Neotys

Load Testing at the Speed of Agile

A Neotys Whitepaper

TABLE OF CONTENTS

INTRODUCTION

Let's face it: Agile is a fact of life. Perhaps you're not a "full Agile" shop and maybe you're not doing Continuous Integration or even talking about DevOps yet, but the reality is that pressures are increasing to realize many of the benefits like quality and speed that are inherent in Agile or "Agile-like" development methodologies.

When you start becoming more Agile, developers churn out code at a rapid pace to try to get as many of their user stories or tasks to "done" as they can before the end of the sprint, and many testers struggle to keep up with this pace. Furthermore, testers in Agile teams often have responsibility for automated testing, unit testing, regression testing, as well as load and performance testing. In this environment, you need to be able to keep up with the speed of development while also meeting heightened expectations of quality.

While the benefits of Agile development are well known (including faster time to market, adapting to changing requirements, constant feedback loop, etc.) the benefits of load and performance testing are similarly well known (including determining how much load an application can handle before it crashes in production, when to add another server, when to reconfigure the network, where code needs to be optimized, etc.). What is less well known is the fact that the combination of the two practices can lead to additional benefits that go beyond just the sum of the benefits of each practice (i.e. 2+2=5).

BENEFITS OF CONTINUOUS LOAD AND PERFORMANCE TESTING

Avoid Late Performance Problem Discovery

When load and performance testing are pushed off until the end of a development cycle, there is often little to no time for developers to make changes. This can cause teams to push back release dates and delay getting features out the door that customers need. Alternatively, if the issues are minor, teams may decide to proceed and launch the application into production while accepting the heightened risks. If the performance problems are more fundamental, they could even require painful architectural changes that could take weeks or months to implement.

Make Changes Earlier When They Are Cheaper

By including load and performance testing in Continuous Integration testing processes, organizations can catch performance issues early before they get much more costly to fix. Developers can instantly know that the new feature in the latest build has caused the application to no longer meet Service Level Agreements (SLAs). They can fix the problem then and there before it becomes exponentially more expensive. This is especially true on Agile teams when discovering a performance problem weeks later could mean that it actually occurred several builds ago which makes the task of pinpointing the root cause a nightmare.

Guarantee Users Get New Features, Not New Performance Issues

In some Agile organizations, changes are happening incredibly fast. It's possible for a new feature or some new functionality to get checked into source control, run through a Continuous Integration build, pass all of the automated tests, and get deployed to the production server in a matter of minutes. But if that code wasn't optimized to handle the number of simultaneous users seen at the worst peak times, it could cause the whole system to crash. Integrating load testing into the process before these changes are deployed to production can ensure that your users get all the goodies they want without the bad user experiences. This can save your company thousand or even millions in lost revenue from users switching to competitors' apps or bashing your brand because of the problems they experienced with your app.

CHALLENGES OF LOAD TESTING IN AN AGILE ENVIRONMENT

In the same way that combining Agile with load testing can provide unique benefits, it can also present your teams with unique challenges they may not have experienced in the past.

Shorter Development Cycles Require More Tests in Less Time

Load and performance testing are usually pushed off until the end of a development cycle. With Agile, development, cycles are much shorter, and load & performance testing can get pushed off until the last day of a sprint or sometimes it's done every other sprint. This can often result in code being released without being adequately tested or user stories slipping to the next release once they have been tested. Conceptually the solution is to do the testing earlier in the development cycle, but that's easier said than done with many teams lacking the resources and tools to make it happen.

"Working" Code Does Not Always Perform Well

So much focus for developers on Agile teams is put on delivering "working" code, but is code really "working" if it fails when the application is under load? Should user stories and tasks really be marked as "done" if the code associated with them causes the application to crash with 100 users? What about 1,000? 100,000? The pressure to get code out the door is high, but so is the cost of having an application crash in production.

Developers Need Feedback NOW

Agile developers need to know more than just the fact that their code is causing performance issues: they need to know when their code started causing problems and what story they were working on when the issue started. It's a huge pain for developers to be forced to go back and fix code for a story they worked on weeks or months ago. It also means they can't spend time working on getting new features out the door. Detecting performance issues early in the cycle so you can deliver important feedback to developers quickly is crucial to saving costs.

Automating the Handoff from Dev to Ops Can Feel Risky

While DevOps and Continuous Deployment are still fairly young practices, the fear felt by operations teams that new changes in code will slow down or even crash the application when it is deployed in production has been around forever. Automating some of the testing in the Continuous Integration process can help to ease some of this fear, but without adequate performance testing included, the risk is still real. Ops teams know well the huge impact application downtime can have on the business.

BEST PRACTICES

The following best practices can help you maximize the advantages while helping you overcome the challenges— of load testing in an Agile environment.

Put Performance SLAs on the Task Board

Every application has minimum performance service level agreements (SLAs) to meet, but Agile teams are often more focused on adding features and functionality to an application than optimizing the application's performance. User stories are typically written from a functional perspective (e.g. "As a user, I can click the 'View Cart' button and view the 'My Cart' page.") without specification of application performance requirements (e.g. "As a user, I can click the 'View Cart' button and view the 'My Cart' page in less than 1 second when there are up to 1,000 other users on the site."). This may not be the best way to write a user story, but it illustrates the point: Performance needs to be somewhere on the task board if the team is going to give it attention.

One way to get performance on the board is to use your SLAs as acceptance tests for each story so that a story cannot make it to "Done" if the changes for that story cause the application to miss those SLAs (this may require new SLAs to be defined for new features and new apps, e.g. all searches for a new search feature must return results in less than 2 seconds). This approach works well when the changes made for the story affect a relatively small section of the codebase () and performance issues would therefore be confined to a small section of the application.

For SLAs that are general across the entire application (e.g. all pages must load in less than 1 second), tests should be added to a larger list of constraints (which may include functional tests) that get tested for every story in order to determine that story meets minimal "definition of done" without breaking any of these constraints.

Work Closely with Developers to Anticipate Changes

One of the benefits for testers working in an Agile environment is that they typically learn about updates on development tasks during daily standups or similar meetings. In order to get the maximum benefit from this level of collaboration, testers should constantly be thinking about how the stories that are currently being coded will be tested. Will these require completely new load tests? Will they cause errors in current test scripts? Can you get away with slight modifications to current test scripts if you plan ahead? Most of the time, these are small changes, so testers can stay ahead of the curve if they keep engaged with the team.

Integrate with Build Server

Even if you aren't completely on the Agile bandwagon yet, you probably have a build server that kicks off some automated tests, unit tests, smoke tests, regression tests, etc. In the same way that performance goals need to be added to the task board, performance tests should be among the tests that occur with every build. This can be as simple as setting up a trigger to have the build server kick off the test, but could include displaying test results within the build tool depending on how sophisticated the integration is. Ideally, you want the person who kicked off the build to instantly see the results and know which changes went into that build so they can be fixed if there is a performance issue.

CI + Nightly Build + End of Sprint Load Testing

The difference between Continuous Integration builds, nightly builds, and the builds produced at the end of sprints can be huge. We're talking the difference between a single change committed to a version control server versus all the changes committed in a day versus all the changes committed during a sprint. With this in mind, you should adjust your load tests to the type of build you're running.

The best practice here is to start small and internal. For CI builds that are getting kicked off every time someone commits a change, you want these tests to run quickly so that you can get results back to that developer about how his/her changes affected the system. Consider running a small performance test with the most common scenarios covered with the typical load on your application being produced from your own internal load generators. For nightly builds, ramp it up to include more of the corner case scenarios and increase the load to what you see at peak times to see if any performance issues were missed during the CI tests. At the end of the sprint, you'll want to go all out: consider generating load from the cloud to see what happens when users access your app through the firewall. Make sure every SLA on the constraints list is passed so that every story completed during the sprint can be marked as "done."

HOW TO CHOOSE AN AGILE LOAD TESTING SOLUTION

While any load testing solution will enable you to do some sort of load testing in your Agile environment, comparatively few enable you to follow all of the best practices outlined here and keep up with the pace of your Agile development teams.

When considering an Agile load testing solution, ask the following questions:

1. How quickly can tests be designed in the tool by any tester or developer (not just your performance testing gurus)?
2. Will my teams be able to share test scenarios and test results?
3. To what extent does the tool integrate with Continuous Integration Servers to automate the testing process?
4. Will everyone on my teams be able to create and understand the test reports and be able to act on them?
5. Will test conditions be able to replicate what happens in production?
6. Does the solution support the technologies we used to build the application and the technologies we will plan to use in the future?

Simple Test Design with Automation

Agile teams rarely have a dedicated performance engineer who show up on demand to script up a new test when changes to an application causes errors in the old test. You may have one or two testers on your team, but they're rarely performance experts. In some cases, you may have developers writing their own tests. In any of these scenarios, these team members are going to be pressed for time, and this is no place for a tool that is difficult to use and configure.

In developing and executing performance tests several key features go a long way in improving test productivity, including support for:

- Easily launching the recording of a virtual user profile (preferably in one click)
- Defining advanced behaviors (with structures such as conditions and loops) via a graphical interface
- Automatic handling of dynamic parameters. This includes a set of correlation rules for well-known server frameworks. Ideally, the solution will dynamically detect and handle custom parameters specific to your application
- Sharing common script parts, such as login or logout transactions, between multiple virtual user profiles

This is not an exhaustive list of usability features that can help team members work more efficiently; rather it should be considered as a baseline of minimum required capabilities for an efficient load testing solution.

Collaboration Capabilities

Working on an Agile team is all about collaborating, and you'll likely want to share things like virtual user profiles, groups, monitoring configurations, load profiles, and test results with team members. You should look at load testing solutions that allow several team members to be involved in the creation and analysis of test cases and enable testers to quickly share results with developers to help them understand what went wrong and what needs to be fixed.

Integration with Continuous Integration Servers

You should consider a load testing solution that integrates with Continuous Integration servers so that you can both automate the triggering of a performance test with every build and also quickly see the results of those tests along with the other tests kicked off during the build to provide instant feedback to development. The ability to the see performance

trends over several builds is crucial for performance regression testing. Testers need to be able to pinpoint exactly which build and code changes started a trend of performance degradation.

Automating the interaction between the CIS and the load testing is crucial to ensuring that performance becomes a regular goal for every iteration. When load tests are only done manually on Agile teams, it becomes easy for them to become tasks that are put off until the next sprint.

Reports that are Easy to Build and Understand

Because not every member of your team is a performance guru, you should look for load testing solutions that provide results that are easy to create, understand, and act on. This includes reports developers can interpret to make important code changes, ops teams can use to change infrastructure configurations, and management can read to get a view of overall performance metrics. Functionality like drag-and-drop metrics and filtering will significantly reduce the time needed to build out reports. Because reporting needs change, it is a good idea to keep your options open with a tool that supports multiple formats, including PDF, Word, HTML, and XML for integration with other systems.

Realistic Tests

Most web applications and mobile applications these days are going to be used by a number of users in many locations on different devices with multiple browsers over various network conditions. You should look for a load testing solution that can do the following:

- **Emulate real world network conditions.** When choosing a load testing solution, look for one that provides WAN emulation that can limit bandwidth and simulate latency and packet loss to ensure that the virtual users download the content of the web application at a realistic rate. This capability is particularly important when testing mobile applications, because mobile devices typically operate with less bandwidth than laptops and desktops and are greatly affected by changes in latency and packet loss especially when signal strength is weak.
- **Device and browser support.** Similarly, look for a solution that can record from any browser or mobile device and simulate those back during the load tests. Simulating devices is important because you need the appropriate server content and number of parallel connections for realistic response times and server load. Additionally, modern browsers have the ability to parallelize HTTP requests as they retrieve a web page's static resources. These parallel requests require more connections with the server and can lengthen response times. Load testing solutions that do not parallelize requests are incapable of producing truly realistic performance tests for web applications.
- **Test from outside the firewall.** The overwhelming majority of web and mobile apps are accessed by users from outside the firewall. While it is absolutely crucial to be able to test internally in an Agile environment, in order to truly understand how location will affect the performance for your users, you need to look at a solution that is also capable of generating load from cloud servers around the world. You might also consider a solution that offers the ability to generate load from multiple cloud providers to reduce the risk of relying upon a single source of cloud load generation.

Support for the Latest Technologies

To test applications built with Adobe Flex, Microsoft Silverlight, Real-Time Messaging Protocol (RTMP), AJAX Push, HTML5 or WebSocket technologies, you need a load testing tool with built-in support for the technologies you're using. Without this specialized support it can be very difficult, if not impossible, to effectively test the performance of your applications.

This can be especially problematic for Agile teams that are adding new features and functionality at a rapid pace since developers often want to want to use the newest technologies available to them. One minute your developers could be experimenting with Google SPDY and another minute they are adding streaming video to an app; you need to know that your load testing solution will be able to test how these changes affect performance.

CONCLUSION

Agile development and Continuous Testing are increasing the productivity of teams and the quality of applications. When adding load and performance testing into those processes, careful planning should occur to ensure that performance is a priority in every iteration. The goal of load testing at the speed of Agile is to deliver as much value to the users of an application through evolving features and functionality while ensuring performance no matter how many users are on the app at any one time.

To ensure you are getting the most value from combining Agile methodologies and load testing, you need to:

- Make sure performance SLAs are on your task board or constraints list to guarantee the code associated with a task performs well before that task is marked "Done"

- Collaborate with developers to anticipate when new code changes require changes in performance test scenarios
- Have performance tests automatically kicked off with every new build and track performance trends from build to build
- Ramp up complexity and load for tests with the size of the build to keep build times down (CI => performance smoke test, Nightly Build => full load test, End of Sprint => stress test with cloud load generators)

In selecting a load testing solution for Agile development, you want a tool that can adapt to the needs of your team and your application. The solution should be able to:

- Allow both novice and expert performance testers to design and edit testing scenarios quickly
- Enable everyone on the team to share test scenarios and test results
- Fully integrate with Continuous Integration Servers to both kick off tests and display result trends over several builds for regression testing
- Produce reports that everyone on the team can understand and act on
- Run tests that resemble the real world by accounting for users' devices, browsers, network conditions, and geographic locations
- Support protocols and other technologies you are using to build your apps now and in the future

About Neotys | www.neotys.com

Neotys is the leading innovator in Continuous Performance Validation for Web and Mobile applications. Neotys load testing (NeoLoad) and performance monitoring (NeoSense) enable teams to produce faster applications, deliver new features and enhancements in less time and simplify interactions across Dev, QA, Ops and business stakeholders. Neotys has helped over 1600 customers test, monitor and improve performance at every stage of the application development lifecycle, from development to production, leveraging its automated and collaborative tooling. For more information about Neotys, NeoLoad and NeoSense visit: www.neotys.com or contact sales@neotys.com

Contact for More Info

US: Tel: +1 781 899 7200

EMEA: Tel: +33 442 180 830

Email: sales@neotys.com

Learn More: www.neotys.com