Testing Strategy & Management Series

by:

R. Scott Barber

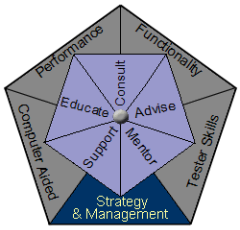
SOA Driven Testing?

By now, I suspect that most folks who are involved with designing, writing, maintaining and/or supporting software have at least heard of the newest addition to the industry's "buzz-acronym alphabet soup." It's not XP (Extreme Programming), OO(Object Oriented) or even TDD (Test Driven Development). This time the buzz-acronym is SOA (Service Oriented Architecture). And in fashion with many of the more recent buzz-acronyms, the expanded phrase sheds little, if any, light on what the term really means. When I checked in with a few friends of mine to make sure I had my terminology straight, one of them pointed me to Martin Fowler's Blog where he writes…

> "…one question I'm bound to be asked is "what do you think of SOA (Service Oriented Architecture)?" It's a question that's pretty much impossible to answer because SOA means so many different things to different people.
>
> - For some SOA is about exposing software through web services…
> - For some SOA implies an architecture where applications disappear…
> - For some SOA is about allowing systems to communicate over some form of standard structure… with other applications…
> - For some SOA is all about using (mostly) asynchronous messaging to transfer documents between different systems…
>
> I've heard people say the nice thing about SOA is that it separates data from process, that it combines data and process, that it uses web standards, that it's independent of web standards, that it's asynchronous, that it's synchronous, that the synchronicity doesn't matter....
>
> I was at Microsoft PDC a couple of years ago. I sat through a day's worth of presentations on SOA - at the end I was on the SOA panel. I played it for laughs by asking if anyone else understood what on earth SOA was. Afterwards someone made the comment that this ambiguity was also something that happened with Object Orientation. There's some truth in that, there were (and are) some divergent views on what OO means. But there's far less Object Ambiguity than the there is Service Oriented Ambiguity…"

Service Oriented Ambiguity?!? No *WONDER* I got confused sometimes while reading the articles in CRN, SD-Times, CTO Source, TechRepublic

© PerfTestPlus, Inc. 2006



and others about the technologies behind SOA! This is just one more reason I'm thrilled with my choice to be a tester. Compared to figuring out all of the enabling technologies, testing SOA is a piece of cake. Don't get me wrong, testing SOA has its challenges, but we at least have some experience with the concepts. Allow me to explain.

SOA Concept

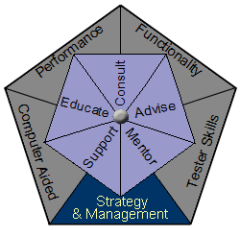
Let's start by taking a look at what SOA is *conceptually* all about from a tester's point of view – without creating a paper that is certain to win us a round of "Buzzword Bingo." Ignoring the hype around the phrase and the acronym, SOA is nothing more than the most recent step in the natural evolution of software and software development that started with the GOTO statement. I'm not being cynical; I'm serious! The dreaded GOTO statement started the evolution of abstraction of code decades ago. Some of you may remember being chastised for using the GOTO statement in line numbered BASIC and upgrading to GOSUB-RETURN to save face before becoming empowered by Functions, Procedures and eventually Java Beans or Objects. Even if your programming background doesn't include first-hand experience with that evolution, you probably recognize all of these programming concepts as methods of minimizing code redundancy and abstracting sections of code to maximize code re-use.

This concept of abstraction and code re-use (the basic concept behind what Fowler called the Object Ambiguity) is what paved the way for the software industry to think in terms of not just reusable segments of code but eventually entire mini-applications that can be used in many different contexts to provide the same, or similar, functionality. Possibly the most well known of this breed of mini-application, as I think of them, are those that process credit card purchases over the web.

I'm sure that it's no surprise to anyone reading this article that once you get beyond the service providers and resellers, there are really only a small handful of organizations that actually publish and maintain the vast majority of credit card processing software. In fact, virtually all of us with Web Sites that sell products (often referred to as B2C or Business to Consumer sites) simply "plug-in" to one of those pieces of software (for a small fee, of course) to make our once-innocent web site into an E-Commerce web site! Of course, this particular type of mini-application has its own buzz-term – it's a called a Web Service. Web Services have been around for several years and are actually the direct predecessors, or maybe the earliest adopted subset, of SOA.

For years I struggled with the question of "What's the difference between a Service and an Object on Steroids?" It took me almost four years to navigate my way through the implementation technologies and coding patterns to figure out that the fundamental difference is that Objects are programmer-centric abstractions of code and Services are user- or business-centric abstractions of code. Basically, a programmer may write code to reference a number of objects that the user is completely unaware of while that user performs an activity, like logging into a secure web site. If, instead, the "log into a secure web site" activity were to be written as a Service, it would be a single entity that accepted certain input and responded with certain output. Not only is the user unaware of the Service, but the programmer writing the application need only be aware of the format and contents of the input and output parameters. In fact, SOA is really nothing more than building software applications in such a manner as to be able to take advantage of Services, whether they are available via the web or the next server down on the rack. Independent of all the ambiguity about technologies, protocols and degrees of abstraction, that is really all there is to SOA.

© PerfTestPlus, Inc. 2006



Testing SOA

That said, there are several things about SOA that are going to present challenges that many testers are not used to facing, at least not in the volumes and combinations that we will see with SOA. First, testing Services in an SOA environment is fundamentally different from testing the Objects that inspired them in at least one significant way. Objects were (and are), as we mentioned, programmer-centric segments of code that are likely to be used in more than one area of one or more applications. An object is generally tested directly via unit-tests written by the developer and indirectly by user-acceptance and black-box testers.

Services however, require a different testing approach because they encompass entire business processes and can call dozens of objects and are unlikely to have been developed or tested by anyone you will ever meet or speak to. As testers, we have little choice but to validate the service as a black-box, probably through some kind of test harness, focusing on input values, output values and data format. Sounds a lot like a unit-test, doesn't it?

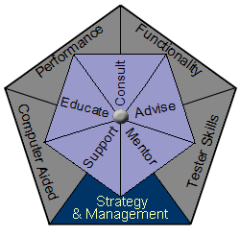
A New Approach

The next challenge we testers face is that with SOA, we can no longer get away with thinking about applications exclusively from just unit and black-box perspectives. We absolutely must think about SOA applications in (at least) three logical segments: the services themselves, the user interface, and a communication or SOA interface segment (sometimes referred to as a "service broker"). Sounds easy enough, but here's the kicker: we need to test each of these segments both independently and collectively and we need to test each of these segments at both the unit-level as well as a black-box. This means more testers pairing with developers, more testers writing test harnesses, and more automation via API versus UI.

The testing challenges that SOA present that I am most excited about (yes, I am aware that makes me a geek) are the challenges related to performance testing. We as an industry already have enough trouble finding the time and/or the money to performance test the way we'd like, even when we are lucky enough to be part of an organization that thinks about performance testing at all. Now we're intentionally building applications so we can plug in code that we will likely never see that was probably written and is certainly hosted elsewhere on some machine we are unlikely to have access to, that takes our data in and magically spits out the "answer" (we'll assume it's even the "correct answer"). How, exactly, are we to trust that this magic service is going to handle our holiday peak? Even more frightening, how are we going to trust that a whole bunch of these magic services are going to all perform well together like the "well-oiled machine" we'd have built (or would like to believe we'd build) on our own?

Why am I *excited* about this you ask? No, not because I think these challenges are going to make me rich (though, that would be nice). I'm excited because I think that SOA is going to force the industry to bridge the gap between top-down (black-box/user-experience) performance testing and bottom-up (unit/component/object-level) performance testing that has needed to be bridged for as long as I've been involved with performance testing.

© PerfTestPlus, Inc. 2006



Performance Testing as it was Meant To Be

Rather than having to figure out the logical segments for decomposition and recomposition, they have already been defined for us. Rather than having to build test harnesses exclusively for performance testing that no one else has thought of, we can piggy-back on the test harnesses used by the functional and unit testers. Rather than starting from a belief that "We wrote it, therefore it will perform well," we will be starting from a position of "Someone else wrote it and we need validate their performance claims and make sure that it actually works that well with our UI/data/configuration."

Facing the Challenge

I'm certain that some of these challenges may seem pretty, well, challenging to folks who haven't faced them before, but it's not completely uncharted territory. Keep your eyes open for more articles, presentations and tools focused on this kind of testing. Pay particular attention to the folks who talk about how testing SOA relates to testing EAI (or EDI for that matter), Middleware and Web Services. They are the ones who have taken on similar challenges before.

Martin Fowler closed the blog I referenced earlier this way:

> "…many different (and mostly incompatible) ideas fall under the SOA camp. These do need to be properly described (and named) independently of SOA. I think SOA has turned into a semantics-free concept that can join 'components' and 'architecture'…"

So while the developers, vendors, marketers and architects are sorting out the "Buzz-Acronym Soup" Martin alludes to, we testers can be digging into the concept of testing applications that are segregated cleanly enough for us to effectively apply all of our tester skills at more points in both the application and the development process. As a tester, I really can't think of a better way to spend my time until we switch to the wave made by the next acronym that sticks!

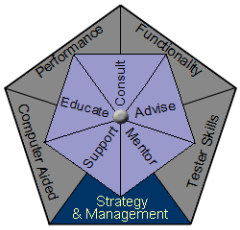
Acknowledgments

This article was first written in support of a webinar presented by Software Quality Engineering on May 9, 2006.

About the Author

Scott Barber is the CTO of PerfTestPlus (www.PerfTestPlus.com) and Co-Founder of the Workshop on Performance and Reliability (WOPR – www.performance-workshop.org). Scott's particular specialties are testing and analyzing performance for complex systems, developing customized testing methodologies, testing embedded systems, testing biometric identification and security systems, group facilitation and authoring instructional or educational materials. In recognition of his standing as a thought leading performance tester, Scott was invited to be a monthly columnist for Software Test and Performance Magazine in addition to his regular contributions to this and other top software testing print and on-line publications, is regularly invited to participate in industry advancing professional workshops and to present at a wide variety of software development and testing venues. His presentations are well received by industry and academic conferences, college classes, local user groups and individual corporations. Scott is active in his personal mission of improving the state of

© PerfTestPlus, Inc. 2006



performance testing across the industry by collaborating with other industry authors, thought leaders and expert practitioners as well as volunteering his time to establish and grow industry organizations. His tireless dedication to the advancement of software testing in general and specifically performance testing is often referred to as a hobby in addition to a job due to the enjoyment he gains from his efforts.

About PerfTestPlus

PerfTestPlus was founded on the concept of making software testing industry expertise and thought-leadership available to organizations, large and small, who want to push their testing beyond "state-of-the-practice" to "state-of-the-art." Our founders are dedicated to delivering expert level software-testing-related services in a manner that is both ethical and cost-effective. PerfTestPlus enables individual experts to deliver expert-level services to clients who value true expertise. Rather than trying to find individuals to fit some pre-determined expertise or service offering, PerfTestPlus builds its services around the expertise of its employees. What this means to you is that when you hire an analyst, trainer, mentor or consultant through PerfTestPlus, what you get is someone who is passionate about what you have hired them to do, someone who considers that task to be their specialty, someone who is willing to stake their personal reputation on the quality of their work - not just the reputation of a distant and "faceless" company.

© PerfTestPlus, Inc. 2006