Level: Beginner
Works with: QuickPlace 3
Updated: 04-Nov-2002

Behind the scenes of QuickPlace automation testing

Interview by
Tara Hall

This month we speak with Anthony (Tony) Venditti about software automation testing and how it's used to improve a product, in particular QuickPlace. Automation testing can be used with both regression and smoke testing. Regression testing is a form of comprehensive testing that tests each feature and function of a product. Smoke testing uses a subset of the regression tests to test a product. In addition to automation testing, Tony talks to us about smoke testing, regression tests, and how well QuickPlace 3 is doing under the pressure of taking all these tests.

Can you describe to me what you do on the QuickPlace team—what your role is?
I am the QuickPlace test automation architect. I design and develop QuickPlace test automation, which involves designing automation framework and function libraries, writing automated test scripts, configuring test servers, executing test scripts against test servers, reporting any problems flagged in QuickPlace by the automation, and maintaining the script libraries for a product that changes daily.

What is automation testing and how is it different from other forms of testing that you normally perform for a product?
Essentially, software automation testing is using a computer system instead of a human to test a software application. Most other forms of software testing require human interaction with the software product under development. For example, we have QuickPlace product experts who systematically, manually test the product functionality. Our team also uses QuickPlace test servers to collaborate on a daily basis, so we're eating our own dog food. We also test the product by releasing beta versions of the product to our customers, and we do in-house usability testing and a lot of other types of product testing. All of these methods require human interaction with the product. Automation is different; it uses a personal computer to run programs that make the computer itself a virtual tester. It drives the user interface (UI) of a product like an end user would. Automation testing is unattended testing. If an automated script is designed to test a certain feature of the product, it exercises that feature, verifies whether the feature functioned properly or not, and logs the results to a log file—all without any human intervention.

What benefits does automation give you over manual testing?
Automation saves execution time, which results in more time for analysis, design, and development of new tests. Automation allows for easy repeatability of tests, which results in faster test regression cycles. Automation provides increased product coverage and deeper stress testing of product features, which allows manual testers more time to test fringe areas of the product and to do more ad hoc testing.

If I am the test engineer responsible for smoke testing the daily build, I would need to manually execute about 50 test cases every day. Let's say that takes an hour per day. By automating that procedure, we have now saved a test engineer one hour per day—about 250 hours per year to work on other areas of testing—not to mention that it would save the human tester from going absolutely bonkers from manually repeating the same steps every day.

Our test automation provides a way for a computer system to quickly verify that basic product features are working properly. For example, we use automation to install multiple test servers and to smoke test new QuickPlace builds overnight on a daily basis. The automation provides for completely unattended overnight testing. In the morning, a report is automatically generated that lets us know how things went overnight. We know instantly, upon coming into our offices in the morning, if a build passed or failed, and if it failed, why it failed. Automation gives us a sense of confidence in our daily builds. When automation is programmed right, it can save huge amounts of time verifying product functionality and allows for greater product test coverage. Automated testing does not replace the need for manual testing—it complements it.

Because automation testing doesn't require you to be there at the computer, what happens if something goes terribly wrong? How do you know if it's the build that you're testing or a problem with the script that's been executed?
Our automation has built-in error recovery that can trap failures and logging mechanisms that allow us to analyze what failed at a later time. An automation result log displays trace-back information about what passed and what failed. Within minutes, you can usually tell if it is something that needs to be refined in the script or if it is a bug in the product.

You use automation for smoke testing. Do you also use it for the regression testing?
Yes, absolutely. We have 140 functional regression scripts that contain over 1,400 test cases that execute over 40,000 test actions on multiple server configurations on a daily basis. It takes over 24 hours to run the full regression suite on one system. So I divide the full regression suite into four separate suites, which run on four different test client/server configurations. By splitting the regression suite and running it on four clients instead of one, I can turn the entire suite over in five or six hours instead of 24 hours. The suite starts at 5 AM on the four test systems and usually finishes by 11 AM daily. We are also testing multiple server configurations (NT, 2000, AIX, and Solaris) at the same time. Another great thing about automation is that once you automate a feature, it's like putting that feature in a lockbox. If there are any changes to that feature or if the feature ever breaks, we are going to know about it right away. The more functionality and features that you can cover with automation the more robust and stable the product will be as an end result.

Do you test on Linux at all?
No. I think there's been some interest in doing that, but our team is fairly limited. It's a really small team working on

what is turning out to be a very complex product. So, we are stretched thin right now.

When you are determining your scripts for automation, do you involve customers in some way?
Customers are indirectly involved. Several of our scripts are based on Software Problem Reports (SPRs) that were originally customer support issues. But most of the automated scripts are based on manual test plans and product specifications.

When measuring the quality of the product, how do you determine what's acceptable and what's unacceptable? And how do you do that with QuickPlace, a product that's so new—one in which you don't have many years of previous testing?
There are several ways that we measure the quality of the product. In my little automation niche, it's how well the smoke test and regression suites run. Usually at the beginning of the product cycle, there are many crashes and failures that occur during the regression suite, but toward the end of the product cycle, the regression suite usually runs to completion without any failures. That is the way I measure the stability and quality of a build. Our SPR system is another good indicator. When the number of new SPRs gets smaller and smaller, we know that we are getting closer. We also have a Test Plan database that tracks metrics per build. The entire team also uses the product on a daily basis for everyday tasks like calendar and scheduling, build information, messaging, and collaboration.

The test servers that we use are instrumented with mean time between failure (MTBF) statistics. When we can run for several days under a heavy load without downtime, we know the product is getting there. Our performance lab gives us metrics on load testing, performance, and MTBF as well. We also get a lot of input from large companies that use the product, like GE (General Electric), so we get reaction from our customers as well. Those are some of the factors that we use to determine the quality and acceptability of the product.

As you move forward, how will you use those metrics? Will they become the standards by which you measure future releases of QuickPlace or do you start over with every release because the product is constantly evolving?
It is evolving. This release is a very major release with several features that really make it an enterprise release. Going forward, the metrics we collect in previous versions will be applied to future releases. As far as automation is concerned, it really parallels the development environment. As changes are made to the product, modifications will need to be implemented on the automation framework. Automation is a very important resource to maintenance releases because it provides a large foundation of tests that you can quickly run on new releases of the product. It's very important to maintain and to upkeep the automation for future releases and to build upon it.

The automation has accumulated significantly for release 3. We had 734 automated test cases for release 2.0.8. We now have over 1,400 test cases. When new features get implemented, new scripts need to be written. You can always increase coverage and do deeper testing with automation. There's always plenty to do there. It is important to maintain the automation with each release.

In your testing, how do you determine if a problem is caused by the QuickPlace server, the Domino Server, or maybe the client? Can automation reveal if it is a problem with the hardware?
If the problem is reproducible, then it will cause a failure every time the script is executed. When bugs are reproducible, it is quite easy for development to understand where the bug is occurring and how to fix it. It gets a little tricky when server crashes occur that are not reproducible; then we need to rely on crash logging tools. Developers can usually tell by looking at a crash log if the crash was caused by the QuickPlace server or the Domino server.

How do you work together with the Domino development team to resolve those issues when you find that your product and the core product aren't always working together as well as they should?
From the QA perspective, we need to get the information in an SPR, including the crash log; then it goes to the QuickPlace developer. The QuickPlace developer does the initial investigation work to find out where the crash occurred in the code. If the crash occurred in Domino, we will make the necessary updates to the SPR and enter it in the Domino SPR system.

Do you have any recommendations for customers about the types of configurations they might want to consider? Which ones work best for you?
The best server configuration for our customers depends upon each customer's individual needs and how the customer's organization uses QuickPlace. We have customers who use QuickPlace in a variety of roles and in a surprising amount of configurations, so it is difficult to assess what would be the optimal configuration for all customers. Generally speaking, breaking the various service components down so that they are running on dedicated servers provides for the best performance, reliability, and scalability. With regards to directory types,

authentication models, clustering, SSL, or any of the other various configurations that QuickPlace can utilize, again, this is going to depend upon our customer's organizational security policies and existing infrastructure and what implementation of the product best suits their needs.

Can you give me a quick rundown of the servers that you have in your test lab?
There are about 30 servers in our test lab. About a dozen of them are involved in our core test cluster that we use to test QuickPlace. The core production QuickPlace servers our team uses are of various operating systems—a mix of W32 and UNIX platforms—that are load balanced through Network Dispatcher. These systems also use HTTPS; have Sametime awareness, meetings, and chat (provided by a dedicated Sametime 3 Service Pack 1 server in the same domain); use Domino Web Single Sign-On (SSO); use a Domino LDAP server as a user directory (a dedicated server with over 500,000 records); and utilize QuickPlace's DOLS features. We use a cluster of dedicated Place catalog servers and a dedicated Domain Catalog Server to provide the respective features to the service.

Also in our lab, we have several systems configured to test different LDAP directory sources, such as IBM BluePages, Tivoli Secureway Directory, and Netscape iPlanet Directory. We have a dedicated system to test Domino Directories using NRPC. We test portals using Secureway Policy Director, and we also have an iPlanet Portal server for the same purpose.

You use a third-party tool in your testing. Can you talk a little bit about that tool?
The tool that we use to develop automated tests is called SilkTest by Segue Software. There are several automation test tools on the market, but I feel SilkTest is the most powerful test tool available. I've been developing automation ever since I started at Lotus in 1989. I had used several different tools to develop automation. When I started working on Notes back in 1993, I began using SilkTest (known as QA Partner back then) to automate Notes formula testing. SilkTest has a scripting language called 4Test that is very similar to the C programming language. The tool works by recognizing objects in the GUI of an application or Web page and mapping the objects to a window declaration file. After the GUI objects are defined in the mapping process, they can be referenced by the 4Test scripting language and used to drive the application under test. You can then use the 4Test language to build sophisticated function libraries to systematically test an application.

With all this testing that you do nightly, has it slowed down the development process or do you find that it really helps to speed up the development process?
It speeds the development process considerably. Before the nightly automated smoke test process was in place, it took developers a week to master a build and required QA to manually smoke test each build. Now it is built, installed on multiple server configurations, and smoke tested on multiple clients automatically overnight. I believe this automated process has helped us deliver a higher quality product in a shorter amount of time. If the build passes the smoke test process overnight, we deploy the build on all of our test servers in the lab. People can begin testing the latest and greatest build in the morning of the same day it was built! That's quick turnaround time.

Do you have different sets of automation scripts, ones that run daily and ones for a different set of testing, perhaps?
I try to run everything, the entire regression suite—140 scripts/24 hour execution time—on each new daily build. In order to do this in a day, I need to split the full regression suite into four parts and run it on four separate systems. There are some things like stress testing scripts that I will run only occasionally. For example, I have scripts that will stress the product by creating hundreds of rooms in a QuickPlace or by continuously adding members until it can't add any more.

Is stress testing like load testing?
The stress tests I referred to are not like the load testing you're thinking of. The performance and load testing you're thinking of is done for QuickPlace by the Product Introduction Engineering team. My stress scripts perform limits testing like how many pages can be added or how many columns can be displayed in a folder view.

How is testing used to set the criteria for a beta candidate? How do you determine that you've got a build that is stable enough to start giving it to customers?
As I mentioned earlier, there are several ways that we determine a build to be of ship quality. Things like automated regression results, test database metrics, SPR metrics, performance statistics, and MTBF. With a beta build, the same sorts of metrics are used, but the bar is a little lower. For me, the first indicator of quality and stability is the smoke test results. After that, it is the regression suite and how well it runs.

What is the point at which you stop accepting daily builds to prepare for a gold build?
Usually a month or so before a ship date, we will stop daily builds and focus on one gold candidate build. Very few

changes are made to a gold candidate. Only critical bugs are fixed at this point. There is a triage team that goes over each new bug and determines whether it needs to be fixed now or if we can wait until a future release to fix it.

And does automation testing continue with the gold candidate build?
The daily build process automation stops with the first gold candidate. At this point all of our testing is focused on this one build—the gold candidate. All of the automation is executed against this one build using every different test server configuration possible—different portals, different directories, different platforms, and all of the combinations in between. We really pound on the gold candidate to make sure it is good to go.

So, overall, how is QuickPlace 3 holding up against previous releases?
I think our customers will be thrilled with QuickPlace 3. It has many new features, it is faster, more stable; and it is far more "enterprise-ready" than earlier releases.

ABOUT TONY VENDITTI
Tony is a QA Automation Architect for the QuickPlace team. He joined Lotus in 1989 and has developed test automation for several Lotus products. In 1996, Tony left Lotus to design automation infrastructures for Fidelity Investments in Boston, MA and Avid Technology in Tewksbury, MA. He returned to Lotus/IBM in 1998 to join the QuickPlace team. Tony lives in Westborough, MA with his wife and four children. He enjoys playing ice hockey and coaching his kids' ice hockey team.