28

December 2014

testing experience

The Magazine for Professional Testers

The Three Pillars of Agile Quality and Testing
by Robert Galen

Testing the Internet of Things – The Future is Here
by Venkat Ramesh Atigadda

… and many more

printed in Germany
print version 8,00 €
free digital version
www.testingexperience.com
ISSN 1866-5705

Exploratory Performance Testing

It looks like exploratory performance testing has started to attract some attention and is getting a mention here and there. Mostly, I assume, due to the growing popularity of functional exploratory testing [1]. A proponent of exploratory testing probably would not like my use of the word "functional" here, but not much has been written, for example, about performance exploratory testing – and even what has been written often refers to different things.

There have been attempts to directly apply functional exploratory testing techniques to performance testing. SmartBear blog posts [2, 3] contrast exploratory performance testing with "static" traditional load testing. My view is probably closer to Goranka Bjedov's understanding as she described it back in 2007 [4].

It was clear to me that a traditional, waterfall-like approach to performance testing is very ineffective and error-prone. I presented a more agile/exploratory approach to performance testing in a traditional waterfall software development environment at CMG in 2008 [5]. I intended to apply the original principals of Manifesto for Agile Software Development [6] (valuing "Individuals and interactions over processes and tools. Working software over comprehensive documentation. Customer collaboration over contract negotiation. Responding to change over following a plan.") to performance engineering.

Performance testing in projects utilizing specific agile development methodologies is a separate topic. Having become more involved in the agile development environment, I added some aspects of it to my presentation at the Performance and Capacity 2013 conference by CMG [7].

The words "agile" and "exploratory" are periodically and loosely used in relation to performance testing – but it does not look like we have any accepted definition. Both terms are, in a way, antonyms of traditional waterfall-like performance testing – so their meaning may somewhat overlap in certain contexts. I explained my view of using the word "agile" for performance testing in the above-mentioned presentations. Now it is time to contemplate the use of the word "exploratory" in the context of performance testing.

If we look at the definition of exploratory testing as "simultaneous learning, test design and test execution"1, we can see that it makes even more sense for performance testing, because learning here is more complicated, and good test design and execution heavily depend on a good understanding of the system.

If we talk about the specific techniques used in functional exploratory testing, some can be mapped to performance testing – but definitely should not be copied blindly. Working with a completely new system, I found that I rather naturally align my work to a kind of "session" – so session-related techniques of functional exploratory testing are probably applicable to performance testing. I would not apply such details as session duration, for example – but the overall idea definitely makes sense. You decide what area of functionality you want to explore, figure out a way to do that (for example, create a load testing script) and start to run tests to see how the system behaves. For example, if you want to investigate the creation of purchase orders, you may run tests for different numbers of concurrent users, check resource utilization, see how the system will behave under stress load of that kind, or how response times and resource utilization respond to the number of purchase orders in the database, etc. The outcome would be at least three-fold: (1) early feedback to development about the problems and concerns found; (2) understanding the system dynamic for that kind of workload, what kind of load it can handle, and how much resource it needs; (3) obtaining input for other kinds of testing, such as automated regression or realistic performance testing to validate requirements. Then we move to another session exploring the performance of another area of functionality or another aspect of performance (for example, how performance depends on the number of items purchased in the order).

The approach looks quite natural to me and maximizes the amount of early feedback to development, which is probably the most valuable outcome of performance testing for new systems. However, there are different opinions. Objections mostly align along three notions, which, if taken in their pure form, are not fully applicable to performance testing of new systems:

- Creating a detailed project plan (with test design, time estimates, etc) in the beginning and adhering to it.
- Fully automating performance testing.
- Using scientific Design of Experiments (DOE) approaches.

I mention all three of them here because (1) they are often referred as alternatives to exploratory testing; (2) they all are rather idealistic for the same reason – we do not know much about new systems in the beginning and every new test provides us with additional information. And often this additional information makes us to modify the system. Somehow the point that the system *is* usually changing in the process of performance testing is often overlooked.

For example, if your bottleneck is the number of web server threads, it does not make much sense to continue testing the system once you realize it. As you tune the number of threads, the system's behavior will change drastically. And you would not know about it from the beginning (well, this is a simple example and an experienced performance engineer may tune such obvious things from the very beginning – but, at least in my experience, you will always have something to tune or optimize that you have no idea about in the beginning).

So, actually, you probably do exploratory testing of new systems one way or another even if you do not recognize it. And it would probably be more productive to fully acknowledge it and make it a part of the process, so you will not feel bad facing multiple issues and will not need to explain why your plans are changing all the time. I would concur here with the great post "TDD is dead. Long live testing." by David Heinemeier Hansson [8] discussing, in particular, issues related to using idealistic approaches. ■

References

[1] Exploratory Testing. Wikipedia. *http://en.wikipedia.org/wiki/Exploratory_testing*

[2] Ole Lensmar. 2012. Why Your Application Needs Exploratory Load Testing Today. *http://blog.smartbear.com/loadui/why-your-application-needs-exploratory-load-testing-today*

[3] Dennis Guldstrand. 2013. Should Exploratory Load Testing Be Part of your Process? *http://blog.smartbear.com/load-testing/should-exploratory-load-testing-be-part-of-your-process/*

[4] Goranka Bjedov. Performance Testing. 2007. *http://googletesting.blogspot.com/2007/10/performance-testing.html*

[5] Alexander.Podelko. Agile Performance Testing. CMG, 2008. *http://alexanderpodelko.com/docs/Agile_Performance_Testing_CMG08.pdf*

[6] Manifesto for Agile Software Development. 2001. *http://agilemanifesto.org/*

[7] Alexander Podelko. Agile Aspects of Performance Testing. Performance and Capacity by CMG, 2013. *http://www.slideshare.net/apodelko/agile-aspects-of-performance-testing*

[8] David Heinemeier Hansson. TDD is dead. Long live testing. 2014. *http://david.heinemeierhansson.com/2014/tdd-is-dead-long-live-testing.html*

> about the author

For the last 17 years, **Alex Podelko** has worked as a performance engineer and architect for several companies. Currently he is a Consulting Member of Technical Staff at Oracle, responsible for performance testing and optimization of Enterprise Performance Management and Business Intelligence (a.k.a. Hyperion) products. Alex periodically talks and writes about performance-related topics, advocating tearing down silo walls between different groups of performance professionals. His collection of performance-related links and documents (including his recent papers and presentations) can be found at *www.alexanderpodelko.com*. He blogs at *www.alexanderpodelko.com/blog* and can be found on Twitter as *@apodelko*. Alex currently serves as a director of the Computer Measurement Group (CMG) *www.cmg.org*, an organization of performance and capacity planning professionals.