designed by the evil US to force you back into their clutches. But the charge seems pretty convincing to me (and to much of the rest of the world), and of course the only way you can escape its ramifications is to subject yourself to the trial that the Swedish government and your alleged rape victims want you held to task for. I certainly don't feel sorry for you, hiding out in that tiny Ecuadorian embassy in London, avoiding taking responsibility for your actions.

There's one more thing. You ran for political office here in Australia in the last election, a couple of months back. You of course couldn't campaign here, seeing as how you have locked yourself away in London, and it is important to note that your Australian second-in-command disavowed you and your beliefs and resigned from your campaign (on the very important and relevant grounds that you were being too secretive about your actions!) as the election approached. I think most Australians saw your campaign as a curiosity at best; in any case, you were not elected. Whatever groundswell of support you may have assumed was there, it didn't materialize.

Now Edward Snowden is another matter, of course. Whereas Assange released secrets provided to him by someone else (who eventually and curiously came out as wishing to be sexually changed into a woman, and who now goes by a female name), Snowden was the one with the secret clearance and the one who did the releasing of the data to which he had access. Again, it's easy to choose up sides regarding Snowden. But what I find at least faintly suspicious is that, once he got caught like a deer in the headlights, he ran for the country that is most at odds with the US, the one most likely to love harboring a "whistleblower" from America, Russia. And, of course, that brings us to the final irony. It is likely true that Russia has more secrets, and is more protective of them, than any other country on earth. It is easy to imagine that the Russian government, while welcoming Snowden to their shores for the embarrassment it gives to the US, is pretty nervous about letting him anywhere close to its own secrets. It will, I suspect, be an uneasy (at best) relationship). Just as I can envision Assange eventually bailing out of the Ecuadorian embassy, I can easily envision either Snowden bailing out of Russia, or that country evicting him, depending on how well he can keep his "whistleblowing" habits in check over there.

There you have it, Julian Assange and Edward Snowden I don't think much of you and

your kind of whistleblowing. But I realize that many of my readers will not agree with me, some more vehemently than others. So let me make this offer. If any of you readers want to respond to this, please feel free, and I'll publish the best of the responses in some future issue of the Software Practitioner.

Reference:
Rost and Glass 2011 – The Dark Side of Software Engineering, IEEE Computer Society Press / Wiley, 2011; Johann Rost and Robert L. Glass

(The editor of this publication, who is also a co-author of this book, made me add the following:

There are copies of this book for sale, at $29+$20 postage outside Australia, from Robert L. Glass, 18 View St., Paddington QLD 4064, Australia; rlglass@acm.org)

To the Editor:

Apple's corporate value is $450 BILLION, not million.

– Larry Peters

From the Editor:

(This letter is in response to an SP article in the Sept. , 2013 issue that said "Apple is the world's most valuable company, with a market value of $450 million). I never was very good on numbers any larger than my waist size...)

Gates Is Number One (Again, Again, and Again!)

Speaking of Very Large Numbers, the Software Practitioner would like to pass on this data from Time Magazine (international edition), Sept. 30. "20 is the "number of years that Bill Gates has topped Forbes Magazine's list of the richest Americans; this year, his net worth climbed to $72 billion." How's that for a reason for sticking with the software field as a career choice?!

The Sporting World's Greatest Ever Comeback

Computing hardware and software continue to punch well above their presumed weight in the wider world. Not only have they produced the world's most valuable company (see above)and the nation's richest man (also see above), but now Oracle's come from behind win in the America's Cup competition has been called "the greatest come from behind victory in sporting history" (at one point they trailed 8-1, then came back to win 9-8).

Chaordic Organizations

Linda Rising
linda@lindarising.org • www.lindarising.org

"Geesh, Linda, not another buzzword! We're drowning in them already. We know you love to read about all the latest and greatest, but come on! What the heck is this chaordic stuff?"

Sorry, guys, I've been thinking about this for some time and I've decided to share what little insight I have because I think this is an important topic. I say, "share what little insight" because I've been looking for answers, for a nice set of rules or guidelines and I've finally realized that there aren't any. So, I might as well go ahead and maybe we can work this out together.

Here's the problem. All those agile methods are really taking the software development world by storm. The question in my mind was: I can see how agile approaches work for small teams. It's a no-brainer. But what about large projects? What about the Boeing 777? What about some of the large military projects I've seen? There is no way those could have been done with a team of less than ten—even a team of incredibly great people. Is the answer a collection of small teams? For Scrum users, would you have a Scrum of Scrums? Is there a limit to the number of teams you would have on a really big project—like the "no more than 10" limit on team size? What lessons can we apply from studying the agile approaches? I can't make it work. I keep coming back to a hierarchical, structured, top-down approach and somehow, deep down, I know that's not right. Then I started reading about complexity theory, the edge of chaos, and then, chaordic organizations. I think this is it—I just don't know what "it" is!

This article is just to share information. This is not the answer for all your problems. I don't know any software projects that are run this way. My examples are all from other domains—but when I read the other accounts, I can see how they could apply to software.

The structure of the organizational world in which we develop software is pretty old. In fact, the organizational structure of everything—church, university, corporation, nation-state—has been pretty much the same for more than 300 years. This exists in a time of rapid change. The Web has made more information available than we can handle and it is available instantaneously. As futurist, James Burke, pointed out, it took centuries for the knowledge of the smelting of ore to cross a single continent and bring about the Iron Age. When man stepped onto the moon, it was known and seen in every corner of the world 1.4 seconds later. It seems we're ready for something a little more flexible! All our institutions concentrate decision-making in the hands of relatively few people. The explosion of information chokes these decision-making systems, making them slow to respond and certainly not agile. [Hock98]

The world-shaking (in the 17th century, at least) ideas of Newton and Descartes led to a machine metaphor that is still used today. We see the entire universe and everything in it as a giant mechanism where each and every component acts on the other components in a precise way with clearly understood cause and effect. We don't realize how powerful this metaphor is and how it affects our thinking about people, who, we know are not components and do not behave in a mechanistic fashion. Our organizations are built to treat the people who work in them as cogs in a machine. We hire, reward, and fire them in the same way we have been for hundreds of years—even though the world has changed—drastically. [Hock98]

> **"This is not a metaphor. Organizations really are alive."**

Moving away from a hierarchy means more control is given to the teams and to the individuals on the teams. This is more complicated than simply saying, "OK, you guys are in charge. Go for it!" The answer seems to come from some high-powered folks in the research community who are looking at complexity theory. The basic idea is that organizations should be like biological organisms. The researchers say, "This is not a metaphor. Organizations really are alive." Let's try to understand this model to take advantage of the best way to harness the power in this living organism. [Senge+99]

Managers unconsciously follow the second law of thermodynamics, the belief that everything in the universe tends toward disorder, unless it is managed. Modern managers might be better off dropping the title of manager, following the dynamics of complexity, and discovering that natural systems tend to move toward and find their most vital form at the boundary between chaos and order. One step too much to one side or another, and like most companies in a fast-developing market, they will not survive their particular generation. Evidence from the science of complexity says that given certain clear parameters, communities or teams will become self-organizing. [Whyte94]

"Hmmm…complexity science. OK, now you've lost me. I need something to hang on to, Linda. How about some real-life stories?"

> **"...natural systems tend to move toward and find their most vital form at the boundary between chaos and order."**

OK, you're right, this complexity and chaos stuff can get out of hand in a hurry! I'll tell you the Visa story, not only because it's compelling, but it's also what got me started down this road. The Visa credit card is familiar to all of us but we might not know Dee Hock. I've never met him but I've heard he's a dynamic speaker, but that's not why people want to hear what he has to say. They listen because he has a success story to share—a powerful success story. Over 25 years ago, he had a chance to put ideas about restructuring organizations, moving beyond hierarchy and command-and-control to an organization that followed biological principles. He named this structure, "chaordic." [Waldrop96] After searching in vain for a more suitable word, it seemed simpler to Hock to make one up. Since such systems, perhaps even life itself, are believed to arise and thrive on the edge of chaos with just enough order to give them pattern, he borrowed the first syllable of each, combined them and produced—chaord (cha from "chaos" + ord from "order"). [Hock00]

The business that Dee Hock inspired—Visa—has prospered. Since 1970 it has grown by something like 10,000%. It continues to expand at roughly 20% per year. It now operates in some 200 countries worldwide and serves roughly half-a-billion clients. [Waldrop96]

The organization is highly decentralized and highly collaborative. Authority, initiative, decision-making, wealth—everything possible is pushed out to the periphery, to the members. This resulted from the need to reconcile a fundamental tension. On the one hand, the member financial institutions are fierce competitors: they—not Visa—issue credit cards, which means they are constantly going after each other's customers. Members must also cooperate with each other. Participating merchants must be able to take any Visa card issued by any bank, anywhere. Banks follow standards on issues such as card layout. They participate in a common clearinghouse operation that reconciles all accounts and makes sure merchants get paid for each purchase, the transactions are cleared between banks, and customers get billed. This blend of cooperation and competition allows the system to expand worldwide in the face of different currencies, languages, legal codes, customs, cultures, and political philosophies. No one way of doing business, dictated from headquarters, could possibly have worked. The organization had to be based on biological concepts to evolve, to invent, and to organize itself. [Waldrop96]

"The organization is highly decentralized. Authority, initiative, decision-making, wealth - everything possible is pushed out to the periphery..."

"Nice story, Linda. But Visa is a banking study and maybe one of a kind! Got anything else in the treasure chest?"

Sure! Lots more. But realize that the Visa story is more powerful than you might imagine. Our first response to almost any case study or pilot project is to say, "BUT!" This response is just human nature. We hang on to what we know. It's actually, the sane thing to do. Otherwise, we'd be blown around by every new idea that comes along. I respect your skepticism!

Here are some more stories. But remember, my goal is not to convince. My goal is to learn along with you. I'm looking for answers, not presenting the final solution to all our problems. The following story is especially interesting to me as an amateur musician. I see a lot of similarities between software developers and orchestral musicians. See what similarities and differences you can find!

Orpheus Chamber Orchestra is an orchestra with a difference: it has no conductor. The group was founded in 1972 by cellist Julian Fifer and a small group of musicians to bring democracy, personal involvement, and mutual respect into an orchestral setting. Orpheus, considered to be one of the world's great orchestras, comprises 27 permanent members—employees who cannot be fired—and a number of substitute players who fill in where necessary, a board of trustees, and administrative management. In most orchestras, the conductor not only decides what music will be played but how it will be played, with little room for opinions or suggestions from the musicians. Musicians follow the conductor's direction. Anything less invites humiliation before one's colleagues and may be grounds for immediate dismissal. As a result, orchestral musicians are notoriously unhappy employees. When a Harvard Business School professor studied job attitudes, orchestral musicians ranked below prison guards in job satisfaction. [Seifter01]

"...orchestral musicians ranked below prison guards in job satisfaction..."

Orpheus applies collaborative leadership—any member can lead a rehearsal and performance as concertmaster, or lead one of the orchestra's formal or informal teams. This system is extremely flexible—musicians freely move in and out of positions of leadership—and it can quickly adapt to changing conditions in the marketplace or within the group itself. The free flow of leadership positions within the group encourages all the members to give their best. Cellist Eric Bartlett says, "When there's an important concert, everybody feels it, and everybody does their absolute best work, giving it their utmost concentration, playing off of each other, and making sparks fly. In a conducted orchestra, you have a more passive role. You have to play extremely well, but you're not playing off your colleagues—you're playing off the person with the baton. People in regular orchestras are not emotionally involved in the same way." [Seifter01]

Members of Orpheus are energized and responsive to the needs of the organization and to the desires of its leaders. Turnover is extremely low and employee loyalty is extremely high. The result is a better product, increased customer satisfaction, and a healthier bottom line. According to double-bass player Don Palma, a founding member, "I took a year off from Orpheus and went to the Los Angeles Philharmonic. I hated it. I didn't like being told what to do, being treated like I wasn't really worth anything other than to just sit there and be a good soldier. I felt powerless to affect things, particularly when they were not going well. I felt frustrated, and there was nothing I could do to make things better. Orpheus keeps me involved. I participate in the direction the music is going to take." [Seifter01]

How about that story? Inspiring? Next time you hear an orchestra, I'll bet you'll think about the experience in a new light.

OK, one more and this one is impressive because it's the military! Who would have thought that they would be a prime example of an agile organization!In the late 1980s, the U.S. Army's senior leaders studied complexity theory and began to apply them as an alternative to their command-and-control paradigm. According to General Gordon R. Sullivan, retired Chief of Staff of the Army, "The paradox of war in the Information Age is one of managing massive amounts of information and resisting the temptation to over-control it. The competitive advantage is nullified when you try to run decisions up and down the chain of command. All platoons and tank crews have real-time information on what is going on around them, the location of the enemy, and the nature and targeting of the enemy's weapons system. Once the commander's intent is understood, decisions must be devolved to the lowest possible level to allow these frontline soldiers to exploit the opportunities that develop." [Pascale+00]

The Army has improved the quality of the recruits and their training and the electronic communications among the members of a fighting unit. Typically, military services languish during peacetime and then misapply the most recent war's doctrine to the challenges of the next. Since the Vietnam War, the Army has made technological obsolescence "the enemy." The introduction of distributed information allows the foot soldier or tank commander in the

"once the commander's intent is understood, decisions must be devolved to the lowest possible level to allow those frontline soldiers to exploit opportunities that develop..."

field to know roughly as much about what's going on as the generals in the command center. This doesn't mean that tank crews fly helicopters or that soldiers spontaneously decide to support Kurdish rebels. The Army uses the "Commander's Intent," which defines the scope of an engagement. This concept traces its origins to General Patton who said, "Never tell people how to do things. Tell them what to do and they will surprise you with their ingenuity." Combat units are encouraged to improvise and initiate within the structure of the Commander's Intent. When that intent is clearly communicated, fighting units can exploit opportunities or regroup when things don't go as planned. [Pascale+00]

Finally, for those of you who really love stories, here's a link to another great one—how GE builds jet engines. Enjoy! http://www.fastcompany.com/online/28/ge.html

Let's summarize some lessons learned from the stories. Although creating a chaordic organization does not mean following a set of guidelines or procedures these seem to be characteristic of the projects I've studied.

Distribute power and functionality to the lowest level possible. As Dee Hock says, "No function should be performed by any part of the whole that could reasonably be done by any more peripheral part, and no power should be vested in any part that might reasonably be exercised by any lesser part." [Waldrop96]

Instead of a chain of command, create a framework for dialogue, deliberation, and coordination among equals. Authority comes from the bottom up, not the top down. The U. S. federal system is designed so authority rises from the people to local, state, and federal governments. While the system appears to be hierarchical, it is not a chain of command. Instead, each level serves as a forum for members to raise common issues, debate them, and reach some kind of consensus and resolution. [Waldrop96]

Take a minimalist approach to rules. Historically, organizations have relied on rules and standard operating procedures to maintain efficiency and productivity. But as Dee Hock and others have pointed out, specifying many detailed rules intended to cover all situations can lead to behaviors that appear ridiculous to customers and other outside observers, because employees stop thinking and apply the rules arbitrarily. [Just a comment here: remember the horrible story in 1999 where 8,000 passengers were imprisoned on 30 Northwest Airlines planes for as long as 8 hours without food, water, or working toilets? This is a classic example of employees' following rules mindlessly.] In rapidly changing or globally dispersed operation environments, the benefits of efficiency (doing things one way) should give way to effectiveness (doing the right thing using many different ways, tailored to the situation). [Senge+99]

There's currently a lot of research about complexity theory and how its tenets can be applied in organizations, I hope this introduction has inspired you to learn more.

References

[Hock98] Hock, D., 1998 ODN Annual Conference Keynote: An Epidemic of Institutional Failure, "Organizational Development and the new Millennium" *Organization Development Network Annual Conference*, New Orleans, Louisiana, November 16, 1998. http://www.odnetwork.org/odn98/followup/deehock.html

[Hock00] Hock, D., "The Art of Chaordic Leadership," *Leader to Leader*, Winter 2000. http://www.pfdf.org/leaderbooks/l2l/winter2000/hock.html

[Norman99] Norman, D.A., *The Invisible Computer*, The MIT Press, 1999.

[Pascale+00] Pascale, R. T. Pascale, M. Millemann, and L. Gioja, *Surfing the Edge of Chaos*, Crown Business, 2000.

[Seifter01] Seifter, H., "The Conductor-lesszOrchestra," *Leader to Leader*, Summer 2001. http://www.pfdf.org/leaderbooks/l2l/summer2001/seifter.html

[Senge+99] Senge, P., A. Kleiner, C. Roberts, R. Ross, G. Roth, B. Smith, *The Dance of Change: The Challenges to Sustaining Momentum in Learning Organizations*, Doubleday, 1999.

[Waldrop96] Waldrop, M.M., "The Trillion-Dollar Vision of Dee Hock," *Fast Company*, November 1996, 76-86. http://www.fastcompany.com/online/05/deehock.html

[Whyte94] Whyte, D., *The Heart Aroused: Poetry and the Preservation of the Soul in Corporate America*, Currency Doubleday, 1994.

REVIEWS REVIEWS REVIEWS REVIEWS REVIEWS REVIEWS REVIEWS

Peopleware: Productive Projects and Teams, Third Edition

By Tom DeMarco and Timothy Lister
Published by Addison-Wesley and Dorset House, 2013 (previous editions published in 1987 and 1999)

Review by Robert L. Glass

Usually, I don't review subsequent editions of computing/software books. All too often, the first edition contains the essence of the material to be presented, and subsequent issues are more like warmed-over meals, worthwhile but not presenting enough that is new.

But how could I resist reviewing the newest edition of Peopleware? That's the book that I consider one of the two most important books in our field (the other is Fred Brooks' The Mythical Man-Month) (*). If the authors of this important book think they have something new and worthwhile to say, I actually wait with baited breath to see what that is!

Now, for those who don't know about even the first edition of the book (how can that be, I mumble to myself!), what the book's fundamental premise is "Our main problems [in software development] are more likely to be sociological than technological." And in this edition, the authors expand their material by visiting a collection of sociological issues that have evolved since the earlier editions came out:

In the new chapter "Let's Talk about Leadership," they point out that "Leaders make it possible for the magic to happen," note that leaders are "a catalyst, not a director," and point out that "Leadership as a service always operates without official permission," and goes on to note that this is how innovation most often comes about.

In "Childhood's End," which addresses the generational change in software folk, the authors speak of the tendency of today's young to engage in "continuous partial attention " – timesharing themselves over various tasks – and note that this gets in the way of "flow," which is about concentrating on the task at hand.

In "Human Capital," they note that "Companies with knowledge workers have to realize that it is their investment in human capital that matters most," and go on to explain how so often that investment is frittered away.

In "Teamicide Revisited," they note ways that team productivity can be inadvertently diminished, saying "Extended overtime is a productivity-reduction technique..."

In "Competition," they say that "Competition [within a group] is certain to inhibit team jell."

In "Dancing with Risk," they note that management says things like "This work is so important that we need to have it finished by Jan. 1" when what they really mean is "This work is so unimportant that we don't want to extend if beyond Jan. 1."

In "Meetings, Monologues, and Conversations," they note that meetings are all too often about "competitive windbagging," and go on to say that "A meeting that is ended by the clock [is not a working meeting], it's a ceremony."

And finally, in E(vil) Mail, they decry the tendency to cc everyone on an email, saying that some sort of "need to know" test should be applied before adding a cc to the mailing list.

Rereading this book was a wonderful and nostalgic experience, like meeting an old friend after 26 years! If you don't know Peopleware, correct that flaw immediately. If you do, I can still recommend adding this third edition to your library.

* And for a guy who has written more than a couple of dozen software-related books, that's a confession that's painful to make!

> **"Our main problems [in software development] are more likely to be sociological than technological"**

Software Requirements (Third Edition)

By Karl Wiegers and Joy Beatty
Published by Microsoft Press, 2013

Review by Robert L. Glass

Subsequent editions of computing books trouble me. I'm never quite sure whether I ought to be reviewing what's new about the book, or the whole book, original parts and new parts together. And, to make that issue more complicated, I'm never quite sure what it is that's new about the book, so I'm normally condemned to reviewing the whole thing. And, to be honest, I suppose I'm a little bit jealous. None of MY books has ever made it to a third edition!

But that dilemma is eased somewhat with this book. It starts right out by explaining what's new in its subject field since the previous edition. That's slightly different from what's new about this edition, but I think one can assume that there's a correlation between what the authors see as new about their field, and what they've done to the previous edition to bring it up to date.

OK, so what's new about the requirements field that prompted the authors to produce a third edition?

- The field has become a professional discipline, with certification and support organizations
- Requirements support tools are maturing
- Increasingly, agile approaches impact everything about the computing field, requirements especially included
- There is increasing use of visual models

I don't know how many pages were in the previous editions, but this one is HUGE! Getting up toward 700 pages, enough that just holding the book up to read and review it becomes a chore! Still, that's a good thing, right? You'd rather such a book contains too much information than too little.

And what's my bottom line here? Two things – I have to admit that I personally know and like the lead author of this book, and I would be embarrassed if I had to say nasty things about it (although I have been frequently known to do just that!) But fortunately, I don't have anything nasty to say – I in fact like the way the book is organized, and I like what it contains.

Some particularly likeable things:

Each new topic begins with a realistic and often contentious conversation between two or more principals on a relevant project. It's a comfortable way to be introduced to a topic.

The book focuses not just on relevant conversations, but relevant projects / case studies. It has a feeling of realism.

It feels astonishingly thorough. Here are just some of the topics it covers – use cases, business rules, requirements specifications (note – that is not your standard Computer Science formal specs discussion), representation techniques, quality requirements, prototyping, prioritizing, validation, requirements reuse (that's a fascinating concept all by itself!), and requirements management.

It presents a bill of rights, and a bill of responsibilities, for the customers that requirements analysts deal with.

Rights:

Business Analysts (BAs) should learn the customer's language
BAs should learn about the business itself and its objectives
BAs should record the requirements appropriately
BAs should explain their practice and the expected deliverables
You the customer have a right to change your requirements
There must be mutual respect
BAs must be prepared to listen to what's wrong with the current solution
BAs must be prepared to provide increased ease of use vs. the current approach
BAs should be prepared to suggest reusable approaches
BAs must provide a new system that meets needs and expectations

Responsibilities:

Customers must educate BAs as needed
Must provide sufficient time to provide/clarify requirements
Must be specific and precise
Make timely decisions
Respect developers' assessments
Set realistic priorities
Review work products
Establish acceptance criteria
Promptly communicate changes
Respect the requirements development process

It provides/describes a list of 50 requirements engineering good practices, and imbeds them in a process framework within which they can be applied.

It discusses approaches that can be used for six different kinds of projects – agile, enhancement, packaged, outsourced, business process automation, business analytics, and embedded real-time (that's quite a comprehensive spectrum of possible projects!)

There are lots of appropriate warnings about the result of not following good requirements practices – resulting rework can consume 30-50% of project cost, and errors in requirements drive 70-85% of rework cost.

This book says it is focused on "principles that work in practice," and I am happy to say that I believe it has accomplished precisely that. For example, wouldn't you love it if every project on which you worked applied those rights and responsibilities provided above?!

Professional Wordpress Plugin Development

By Brad Williams, Ozh Richard, and Justin Tadlock
Published by Wrox,
an Imprint of Wiley, 2011

Review by Johann Rost

I read a number of enthusiastic reviews of this book before I bought it. And many things that the other reviewers said are true. It is a good book. And it helped me more than the money I paid for it. However, the book made me angry because it is done carelessly.

For example, the connection to Twitter in Chapter 9: This code does not work any more. Version 1.0 of Twitter's API is deactivated. The book was printed quite a while ago and the Twitter API changed after the book was already printed. But I downloaded the sample code from the Website - and it was not updated. Well, I can check out what is new in Twitter API 1.1. But I can as well download something for free which works immediately. If I pay money I expect to get something that is not worse than something that I can get for free.

The Twitter Plugin has more problems: I did not understand how this plugin should be used in a Wordpress blog. Finally, I found my own solution. To do this, I had to modify the authors' code. And I still don't know how they thought it should be done.

Another example is the CRON plugin of chapter 13. The plugin works. However, it took me a while to find out that I have to click on the CRON menu item in the admin menu. This menu item does not show anything on the screen, but this is what makes the plugin finally work. Perhaps I was expected to know this - but I did not. Note that debugging CRON is a bit tricky: If it does not work immediately there is little advice regarding what is going wrong.

Last but not least: Usage of $() vs jQuery(): In many other sources, I read that we should use jQuery() instead of $() in the context of Wordpress plugins. The authors use $() - for example on page 335. Is this a typo or do the authors disagree with the mainstream? If it is a typo they should fix it; if they disagree with the mainstream they should say why.

Still it is a good book and I would buy it again. I give it a bad review because I feel that authors and publishers of a bestselling book get enough money that they should apply due care to provide something that is clearly better than what we can download for free. After all: Most of the information in this book is freely available on wordpress.com or stackoverflow - completely up to date and free of any typos.

Perhaps I should not be angry. The experience might highlight a more general phenomenon: Twenty years ago we would have said "It is a great book". No "however", no "despite..." - nothing, but great. Now we compare it to the free content which is great as well and which we can readily download. This raises the question if we should buy books on programming technology any more?

The Laws of Software Process: A New Model for the Production and Management of Software

By Phillip G. Armour (*)
Published by Auerbach Publications, 2004

Review by Robert L. Glass

This book may be quite profound. But note the "may be" – what that should tell you is that I don't understand this book very well. Note also the publication year. If the book truly were profound, you'd think, 10 years later, that someone would have found that out, and its fame would have grown enormously!

OK, so what's the book about? "We can, and should, rigorously define process for those aspects of our work that we can define rigorously. However, we cannot rigorously define process for those aspects of work that are discovery-based." And, perhaps to clarify the murkiness of that statement, the author adds his "Usefulness Dilemma" – "software process is only really useful for things that we don't want to do anyway." And, if that doesn't leave you confused enough, here's the capper – software is not a product, the author says, it is a medium; and the product is the knowledge contained in the software. And then, since the book is about process (and perhaps a reaction against the Software Engineering Institute's CMM Process Initiative, which was at the time of this publication becoming a force to be reckoned with in the software field), he concludes "all attempts to define software process are wrong, because the basic premise is wrong."

Given the title of the book, the author presents his own Laws of Software Process:

1. Process only allows us to do things we already know how to do.
2. We can define software process at two levels – too vague, and too confining.
3. The very last type of knowledge to be considered as a candidate for implementation into an executable software system is the knowledge of how to implement knowledge into an executable software system.

(I think it was somewhere around this third law that I began thinking this book was either profound, or something else entirely!)

The book also toys around with what it calls the "Five Orders of Ignorance:"

0. I know something.
1. I don't know something, but I know that I don't know it.
2. I don't know that I don't know something.
3. I don't have an efficient way of finding out that I don't know that I don't know something.
4. I don't know about these five orders of ignorance. (The author tells me that he added this order of ignorance to the list "because I thought it was cute, and it highlights the intensely recursive matter of knowledge.")

I do indeed find these profound. But I'm not at all sure how they work their way into the Laws of Software Process, even though the author seems to think he has done so.

The book does have some clear, and quite amusing, pithy moments:

"What all developers really want is a rigorous, iron-clad, hide-bound, universal, absolute, total, definitive, and complete set of process rules that they can break."

"Adults learn primarily from failure."

Given all his thoughts on ignorance, process, and everything else in the software field, he offers – toward the end of the book – his thoughts on the future of the field. He sees

- the demise of software engineering, to be replaced by the fields of Knowledge Engineering and Domain Engineering (because his data shows that the role of, especially, domain knowledge is growing much more rapidly than the role of software construction knowledge)
- the demise of code and coding languages, in favour of domain specific languages and packages.
- these roles in the software field – organizational resource coordinator, process engineer, ontologist, model linguist, methodologist, domain engineer, user/customer representative, repository engineer, anthropologist, tester, system test representative, learning systems expert.

The book concludes with a collection of typical mornings for the participants in a fictional future software project.

I suppose, when I finished the book, I felt disturbed by its unusual thoughts. But I couldn't see what I wanted to do about it, and I suspect that same is true of everyone else who has read the book since its publication nearly a decade ago. Here's a thought – read the book yourself and let me know what you think!

(I submitted this review to the author before publishing it here in the Software Practitioner, and he added some thoughts, including this one on a further elaboration of the orders of ignorance:

"Later in the book, I tackled the issue of intent: what if I don't know, and I don't care to find out? Or what if I don't know and I actively resist finding out? [Too many people] don't know the things they don't know, and are manfully resisting any impulse to (a) recognize that and (b) do something about it. It is ignorance, but it is far from blissful.)

* - A most unusual thing about this look is that the publisher (Auerbach) misspelled the author's name, using only one letter l in Phillip. Because the author told me about the problem, I have intentionally spelled it here consistently with how the author wants it spelled. Throughout the book itself, his name is spelled (erroneously) Philip.

Mars Software: Some Curious Statements

How big is BIG?

In the paper [Holzmann 2013], the author describes the software system for the most recent Mars spacecraft, and notes two apparently contradictory things:

"The software was written by a relatively small team of about 35 developers..."

Each new [Mars] mission "uses more control software than all missions before it combined."

Things to consider: is a team of 35 developers considered "small" these days? If this software is larger than all previous missions combined, how could it be developed using a "small team"?

Reference:

Holzmann 2013 – "Landing a Spacecraft on Mars," IEEE Spoftw3are, March 2013; Gerard Holzmann

Engineers and Management

Gary Stringham,
Gary Stringham & Associates, LLC

In the July, 2013, issue of IEEE's Computer magazine, David Alan Grier discussed in his article, "Short –Term Loan" (pg 112) how engineers moved into management after five years. I have seen cases where engineers feel pressure to move into management but am concerned of the side effects. I was an engineer with Hewlett-Packard for 21 years, which gave me lots of exposure to engineers and managers, in particular, engineers who move into management.

We joke how the stereotypical engineer does not like meetings for the sake of meetings. Years ago before one such monthly meeting, my manager, Spence, commented how he didn't like these monthly meetings, how they were a waste of time for engineers.

They were section meetings lead by Greg, the section manager, who was Spence's manager. All the teams in that section (including engineers like me) were expected to be there. All team managers had to show a slide (this is before the PowerPoint days) showing the status and activities for their respective teams. I observed something interesting during the meeting. Greg had a smile and a contented look on his face. He was "managing"! All of his managers were giving their respective reports in some pre-determined order covering some pre-defined list of topics. You could tell he thought he was doing an excellent job of "managing."

While I didn't think the meeting was of much benefit for me, Greg, however, was very successful as a manager. In the subsequent years, he moved up the HP management chain in various locations, left them after 30 years and was a COO at a small firm with an international presence. Greg was definitely management material.

I don't know what Spence did in the following years but while he was my manager, he was very influential in helping me in my job, though I did not realize that until years later. His team was manufacturing engineering tasked with supporting the manufacturing line including getting set up for new printed circuit board assemblies as designed by the lab. I had one of those boards. I had to get the artwork to the blank board manufacturer, to the solder mask maker, and to the bed-of-nails test developer. I had to get the list of parts to the materials team and to finance. I had to organize meetings, find a room, send out nag-o-grams, and write up meeting minutes. I was doing managerial tasks. My ranking went down; I was not cut out to be a manager. Spence could see that and tried to encourage and help me. I found another position I wanted that was a better fit and he put in a good word for me, in spite of my lower ranking.

Several years later I was toying with the idea of moving into management. At a section meeting, the section manager, Tracy, announced that one of the managers had been promoted to section manager and there was now an opening for a manager, creating an opportunity for any interested engineer to move into management. On the way back to our desks, I asked my manager, Phil, what it was like being a manager. He said, "The first thing they do is take away your compiler." He hit me right to the core. He knew me so well that he knew I would not be happy if I wasn't writing code. I never looked at management again.

I had reflected on my many years at HP. When I was designing and coding, I was happy and successful. My ranking improved. But when I had managerial assignments, I was not as happy and my ranking went down. Why should I move into management and compete against the likes of Greg and Tracy who were good at it, who enjoyed it, who had the natural abilities to manage? Their rankings went up easily; I would have had to work very hard at something I didn't like just to maintain a low ranking. Though managers' salaries were higher than engineers', a low-ranking manager would not be making more money than a high-ranking engineer. And it would not be fun for me. So I never looked at management again.

" Why force a good engineer to become a bad manager? What benefit is there (besides social status and acceptance?) "

I stayed in engineering and was successful. I had a part in producing some very important products. I designed tools, techniques, and concepts that are still being used today. I have 12 US patents to my name. And I am known among my peers as a good engineer and my ranking was high.

I succeeded because I stayed in engineering. That worked because I was within HP and in the United States. I have, however, observed that the culture in other countries, such as Mexico and India, is such that one is a failure if one is not moving up the management chain. I think that is a mistake. Why force a good engineer to become a bad manager? What benefit is there (besides social status and acceptance?)

This creates a bad situation for trying to make a quality product. Instead of having teams with experienced (and novice) engineers led by good managers, you now have teams with novice engineers led by bad managers. In presentations that I have given in the US and India, I have stated that in order to put out a quality product, you have to have a fairly stable team of engineers that have been through several product cycles. The pressure and culture found outside the US to move into management is hampering their ability to produce quality products.

I suspect that the better products, the new inventions, and the major advancements in technology come primarily from engineers who stayed as engineers past their first five years. It is my hope that companies will resist the culture pressure and encourage and reward engineers with the interest and skill to stay in engineering and not be forced into management.

Note from the editor: This article was created in response to an email request from me. Gary had sent a letter to IEEE Computer in which he talked about the decision he had made to avoid moving into management during his professional career in industry.

I was intrigued by his letter, because I had made essentially the same decision in my career. The expectation, as it is in most of industry, I think, was that after a certain period of time I would move into management. But I loved the technology of software; I loved the feeling that I could produce products with my mind (I was no good at doing that with my hands!); I loved the feeling of getting a piece of software to work and to solve the problem it was designed to address; I loved the feeling of finding and eliminating that "last bug" in a product! Did I want to give up all of that pleasure in exchange for – I wasn't sure what, because I suspected, because of who I was, that I would really not be good management material. So I stayed in the technical end of the field until my retirement years.

Was that a good decision? Mostly, I think so, and in fact I wouldn't have wanted to pursue my career in any other way. Did I pay a price? Definitely, yes. Even in the enlightened aerospace industry world where I was working at the time, the salary ladder for technologists was far lower than for managers. I made a decent living, I am pleased to say, but my income would have undoubtedly been higher if I had gone into management. And there was another problem, more subtle but in the end more real and more troubling. As a highly-paid technologist, expectations for my performance were higher. On one project where I was acceptable but not stellar in my contributions, I was the first to be removed from the project when cost-cutting became necessary (I had never been "low man on the totem pole" on any previous project, and it hurt). When I went to change jobs to another company, those companies were reluctant to hire someone whose salary history was as high as mine. You would think, of course, that at that point I could have happily chosen to accept a reduced salary from a new company, and in fact I would have. But companies are reluctant to hire anyone while reducing their salary; their fear is that as soon as something paying more like what you are accustomed to comes along, you would jump ship, and therefore you are an unacceptable risk if they are looking for someone long-term.

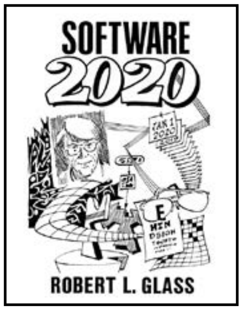
I thank Gary Stringham for allowing me to revisit this particularly important issue; and I hope all of this will be useful to anyone else who is encountering the same dilemma!

What does Computing Trends produce?

BOOKS AND NEWSLETTERS

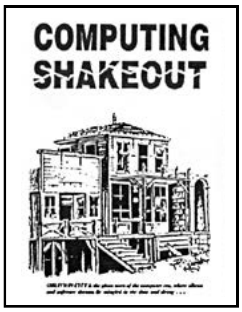
SOFTWARE 2020 is a contrarian view of the future of software development as seen with 20/20 hindsight from the year 2020. This is not your average "gee how great it's going to be" view, in which today's research becomes tomorrow's state of the practice. Rather, it's a pragmatist's view of realistic possibilities.

$9/copy



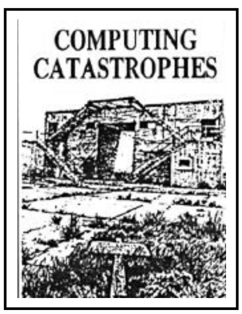
COMPUTING SHAKEOUT, real "what happened and why" stories about some well-known microcomputer companies that failed. What happened to the pioneers, like MITS and IMSAI? Why did Texas Instruments get out of the business? These questions and many more are answered here.

$9/copy



COMPUTING CATASTROPHES, more real stories, about the failures of some mainframe companies. The computing demise of RCA, GE, Xerox, and others was astonishing. Why was there so much failure in the midst of so much success?

$9/copy



SOFTWARE FOLKLORE, stories about some of the very special people in computing - "the intentionally strange boss," the "software thief," "the computer that never computes," and "every programmer's dream." Read about real computing people who've done some pretty weird and wonderful real things.

$9/copy

THE UNIVERSAL ELIXIR, AND OTHER COMPUTING PROJECTS WHICH FAILED, fictionalized tales about real failed projects. Software Practice and Experience called it "Compulsive and essential reading…the perfect Programmer's Bedside Book." ACM Computing Reviews said "Read it! Remember it the next time some wild new project is organized."

$9/copy

THE SOFTWARE PRACTITIONER

The Software Practitioner (SP) is a newsletter written by and for people who build software for a living. It is not written by journalists who know too little about software – or theorists who know too little about practice. We publish material straight from the real world:

- "best of practice" methods
- lessons learned using new technologies
- (often contrarian) views of the scalability of theoretical approaches like formal methods, object-orientation, and radical practical approaches like Agile and Open Source


THE SOFTWARE PRACTITIONER
Ig Nobel?
Boondoggles in Computing-Land
Outsourcer Satyam In Major Financial Trouble
Also in This Issue

You'll find SP a refreshing dose of honesty and reality in an all-too hype-filled software world.

$39/year individual subscription,
$99/year institutional (such as a library). **Bimonthly.**

ORDER FORM

Books: (all by Robert L. Glass)
only a few copies left of each title

- ☐ Software 2020 ....$ 9.00
- ☐ Computing Shakeout ....$ 9.00
- ☐ Computing Catastrophes ....$ 9.00
- ☐ Software Folklore ....$ 9.00
- ☐ The Universal Elixir ....$ 9.00
- ☐ Facts and Fallacies of Software Engineering ....$ 29.00
- ☐ An ISO 9000 Approach to Building Quality Software ....$ 19.00

Subscription to The Software Practitioner

- ☐ Individual .... $ 39/year
- ☐ Institutional .... $ 99/year

TOTAL ORDER $________
Add $5.00 shipping/handling (books only) $ 5.00
Amount enclosed $________

Checks may be in U.S. or Australian dollars, payable to
Robert L. Glass,
18 View Street, Paddington QLD 4064, Australia

Send to:
Name ____________________
Address ____________________
____________________ Email address ____________________
City ____________ State ______ Zip ______