Editor in Chief: **Forrest Shull**
Fraunhofer Center for Experimental Software Engineering, fshull@computer.org

Progression, Regression, or Stasis?

Forrest Shull

HOW OPTIMISTIC SHOULD we be about the profession of software engineering and our current ability to credibly deliver quality systems within expected cost and schedule?

In this issue, you'll find a selection of articles that cover recent research on software quality assurance, all of which go beyond the "traditional" verification and validation (V&V) toolkit to look at recent advances focused squarely on the needs of contemporary projects and important application domains. These articles look at quality in the context of today's systems and their reliance on mobility, sophisticated GUIs, privacy and other data quality needs, agility, and so on. While compiling this issue, I wondered about the big picture: Are technologies like these allowing us to progress as a profession in terms of being able to deliver better systems? Or are we losing the arms race with stakeholder expectations for more ubiquitous and sophisticated functionality?

> Does the evidence suggest that on balance our profession has progressed, regressed, or stayed in stasis?

I was thinking about this question largely because of a keynote I attended earlier this year at the IEEE Software Technology Conference (STC) titled "The Beginning vs. Today," given by Lloyd Mosemann, a former senior vice president of science at SAIC, as well as a deputy assistant secretary of the US Air Force. Mosemann had been instrumental in creating STC 25 years earlier as a response to the recognition of the "software crisis" at that time. Now, he asked, does the evidence suggest that on balance our profession has progressed, regressed, or stayed in stasis?

His answer to that question, based on several reports and analyses, was quite pessimistic in terms of whether we can claim much progress. Appropriately for a keynote, he used this as a launching pad to encourage the audience to engage in reflection and a call to action to improve the rigor of engineering within our field.

Is Government the Problem?

Since attending this keynote last May, I've periodically been thinking about my own answer to this question. Being optimistic by nature and fascinated by all things software, I lean toward a less harsh assessment and wondered how to justify such an answer. Unfortunately, however, it's hard to retain much optimism when being bombarded daily (as I write this article in mid-November, 2013) with news about another high-profile software failure story, the disastrous rollout of the US Health Insurance Marketplace through the healthcare.gov website.

For those readers who aren't US citizens, or who have been allowed to escape the ubiquitous news stories for

IEEE Software Mission Statement: To be the best source of reliable, useful, peer-reviewed information for leading software practitioners—the developers and managers who want to keep up with rapid technology change.

 0740-7459/14/$31.00 © 2014 IEEE

some other reason, the Health Insurance Marketplace is the technical centerpiece of President Obama's hard-won healthcare reform initiative. Because the reform introduces, for the first time, penalties on US citizens who don't have health coverage, the Marketplace was designed to assist people in finding coverage if they need it or want to improve their current coverage. The online Marketplace is intended to let users across the US find insurance plans according to their geographic area and then do the kind of comparison shopping that has previously been difficult in health insurance owing to the many different types of coverage and different pricing options. This vision entails a high level of technical complexity because the back-end of the site has to communicate with the numerous other systems maintained by healthcare insurers, none of which were ever designed to work with such a site.

Unfortunately, since its rollout, the Marketplace has been plagued with poor responsiveness and site errors that have made it impossible for many people to use the functionality. Currently, it's acknowledged that the site can malfunction when experiencing more than 20,000 concurrent users—less than half of the planned capacity.[1] Also, security flaws have been found and fixed, but not until after the site went live.[2]

Are such results inevitably the result of government acquisition practices? I would argue not, but President Obama has perhaps added fuel to the fire with his statement that information technology procurement is not one of the things that the federal government does well.[3] And there are other software issues, especially within the US Department of Defense, which one could point to as problematic; numerous articles point to budget and quality problems in acquiring systems, many of which are software-intensive, for the DoD.[4] Although we can look at this as a system engineering problem rather than a software issue, Mosemann appropriately pointed out in his keynote the important role that software plays in those systems: "Software is on or impacting the critical path in almost 25 percent of major systems in [the Defense Department's] ~$1.5 trillion acquisition portfolio." And his analysis of the status of software engineering in these systems is one that even I have a hard time being optimistic about.

From these examples, some will conclude that the federal government is the problem. But like many sweeping conclusions, that strikes me as a bad one. It leaves out the successes, many of which generate less publicity—for example, it ignores the fact that US government code got *Curiosity* safely to the surface of Mars and continues running well there to date.[5] It's also worth mentioning that the government's watchdog agency, the Government and Accountability Office, actually cites progress this year by NASA in improving cost and schedule performance on recent projects,[6] and NASA is nothing if not a high-reliability, mission-critical environment.

Is Limited Competition the Problem?

The online Health Insurance Marketplace is a high-profile software development failure with significant effects for citizens. Sadly, it's hardly the only software industry problem with such effects, even if the others are less visible (so far).

The state of practice on moving to electronic health/medical records

IEEE Software

STAFF

Lead Editor
Brian Brannon
bbrannon@computer.org

Content Editor
Camber Agrelius

Manager, Editorial Services
Jenny Stout

Publications Coordinator
software@computer.org

Production and Design Editor
Jennie Zhu-Mai

Webmaster
Brandi Ortega

Illustrators
Robert Stack and Alex Torres

Cover Artist
Peter Bollinger

Director, Products & Services
Evan Butterfield

Senior Manager, Editorial Services
Robin Baldwin

Senior Business Development Manager
Sandra Brown

Membership Development Manager
Cecelia Huffman

Senior Advertising Coordinator
Marian Anderson
manderson@computer.org

Editorial: All submissions are subject to editing for clarity, style, and space. Unless otherwise stated, bylined articles and departments, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *IEEE Software* does not necessarily constitute endorsement by IEEE or the IEEE Computer Society.

To Submit: Access the IEEE Computer Society's Web-based system, ScholarOne, at http://mc.manuscriptcentral.com/sw-cs. Be sure to select the right manuscript type when submitting. Articles must be original and not exceed 4,700 words including figures and tables, which count for 200 words each.

IEEE prohibits discrimination, harassment and bullying: For more information, visit www.ieee.org/web/aboutus/whatis/policies/p9-26.html.

WELCOME TO NEW BOARD MEMBERS

I'm very happy to welcome several new members to our editorial and advisory boards. I rely on our energetic board members for the work they do on our special projects that keep the magazine active and visible in so many different domains.

Jeromy Carriere is engineering director at Google, where he manages a group of teams working on core infrastructure. Carriere has held prior positions at eBay, where he was the technical lead for design and development of an open commerce platform, as well as Yahoo!, Fidelity Investments, Microsoft, America Online, the SEI, and Nortel, among others.

Davide Falessi, a research scientist at the Fraunhofer Center for Experimental Software Engineering, joins the board as the department editor specializing in multimedia. This is a special position for *IEEE Software* because the multimedia editor works across our existing departments and special issues and covers multiple topic areas. Falessi's research work is in the area of empirical studies, which has allowed him to collaborate with researchers in many different technical areas, an important asset for the cross-cutting work he will do for us.

Evelyn Tian is a lean and agile coach lead with Ericsson in northeast Asia. She joins our advisory board in part to help us with outreach efforts in China. Tian is an experienced system and software designer with a proven track record in the telecom industry and is known for her experience with lean and agile coaching at individual, team, management, and organization levels.

Grigori Melnik, a principal program manager with Microsoft, has switched roles, leaving the advisory board to take a new role as our associate editor covering submissions in the area of agile practices. His career has spanned a number of roles (including researcher, software engineer, and educator) with extensive experience in the fields of software engineering, applied mathematics, and economics. Melnik's career has included multiple leadership roles at start-ups, enterprises, and government. Especially important to us, he's an internationally recognized expert and speaker on agile methods, test-driven development, and test automation, as well as the author of five books and numerous scientific publications.

I'm looking forward to working with all of our new members.

in the US has recently been the subject of increasing discussion as well, with much agreement that progress should have been better and that corrective action is needed. Many of the problems stem from the large number of competing commercial solutions on the market. In this case, government defined the problem and looked to the private sector for solutions, resulting in a large number of commercial software products that don't communicate with one another and have incompatible interfaces. Although they're being adopted, these competing products have led to unexpected costs (because medical professionals have to be trained on every system in use at the organizations where they work), unexpected time pressures (because data entry and dealing with warnings and recommendations from poorly designed software takes time away from patients), and possible user error (resulting from confusion when the manner of entering data changes radically from one system to another).[7]

When pointing out the problems in the development or acquisition of federal software-intensive systems, critics will sometimes explain that they're really blaming the lack of free market competition. Although the original contract is completed, the winner tends to be locked in for the life of the system, barring some kind of truly disastrous development. But the issues surrounding the development of electronic medical records systems are a definitive counterexample, with many of the worst outcomes stemming plausibly from too much competition and too little oversight. Recent and proposed fixes tend to focus on more government oversight related to improving quality, compatibility, and a better understanding of the needs of the special-

ized medical professionals who will be the users.

Critiques related to a lack of competition often seem to mean, at heart, that the government has tried to develop a software solution that would have been better left to Silicon Valley, where companies are implicitly understood to be agile and innovative. I have a hard time buying this argument as well. In the case of healthcare.gov, for example, I find it difficult to understand how agile practices per se could have addressed the problem of interoperability with literally thousands of legacy insurance systems for inclusion in the Marketplace. Moreover, while many commercial software-intensive products have excellent reliability today, there doesn't seem to be any magical capability on the part of agile software development teams to accurately understand users' needs; if there were, Yahoo! and Windows 8 would presumably be in stronger positions.

Technical vs. Managerial

Also part of the discussion of almost every software-related failure is the issue of whether the root cause was a failure of management or a technical problem. This has a direct bearing on how people answer the question of progression, regression, or stasis, because it tends to reflect something fundamental to one's view of the profession: Are the methods of producing quality software already well understood with the challenge being just providing the needed time and effort for teams to do it right? Or are we still truly in need of increasingly sophisticated technical solutions that work at the scale and complexity of today's systems?

In the case of healthcare.gov, for example, there have been two major trains of thought on the cause of the problem. Some articles, such as *Slate*'s accessible series aimed at explaining the cause to general readers, emphasize the complexity of interfacing with such large numbers of legacy systems.[8] Other authors, such as Johanna Rothmann in her blog, have emphasized more the managerial aspects, especially the need to create a culture that can produce effective results.[9] And let's not forget the risks inherent in contractual issues, as I discussed in my last column[10]; almost inevitably, the lead contractor for the development of the online Health Insurance Marketplace was contracted using a cost-plus vehicle that often puts the financial risks for underperformance onto the government.[11]

Let's Call the Whole Thing Off

Not surprisingly, trying to diagnose the health of the domain from a few anecdotes is a dangerous undertaking. The field is so broad that there are sufficient examples to be found for both optimism and pessimism. In the end, I'm not sure that the question is useful, and I'm convinced at least that there are no easy answers regarding goodness or badness caused by top-down decision making, the degree of competition, the degree of government involvement, or other similar factors.

What is useful is the individual soul-searching and introspection that result from high-profile failures in our field. We certainly have no lack of best practices and lessons learned, but for many of us, these high-profile

Erratum

Regarding the November/December 2013 issue of *IEEE Software*, we regret posting an incomplete biography for Rich Hilliard, one of the guest editors. His biography should have read as follows:

Rich Hilliard is a freelance software systems architect and softvware engineer. He's also editor of ISO/IEC/IEEE 42010, *Systems and software engineering—Architecture description* (the internationalization of the widely used IEEE Std. 1471:2000). Hilliard is a member of IFIP WG 2.10, Software Architecture, the IEEE Computer Society, and the Free Software Foundation, and is an officer of the League for Programming Freedom. Contact him at r.hilliard@computer.org.

disasters that show up in the popular media can be some of the few times that we attach significant meaning to those lessons and pay significant attention to those nagging worries. Do we feel that similar criticisms could be attached to our own projects, if only people were paying attention, or can we feel smug that we're at least making a different set of mistakes?

Like any other bit of reflection in practice, such analyses are no good unless we act on them. Over time, I've become convinced that the successful organizations—the ones with real staying power—are the ones where it's possible to have an open and honest discussion about policies and decisions. What's striking to me now about healthcare.gov is the number of misgivings expressed by people on both the government and contractor sides about project success during the development, and yet nothing seems to have happened as a result. We know about those misgivings now because of the intense scrutiny being applied to the project from all sides. How confident are we that we would detect similar warning signs in our own projects—and raise the issue for other people to hear? And how confident are we that are own organizations provide realistic and useful conduits for such information when it needs to be heard?

Your Turn

My own answer to Mosemann's question is that we have indeed made progress; there are many centers of excellence we can find for software being done well. What does make me pessimistic is that the lessons learned haven't been more broadly disseminated across the industry and that business and contractual issues too often work against the deployment of best practices. But what would you say?

I'd actually like to apologize for the US-centric nature of the examples in this column; while I'd have loved to cast a broader net, I also felt it's important to write about what I know well. How do you perceive the health of software engineering in your part of the world? If you have thoughts to share (to agree, to highlight a different set of problems, or to argue that "it would never happen here"), I'd love to hear from you. Share your thoughts by writing to me at fshull@computer.org.

References

1. A. Goldstein and J. Eilperin, "HealthCare.gov Goal is for 80% of Users to Be Able to Enroll for Insurance," *Washington Post*, 16 Sept. 2013; www.washingtonpost.com/national/health-science/healthcaregov-goal-80-percent-able-to-enroll-for-insurance-through-web-site/2013/11/16/04fa02a2-4e1a-11e3-ac54-aa84301ced81_story.html.
2. B. Fung, "HealthCare.gov Had a Glaring Security Flaw that Wasn't Patched until Last Week," *Washington Post*, 30 Oct. 2013; www.washingtonpost.com/blogs/the-switch/wp/2013/10/30/healthcare-gov-had-a-glaring-security-flaw-that-wasnt- patched-until-last-week.
3. A. Medici, "Obama: Federal IT Procurement a 'Systemic Problem,'" *Federal Times*, 14 Nov. 2013; www.federaltimes.com/article/20131114/IT/311140006.
4. S.I. Erwin, "In Defense Acquisitions, Prevention Is Better Than Cure," *National Defense*, blog, 11 Nov. 2013; www.nationaldefensemagazine.org/blog/Lists/Posts/Post.aspx?ID=1329.
5. G. Holzmann, "Landing a Spacecraft on Mars," *IEEE Software*, vol. 30, no. 3, 2013, pp. 83–86.
6. "NASA: Assessments of Selected Large-Scale Projects," *Highlights of GAO-13-276SP*, US Government Accounting Office, Apr. 2103; www.gao.gov/assets/660/653866.pdf.
7. K. Nnamdi, "Improving Electronic Health Records," *WAMU 88.5*, 30 Oct. 2013; http://thekojonnamdishow.org/shows/2013-10-30/improving-electronic-health-records.
8. D. Auerbach, "Err Engine Down: What Really Went Wrong with healthcare.gov?" *Slate*, 8 Oct. 2013; www.slate.com/articles/business/bitwise/2013/10/what_went_wrong_with_healthcare_gov_the_front_end_and_back_end_never_talked.html.
9. J. Rothman, "Creating a Healthy Project Culture," *Managing Product Development*, blog, 4 Nov. 2013; www.jrothman.com/blog/mpd/2013/11/creating-a-healthy-project-culture.html.
10. F. Shull, "A Lifetime Guarantee," *IEEE Software*, vol 30, no. 6, 2013, pp. 4–8.
11. R. Pear, "Troubleshooter Reports Progress and Barriers in Bid to Repair Health Portal," *The New York Times*, 1 Nov. 2013; www.nytimes.com/2013/11/02/us/politics/day-1-on-healthcaregov-fewer-than-a-dozen-signed-up.html.

FORREST SHULL is a division director at the Fraunhofer Center for Experimental Software Engineering in Maryland, a nonprofit research and tech transfer organization, where he leads the Measurement and Knowledge Management Division. He's also an adjunct professor at the University of Maryland College Park and editor in chief of *IEEE Software*. Contact him at fshull@computer.org.

See www.computer.org/software-multimedia for multimedia content related to this article.