WHITE PAPER | APRIL 2016

Surviving Shift-Left as a Tester

In a series of articles, Paul Gerrard, a testing guru and consultant, discusses a range of testing topics. Shift-left is mostly about bringing the thinking about testing earlier in the process. In this article, Paul talks about what is driving the move to shift-left, how the role of testing is changing and how you will be affected in a more agile world.

Paul Gerrard
Gerrard Consulting

Sponsored by

ca technologies

Surviving Shift-Left as a Tester

Background

Some five years ago, I coined the term "redistributed testing" to describe a change in the testing business. Users, analysts, developers and testers would redistribute responsibility for testing and collaborate more effectively. The change would involve moving test activities (and possibly responsibilities) to the left and the term used for this approach is commonly called "shift-left."

Shift-left can mean developers take more ownership and responsibility for their own testing; it can also mean testers get involved earlier, challenge requirements and feed examples through a behaviour driven development (BDD) process to developers. It can mean users and BAs together with developers take full responsibility for testing, and it can mean no test team and no testers. We have seen all configurations and there is no "one true way" of course.

This article describes how the role of testing is changing and how you will be affected in a more agile world.

Shift-Left is Not New

For as long as I've been in the testing business (1992) and certainly earlier than that, testing advocates have preached the mantra "test early, test often". The W-Model,[1] introduced by my friend and colleague Paul Herzlich as long ago as 1993 suggested that all artefacts in a staged process—both documentary and software—can (and should often) be tested.

Although Waterfall was the dominant life-cycle approach at the time, the number or duration of stages are not what is important. The underlying principle was that the *sources of knowledge that provide direction for the design and development of software should be challenged or tested*. In a staged project, this might involve formal reviews. In an agile project, the tester (or developer or BA or user) can suggest scenarios (or examples) that challenge the author of a requirement or story to think-through a concrete examples and discussion before code is written.

Shift-left is mostly about bringing the thinking about testing earlier in the process.

So, all that shift-left requires is that the testers get involved earlier and ask awkward questions? Is it really as simple as that? Well, not quite.

What is Driving Shift-Left?

Several changes in the market are at play and are driving new behaviours in our industry. Some started around five years ago but others are more recent. But what is meant by "started?" Started means there were enough people promoting new approaches that showed evidence of success and were sufficiently credible for others to adopt them. These approaches "crossed the chasm" as Geoffrey Moore describes it,[2] and became viable for the wider business and IT community.

These are the main changes involved in the shift-left phenomena:

1. The Behaviour Driven Development (BDD) approach has allowed developers, users/BAs and testers to collaborate using business stories. BDD is being adopted more widely because it encourages better collaboration in Agile teams.

2. The concept of continuous delivery (CD),[3] again, has been around for 5-10 years and its roots are in the highly automated build and release automation approaches pioneered by large online businesses. It is now adopted by most organisations with an online presence.
3. CD systematised and accelerated the release process through automation. But it then highlighted the delays in production deployment and infrastructure change that had previously been masked by slow build, test and release processes. DevOps is a cultural and mind-set change whereby developers collaborate much more closely with operations staff, in particular. Right now, new tools appear almost daily and vendors promote DevOps as the "next big thing." It is a very hyped and dynamic situation.
4. SMAC, or social, mobile, analytics and cloud represents a shift in the way organisations are managing business and systems' change in the mobile space. Experimentation in business, implemented as production systems changes, are monitored at a detailed level. The "big data" captured is processed and business decisions are made on the basis of the analytics obtained.

Frequent experimentation with production systems enables business innovation "at the speed of marketing." Experimentation is at the heart on what seems to be the most important bandwagon of the 2010s—"digital transformation." Digital transformation, or "digital," is getting most of the attention (and budget) right now. Marketers are promising faster and better access to consumers through more channels—mostly mobile.

My paper "Digital Transformation, Testing and Automation"[4] describes the digital revolution and proposes some responses, so might be of interest.

What Does Shift-Left Mean to Testers?

Shift-left implies that whenever it is possible to provide feedback that will help the team to understand, challenge and improve, goals, requirements, design or implementation, that feedback should be provided. This behaviour comes as second-nature to many, but not all, testers. Users, BAs, developers and the entire team should be ready to both provide and accept feedback in this way. There might be resistance, but the overall aim is to run a better, more informed project—that's all.

What does the tester do in the shift-left world of testing? Well, the easiest way to summarise this behaviour is "get involved early" as early as possible. Engage in the discussion and collaborate on ideas, requirements and every stage where the outcome of that stage has a bearing on the value of the final deliverable of the project. Put simply, the tester challenges sources of knowledge, whether these sources are stakeholders, users, developers, business stories, documents or received wisdom.

The most common approach is to "challenge through example." At all stages, these examples can be regarded as tests. They might be discarded quickly after use, or be codified into test automation or manual checks. These examples could just be used tactically to point out flaws in peoples' thinking or be provided to developers, say, as ideas or seeds for developer tests. They might also be used as a coaching aid to help users or developers to see how better tests can be created.

Software projects have been described as a knowledge acquisition process.[4] This knowledge is gathered throughout the project and often evolves over time. The goal of shift-left is to assure this knowledge through challenge and testing close to its source and to ensure, where possible, that it is trusted before it is frozen in code.

Shift-left takes the test-first philosophy further. Agile has always promoted collaboration and rapid feedback and shift-left could be viewed simply as the ultimate rapid-feedback approach.

If you adopt it, shift-left has a profound effect on how you work as a tester.

As a Tester, How do I Apply Shift-Left?

It looks like shift-left is not just a fad, and it's coming your way. How will it affect you if you work in a system test team? If you are part of an agile team, does it still matter? What should you do?

We have advocated the shift-left approach as core to test strategy in agile projects for some time. In an agile context, test strategy can be viewed as a series of "agile interventions." There are critical moments in all projects where opportunities to gather and present feedback present themselves. The tester needs to focus on these critical moments and be ready to contribute at those times.

I presented the thinking behind this approach in a recent webinar[6] and I use a client case study to illustrate where these interventions might occur. In your own projects, you need to identify your "critical moments," and identify the choices that you and your team can make. For example, should you write unit tests for developers, provide examples to get them started, or should you coach them to improve their testing ability?

Your role will almost certainly change. It may be that testers are not just thinking but are shifting left, and you will become the testing servant of developers. This probably is not the best outcome for you or your project. We suggest you identify the critical moments, propose your contribution and negotiate with your team. You offer more test leadership and guidance rather than volunteering simply to take on responsibility for the testing work. It will be much easier to demonstrate your value to the team if you take this approach—the team won't need as many testers.

I'm a Test Lead/Manager. What Should I Do?

If you are a test manager or a test lead now, it might be harder to justify your role if the intent of management is to reduce the cost of testing by shifting left. If your organisation is moving in this direction, you probably have a longer term career decision to make. Where do you want to be in five years? In six months? We have identified five broad choices that might be open to you.

1. **Providing test and assurance skills to business**—moving up the food chain towards your stakeholders, your role could be to provide advice to business leaders wishing to take control of their IT projects. As an independent agent, you understand business concerns and communicate them to projects. You advise and cajole project leadership, review their performance and achievement and interpret outputs and advise your stakeholders.
2. **Managing Requirements knowledge**—In this role, you take control of the knowledge required to define and build systems. Using your critical skills, you ensure clarity and precision in requirements and the examples that illustrate features in use. You help business and developers to decide when requirements can be trusted to the degree that software can reasonably be built and tested. You manage the requirements and glossary and dictionary of usage of business concepts and data items. You provide a business impact analysis service.
3. **Be a TestMaster**—providing an assurance function to teams, projects and stakeholders: A similar role to 1 above—but for more agile-oriented environments. You are a specialist test and assurance practitioner that keeps agile projects honest. You work closely with on-site customers and product owners. You help projects to recognise and react to risk, coach and mentor the team and manage their testing activities and on occasion, do some testing too.

4. **Be a DevOpsMaster**—managing the critical information flow to and from DevOps processes (automated build, test and deployment processes). This information flow is critical. Perhaps you could define and oversee the processes used to manage the flows that enable control of change, testing and delivery.
5. **Managing outsourced/offshore teams**—In this case, you relinquish your onsite test team and manage the transfer of work to outsourced or offshore suppliers. You are expert in the information flow and manage the relationship with the outsourced test team, monitor their performance and assure the outputs from them.

If you haven't been shifted left yet, then you should take a look around both inside and outside your team and think about how your role might change in the near future. Your role will eventually change, but you should have some choice of how it evolves. I wish you luck in your choices.

About the Author

Paul Gerrard is a consultant, teacher, author, webmaster, developer, tester, conference speaker, rowing coach and a publisher. He has conducted consulting assignments in all aspects of software testing and quality assurance, specialising in test assurance. He has presented keynote talks and tutorials at testing conferences across Europe, the USA, Australia, South Africa and occasionally won awards for them.

Educated at the universities of Oxford and Imperial College London, in 2010, Paul won the Eurostar European Testing excellence Award and, in 2013, won The European Software Testing Awards (TESTA) Lifetime Achievement Award.

In 2002, Paul wrote "Risk-Based E-Business Testing" with Neil Thompson. Paul wrote "The Tester's Pocketbook" in 2009. Paul co-authored "The Business Story Pocketbook" with Susan Windsor in 2011 and wrote "Lean Python" in 2014.

In 2014, Paul was the Programme Chair for the EuroSTAR Conference in Dublin.

He is Principal of Gerrard Consulting Limited, Director of TestOpera Limited and is the host of the Test Management Forum.

Mail: paul@gerrardconsulting.com
Twitter: @paul_gerrard
Web: gerrardconsulting.com

For more information visit **Develop & Test** with CA Technologies.

Connect with CA Technologies at ca.com

CA Technologies (NASDAQ: CA) creates software that fuels transformation for companies and enables them to seize the opportunities of the application economy. Software is at the heart of every business, in every industry. From planning to development to management and security, CA is working with companies worldwide to change the way we live, transact and communicate – across mobile, private and public cloud, distributed and mainframe environments. Learn more at **ca.com**.

References

1. "The W-Model," http://blog.gerrardconsulting.com/?q=node/531
2. "Crossing the Chasm" and other books by Geoffrey A Moore, http://www.chasminstitute.com/
3. Continuous Delivery definition, Martin Fowler, http://martinfowler.com/bliki/ContinuousDelivery.html
4. "Digital Transformation, Testing and Automation," Paul Gerrard's blog, http://blog.gerrardconsulting.com/?q=node/660
5. "The Laws of Software Process," Philip G Armour.
6. Webinar: "Agile Test Strategy," Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/627

CS200-181168_0416