

Username login register

Search...

Local
 Web
 Videos

[HOME](#)
[ARTICLES](#)
[EVENTS](#)
[BLOGS](#)
[COMMUNITY](#)
[FORUMS](#)
[MEDIA CENTER](#)
[WHITEPAPERS](#)
[RESOURCES](#)
[AGILE NEWS](#)
[JOBS](#)

Home > Articles > Columns > Articles > Integrating Performance Testing into the Agile Development Process

Integrating Performance Testing into the Agile Development Process

1
tweet

retweet

WRITTEN BY GRAHAM PARSONS
TUESDAY, 07 SEPTEMBER 2010 11:25



One of the key purposes and benefits of Agile development processes ("Agile") is that they emphasize working software as the primary measure of progress. To ensure that the current state of the software is working, every iteration includes unit and acceptance testing.

Many organizations report that, following the adoption of Agile, the quality of their released systems has been improved.

When talking about 'quality' in this respect, the focus is solely on functionality; non-functional quality, and particularly the ability of the application to perform and scale at times of peak usage, is excluded from testing that occurs within iterations.

The main reason for excluding this type of testing is that performance tests have traditionally required weeks to set-up and hence the popular belief is that there is no time for performance testing within an Agile project's iterations.

Why is this a problem?

Let us presume that an Agile project's iterations last 3 weeks and needs 10 iterations before enough software has been developed and signed off to form a release to the business/client.

Following the commonly believed approach that the only time to performance test applications delivered by Agile projects is just before the software is released, that would mean that the performance testing would occur 30 weeks after the first line of code was written. In addition, using traditional performance test tools, the performance test project itself could easily take 4 weeks to deliver.

So, 34 weeks after the first line of code was written, and therefore potentially the first 'performance defect' implemented into the code base, the problems caused by the defect would be detected. Of course it is highly unlikely that the first day of coding would implement problems, but it is definitely feasible that defects could be implemented in the first 3 or 4 iterations; many weeks, and potentially months, before the first round of performance testing delivers the 'bad news'.

The obvious risk with the above, and an all too common approach, is that performance defects discovered towards the end of the project, long after they were implemented, can significantly delay the actual project delivery. Identifying and fixing the problems will often require significant amounts of unscheduled effort.

If the software's purpose was to reduce the business' costs or drive more revenue, problems discovered at such a late stage will delay the business receiving the benefits the software was meant to deliver.

Some of you may be thinking that it's not your fault, as performance testing is not possible during the main part of an Agile project. This statement is correct when referring to traditional performance testing tools that require extensive tool experience, and a large amount of time and effort to write and maintain test scripts.

So, the widely accepted conclusion is that Agile projects have to live with the fact that they run the risk of finding major performance or scalability bottlenecks late in the project and, if performance issues are identified, project delivery timescales and budgets will slip, possibly significantly.

A solution exists

Within the last year or so a new breed of performance testing tool has appeared and has quickly gained significant numbers of adopters. The new tools are innovative, easy to use and reduce performance testing timescales from weeks to days.

Adoption of one of these tools into the Agile development process allows performance testing to be treated with the same importance as unit and acceptance testing, and integrated into every iteration of the project. By doing this, projects ensure that at the end of the development, any performance problems detected will be due to infrastructure software configuration or hardware limitations as they will know that their application software contains no performance defects. Therefore, any detected problems will be much quicker and more straightforward to correct and any knock-on delay to the software's release will be minor.

The new tools all share similar attributes and it is important that any chosen tool provides the following features:

As little scripting as possible. Scripting is effectively programming and means the tester spends most of their time writing code. Test scripts can contain defects of course, so the scripts themselves need to be tested, corrected and retested before the performance test can be executed.

By choosing a tool that requires as little scripting as possible – with some tools requiring zero scripting - the complexity and effort associated with setting up a performance test are significantly reduced.

Easy to learn. Removing the need to program test scripts greatly simplifies the task of establishing a performance test; however, some of the new tools do this better than others. These tools also try to remove the need for formal training. By choosing a tool that is easy to learn, more of your Agile project team can be involved in performance testing and expert resource scheduling challenges are no longer an issue.

Quick to use. A requirement for testing within an Agile project iteration is that the time to update the test assets (from those used in the previous iteration), repeat the test and analyze the results needs to be as short as possible. The new breed of tools often allow this to be achieved in a matter of hours whereas traditional tools would require many days or weeks to do the same – and that's presuming a tool expert was available!

Encourage correct testing. Why do systems that have apparently been properly performance tested still fail when user load

We have 2500 guests and 7 members online



Agile Sponsors

[Streamstep](#)

[Thoughtworks](#)

[CollabNet](#)

Recommendations

[Log in](#) You need to be logged into Facebook to see your friends' recommendations

<http://agilejournal.com/events/webcasts/3176-scrum-for-project-managers>

3 people shared this.

[The 5 Key Benefits Agile Brings to Distributed Development](#)

2 people shared this.

Facebook social plugin

Featured Whitepapers

- [IBM Rational Workbench for Systems and Software Engineering](#)
- [A New Perspective on Application Release Management](#)
- [Lean Development Governance](#)
- [The IBM agility@scale e-Kit](#)
- [Seven Deadly Sins of Slow Software Builds](#)

Agile ALM from PureCM.
Integrate planning and configuration management.

[GET THE FREE TRIAL NOW](#)

peaks? The answer is simple. The performance tests executed did not simulate the way the application would be used by the real users. It is therefore imperative that any tool chosen allows the tester to correctly and fully simulate everything the real world user will do. Only by correctly testing an application can the test results be trusted and the decisions to release the software made with confidence.

Easy maintenance. In the example project used above, a performance test would be run at iteration end, every 3 weeks. Agile processes actively encourage major changes in requirements and therefore any selected tool should allow test assets to be updated easily and quickly.

Immediate diagnosis of problems. When testing identifies poor performance or scalability bottlenecks, it is imperative that you can quickly identify the real underlying causes (as opposed to the symptoms) of the problem. Therefore, a selected tool must also monitor 'deeply' into your system and provide a mechanism for quickly correlating performance problems and the monitor data to identify the true underlying cause(s).

Performance testing within the Agile process

So, how do you now implement performance testing within the Agile process with a performance testing tool that can correctly simulate how real users will interact with the software, is easy enough for many Agile team members to use, and allows rapid configuration, execution and analysis of performance tests?

If we again consider the example above – a 10 iteration Agile project with each iteration lasting 3 weeks - the goal is to treat performance testing with the same importance as unit and acceptance testing and therefore to execute a correct performance test at the end of every iteration.

The initial test would be at the end of the first iteration when only a small amount of software will have been developed; of course this may be the implementation of the applications architecture which is very important software as it will affect every business transaction subsequently developed. It doesn't actually matter how much code has been developed – it should be performance tested.

Using your new breed of performance testing tool, within approximately a day, the test environment (see below) can be configured, the tool installed, the initial 'test scripts' developed (of course you want to be scripting as little as possible) and a performance test executed.

Then, as part of every subsequent iteration, it will be literally a matter of hours to update the test assets and repeat the tests.

But what about the test environment?

It is often stated that performance tests must be executed on environments as similar to the production system as possible. This is correct for system performance testing (where the system includes the hardware and layered software as one item), but is not the case for application performance testing.

To quickly jump into some performance testing theory, there are four main reasons a system will fail to perform under peak load: the hardware is insufficient, the infrastructure software (database management systems, web servers, etc.) is not up to the job, the infrastructure software is incorrectly configured, or performance defects exist within the application code.

These causes are placed in the order they are least likely to occur i.e. very rarely nowadays are performance problems due to insufficient hardware resources and, conversely, the most common cause for poor performance is application code. In addition, in our experience of testing literally thousands of application releases for hundreds of organizations, the vast majority of application performance defects are identified with a simulated load of between 50 and 100 concurrent users.

Using this knowledge, it is obvious you don't need hardware as powerful as your production servers. It does however make sense to have the same infrastructure software in your test environment as will be installed in production.

Therefore, the test environment during an Agile project can consist simply of allocating two developer machines for the period of the test – one will host the performance testing tool and the other the latest build of the application. Alternatively, a couple of average specification desktop machines can be dedicated for this purpose for the duration of the project. The key point is that you do not need anywhere near the specification of the machines in your production environment.

No excuses

Agile processes promote regular testing and the goal of every iteration is to produce working software. Up until recently it has been believed that it was impossible to adopt performance testing within the project's iterations, and hence when it was stated that software was 'working', a 'subject to performance testing' caveat had to also be used.

The potentially high cost drawback of this approach was that performance testing was deferred until near the end of the project, and any problems detected during testing at this point often caused significant delays to the software being released and required extra effort and cost to fix. Of course, any benefits the business would receive from the software being released on time would be delayed as well.

Today, however, there is a new breed of performance testing tools available; the tools are lower cost, easy to learn and significantly quicker to use. These benefits mean the tools can be used within Agile iterations without delaying the projects, there is no need to hire or 'borrow' tool experts, and the danger of performance defects causing significant problems later in the project is eradicated.

Case Study: StressTester™ in action

One of the new breed of performance testing tools available is StressTester™ from Reflective Solutions. The tool was designed from first principles to be easy and quick to use but still allow organizations to correctly test their applications and understand the causes for any problems identified. Independent performance testing consultancies and adopting organizations have stated that the latest releases of StressTester™ are suitable for adoption into Agile projects; the time to configure and execute performance tests has been reduced from weeks to hours.

So let's look at StressTester™ in action and see how it helps the Agile project team configure and execute performance tests, and understand the causes for any performance problems, with no specialist tool experience and minimal associated effort.

No test scripts! Everything you need to do to correctly simulate real world users interacting with your software is configured within a single innovative GUI and you are never required to write a single line of test script.

The GUI is split into four workspaces. The majority of the time saving compared to traditional tools is in the User Journey workspace (a User Journey is a simulated business transaction) as there is no need for any scripting.

In this workspace (see Figure-1) the GUI presents a tree view of the User Journey with icons depicting key information.

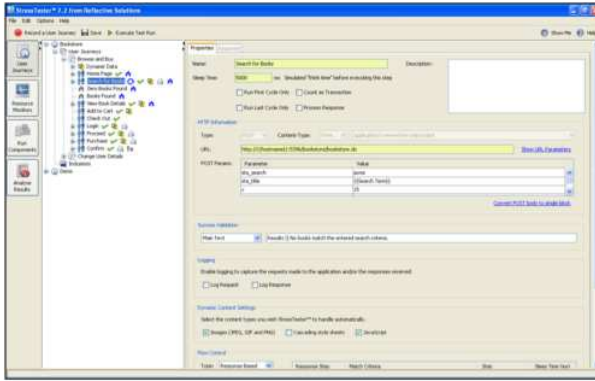


Figure-1: Properties of a Step

Clicking on a step (a web page or a request within a web page) shows the details about the selected item (see Figure-2) represented as fields on a screen.

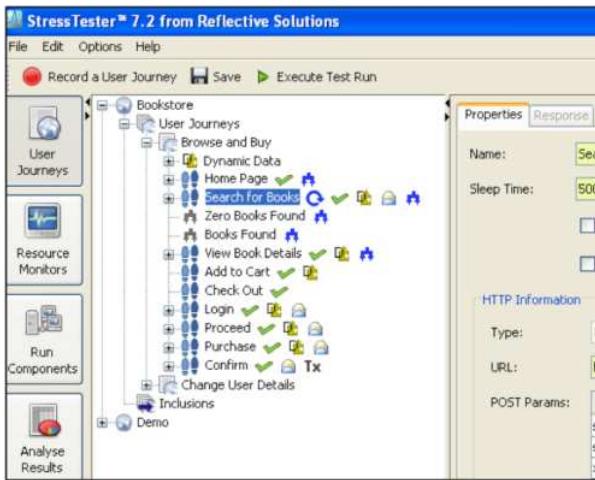


Figure-2: Easy to Navigate Tree View.png

In-tool wizards exist for changing recorded data into parameters, selecting data from application responses to use in subsequent requests (e.g. the item from the search results your simulated user will view in more details), varying the user's flow through the User Journey (there are three ways to check out of Amazon.com and if you did not simulate them all, your test would be incorrect and you may not discover hidden application performance defects) and many more commonly required features.

Easy to learn. The free online StressTester™ learning centre takes between 4 and 8 hours to complete and consists of various tutorials.

In addition, there are over 20 in-tool context sensitive 'Show Me' videos; when on a screen and needing assistance, simply click the 'Show Me' button and a list of applicable videos is displayed (see Figure-3).

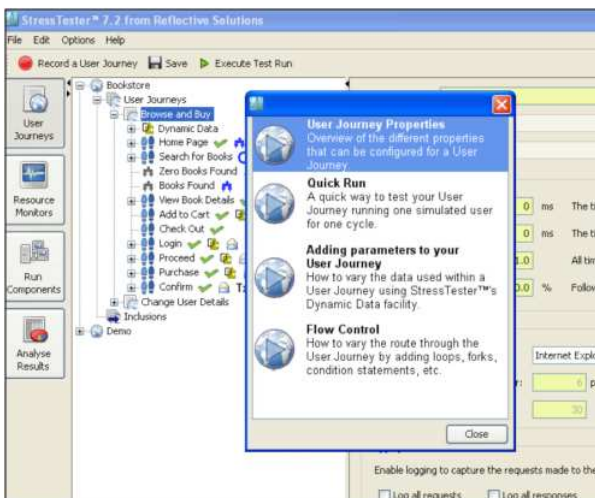


Figure-3: In Tool Video Tutorials

You can then watch the video(s) of interest and will understand how to proceed with configuring your performance test.

Therefore, any member of the Agile team, after dedicating a few hours to follow the online learning center tutorials, can now start to configure and execute performance tests.

Quick to use. Adopting organizations state that it now takes days of effort to configure and execute a performance test as opposed to weeks when using traditional tools.

'Days' refers to full performance tests where no User Journeys already exist for the application. As you will be testing from the first iteration, the initial configuration is much shorter. Then, as your project proceeds through the iterations, you will maintain your User Journeys and probably should allow half a day's effort per iteration to do so.

An example of time saving in action is when you need to extract data from an application response. In the example shown in Figure-4, the tester wishes to extract both a book id from a web page showing a list of books (so that the next step can request the details of a book) and the book name (which will be used to validate the response of the request and ensure the correct book's details are returned).

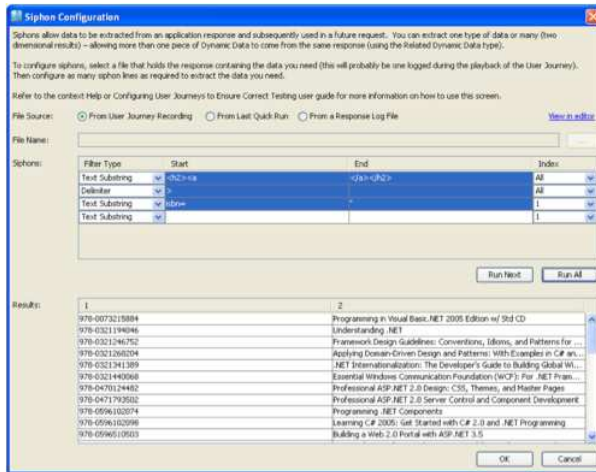


Figure-4: Extracting Data Values from a Response

Without any scripting, and with the ability to test the configuration without having to execute a performance test (using the 'Run' buttons shown), something that could take anywhere between 10 minutes to 4 hours to have fully working using a traditional tool, and which would require complex scripting, takes just a couple of minutes using StressTester™.

Correct testing. It is no use a tool significantly simplifying the performance testing process if it does not allow you to simulate everything that will happen in the real world.

StressTester™ allows you to configure all of the following (every item is as important as each other if you are going to correctly simulate real user behavior):

- Scaling to the load required without the need for large injection hardware; a typical desktop will simulate 1000+ users
- Specifying and varying sleep / think times between user requests
- Correct handling of hidden application variables, cookies and headers automatically – in exactly the same way the browser does
- Parameterization of all data that will change in the real world
- Ability to extract data from applications responses to use in subsequent requests
- Exact validation of application responses
- Variation of the route a simulated user takes through the User Journey
- Ability to simulate different browsers. Although last, this is not to be underestimated - especially for software that will be used by the public over the internet. Are you aware the Internet Explorer 8 can open 3 times as many connections to a web server than the previous version did? Potentially that is 3 times the number of concurrent requests and therefore 3 times the total load.

StressTester™ offers the ability to customize all of the above using intuitive screens that present the relevant fields in a logical order so even non-experienced performance testers are led through the thought process to ensure their User Journeys correctly simulate real world users.

Easy maintenance. There are tools that are quick to initially configure but you have to repeat all or the majority of the configuration every time your application changes. In the Agile world this is a non-starter as we expect the application to change, sometimes significantly, between iterations.

StressTester™ offers the ability to reorganize the flow of pages in a User Journey, edit what has been previously configured (including inserting new steps) or re-recording parts of a User Journey and replacing the equivalent in the old Journey. In addition, StressTester™ possesses full search/replace and copy/paste functionality, and the ability to mark one or more steps as re-usable between User Journeys (effectively sub-routines but without the need to write the script code).

These features ensure adapting User Journeys for application changes is a simple and quick task.

Immediate diagnosis of problems. If a performance testing tool tells you that a problem exists, but not the cause of the problem, it is no use within a fast moving Agile project.

StressTester™ allows you to monitor deeply into the internal workings of your whole application stack including operating systems, database servers, Java/.NET/etc. environments, web servers and much more.

All the monitoring information is reported in real-time during tests, and is automatically correlated with the performance test data by StressTester™'s analysis engine. When a performance problem occurs, you can quickly identify which monitors have exceeded their thresholds so you immediately know the cause for the problem(s).

Looking at the results from a performance test of a Java application as depicted in Figure-5, it is obvious that the performance degraded once 25 - 30 users were simulated accessing the system (the response times of the web pages are the dark blue and green lines on the graph). StressTester™ has detected and automatically correlated all monitors that exceeded their configurable thresholds and this data is displayed on the same graph.

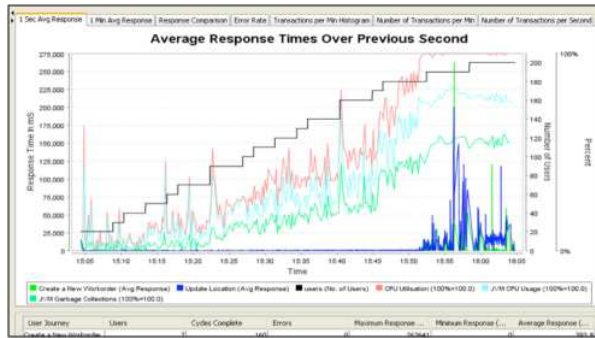


Figure-5: Identifying Cause of Problem

The statement "we need more CPU" seems correct due to the CPU usage hitting 100% (red line) - but is actually misleading. The CPU exhaustion is a symptom as opposed to the underlying problem. StressTester™ has identified the correct problem; a high number of Java garbage collections (dark green line) caused the JVM to use excessive CPU (light blue line) and hence the application server's CPU hit 100% usage. So, the problem is actually a code problem and the solution is application refactoring – not buying more hardware!

Quick ROI. The cost of an application performance defect discovered late in a project depends on the effort and time required to identify the cause, correct the problem and retest to ensure the software now performs and scales. The effort and time associated with such problems is often significant and it has even been known for projects to have been completely cancelled, as they had a fixed delivery date and it was obvious the required refactoring meant the date was unachievable.

Being able to adopt tools, with starting prices of a few thousand dollars, into the Agile development process provides an organization with the assurance that never again will a system performance test at the end of a project identify application code problems and require significant rework, cost and delay to implement the required solutions.

About the Author

A recognised expert in the field of web application performance and load testing, Graham Parsons is co-founder and CEO at Reflective Solutions, responsible for business strategy and growth. Reflective Solutions has worked with global brands, including Barclays, EDS, Harrods department store, JP Morgan, Nissan Motors, TNT, UBS and Vodafone, to ensure their applications are scalable and deliver the response times required by website users.

Graham has played a key role in improving the efficiency and effectiveness of performance testing tools, through the advancements of Reflective Solutions' industry-leading product StressTester™ – an enterprise class performance testing tool used by global organisations to verify the performance of their mission critical systems. He is regularly invited to share his expertise and insight into the future of the performance testing landscape at industry conferences globally.

Trained as a specialist in Java architecture, Graham co-founded Reflective Solutions in 1998 to offer high level strategic consultancy to adopters of the programming language and computing platform. In subsequent years, the company developed specialist knowledge in performance testing and tuning of enterprise systems before focusing on developing an innovative approach to the design of performance testing tools - totally removing the need for writing test scripts.

Prior to Reflective Solutions, Graham was employed by major financial organisations including Prudential and Barclays Funds. Graham has an Honours degree in Computer Studies from Sheffield University.

<http://www.reflective.com>

Hits: 186 [Email this](#) [Bookmark](#) [Set as favorite](#)

Comments (0)

Write comment

[Show/hide comment form](#)

You must be logged in to post a comment. Please register if you do not have an account yet.

Be the first of your friends to like this.

Agile Marketplace - Announcements and Special Offers

Rally Software Extends Agile ALM Platform to Meet the Unique Needs of Global Organizations

Rally Unlimited Edition – Promote Agile practices throughout your organization by providing a complete system-of-record of each product's status, progress and quality across the full idea-to-deployment lifecycle. [Sign-up today for your free trial!](#)

iPhone iPad Developers Conference

The iPhone iPad Developers Conference, September 27-29 in San Diego, is the world's premier independent event dedicated to building and marketing apps for Apple's iPhone, iPad and iPod Touch. The format includes 45+ technical classes, workshops and breakout classes. It will also be the first major developer conference after the release of iPhone OS4. CMC subscribers can receive a \$100 discount off the Full Event Passport and/or gain free admission to the exhibits (first time registrants only - cannot be combined with other offers) by inserting the code MEDIASPONSOR when prompted on the eRegistration page linked from www.iphonedevcon.com

AgilePalooza - Serious Learning in a Fun Atmosphere

AgilePaloozas are community events sponsored by VersionOne and Agile Journal. These one day conferences provide serious learning in a fun atmosphere. Two tracks are included: Learning Agility and Advancing Agility. Speakers include internationally recognized agile coaches and trainers. The next seminar will be held August 27th in Dallas, TX – use discount code [agilejournal](#) and save \$20!

[Register Here](#)

CollabNet Subversion Edge Improves Governance, Security, Administration

Quickly configure SVN, Apache, and ViewVC with one certified stack, fronted by a powerful UI.

Try our beta version and let us know what you think!

Most Popular Articles

- Go For The Low Hanging Fruit!
- AgileEVM Earned Value Management The Agile Way
- Scaling Agile Development Via Architecture
- What is Best, Scrum or Kanban?
- Agile at Scale: 7+7 Practices for Enterprise Agility
- FEATURED BOOK: Collaboration Explained: Facilitation Skills for Software Project Leaders
- FEATURED BOOK: Practical Development Environments - by Matthew B. Doar

Industry News

- Psion Powers-On its Innovation with Qualify
- USB lockdown capability of 3ami MAS saves the day at agile New Zealand ... - Newsblaze.com
- The what—and why—of going agile - SDTimes.com
- TRADE NEWS: Agilent Technologies' New LTE Base-Station Emulator Speeds Development ... - PR Inside
- TRADE NEWS: Agilent Technologies' New LTE Base-Station Emulator Speeds ... - Yahoo Finance
- Agile drivers for new project management tools - Zd Net Asia.com

Copyright © 2006 - 2009 CMC Media, Inc. All rights reserved. All marks are trademarks of CMC Media. Reproduction in whole or in part in any form or medium without the express written permission of CMC Media, Inc. is prohibited.

[Home](#) | [Articles](#) | [Login](#) | [Blogs](#) | [Webcasts](#) | [Resources](#) | [Forums](#) | [Media Center](#) | [About Us](#) | [Contact Us](#) | [Privacy Policy](#) | [CM Crossroads](#)