May/June
2009
Volume 2

SOFTWARE TESTING

Testing strategies for complex environments: Agile, SOA

TechTarget
The Technology Media ROI Experts

EDITOR'S LETTER

Testing in the new worlds of SOA and agile

→ **UNLESS YOU LIVE** in a sci-fi novel, there's one rule of thumb for any new place you go: some things are different; some are the same. Usually, success, fun or survival in that new place depends upon how well you handle the differences. In this issue of SearchSoftwareQuality.com's Software Testing E-Zine, the new "places" are a service-oriented architecture (SOA) and an agile software development environment.

Examining the ins and outs of software testing in SOA environments in "An SOA view of testing," consultant Mike Kelly focuses on the "subtle differences." What's the same is that testers here must start with the basics. In SOA, the basics are connectivity, basic capacity, and authorization and authentication—not that different from other environments. The level of complexity of data models, however, is very different from that of other architectures and requires some new methods of testing.

While agile development requires different testing methods than the waterfall model, those differences lie as much in human behavior as in technology, according to consultant Karen N. Johnson in "Agile testing strategies: Evolving with the product." Agile is a more collaborative process and calls for seizing "iterations as a chance to evolve test ideas," Johnson writes. Fundamental testing tasks, like exploratory and investigative testing, stay the same, but the productivity of completing those jobs can increase.

Johnson and Kelly discuss the finer points of testing in agile and SOA, respectively, in this issue's articles. They also describe revelations they've had while working in those environments and best practices they've learned and continue to use.

The theme of connectivity runs through these discussions of SOA and agile testing, as the former focuses on system connectivity and the latter on human collaboration. Both approaches, when done well, can help development and IT organizations achieve lower costs and better software. ■

JAN STAFFORD
Executive Editor
jstafford@techtarget.com

CHAPTER 1

A SOA view of testing

The complexity of a service-oriented architecture can complicate testing and make choosing and implementing the right testing tools more difficult. **BY MICHAEL KELLY**

→ **SERVICE-ORIENTED** architectures (SOAs) are stealthily becoming ubiquitous. Large and small enterprises leverage them to integrate wide arrays of disparate systems into a cohesive whole. Most companies and project teams that implement SOA do so to reduce the cost, risk and difficulty of replacing legacy systems, acquiring new businesses or extending the life of existing systems.

Testing a SOA requires a solid understanding of the SOA design and underlying technology. Yet, SOA's complexity makes figuring out the functional purpose of the SOA difficult and choosing and implementing the right testing tools and techniques to use with it more difficult.

SOA is not simply Web services. According to iTKO (the makers of the TechTarget award-winning SOA testing tool LISA), nine out of 10 Web services also involve some other type of technology. In addition, they state that most testing for SOA isn't done at the user interface level. It's done using either specialized tools or as part of an integration or end-to-end test. This means people testing SOA need to be comfortable with varied and changing technology, need to understand the various models for SOA, and should be familiar with the risks common to SOA.

WHY SERVICE-ORIENTED ARCHITECTURES?

Large companies typically turn to SOA because their existing systems can't change fast enough to keep up with the business. Because business operations don't exist in discrete or finite units, there are a lot of dependencies built into existing systems. SOA is an attempt to separate the system from the business operations it supports. This allows companies to incrementally build or decommission systems and minimizes the impact of changes in one system to changes in another.

Implementing a SOA removes the requirement of direct integration, which results in business operations that are easier to understand. That translates into a more testable system

EDITOR'S LETTER

CHAPTER 1 A SOA VIEW OF TESTING

CHAPTER 2 AGILE TESTING STRATEGIES: EVOLVING WITH THE PRODUCT

overall. Over time, this makes it easier to support a broader technology base, which in turn makes it easier to acquire and integrate new systems, extend the life of existing systems and replace legacy systems.

There are a number of different model architectures for implementing SOA, including publish and subscript (often called "pub/sub"), request and reply, and synchronous versus asynchronous. In addition, they can be implemented in a broad range of technologies, including message-oriented middleware (MOM), simple HTTP posts with DNS routing, Web services, MQ series, JMS (or some other type of queuing system) and even files passed in batch processes.

If you're testing a SOA you need to understand the implementation model and technologies involved, because these typically combine to give you some common features of a SOA. These features often become a focal point for your testing. They commonly include:

- **Transformation and mapping:** As data moves through a SOA, it's often transformed between data formats and mapped to various standard or custom data schemes.

- **Routing:** The path information takes as it moves through a SOA is often based on business rules embedded at different levels of the architecture.

- **Security:** SOAs often use standards such as application-oriented networking (AON), WS-Security, SAML, WS-Trust, WS-SecurityPolicy, or other custom security schemes for data security as well as authorization and authentication.

- **Load balancing:** SOAs are often designed to spread work between resources to optimize resource utilization, availability, reliability and performance.

- **Translation:** As data moves through a SOA, it's often converted or changed based on business rules or reference data.

- **Logging:** For monitoring and auditing, SOAs often implement multiple forms of logging.

- **Notification:** Based on the model of SOA implemented, different notifications may happen at different times.

- **Adapters:** Adapters (both custom and commercial) provide APIs to access data and common functions within a SOA.

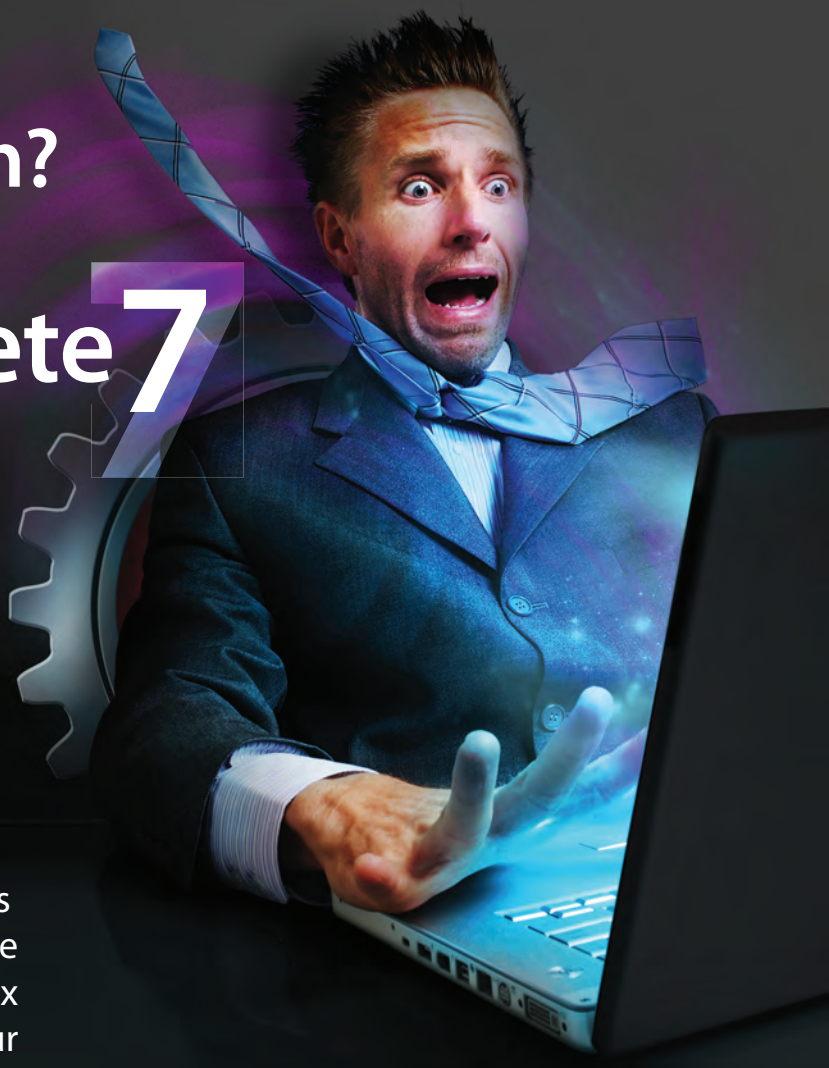
FIGURING OUT WHAT TO TEST

The first thing you need to do when you start figuring what to test for your SOA is to develop (or start developing) your test strategy. When I'm faced with a new project and I'm not sure where to start, I normally pull out the Satisfice Heuristic Test Strategy Model and start there. Each sec-

Feeling the Agile Rush?

Test Faster With TestComplete 7

The Easiest TestComplete Ever!



Agile development moves fast, too fast for manual testing alone. In an agile environment you need automated testing tools to succeed. Check out TestComplete from AutomatedQA. TestComplete lets anyone automate tests for the widest range of technologies like Web, Ajax, Windows, .NET, Java, Flash, Flex and Silverlight. TestComplete has won four Jolt awards, been voted Best Testing Tool and yet it's still affordable.

TestComplete's extensive feature set and great price has long made it the choice of expert testers. Now version 7 adds script-free test creation and a simplified user interface so new testers can get productive fast. TestComplete 7's easy automated testing lets your entire QA team test more, test faster and ship on time with confidence.

Script-Free Test Creation
Simple to learn and easy to extend

Download and Customize Extensions
Pre-built and custom add-ons make complex tests point and click easy

Unmatched Technology Support
New technology? No problem! Rapid support for the latest software

Price and Performance Leader
Unsurpassed features and a low price to make you smile.

Start Testing In Minutes!
Download TestComplete now and start testing today.

Download TestComplete FREE
www.testcomplete.com

AutomatedQA
(978) 236-7900

↘ EDITOR'S LETTER

↘ CHAPTER 1 A SOA VIEW OF TESTING

↘ CHAPTER 2 AGILE TESTING STRATEGIES: EVOLVING WITH THE PRODUCT

tion of the Satisfice Heuristic Test Strategy Model contains things you'll need to consider as you think about your testing:

- **Test techniques:** What types of testing techniques will you be doing? What would you need to support those types of testing in terms of people, processes, tools, environments or data? Are there specific things you need to test (like security, transformation and mapping, or load balancing) that may require specialized techniques?

- **Product elements:** What product elements will you be covering in your testing? What's your scope? To what extent will different data elements be covered in each stage? Which business rules? Will you be testing the target implementation environment? How much testing will you be doing around timing issues? How will you measure your coverage against those elements, track it and manage it from a documentation and configuration management perspective?

- **Quality criteria:** What types of risks will you be looking for while testing? Will you focus more on the business problem being solved or the risks related to the implementation model or technology being used? Will you look at performance at each level? What about security? How will you need to build environments, integrate with service providers, or find tools to do all the different types of testing you'll need to do?

- **Project environment:** What factors will be critical to your success and the success of the in-house team as you take over this work? How will you take in new work, structure your release schedules, or move code between teams, phases or environments? What are the business drivers to moving to a SOA and how will those come into play while you're testing?

Recognize that as you think of these questions, it's a matrix of concerns. A decision (or lack of decision) in each of these categories affects the scope of the decisions in the other three. Therefore, you will most likely find yourself approaching the problem from different perspectives at different times.

Once you have an understanding of what features you'll be testing and you know which quality criteria you'll be focused on, you're ready to dig in. Don't be surprised if in many ways your testing looks like it does on a non-SOA project. SOA isn't magic, and it doesn't change things all that much. However, I've found that there are some subtle differences in where my focus is when I'm working on a SOA project.

COVER THE BASICS AS SOON (AND AS OFTEN) AS YOU CAN

The first area to focus on is typically connectivity. Establishing a successful round-trip test is a big step early in a SOA implementation. The first time connecting is often the most difficult.

EDITOR'S LETTER

CHAPTER 1 A SOA VIEW OF TESTING

CHAPTER 2 AGILE TESTING STRATEGIES: EVOLVING WITH THE PRODUCT

Once you have that connectivity, don't lose it. Build out regression tests for basic connectivity that you can run on a regular basis. While a SOA evolves, it is easy to break connectivity without knowing it. If you break it, you'll want to find out as soon as possible so it's easier to debug any issues.

As data moves through a SOA it's translated, transformed, reformatted and processed. That means you'll have a lot of tests focused on testing data.

Next you may want to look at basic capacity. Capacity at this level can be anything from basic storage to connection pooling to service throughput. I've been on a couple of projects where these elements weren't looked at until the end, only to find that we had to requisition new hardware at the last minute or end up spending days changing configuration settings on network devices and Web servers. Figure out what you think you'll need early on, and run tests to confirm those numbers are accurate and consistent throughout the project.

In addition, don't put off testing for authorization and authentication until you get into the target environment. I've worked on several teams where development and test environments had different security controls than production, which led to confusion and rework once we deployed and tried to run using an untested permissions scheme. If you use LDAP in production, use it when testing. If you're handling authorization in your SOAP request, do it consistently through your testing, even if you think it's easier to disable it while doing "functional" testing.

FOCUS ON THE DATA

Service-oriented architectures aren't just cleaner interfaces between systems, they are also complex data models. As data moves through a SOA it's translated, transformed, reformatted and processed. That means you'll have a lot of tests focused on testing data. These tests typically make up the bulk of the regression test beds I've seen on projects implementing a SOA.

Tests focused around testing data in an SOA typically happen at the component level and leverage stubs and harnesses to allow for testing as close as possible to where the change takes place. This simplifies debugging and allows for more granular tracking of test coverage (based on a schema, mapping document or set of business rules). SOA teams are constantly thinking about regression testing for these tests. Small changes can have a large impact on the data, so these tests become critical to refactoring or when adding new features.

For many teams, these tests can

↘ EDITOR'S LETTER

↘ CHAPTER 1 A SOA VIEW OF TESTING

↘ CHAPTER 2 AGILE TESTING STRATEGIES: EVOLVING WITH THE PRODUCT

be part of the continuous integration process. While building these regression test beds can be very easy, once you have libraries of XML test cases lying around, maintaining them can be very time consuming. That makes a team's unit testing strategy a critical part of the overall testing strategy. What will be tested at the unit level, component level, integration level and end-to-end? Starting as close to the code as possible, automating there, and running those tests as often as possible is often the best choice for data-focused testing.

UNDERSTAND YOUR USAGE MODELS

When I test a SOA, I develop detailed usage models. I originally used usage models solely when doing performance testing, but I've also found them helpful when testing SOAs. When I build usage models, I use the user community modeling language (UCML) developed by Scott Barber. UCML allows you to visually depict complex workloads and performance scenarios. I've used UCML diagrams to help me plan my SOA testing, to document the tests I executed, and to help elicit various performance, security and capacity requirements. UCML isn't the only way you can do that; if you have another modeling technique you prefer, use that one.

Usage models allow us to create realistic (or reasonably realistic) testing scenarios. The power behind a modeling approach like this is that it's intuitive to developers, users, managers and testers alike. That means faster communication, clearer requirements and better tests. At some level, every SOA implementation needs to think about performance, availability, reliability and capacity. Usage models help focus the testing dialogue on those topics (as opposed to just focusing on functionality and data).

When testing for SOA performance, I focus on speed, load, concurrency and latency. My availability and reliability scenarios for SOA often focus on failover, application and infrastructure stress, fault tolerance and operations scenarios. And conversations around capacity often begin with a basic scaling strategy and spiral out from there (taking in hardware, software, licensing and network concerns as appropriate). Because the scope is so broad, effective usage modeling requires balancing creativity and reality.

DEMONSTRATING YOU CAN IMPLEMENT BUSINESS PROCESSES

At the end of your SOA implementation, you will have delivered something that implements or supports a business process. If the bulk of your testing up to that point has focused on various combinations of service-level processes and data functions, then your end-to-end acceptance tests will focus on those final implemented business processes. These tests are often manual, utilize the end systems (often user interface sys-

↘ EDITOR'S LETTER

↘ **CHAPTER 1** A SOA VIEW OF TESTING

↘ **CHAPTER 2** AGILE TESTING STRATEGIES: EVOLVING WITH THE PRODUCT

tems), and are slow and costly. All the testing outlined up to this point has been focused on making this testing more efficient and effective.

Make sure you're selecting tools to fit the types of testing you want to do. Many tools will do more than you need, or may force you to do something in a way that's not optimal for your project.

If you've successfully covered the basics, early and often, then you should have few integration surprises while doing your end-to-end testing. When you deploy the end-to-end solution, you don't want to be debugging basic connectivity or authorization/authentication issues. If you've successfully covered the data at the unit and service level, then you won't have thousands of tests focused on subtle variations of similar data. Instead, you'll have tests targeted at high-risk transactions, or tests based on sampling strategies of what you've already tested. If you've done a good job of facilitating discussions around performance, availability, reliability and capacity, then you likely already reduced most of the risk around those quality criteria and have clear plans for migration to production.

If you're testing your first SOA implementation, take some time to learn more about the underlying technologies and implementation models. Once you're comfortable with what's being done, start looking at the tools that are available to support your testing needs. There are many great commercial and open source tools available, and there is always the option to create your own tools as needed. Just make sure you're selecting tools to fit the types of testing you want to do. Many tools will do more than you need, or may force you to do something in a way that's not optimal for your project.

Finally, remember that whatever you're building now will change over time. Make sure you're working closely with the rest of the team to ensure regression capabilities have been implemented in the most effective and cost-effective ways possible. When you come back a year later to make a small change, you don't want to have to re-create all the work you've done the first time around. ■

ABOUT THE AUTHOR:

MICHAEL KELLY is currently the director of application development for Interactions. He also writes and speaks about topics in software testing. Kelly is a board member for the Association for Software Testing and a co-founder of the Indianapolis Workshops on Software Testing, a series of ongoing meetings on topics in software testing. You can find most of his articles and blog on his website, www.MichaelDKelly.com.

Outsource Your Software Development?

Insource Your Quality!

Making the decision to outsource the development of your critical application is not easy. There are pros and cons, but, once the decision is made, it is imperative that the **quality you paid for** and expected is the **quality you receive**.

Ask yourself these questions:

1. **What is the quality of the code that has been developed?**
2. **Is it too complex, making it unmaintainable and unreliable?**
3. **How thoroughly was it tested prior to delivery back to you?**
4. **Are you convinced that the most complex areas of your critical application are being tested?**
5. **Is the code quality and test coverage trending in a positive direction?**

If you do outsource, you owe it to yourself, your organization, and most importantly your customers to know the answers to these questions.

McCabe IQ provides those answers using advanced static and dynamic analysis technology to help you focus your attention on the most complex and risky areas of your code base.

McCabe IQ's breakthrough visualization techniques, enterprise reporting engine, and executive dashboard provide you with a complete picture of what is being produced.

It's time you made the most of your outsourcing investment and benefitted from our 30+ years of software quality research and development.

Don't just think your code is good.
Be sure of it, with McCabe IQ.

Download "*Code Quality Metrics in Management of Outsourced Development*" at www.mccabe.com/techtarget.

McCabe IQ
Your Software…Better™

McCabe SOFTWARE

www.mccabe.com / 800-638-6316

CHAPTER 2

Agile testing strategies: Evolving with the product

Software testing in an agile environment is the art of seizing iterations as a chance to learn and evolve test ideas.

BY KAREN JOHNSON

→ **YOU CAN ACHIEVE** successful testing in agile if you seize upon iterations as a chance to learn and evolve test ideas. This is the art of software testing strategy in agile. The challenge: How can testing be planned and executed in the ever-evolving world of the iterative agile methodology? This article discusses some of philosophies, pros, cons and realities of software testing in the agile iterative process.

One aspect of an agile software development project that I believe benefits testers is working with iterations. Iterations provide a preview opportunity to see new features and a chance to watch features evolve. The early iterations are such sweet times to see functionality flourish. I feel as though I'm able to watch the process, to see the growth and the maturity of a feature and a product. I feel more a part of the process of development with an iterative approach to building than I do when I'm forced to wait through a full waterfall lifecycle to see an allegedly more polished version of an application.

But there is an opposing point of view that sees the iterative approach as frustrating. Early iterations could be viewed as a hassle; the fact that features will evolve and morph is potentially maddening if a tester thinks their test planning could become obsolete. What if the features morph in such a way as to crush the test planning and even crush the spirit of a waterfall-oriented tester? I

Iterations provide a preview opportunity to see new features and a chance to watch features evolve.

↘ EDITOR'S LETTER

↘ **CHAPTER 1** A SOA VIEW OF TESTING

↘ **CHAPTER 2** AGILE TESTING STRATEGIES: EVOLVING WITH THE PRODUCT

think it's a matter of making a mind shift.

Early iterations of functionality provide time to learn and explore. Learning is opportunity. While I'm learning the product, I'm devising more ideas of how to test and what to test. Cycles of continuous integration are chances to pick up speed with manual testing and fine-tune test automation.

The iterative process also gives me time to build test data. I'm able to take advantage of the time. I also see my learning grow in parallel to the maturity of a feature. As I'm learning, and the functionality is settling down, I can't help but feel that both the product as a whole and I as a tester are becoming more mature, robust and effective. I'm learning and devising more ideas of how to test and what to test and potentially building test data to test. We're growing together.

Cem Kaner outlined the approach of investigatory testing, and Scott Ambler discussed it in an article called "Agile Testing Strategies". Having been through a couple of agile projects before reading Scott's article, it was interesting to find myself nodding in agreement throughout. One of Kaner's ideas on investigative testing that I especially like and have experienced is finding both "big picture" and "little picture" issues.

I enjoy the early iterations to stomp about an application so that I can understand it. I like seeing my knowledge escalate to the point where I'm asking good, solid questions that help the development process move along. I especially enjoy asking development a question and receiving a pensive look from a developer, telling me I've just found a bug in code that I've not even seen or that hasn't even been built—talk about finding your bugs upstream. Frankly,

I've long loathed the idea that a fat stack of requirements could give me everything I needed.

there's also a greater chance that my own ideas will be incorporated, since agile gives me the opportunity to provide feedback about features early enough that my ideas can actually make it into a product. This is far more likely with agile than if the product was fully baked before I had my first whiff.

I've long loathed the idea that a fat stack of requirements could give me everything I needed. "Here you go," I've been told. "Now you can write those test cases you testers build, because if you only test to requirements, the requirements and specifications are all you need." I want to holler, "No, it isn't so!" An hour at the keyboard has more value to me than a day reading specifications. I want time with an application, I want time to play, and I want time to learn. And the agile iterative process gives me that.

↘ EDITOR'S LETTER

↘ CHAPTER 1 A SOA VIEW OF TESTING

↘ CHAPTER 2 AGILE TESTING STRATEGIES: EVOLVING WITH THE PRODUCT

PUTTING STRATEGY INTO PRACTICE: OUTLINE TEST IDEAS

Review the user stories created for upcoming features. Read each story closely, envisioning the functionality. Write questions alongside the user stories; if possible, record your questions on a wiki where your comments and the user stories can be viewed by the team. Sharing your early thoughts and questions on a team wiki gives the entire team a chance to see the questions being asked. Raising more questions and conversations helps the entire team's product knowledge evolve. Also record your test ideas. This gives the developers a chance to see the test ideas that you have. Miscommunications have an early chance for clarification.

When I write my early test ideas, they're usually formatted as a bulleted list—nothing glamorous. I raise my questions with the team and the developers on team calls or at times I arrange for that purpose. I openly share my test ideas and hunches at any possible time that I can. I tell every developer I work with that my work is not and does not need to be mysterious to them. I share as much as anyone is willing to listen to or review as early and as often as I can. To be honest, often the developers are busy and ask for less than I would prefer, but that's OK.

I sometimes mix how I record my test ideas, but the recording isn't the point. The idea is to be constantly thinking of new and inventive ways to find a break in the product. Find a way to record test ideas. Be prepared to record an idea at any time by having a notebook, index cards, a voice recorder—whatever the medium, make the process easy.

GAIN A SENSE OF PRIORITY

After reviewing all of the user stories, develop a sense of priority within each story and then across all the stories. What do you want to test first? What have you been waiting to see? And what test idea is most likely to evoke the greatest issue? These are the first tests.

The theory behind the first-strike attack is the old informal risk analysis. Risk analysis doesn't go away with agile.

When the next iteration comes, I'm not in pause mode. I know where I want to be, I know which tests I want to run. I don't have any long test scripts to rework because of changes. I expect change. As I test these early releases, I take notes. I adjust my ideas as I learn the application. I find that some early test ideas no longer make sense, no longer apply. I generate new ideas as I learn with each iteration.

USE SWEEP TESTING

In 2007, I blogged about my approach to exploratory testing through a concept I call sweep testing. The practice is one of accumulating test ideas, often using Excel. I could say I build test ideas into a central repository, but the term "repository" has such a heavy sound to it, when what I really

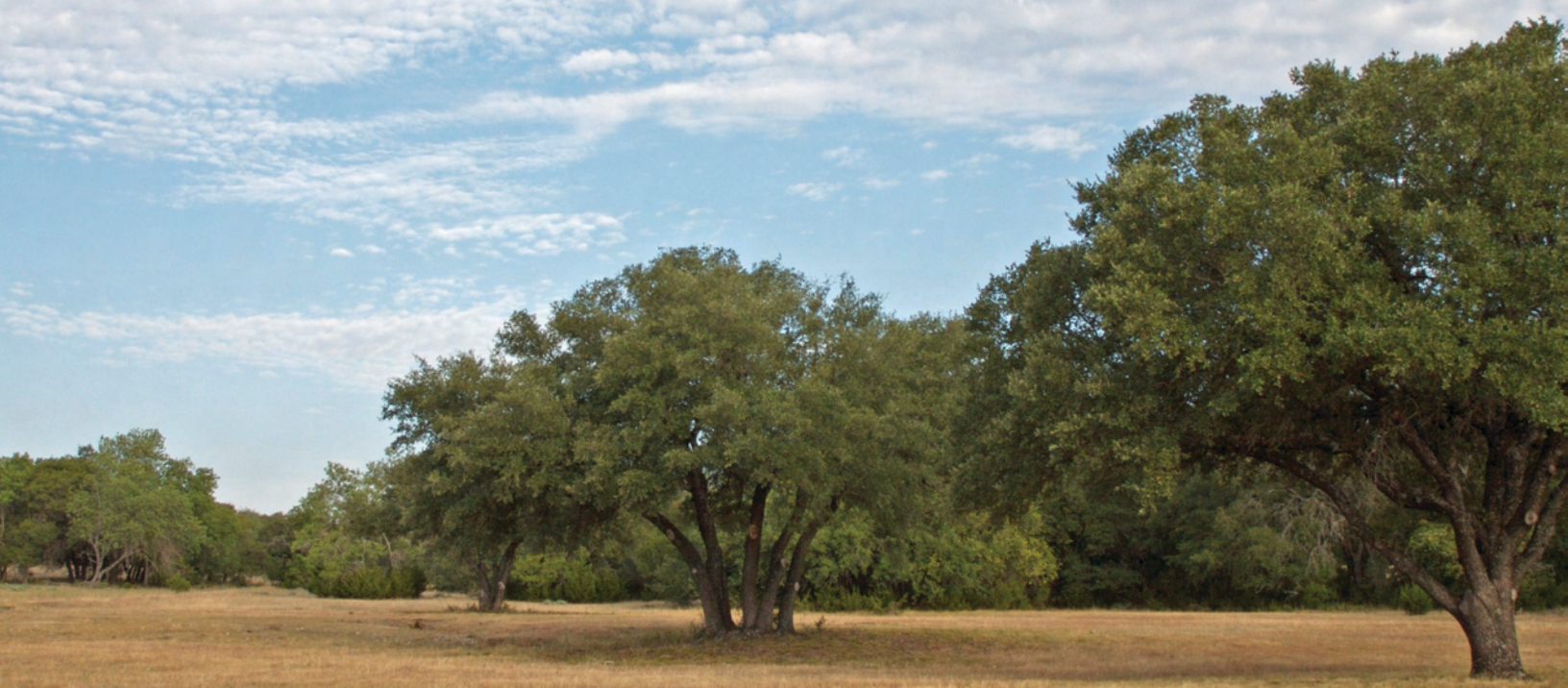
FEELING CONSTRAINED?

iTKO's LISA Removes the Constraints to Software Quality and Agility.

Constant change and growing complexity makes enterprise software testing and delivery more than just difficult... it can be downright impossible without the right tools. Add the challenge of reducing costs and risk under compressed release cycles, and you can quickly feel trapped in a corner.

iTKO's LISA product suite eliminates software test and delivery constraints and puts quality, agility and cost savings back into the lifecycle. And that includes the most complex multi-tier, distributed and heterogeneous technology environments you can imagine. Go ahead... test us.

iTKO LISA™



Find out how LISA can lower your QA costs, reduce risks, and eliminate constraints from software lifecycles by virtualizing IT resources.
Visit http://www.itko.com and sign up for a free Business Case Assessment.

↘ EDITOR'S LETTER

↘ **CHAPTER 1** A SOA VIEW OF TESTING

↘ **CHAPTER 2** AGILE TESTING STRATEGIES: EVOLVING WITH THE PRODUCT

mean is that I take the time to collect those scraps of test ideas and early hunches and pool them together in one place. As the early iterations continue and my test ideas come to me at random, unplanned times, I've jotted down ideas on index cards in my car, as voice recordings on my BlackBerry or as notes on the team wiki.

As the project continues, early test ideas fall by the wayside and new ideas come into place. Taking the time to list my ideas in one place helps me rethink ideas and gain that revised, better informed sense of order and importance. Once I've built my list and my list sweeps across all the functionality, I'm ready. I feel prepared, and I can launch into testing rapidly once the build hits.

I function as the only tester on projects fairly often, so I'm able to use my concept of sweep testing without needing to share, do version control or address the issues that come into play when I'm not testing alone. On projects where I have had other testers, I have used the concept of sweep testing to assign different worksheets or sections of a sheet from the same Excel file and had no issues with the divide-and-conquer approach. I write in bulleted lists. The lists are ideas, not steps or details.

The idea is to be constantly thinking of new and inventive ways to find a break in the product.

Now it's not just the product that has evolved through the iterations—I have evolved as a tester as well.

COLLABORATIVE SPIRIT: PEOPLE WORKING TOGETHER

The shortage of people, money and time, and the twisted situations each of these can present—agile doesn't cure all the blues. So it's best not to be too naïve, to think a process change or a change in approach is going to suddenly resolve all the murky events and behaviors that can happen on a project.

I still believe that the most powerful advantage on any project is people working together who want to work together. This is one of the core principles of the context-driven school of testing.

Bret Pettichord discusses several ways to build a collaborative spirit in his presentation "Agile Testing Practices." The concepts of pairing together and no one developing specialized knowledge are two constructs that I find especially insightful. I just seem to work alone more often than not and have had fewer chances personally to put some of his collaborative test team concepts into practice. But as the solo tester, I can certainly relate to and find his thoughts on testing "half-baked" code highly practical.

A small benefit I've seen in working with scrum has been a side effect of the daily scrum meeting that I hadn't expected. The frequency of the meet-

EDITOR'S LETTER

CHAPTER 1 A SOA VIEW OF TESTING

CHAPTER 2 AGILE TESTING STRATEGIES: EVOLVING WITH THE PRODUCT

ings helps to drive a friendly familiarity. Given the short duration of the meetings, people seem less inclined to open a laptop and ignore each other as they check emails and the daily news. The forced frequency seems to help. A more practical aspect is hearing from team members what activities people have been pulled into, how time has been spent or misspent. People have a place to explain how hours have been burnt over disruptive, unexpected and expensive tasks. These daily meetings bring people together; they build a sense of team and, for that reason alone, add great value.

These daily meetings bring people together; they build a sense of team and, for that reason alone, add great value.

In his "Agile Test Strategies and Experiences" article, Fran O'Hara talks about "independent testers" needing to be integrated with the team. This concept resonates deeply with me; I have long felt it is the team and not a single stakeholder who will know what value I have brought to the project. As an independent consultant, this has been interesting; I often have a stakeholder who owns my contract, but it is the team that knows what I deliver.

CONCLUSION

While some of the test practices in agile may be different than working on a waterfall project, the fundamental ideas in testing still apply. Exploratory and investigative testing produce more bugs that matter in a timelier manner than detailed test scripts can provide. Risk-based thinking helps steer testing to find the defects that matter most. Testing will always be affected by the reality that time at the end game is tight, and the best way to cope with tight time is to be flexible. And people who want to work together can produce great products, so finding ways to build healthy, constructive relationships is likely the most valuable strategy of all. ■

ABOUT THE AUTHOR:

KAREN JOHNSON is an independent software test consultant with 14 years of experience in software testing and software test management. She views software testing as an intellectual challenge and believes in the context-driven school of testing. Karen frequently speaks at software testing conferences and participates in software testing workshops, in addition to authoring articles in software testing publications.

Learn More and Download TestComplete Free

About AutomatedQA: AutomatedQA has been making award-winning software products for quality assurance and software development worldwide since 1999.

AutomatedQA's flagship product is TestComplete, the automated software testing solution that's easy to use and affordable for any size team. TestComplete makes testing fast and simple for all Windows software, including Web, .NET, Java, Windows Desktop, Flash/Flex, Ajax and Silverlight.

TestComplete's easy script-free keyword testing and unified interface enable testers to learn just one product and automate the full range of test types including functional testing, load testing, distributed client/server testing and unit testing for Windows and Web.

EDITOR'S LETTER

CHAPTER 1 A SOA VIEW OF TESTING

CHAPTER 2 AGILE TESTING STRATEGIES: EVOLVING WITH THE PRODUCT

FROM OUR SPONSOR

- **Improved Software Testing Using McCabe IQ Coverage Analysis**
- **More Complex = Less Secure: Miss a Test Path and You Could Get Hacked**
- **Using Code Quality Metrics in Management of Outsourced Development and Maintenance**

About McCabe Software, Inc.: McCabe Software provides Software Quality Management and Software Change & Configuration Management solutions worldwide. "McCabe IQ" is used to analyze and visualize the quality and test coverage of mission, life, and business critical applications, utilizing a comprehensive set of advanced software metrics including the McCabe-authored Cyclomatic Complexity metric. Our configuration management solution, "McCabe CM", utilizes exclusive Integrated Difference technology to manage software changes faster and more efficiently, ensuring quality throughout the Application Lifecycle. McCabe Software has offices in the United States, distribution worldwide, and can be found online at www.mccabe.com.

FROM OUR SPONSOR

- **iTKO Whitepaper: Service Virtualization in Enterprise Application Development**
- **iTKO Whitepaper: Minimize IT Outsourcing Risk with Collaborative Quality**
- **Free Analyst Report from Butler Group: LISA 4.6 Technology Audit**

About iTKO: iTKO helps our customers transform the software development and testing lifecycle for greater quality and agility in an environment of constant change. iTKO's award winning LISA(tm) product suite can dramatically lower quality assurance costs, shorten release cycles, reduce risks, and eliminate critical development and testing constraints by virtualizing IT resources to provide accessibility, capacity and security across interdependent teams. LISA enables test, validation, and virtualization solutions optimized for distributed, multi-tier applications that leverage SOA, BPM, integration suites, and ESBs. iTKO customers include eBay, American Airlines, Allstate, Time Warner, SwissRe, Bank of America and the U.S. Department of Defense. Visit http://www.itko.com.

EDITOR'S LETTER

CHAPTER 1 A SOA VIEW OF TESTING

CHAPTER 2 AGILE TESTING STRATEGIES: EVOLVING WITH THE PRODUCT