# Agile Development Tests the Testers

**Dennis Stam MSc CISA and Thomas Beekman MSc**

**Anyone looking back at the history of software development can see how radically the world of development has been changing. From traditional development methods such as SDM, which follows a pre-defined structure and approach, towards adaptive Agile methods, such as Scrum (that appear less structured to the untrained eye), the testing landscape seemingly has not changed much. Does that same (proven) approach to software testing still work for current projects?**

D. Stam MSc CISA
is a manager at KPMG Advisory N.V.
stam.dennis@kpmg.nl

T.S. Beekman MSc
is a consultant at KPMG Advisory N.V.
beekman.thomas@kpmg.nl

## Introduction

The Agile approach is often misused as a convenient excuse for "cowboy coding": a sloppy, ad-hoc, *do-whatever-feels-good* approach to software development. However, contrary to popular myth, Agile methods are far from that. As Mary Poppendieck stated: "*speed requires discipline*" ([Popp04]).

As design and development (coding) strategies are embracing Agile methodologies, so must testing. Agile projects need a rigorous approach to testing: a structured, documented approach to carrying out testing. This article outlines implications of the Agile movement on the testing domain within a software development project.

## What is Agile testing?

"Agile testing" means the testing activities executed in an Agile-based project. In the world of software development, the term "agile" typically refers to an approach based upon the principles of flexibility and responsiveness to feedback that is applied throughout the process of system development. Integrated testing within the iterations of the Agile development approach is necessary to make full use of the benefits of the methodology. For example, early realization of business value requires the newly developed functionality to go live. Without properly testing the release, the business (and most likely IT) would either face significant risks going live, or would need to postpone the go-live while testing is completed (postponing the realization of business value). Since Agile development and testing necessarily go together, Agile testing generally means the practice of testing software within the context of an Agile project.

The main objective for any testing, regardless of the software development methodology applied, is to verify that the system works correctly (according to the business needs) before being delivered to the end-user or customer. This is no different in Agile environments.

*The main objective for any testing is to verify that the system works correctly. This is no different in Agile environments.*

The main difference from a testing perspective comes from the phasing within Agile projects. Agile development recognizes that testing is not a separate phase, but an integral part of the development cycle. Within a single iteration, the envisioned functionality needs to be designed, developed and then tested for the team to be successful and to be able to deliver a new (potential) release to the customer. As the success of the team depends on all these elements, the Agile approach requires the entire team (including developers, analysts and testers) to focus on quality (and thus testing), instead of leaving that to the test team as the product-quality custodians at the end of the cycle. As such, testing is no longer regarded a separate phase but is integrated as a way of working, leading to continuous testing to support continuous progress. The testing role is performed throughout, guiding the development team towards controlled quality checks.

Without this integrated approach, none of the iterations would result in delivering a potentially shippable product, and hence the product's promised business value could not be realized until Testing had caught up with Development. Of course there are many variants of Agile methodologies, sharing the same principles as well as many of the same characteristics and practices. And although each implementation has its own advantages and disadvantages, those core principles direct how testing needs to adapt and respond to these new Agile processes.

## Changes to testing

Software testing has traditionally relied on project deliverables. A traditional project would commence with identifying the requirements, leading to detailed designs of the planned system. Based on these designs the developer would develop code, which ultimately would be delivered to the tester to assess whether the system had been developed according to the specifications and would be suitable for the business. As a result, testers would prepare and specify the tests to be executed on the system while the analyst and developer were completing the system for delivery to the testers. Then the tests would be executed and the findings returned to the developer for resolution. After the identified defects were assessed and resolved, the tester would re-test the system to ensure that the quality of the system would meet the customer's expectations. All this in sequential phases.

Agile principles introduce a paradigm shift here. Shorter iterations and more frequent deliveries (ideally in a production state) require changes in two areas: the testing process and a mentality change of the tester.

## Testing process

### Test approach per iteration

Following internationally accepted testing frameworks such as ISTQB and TMap Next, the approach to testing is often described in a test strategy at the organizational level, the division level or (most often) the project / product level. Especially in the latter case, the testing approach is tailored to the specific situation.

This tailored approach depends upon there being sufficient insight as to the project goals, insight that, due to the iterative nature of Agile development, often does not exist until the start of the iteration. In those cases, the overall test approach can be documented in the release plan, with iteration-specific plans describing more detailed activities to target specific functionality. The overall test approach would then cover the types of (product) risks associated with the (planned) functionality as well define the different test types, the test organization, infrastructure and test management framework.

| | Traditional | Agile |
|---|---|---|
| Test approach | Describing approach for release | Describing overall functionality but detailed description per iteration |
| Test execution | Manual testing, sometimes automated | Manual testing, plus mandatory automation to keep up the pace |
| Positioning of testing | Testing as verification near the end of process | Testing as guidance throughout process |
| Communication | Spotting bugs and deviations between specifications and (final) product | Providing information about current state and context |
| Role | Quality custodian | Coaching entire team to maintain quality awareness |
| Mentality | Rigorous, structured and process-oriented | Adaptive, pragmatic |
| Expertise | Specialist | Jack-of-all-trades |

Table 1. **Comparison of traditional and agile testing.**

*Automation is not optional anymore; it is an essential aspect, especially when trying to improve the time to market*

The Agile approach "postpones" detailing the test approach and demands more insight from the actual tester, rather than the test manager or coordinator. This happens because identified (product) risks will evolve or only become apparent close to the actual realization of the functionality. This means the risk assessment happens in or near the testing phase and is often performed by the tester rather than the test manager.

Although this ongoing insight is captured in the detailed iteration-specific test documentation, it goes without saying that the overall test approach should be kept evergreen for adaptation in future iterations.

**Automation is no longer optional**

Traditional development methods required a limited number of testing cycles. As a result, regression testing (testing to verify that recent changes to existing functionality did not introduce new issues) needed to be executed only once or twice in the entire testing project, for which sufficient time could be allocated.

The short iterations within Agile development, however, mean that the test team is not in such a luxurious position. At the end of every iteration (usually 2 to 4 weeks), the expectation is that the team will deliver a fully functional release that (potentially) could go live. Although the team may be able to cope with this the first few iterations, the total scope of the system grows with every additional iteration, and the problem in doing a full regression test grows to a point where the team can no longer add functionality and test the system adequately within a single iteration.

Manual (regression) testing is therefore only a short-term solution in most Agile projects. Automation is not optional anymore; it is an essential aspect, especially when trying to improve the time to market. Agile teams working at peak velocity adopt such practices as continuous integration, automated development tests, and automated acceptance tests. Without automation and application of tools, the team cannot achieve the desired agility ([Expe12]).

Automation occurs in various ways, ranging from automated deployment of a new build to automated execution of tests (unit testing). The more the team can automate, the faster they can move on to the next developments, cutting delivery times and removing some of the mundane, repetitive work, leaving skilled resources to focus on the more difficult and valuable tasks. While it is impossible to automate everything, automation does free up more time to spend on manual testing.

## Mentality change

Optimizing results with Agile testing requires a change in the mentality within a team. The following paragraphs outline a select number of guidelines for testers moving towards Agile environments ([Hend08]).

**Testing as the headlights of the project**

Traditional testing was based on testing a product near to the end of the development phase. Testers were given the original requirements and the product, and their assignment was to verify if the product satisfied the requirements. Worst-case scenarios in which critical but lengthy development phases were substantially prolonged were not uncommon.

Testing during the development gives insight into questions like, "*Where are we now?*" and "*Where are we headed?*" Especially in Agile development processes, the end goal may not always be clear, due to the flexible nature of the methodology: there is always the ability to change the course of development due to new insights. Testing during development gives insight into quality as well as integration of all realized functionalities, and can reveal potential gaps. It gives a total overview of the current state of the product in contrast to what could be defined as a fully working, mature product. As the test results are available sooner than with traditional testing, the team and product owner are able to adjust the scope and direction at an earlier stage, to prevent massive subsequent development phases.

For the tester this requires a flexible approach towards the specifications and the resulting product. The tester can no longer rely entirely on (pre)written specifications, but instead must deal with ongoing change in both the product and its supporting documentation. As a result, the tester should be thoroughly aware of the envisioned result, with a full understanding of what a fully working, mature product requires, and be able to oversee the gap between this end result and the current state of the system.

### Testing provides information to the team

Traditional testing provided information about the developed product after its initial development. Defects and inconsistencies were reported in lengthy documents returned to the development team for correction.

Agile testing has, with its continuous nature, a direct link to the development. Not only are defects found at earlier stages, but specific information is also instantaneously available. A developer is able to discuss the defects directly with the tester or even the whole team, so that much more information is shared and a higher level of understanding of the product can be reached, which enables the team to make better-informed decisions.

### A "bug" is anything that could bug a user

Where traditional testing is focused on defects and deviations between the specifications and the final product, Agile testers also focus on less obvious quality attributes such as user experience. After all, at the end of every iteration, the product could migrate into a live situation. A "bug" is therefore not by definition a defect, but can be anything that causes the product to work in a less than optimal manner, and is to be discussed within the team during the development. These "issues," like feature requests, represent work that should be prioritized against other envisioned work. As such, when a solution is conceived in collaboration with the whole team, a final decision is made by the product owner whether or not to fix the issue or implement the improvement.

### Testing is no longer the sole quality custodian

Traditionally, testing had the main purpose of determining whether the quality of the product was at an acceptable level. If this was not the case, the development team was required to keep working on the quality of the product until it reached an acceptable level.

In Agile development methodologies, delivering a quality product is seen as the responsibility of the whole team: testers as well as developers. As such, testers cooperate with developers in a joint effort to provide the level of quality requested by the client. The tester should be focused on taking ownership of the product and its quality, not solely on examining its quality. Similarly, the tester needs to accept that fellow team members will need to "test" as well. This in itself prompts an interesting discussion on the objectivity and independence of the tester and Agile testing activities in general (see also "Challenges").

### Work in a less ideal world

To become truly Agile, one must gain speed: speed in both making progress as well as in adapting to a changing situation. The same goes for the tester.

Throughout the project, the tester is bound to encounter numerous hurdles that will render the test environment unusable or prevent access to a particular functionality. In those cases, the traditional tester would write down the issue at hand, drop the pen and wait for a new release: a second chance for development to fix the issue. Agile testers, on the other hand, feel a need to continue, due to the shared responsibility to deliver a quality product at the end of the iteration. They will pursue alternatives. For example, they might bypass the test environment for the time being and work in the development systems instead while the issue with the test environment is being determined.

### The jack-of-all-trades: not just functional testing anymore

Traditional projects allowed for specific expertise to be bundled into a limited timeframe. For example, performance or security testing requires experts in their domain rather than the average (often functional-oriented) tester. Throughout the project, the test team would consist of testers focused on functionality, and as the project approached release, specialists would be on-boarded to run their tests.

Agile development projects require a quality product delivered with each iteration. As every code-change potentially introduces new security or performance flaws, the iterative characteristics of Agile development require constant testing against these aspects as well. However, the experts for these types of tests are typically rare and expensive.

The solution is simple in theory, but hard to accomplish in practice: there is a need for generalists, jacks-of-all-trades, testers who can help the team identify the functional issues but also investigate non-functional aspects such as (and not limited to) performance, security, usability and accessibility. The focus is on minimizing project risks. Even though it may be almost impossible to perfect all functional and non-functional issues, every effort is made to allow the team to act on findings and thus minimize the risks of going live with severe issues. For example, having a team member with thorough knowledge of cross-site scripting (XSS[1]) attacks on web applications could identify such possible threats at an early stage. Nonetheless, periodic testing by specialists will be required to safeguard against unacceptable risks.

1 Cross-site scripting is one of the OWASP top-10 web vulnerabilities ([OWAS13]).

## Challenges

The main challenges in adapting Agile methodologies from a testing perspective lie in allowing the team to work as a single unit towards a common goal. This requires the team to synchronize their efforts and work at a sustainable pace to deliver software while maintaining the core competences of the team.

*There is a need for generalists, jacks-of-all-trades, testers who can help the team identify the functional issues but also investigate non-functional aspects*

## Working at a sustainable pace

As with many other aspects of life, the team delivering the system can only go as fast as its slowest component. If the team develops more code than can be properly tested, the team may be unable to comply with the quality checks required before delivery.

In many cases the bottleneck is in testing the developed code (and retesting after defects have been fixed). The challenge here lies in increasing test speed without compromising quality. The answer may lie in adopting Risk-Based Testing (prioritize testing based on how great a risk failure in any particular area would pose, rather than trying to always test everything), implementing the team-effort concept (have the tester coordinate and oversee testing activities by developers and analysts) and, obviously, automating testing (increase the pace at which developed code can return information on the quality of the system).

## Objectivity

As mentioned earlier, Agile testing requires testers and developers to work together as a team to bring the final product forward. The tendency is for testers to develop (partial) ownership of the product and the (design) decisions behind it. Testers no longer solely report back on the gap between specifications and the system, but give feedback on the specifications and might even work together with the developer to resolve issues. Many question whether this affects the testers' objectivity in assessing the system, as would be the case with IT auditors auditing controls they had developed or implemented themselves.

## Test planning and motivation

Recent research has found that happier people are up to 12% more productive ([Oswa14]). Team morale could suffer by spending too much time on (repetitive) testing, resulting in slow progress and communication challenges ([Borl12]).

Traditional projects usually define one or two testing phases. The iterative nature of Agile development means that there are multiple testing phases, with the product tested at every iteration before being released to the customer. As a result, the general testing costs for Agile projects are often substantially larger than in traditional projects, as tests have to be run multiple times and/or automated and maintained.

Evidently it follows that keeping team morale up in Agile projects can be more challenging than in conventional projects, but is crucial. Automating repetitive tasks allows the team to spend more time on more challenging and rewarding activities.

## In conclusion

The world around testers is evolving, so testers and their processes must keep pace. The purpose and process of testing may not be changing much (testing still focuses on verifying and validating that the software meets the requirements, and continues to test the same items). However, the Agile movement is having a large impact on the position of the tester in IT projects as well as on the tester's mindset and skill set. In particular, test automation may have been around since the mid-1980s, but it is playing an increasingly larger role in the success of the testing phase.

In a way, the Agile movement is testing the adaptability of the testing professional.

### Literature

[Borl12] Borland, "Adopting Agile Testing," 2012, https://www.borland.com/_images/Borland_Whitepaper_v7_tcm32-207594.pdf

[Expe12] M. Expedith, "Agile Testing: Key Points for Unlearning," 2012, http://www.scrumalliance.org/community/articles/2012/january/agile-testing-key-points-for-unlearning

[Hend08] E. Hendrickson, "Agile Testing, Nine Principles and Six Concrete Practices for Testing on Agile Teams," 2008.

[Oswa14] A.J. Oswald, E. Proto and D. Sgroi, "Happiness and Productivity," 2014.

[OWAS13] OWASP Foundation, "OWASP Top 10 – 2013," 2013, http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf

[Popp04] M. Poppendieck, "Why the Lean in Lean Six Sigma?" 2004, http://www.poppendieck.com/lean-six-sigma.htm

### About the authors

**Dennis Stam MSc CISA** is a manager at KPMG Advisory N.V. He has extensive experience in the entire software development cycle, with a focus on testing, test management and quality control in particular. He has been involved in numerous development projects based on different development methodologies ranging from traditional waterfalls to Agile Scrum.

**Thomas Beekman MSc** is a consultant at KPMG Advisory N.V. He has a background in Economics & Informatics and IT auditing and has extensive experience in software development as well as software testing and quality control. He has been involved in numerous Software Quality reviews and (mobile) application development projects based on methodologies such as waterfall and Agile Scrum.