Performance Testing Moves Toward Maturity

In last month's column, I began sharing some of the key points that I took from WOPR6, the sixth meeting of the Workshop on Performance and Reliability). In case you missed the column, WOPR is an ongoing series of invitation-only, minimal-cost peer workshops for experienced performance testers and related professionals that emphasizes mutual learning, sharing hands-on experiences, and solving practical problems. WOPR6 (www.performance-workshop.org) was specifically organized to explore evolving perceptions of performance testing. We accomplished this through reports of relevant experiences from past projects and current initiatives that demonstrate or contradict the view that performance testing is undergoing a period of significant, rapid and positive change.


Scott Barber

While it's neither possible nor productive to summarize the six days of conversations among recognized experts and experienced practitioners, there is value in mentioning some of the key points that I took from the workshop. Specifically, I'd like to continue sharing with you points of view that either corroborate or oppose some of the positions I've promulgated in this column and other venues recently.

What About Open Source?

As some of you may know, I'm a big fan of open-source software. That's not to say that I oppose pay software; rather, I have tremendous respect for those who dedicate their time to building and maintaining open-source software, and I make a point to try out open-source software when it appears to be a viable alternative to pay software. Sometimes, I find that the available open-source product meets my needs; other times, it doesn't. When it comes to performance testing, I've built up a significant library of open-source software that I use for everything from load generation to resource monitoring to file parsing when I'm working on projects that use technologies this software supports.

Typically, I've found that "household name" organizations are resistant to using open-source software. Their reasons include limited support and training, small user communities, thin documentation and bad previous experiences. That's why my ears perked up when Goranka Bjedov, a senior software engineer in testing at Google, casually mentioned the following while sharing a recent performance-testing experience. "Google uses open source and they build it out. They figure that they can pay for [a tool] or not pay for [a tool], but it's still [a tool]—and at least with open source, we can modify the code when we need it to do [something that isn't natively supported]."

This is an incredibly insightful statement. While few organizations have Google's pool of talented developers, most of the tweaks that open-source tools need to meet a user's specific needs don't require any rocket science.

More About Open Source...

A day or two after Bjedov shared her experience with us, Antony Marcano, a performance testing consultant in the U.K. and manager of TestingReflections.com, which aggregates some of the best software development and testing blogs on the Web, shared an experience with a related message. On a recent project, the team made the same open-source decision as Google, but his client didn't make the tweaks to the tool themselves. In their case, it was cheaper and more efficient to pay someone to build out the open-source tool than to pay for the licenses of the payload generation tool. Interestingly, they found that the members of the open-source community for this tool charged quite reasonable rates in exchange for permission to put the tweak into the open-source project at the conclusion of the development effort.

Marcano's story gave me a fresh perspective. With some of the top payload-generation tools running six or seven figures to purchase and five or six figures in annual maintenance fees, why not simply hire a member of the open-source development team to enhance or customize the tool (and potentially provide some training in the process)? That could be a whole lot more cost-effective than purchasing a proprietary tool that may or may not meet your current or future needs any better than the open-source solution.

Moving to IDEs

One area of consensus about the testing evolution was evident in the increasing partnership between performance testers and developers. Just a few years ago, performance testers were seen as merely additional members of the "quality police" that many developers considered the bane of their existence. At WOPR6, some performance testers related their experiences of working closely with the

Scott Barber is the CTO at PerfTestPlus. His specialty is context-driven performance testing and analysis for distributed multi-user systems. Contact him at sbarber@perftestplus.com.

development team. Marcano and Neill McCarthy, another top performance tester from the U.K., described finding a high degree of success in working side by side with developers in agile environments.

Coupling these experiences with the tool vendor penchant for piling load generation and other performance-testing tools into the developer's IDEs, the tipping point has apparently been reached. Performance testing as an activity in isolation from development, using tools unfamiliar to developers, will become a thing of the past. In fact, most of the new performance testers I've met at conferences or taught in recent training classes are converted developers.

In further evidence of the shift in the perception of performance testing from an independent task to a job most efficiently accomplished in close collaboration with developers, Bill Barnett, a developer working on the performance-testing component of Microsoft's Visual Studio Team System, shared his experience using VSTS to performance-test a new Microsoft application. Working with developers, he was able to conduct unit performance testing, stress testing, functional testing under load and other varieties of performance testing without switching tools.

Whether you're a fan of this trend or not, both Microsoft and IBM are taking it seriously, and it does present a powerful message.

Maturation, not Evolution

Possibly the most eye-opening realization crystallized during a presentation by Mike Pearl, a friend and senior performance and reliability tester at Intuit. In fact, I was so intrigued by my epiphany that I forgot to record the technical details of his experience, being too busy jotting notes about the light bulb that had just illuminated my brain.

The point of Mike's presentation—and something that I'd subconsciously known for some time—is that maybe rather than evolving, paradigm shifting or undergoing revolution, performance testing is instead *maturing*—adapting, merging, collecting, injecting and infusing lessons from other fields.

Of all the recent advances I've witnessed in performance testing, virtually none are true innovations; rather, they're new applications of well-known and documented research and practices from fields such as human psychology, statistical analysis, information modeling and operational research. If I could influence just one aspect of the field of performance testing, I'd like to encourage us all to reach out to other disciplines to learn what they have to teach us.

Predicting Performance

Over the years, I've been vocal about several pet peeves. Prominent among them: The practice of extrapolating results from a test environment that isn't an accurate replication of production, for the purposes of predicting or even estimating an application's performance, is unwise. In fact, it ranks somewhere between risky and crazy, unless you hire an expert in application performance modeling and then confirm both the production environment and the predictions.

Craig Rapin, a performance test manager for one of the world's largest providers of financial services, came to WOPR6 hoping that the experts in attendance could help him meet his most common challenge: predicting production performance based on results from non-production environments.

Unfortunately for Craig, he received exactly two pieces of advice. First, "Don't do it; it doesn't work," and second, "Find a way to convince your superiors to allow you to run at least a couple of tests in production; it's the only reliable way to gain insight into what production performance will be like."

I guess that shows that no matter how much we seem to be evolving in some areas, in others, the state of the practice has remained essentially stagnant for at least the last six years.

An Indicator of Optimism

I started this two-part column with the intent of sharing some points of view from other respected members of the performance testing community that either corroborate or oppose some of the positions I've shared in this column and other venues recently.

In general, I feel as though most of the leaders in this community who were able to attend this workshop agree on the potential of the advancements that are happening in our field, but I may have been a bit optimistic about how far we've come in applying those advancements to their full advantage.

Rather than evolving, paradigm shifting or undergoing revolution, performance testing is maturing.

But I don't mind being an optimist sometimes. To tell the truth, with the predominance of pessimism surrounding performance testing over the last five or so years, perhaps a bit of vocal optimism on the part of respected members of the community is an advancement in itself.

End Note: The portions of WOPR6 discussed in this column were attended, all or in part, by A.J. Alhait, Henry Amistadi, Charlie Audritsh, Marini Ballard, Scott Barber, Bill Barnett, Goranka Bjedov, Angela Cavesina, Ross Collard, Dan Gold, Corey Goldberg, Linda Hamm, Julian Harty, Dawn Haynes, Gabe Heinsinger, Doug Hoffman, Andy Hohenner, Paul Holland, Pam Holt, Mike Kelly, James Lyndsay, Shelton Mar, Antony Marcano, Neill McCarthy, Michael Pearl, Timmons Player, Craig Rapin, Harry Robinson, Robert Sabourin, Roland Stens, Tony Watson, Brian Warren, Donna Williamson, Steven Woody and Nick Wolf.