An Evolution In Performance Testing

I recently returned from WOPR6, the sixth meeting of the Workshop On Performance and Reliability (www.performance-workshop.org). An ongoing series of invitation-only, minimal-cost peer workshops for experienced performance testers and related professionals, WOPR builds skills in system performance and reliability, and allows people who are interested in these topics to network with their peers. The emphasis is on mutual learning, sharing hands-on experiences and solving practical problems. In these semiannual meetings, groups of approximately 20 subject-matter experts and experienced working practitioners discuss pertinent state-of-the-art topics with the intent of advancing the state-of-the-practice of performance and reliability testing through open collaboration and cross-pollination of ideas.


Scott Barber

WOPR6 was organized to explore the concept of evolving perceptions of performance testing. We accomplished this through reports of relevant experiences from past projects or current initiatives, which demonstrate or contradict the view that performance testing is currently undergoing a period of significant, rapid and positive change.

As you might imagine, participants' perceptions of the field's status varied widely. During the run-up to the workshop, participants expressed positions ranging from "Things are getting worse!" to "Same stuff, different day" to "Performance testing is advancing so quickly, I can hardly keep up!"

This time, Google was kind enough to serve as host to the workshop at the Googleplex in Mountain View, just south of San Francisco. For six straight days, approximately 40 individuals from around the world with a passion for performance testing met and discussed personal experiences that, to them, represented performance testing as stagnating, advancing, evolving, poising for a paradigm shift or something else altogether.

At the Tipping Point

Entering the workshop, I had a sense that in some ways, the state-of-the-art of performance testing was nearing a conceptual and technological tipping point likely to cause a dramatic improvement in how we test systems. However, I sensed that many of these new advancements had not yet permeated the industry as a whole, a feeling that six days with these topics did not dispel.

That isn't to say that significant advancements aren't happening—they are. But they're apparently occurring in small, often isolated pockets. It seems that for every one organization that implements one of these advancements, there are 10 more that consider, but don't implement the same advancements. It appears to me that many companies may be trapped by inertia.

Although it would be neither possible nor productive to summarize the six days of facilitated and passionate conversations among experts and experienced practitioners, I'll share some of the key points that I gleaned. Specifically, I'd like to discuss opinions that either corroborate or oppose some positions I've put forth in this column and elsewhere.

Evolution Isn't Always Good

Brian Warren, a manager of performance testers at a company that is frequently listed in Fortune magazine's top 1,000 companies, shared an experience about how advancements in performance testing in his organization have resulted in an increasing amount of data to process and analyze. On the surface, this appears to be good, since we've been complaining for years about not being able to collect enough of the right data fast enough to properly assess system performance. The challenge? Now that we have the data, we don't always know what to do with it. To illustrate what made him suspicious of some of the findings from all of this data, Warren related a story from his days as a geographer specializing in mapping.

It seems that the field of geographic mapping made some reasonably embarrassing observations based on the tons of new data available from global positioning satellites. With so much of this new satellite data streaming in so fast, to simplify processing, the first step was to "pixelate" or average the data into sections measuring 10 meters by 10 meters.

While this degree of granularity was great for macro-level mapping, averaging over 33 square feet means that things like most houses, streams, drainage ditches and country roads simply disappear. I'm sure this is acceptable for, say, a general topographic map for the state of Colorado, but if the goal is to determine runoff patterns in Florida, you simply aren't going to get an accurate picture with

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

data granularity of 10m x 10m. Warren shared that it took quite some time to realize that some of their applications of the data were simply not accurate and potentially misleading without "de-pixelating" prior to analysis.

I took two important lessons away from Warren's experience. First, I'm reminded that evolution is a set of slow and more or less naturally occurring changes typically brought on by an alteration in environment. Sometimes these evolutionary changes are critical for survival, but at other times, they cause more harm than good. Warren's story reminded me not to get too excited about seeing evolution until I'm sure that the evolution is going some place positive. In this case, the simple existence of more data does not automatically solve many problems.

Second, I'm reminded of just how messy our data often is, and how easy it can be to compile a few averages to make that data easier to understand. The problem is that those averaged averages can easily pixelate some of our best indicators of poor performance right out of sight. Even worse is the fact that most of the tools we use start by presenting us with pixelated data, which makes it that much more tempting to start our analysis from there instead of remembering that that data is a summary and going instead to the raw data before we plunge into our detailed analysis.

Prototyping Performance

In 2001, I presented a paper at the Pacific Northwest Software Quality Conference that described an experiment I'd been working on: I had folks sit in front of a Web site I had created and rate their satisfaction with the time it took each page to download. Using a little JavaScript, I'd made it so that different pages took different amounts of time to display.

The exercise was designed to help me determine a user's tolerance for delay using qualitative methods that I could then directly convert to quantitative data. Mostly this data was used to show the writer of the performance requirements that, for example, an eight-second download time simply wasn't acceptable to the users of this application, no matter what research they quote to the contrary.

I have to admit, until performance testing consultant Roland Stens shared with me his experience using prototypes to collect performance requirements, I had become rather convinced that my little idea made a cool paper, but was pretty much hopeless in industry due to a dearth of prototyping. Stens, however, seems to have found a solution to the prototyping problem—his team uses Axure RP (www.axure.com), which is an affordable and usable prototyping tool. While Axure RP doesn't come with a "How long would you like this page to take to load?" GUI button, it took Stens all of six lines of JavaScript to include this functionality! In my view, that's more than a little exciting.

Oh, did I mention that Stens was completely unaware of my paper and that his approach was very well received by his client?

Paradigm Shift?

Harry Robinson, an expert in model-based testing, spoke to us about his minimal but significant performance testing experience, making some wonderfully quotable statements. First, he remarked that when doing performance testing, you're more likely to find what you're looking for by hunting "for pessimal behavior as opposed to optimal." While this is probably not a new perspective for the folks reading this column, I think that Robinson stated it particularly well.

He continued by making the point that "domain experts can be dangerous." We have all seen that most job posts for performance testers list domain expertise near the top, yet even a performance testing novice realizes this is often a mistake. The fact is that while domain knowledge can improve your test design, it can also tend to create a situation called "inattentional blindness," which occurs when we don't think to test or pay attention to a situation that could turn out to be extremely relevant because, for example, "no *real* user would ever do *that*."

Robinson's next brainteaser was a killer: 'If you use a machine gun, you don't have to aim that carefully.'

Robinson's next brainteaser was, "If you use a machine gun, you don't have to aim that carefully." In other words, if you have the ability to test lots of different combinations and scenarios, it may make more sense to just start testing them instead of spending a bunch of time building a model that has a low probability of being correct anyway.

Finally, after presenting his experience and listening to many of ours, Robinson said, "I don't know if you're experiencing a paradigm shift in how you model performance usage, but you should be." While that's a bigger topic for another day, I will say that I felt more than a little bit validated that the person I have come to know as "The Model Guy" came to the same conclusion as I, along with others, had when faced with the challenge of trying to figure out how to design our performance tests to get the most important or meaningful results efficiently.

It seems that we've reached the bottom of our second page together once again, and I have many more key points from the workshop to share with you. So I'm going to do to you what every one of the few television shows I catch occasionally have done to me this week: get you all interested in what is coming next, only to say...

... To Be Continued