Disbanding the “Process Police”: New Visions for Assuring Compliance

Forrest Shull

ELSEWHERE IN THIS issue, you’ll find a selection of technical articles describing recent research on handling software compliance in domains such as business processes, privacy policies, and even human health and safety. I decided to also take a look at how software compliance is dealt with in exceedingly critical environments: how compliance is assured when the consequences of failure can impact not only human life and multimillion-dollar investments but also national prestige.

I had the opportunity recently to participate in the annual meeting of researchers working for NASA’s Office of Safety and Mission Assurance (OSMA). Their projects are designed to enable more effective and efficient software assurance (SwA) and compliance-checking. Depending on the level of criticality, NASA’s software must comply with a variety of standards and regulations that aim to assure that the software can support mission success as well as the safety of astronauts and people on the ground. Bryan O’Connor, who served as NASA’s chief of safety and mission assurance through 2011, set forth a vision of his organization as a systems engineering partner, engaged with missions early. That “commander’s intent” was embraced by OSMA, which focuses on applied research that provides actionable insight, tools, and support to current NASA missions, while also building the long-term critical understanding that helps catalyze future progress. Almost all this research is undertaken in the context of specific NASA missions aimed at building advanced scientific devices, satellites in low-Earth orbit, and even systems designed to work on Mars and beyond. (Full disclosure: my own research work has received funding from OSMA.)

I won’t report on specific technologies and approaches under study—you can find that type of information in this issue’s technical articles. Instead, I’ll summarize the OSMA researchers’ vision for how software compliance will be done in the future, assuming that current research directions bear fruit. For practitioners, I hope this synopsis will be an interesting prediction of what the development environment will look like in the future—especially as compliance-checking becomes more widespread, not only against requirements for safety and assurance but also for ever-more-important contemporary needs such as security and privacy. It is increasingly important for developers to have a sense of both what assurance is and what it can be.

Perhaps the key vision for the future of software assurance is a desire to end its role as the “process police.”

Process Police versus Process Coaches

Perhaps the key vision for the future of SwA—one that I’ve heard both from teams within NASA and from customers building safety-critical software,
for example, within the US Defense Department—is a desire to end SwA’s role as the “process police.” Too often, development teams perceive assurance in a negative light. Many of the SwA personnel I spoke with expressed frustration at meeting developers who assume that SwA always takes an overly process-driven, adversarial approach that prevents developers from doing the things they really want to do.

In a recent commentary entitled “How Do We Fix Systems Engineering?” (Systems Engineering Research Center 2010 Annual Report, www.sercuarc.org/uploads/files/SERC-InsidePages2010_FINAL.pdf, retrieved 5 March 2012), prior NASA Administrator Michael Griffin provided one particularly memorable and direct anecdote concerning his reaction to an earlier version of the agency’s Systems Engineering Manual:

I looked at the document the Chief Engineer gave me and said, “This thing makes me want to throw up—it’s all about what the rules are for systems engineering and what the process is and it doesn’t have anything to do with how it’s really practiced. The only job I was ever good at was being a spacecraft system engineer. Everything that I ever worked on that really flew worked, and we didn’t do any of this stuff.” The Chief laughingly agreed. So here we were, the Chief Engineer and the Administrator of NASA, the two highest authorities that had anything to say about the release of this document, both agreeing it didn’t have anything to do with what we really did when we did engineering.

Griffin’s diagnosis was that systems engineering had overemphasized the checklist side of the discipline at the expense of the constructive aspects—that is, of actually “teaching students how to do systems engineering.”

Yet I’ve met forward-looking SwA personnel who do think of themselves as stakeholders with a constructive benefit to add to the project, and who are committed to improving the state (and perception) of SwA practice. Within NASA, I’ve heard several SwA folks describe their vision for the research as enabling them to function more as a “process coach,” providing support that helps developers achieve things that they otherwise couldn’t. In this vision, the SwA role is analogous to a basketball coach: although the team members may be skilled, the coach can provide information to improve player and team performance, find weaknesses that the team can fix to improve their performance, or look forward to possible challenges arising from the other team’s play and help develop strategies to counter them. Because the coach isn’t a player, she can bring some independent insight that helps identify issues worthy of attention, and manage the associated risks.

Rather than focusing on developing ever-better process checklists, SwA research at NASA aims to exploit ever-more-available computing power, the ubiquitous prevalence of development tools, and new technological advancements to create SwA approaches that better support developers. The goal of all these approaches is to yield solutions that provide more upfront information with shorter feedback loops (that is, providing insights when they can do the most good). Many of the approaches move beyond process checking to assessment of the software products themselves: rather than assume that mandated processes yield the desired benefits, these technologies let SwA examine both the process be-
Correct by Construction versus Assuring Quality Later

A vision of SwA that is more supportive and constructive to developers cannot diminish the SwA team’s objectivity when checking compliance. At NASA, SwA has an independent chain of command and independent financing from the mission cost for a reason: decisions must be made on the available evidence about what’s best for the mission and whether the project can safely achieve its goals.

But the vision does focus on enabling software that is “correct by construction”—that is, built using methods that support good quality in the first place, not with quality being tested and implemented after the fact. This general goal encompasses two main principles. First, compliance and assurance issues are important across the software life cycle, but with special attention paid to the earliest phases. The best way to avoid wasted effort is to ensure that the system scope is adequately and correctly defined upfront. Thus, research is focused not only on helping project managers get the requirements right but on helping civil servants make sure that the operational concept is sound and the system to be built is feasible, before any technical or contractual commitments are made.

Second, assurance activities must be well integrated into the tools and processes that developers are already using. The most successful research projects (in the sense of having the most impact on practice) tend to be those that utilize the data and materials that already exist in typical development environments. In a world of ubiquitous computing, developers are more averse than ever to anything that seems like “extra work” or documents created only to keep the SwA team satisfied.

But bug trackers, content management systems, developer wikis, and other applications that have become a normal part of the developer environment can be mined for the most accurate insights into developer behavior and project progress.

In short, this vision recognizes that the only way to meet the exceedingly high goals for software quality is to start at the beginning of the software life cycle and leverage the software design platform and accurate measures of software quality throughout.

Constructing Validation Environments

Often, the standards and regulations with which developers must comply mandate various SwA approaches. We envision a future in which the decisions about when and how often to apply such SwA technologies can be tied to accurate assessments of the current state of system quality. In this way, future approaches to compliance should leverage the results of the many other investments in SwA technologies that assess and improve software quality.

Drawing data from the development environment itself to monitor software quality and determine whether compliance objectives are being met would enable another part of the SwA vision: validation environments (analogous to the integrated development environments that developers have available) consisting of a comprehensive suite of analysis tools that support the necessary assurance checks. Imagine dashboards that track defects, bugs, and technical-debt items for developers while aggregating data about those same issues, feeding them into state-of-the-art analyses behind the scenes, and making the results available to both SwA personnel and developers in a timely way.

Model-based development has been the focus of much recent research investment, and it’s also a component of the work done at NASA. Our desired
vision of the future sees such models as built into and underlying our validation environment. We hope to see models with associated tool support that allow developers (with minimal additional training) to perform queries and what-if scenarios based on the model. Models can’t be yet another form of system documentation that may or may not be kept up to date; they have to support useful interactions with important stakeholders.

Thus, this vision suggests using the computational resources available not only to build the system but to analyze the system as it is being built. Testing and validation assets would be constructed incrementally, starting as soon as possible—in short, a new philosophy of “baking in” test rather than doing it after the fact.

**Limits on Automation**

Although the vision for future SwA relies on leveraging more automation in analysis, there are limits to what automation can do. Tool support can help users visualize and understand the degree to which the current system complies with its quality requirements, but we will still need experienced humans in the loop to come up with the right scenarios to ensure that the system is tested under appropriate conditions.

Even in the most optimistic visions for the future, the OSMA group felt it unlikely that we would get past 80 percent automated code generation from models. NASA’s systems will still require substantial hand-coding. Moreover, actually formulating the models correctly requires a deep understanding of the system—which itself requires substantial human skill. Some of the most long-lived assurance activities that we have in software engineering—namely, manual technical reviews—will still have a role in checking that system models accurately reflect the domain.

These points suggest that we will still, for the foreseeable future, need human expertise and involvement in SwA to make sure that testing encompasses the right scenarios, that our analyses are focused on the right issues, and that the domain is adequately and completely modeled.

**Feedback Loops**

One of the key lessons learned from doing research so tied to contemporary mission needs is that mechanisms must be in place for organizational feedback loops. Researchers need the opportunity to engage with new projects on the basis of their prior successes, and when research results are suitably mature, they need to influence the typical state of the practice more widely. Such technology transfer is part of the way that OSMA makes the best use of its investment in research; channels exist by which mature results are communicated back and used to impact the next generation of processes, standards, and guidebooks at the agency.

But perhaps a more important feedback mechanism relies on the people themselves. The more tools SwA personnel have in their toolkit, and the more they receive training on effective methods of systems and software engineering, the more likely they are to find ways to add value to the project that don’t rely on checklists. This is another way in which enforcing the engagement of researchers and project personnel effectively supports the future of NASA’s forward-looking, unique, and challenging missions—and can serve as food for thought regarding research models and attitudes on the part of software developers and researchers alike.

**Acknowledgments**

Thanks to all the researchers in NASA’s Software Assurance Research Program who participated in formulating this vision, but most especially to those whose conversations helped me refine and deepen it: Lisa Montgomery, Madeline Diep, Leila Meskhat, Caroline Wang, and Joel Wilt.

**FORREST SHULL** is a senior scientist 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 is an associate adjunct professor at the University of Maryland College Park and editor in chief of *IEEE Software*. Contact him at fshull@computer.org.

---

**Microsoft** has a job opportunity for Software Dev. Engineer-RAAS or Other in Redmond, WA. Design, implement and/or test comp. s/w apps, sys. or svcs., working with other engineers, working on entry-level problems. Under supervision apply principles and techniques of compisci, engineering and/or math. analysis. Formulate and analyze s/w reqts at an entry level. May be assigned to various entry-level proj. that utilize the rqd. tech. skills to deploy successful prod. releases from early prod. def. and scoping to detailed spec., imp., and roll-out phases. 8AM to 5PM, Mon-Fri; $97,035/yr, std. co. benefits. Req. BA/BS or foreign equiv. in Comp. Sci., Engg, Math., Info. Sys., Physics or rel. field and educ. or exp. in C, CSS, HTML, Web apps & debugging. Any suitable combination of education, training or experience is acceptable. To apply, submit resumes to: Recruitment and Employment Office, MICROSOFT CORPORATION, Attn: Job Ref #: MIC85013, P.O. Box 56625, Atlanta, GA 30303.