Macro to Micro And Back Again

Scott Barber

About 10 years ago, I was at a U.S. military port in Bremerhaven, Germany, evaluating a prototype device and beta software intended to improve the in-transit visibility of military equipment. (I didn't realize it at the time, but in retrospect I realize that was my first software testing job, even though my title was "Information Engineer"—whatever that means.)

One afternoon, the project lead called me over to a table where he'd spread his notes, sketches, flow charts, spreadsheets and other pieces of paper. He waved his hand over the array and asked, "What does this say to you?" I said, "You mean other than you use a lot of paper?" and proceeded to summarize for about 30 minutes the intent of our observations and how each stack of paper supported that intent.

Nodding, he pointed at a specific spreadsheet with a mathematically derived summary of the quantification of the qualitative data we'd collected so far, and asked the same question: "What does this say to you?" I studied the summary and reviewed the supporting data, roughing out some summary data of my own, commenting and asking questions all the while. About the time I'd completely zoned in on the numbers, summaries and data patterns, and was about to go out on a limb and accuse the summaries of "lying with mathematically correct, but highly misleading statistics," he stopped me. I looked at him quizzically, thinking I'd made a mistake or that I'd crossed some kind of line.

On the contrary, he informed me that he'd been trying to determine whether I was a "big picture person" or a "details person"—and that I seemed to be both. I didn't understand what the big deal was; I could have told him that. Before I could so much as shrug, he went on to explain to me how, over his 30-year career, he'd met very few people who could effectively flip back and forth between macro- and micro- tasks. At the time I didn't believe that ability was as rare as he implied, but he was my boss on a government contract, so I figured it was smarter to smile and say "thank you" than to debate the point.

The Stuff in Between

For some reason, I remember that incident. I've always been pretty good with both macro-and micro-level tasks, switching back and forth between the two—it's the stuff in between that has always eluded me. For example, if put on the spot, I can, without hesitation, tell you where I want to be in 15 years as easily as I can tell you what my plans and task list are for today. If, however, you ask me what I'll be up six months from now, I'll stutter, start out with a laundry list of disclaimers and assumptions, and conclude with some vague, dismissive phrase like "I dunno, maybe, if everything turns out the way I think it might, I'd like to be..."

The other day, it dawned on me that every top-notch performance tester who loves his or her job exhibits this trait. We all tend to be good at figuring out conceptual overviews of the applications we're testing—their basic architectures, expected usage, business drivers, testing strategies and so on. We also enjoy the time we spend tracing individual results to source code, conducting detailed retrospective data analysis or writing code to solve some challenging aspect of realistic user simulation. Our frustrations always seem to come in the areas in between, such as, "How long will it take you to test this?" "How much more performance testing do we need?" or "What dates are you going to need access to the network team?"

As I pondered this a little more, I realized that many of the great functional testers I've encountered over the years excel at the micro level, which often earns them names like *nit-picky* and *anal compulsive*. Other members of typical software development teams who spend most of their time in micro-level activities are developers and system administrators; no surprise there. As also makes sense, the great business analysts I've worked with are macro-level savants. They can look past implementation details to focus on overall business intent and items critical to the end user in a way that I envy at times. I ask myself why elite performance testers tend to demonstrate a preference for the traits generally associated with business analysts and functional testers rather than traits, say, of managers at various levels of the team.

More Meta-Analysis

If you're wondering what's the point of all this meta-analysis, I asked that myself—and came up with some interesting answers. First, I realized that when interviewing performance testers, I ask for stories intended to elicit the same kinds of responses that my project lead wanted when he asked me to review his paperwork. Until my epiphany of a few days ago, I had no idea that I was subconsciously trying to determine whether the interviewee was a big-picture person, a detail person or both. And I had no idea that I was actually looking for people who were both. Knowing specifically what I'm looking for is going to make that

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

part of conducting interviews easier in the future. For instance, I can simplify what often felt like a rambling set of questions and answers to two questions: 1) Give me an overview of the last application you performance-tested and the general intent of performance-testing that application, and 2) Tell me about how you collected and analyzed data to support that intent. I suspect that someone who enjoys both the macro and micro aspects of performance testing will launch into clear and conversational stories in response to both questions and will have no hesitation switching gears from one story to the other.

My Macro/Micro Revelation

Next, it dawned on me that classifying various aspects of performance testing as either macro- or micro-level tasks can be extremely useful in teaching performance testing and in describing the purpose of performance testing activities to the rest of the software development team.

As an example, consider usage model development—a macro-level task. To create an accurate and useful model, we don't need to know the implementation details, every piece of variable data, or every button click, screen, hot key, error condition and method for simulating session persistence with our load generation tool.

What we *do* need to know is what the application's actual users do at an activity level. A general description like the following, with some hands-on experience with the application under test, could generate a fairly accurate user activity model valuable for a plethora of micro-level activities: "Most users will log in, do some searches, add some stuff to their cart and eventually check out. Some will add stuff to their cart, but never check out. Some will create new accounts. A few will update account information or check order status."

Some of the micro-level activities this simple model could inform include test data creation, usage-scenario variance analysis, scenario scripting, performance goals and requirements analysis by activity, and sequential dependencies in scenarios. But beware! Determining the entire general-usage scenario before you create the correct number of the right classes of test users for the expected system load may result in the wrong group of people in your working session, skewing the general picture of the application usage and exposing you to "analysis paralysis."

Finally, I realized, as I was explaining my micro/macro revelation to a non-performance tester, that what we performance testers do has just gotten easier to explain. If you say that performance testing involves a cyclical process of decomposition and recomposition of the application, folks who don't already know what that means will remain in the dark.

Classifying various aspects of performance testing as micro- or macro-level tasks can be extremely useful.

However, if you say that performance testing commonly alternates between macro- and micro-level tests to clarify patterns and isolate performance issues, making them easier to resolve, the light bulb will spark. The same is true for my preferred methods of performance test planning and documentation. Micro-level strategies and micro-level plans make sense and intuitively imply the distinction that I once spent much of a magazine article explaining between my "Performance Test Strategies" and "Performance Test Mini-Plans." (Don't you hate it when you come up with a better name for something *after* the document or e-mail has already been delivered?)

The Big Picture and the Details

As I think back over all of the dozens of software projects I've performance-tested, I realize that the vast majority of the insufficient performance testing I've seen, which didn't meet the needs of the team, were overly focused on either the macro- or the micro-level views. Teams that were overly focused on the big picture would encounter frustrations such as identifying that the application has failed to meet the stated performance requirements, but may have significant difficulty in determining what to do fix the problem. Teams that were overly focused on the details would have the opposite frustration; they'd have exceptionally well-tuned servers, algorithms and networks, but have no idea what the performance felt like to an actual user.

In my experience, the key to both determining how the application is really performing and being able to isolate performance issues so they can be effectively resolved is striking a balance between these macro- and micro-level activities.

This is also why I'm a staunch proponent of a performance tester, or performance test team collaborating closely with the developers, testers, analysts and managers throughout the testing effort. It's much easier to find the right balance when working closely with teams and individuals with different, but complementary perspectives of the application being tested as opposed to working in isolation.

Macro- and micro-tests, macro-strategies and micro-plans, macro-level application usage and micro-level usage implementation detail, macro-level results summaries for executives and micro-level test results for developers... it sounds like a day in the life of a performance tester to me.

I guess that's why I kept remembering that particular hour from my first trip to Germany. Now maybe I can get past that pesky "What does this say to you?" and recall something that's far more pressing: the name of the exquisite Hefeweizen I downed on that trip that has had me hooked ever since.