QASymphony

THE AGILE WAY:

A complete guide to understanding Agile testing methodologies

NAVIGATING AN AGILE WORLD

The testing world is going through an agile transformation right now. According to VersionOne's State of Agile survey, from 2012 to 2014, the percentage of respondents practicing agile jumped from 35% to 80%.*

As more and more companies move from the traditional "Waterfall" development methodologies to Agile, this has created both challenges and opportunities for professional testers.

On one hand, development is moving at a much faster pace, putting increased pressure on the testing team to thoroughly assess software quality with dramatically tighter time-lines. And on the other hand, testers are now working more collaboratively with the larger development team than ever before. This gives testers a bigger opportunity to make an impact on the team and the company.

In order to survive and thrive in this new testing world, it's critical to have a comprehensive understanding of what agile really means and the different methodologies that are part of agile development. That's not an easy job. Agile is constantly evolving. Today there are emerging methodologies like BDD, TDD, ATDD, Kanban and many more. For even the most mature agile teams, it's hard to keep up.

That's why we developed this ebook. To help Testers, QA Managers and heads of DevOps, CIOs and anyone else who manages application testing understand the complex world of agile development.

We hope you find the content in this ebook helpful. If you want to stay up-to-date on all things related to agile testing, I encourage you to keep up with our blog, sign up for our monthly e-newsletter and keep up with us on Twitter @QASymphony.

Sincerely,

Kevin Dunne

Kevin Dunne is the Director of Product Strategy at QASymphony, ensuring their continued commitment to innovation and delivering tools to create better software. With a deep interest in the emerging trends in software development and testing, Kevin is dedicated to collaborating with thought leaders in this space.

Kevin comes to QASymphony from Deloitte, where he managed testing on large government and Fortune 500 engagements delivering ERP implementations and custom software development. As one of the first employees at QASymphony, Kevin has seen many facets of the business working in sales, customer support, marketing, and product management.

Kevin holds a Bachelor of Science degree from Vanderbilt University.

If you're in the software development business, there's no doubt you've heard of, dipped your toe into or even experienced the joys and pains of Agile methodology.

The Agile Method promotes adaptive planning, early delivery and continuous improvement, and encourages rapid and flexible response to change. It's quickly becoming the gold standard for software development. And, as the Agile world has heated up, so has the pressure on testers to produce timely releases without sacrificing quality.

BUT WHAT TYPE OF AGILE TESTING IS RIGHT FOR YOUR ORGANIZATION?

In This Guide, You'll Learn:

- ✓ What Agile testing is and why it's an important part of the Agile methodology
- ✓ The different types of Agile testing methodologies and how they're different from a waterfall approach
- ✓ Our predictions on where the Agile movement is headed and what you need to know to stay ahead

WHAT IS AGILE TESTING?

It’s a collaborative, flexible and adaptive approach that requires early participation from testers to deliver working code faster.

With the fast pace of Agile, there is rarely time to test everything so requirements must be prioritized and testing automation is deployed where possible.

Adaptability is key, as features and requirements can change significantly from sprint to sprint. Exploratory testing is used to accelerate the time from code delivery to testing completion — and the goal is not documentation, but to create code that actually works.

This adaptability is why an Agile team needs broader, cross-functional skill sets, compared to a traditional waterfall tester. Testing needs to be collaborative, where communication is regular and fluid, and developers and testers work together to define how the approach to testing will move forward.

Agile testing demands that a complete testing toolkit is available and that testers keep up with the current trends in the space.

During Agile testing, it’s helpful to keep the four central tenets of the Agile Manifesto in mind to help guide the decision-making process.

01 — Individuals and interactions over processes and tools

02 — Working software over comprehensive documentation

03 — Customer collaboration over contract negotiation

04 — Responding to change over following a plan

SCRUM

Scrum is one of the most popular Agile development methodologies and one of the easiest transitions from the waterfall approach, since it's typically requirements-driven like waterfall but involves shorter iterations with higher degrees of collaboration.

OVERVIEW

Similar to waterfall, scrum is driven by a requirement or user story that defines how features should perform and be tested — but with one significant difference: scrum testing is a very iterative process, designed to deliver value in the shorter term.

In scrum, questions are answered up front to avoid the trap of building a product loaded with features, only to realize later that the priorities or requirements may have shifted. Plus, teams are better able to hit the moving targets of evolving customer demands in a complex marketplace, thanks to the flexibility scrum brings.

Overall, scrum is much more collaborative and iterative than waterfall, which typically requires many cycles of testing and bug fixing before a product can be released.

WATERFALL VS. SCRUM

Scrum relies on more frequent touch points between developers and testers, testers and BAs, and BAs and Developers to make sure any changes are properly communicated.

Scrum demands regular touch points across the scrum team, such as daily stand-ups and sprint retrospectives, to keep activities aligned and remove obstacles.

Strong project management is a key need in scrum, and the role of the Scrum Master is to constantly refine the schedule and direction and keep the team on track.

BEST PRACTICES

- ✓ A user story is received from a customer or sales representative and converted into acceptance criteria to minimize potential miscommunication.
- ✓ Code is created against the acceptance criteria, approved by the team and tested in one or many sandbox-like environments.
- ✓ Upon completion, code is tested in a production-like environment, staged into a build and deployed out into production.

KEY TEAM MEMBERS

- Product Owner
- Scrum Master
- Developers
- Automation Engineers
- Testers

KANBAN

Kanban is similar to waterfall in the creation of key stages with gates that are sequential, but specific phases are tracked on a feature level rather than a release level.

OVERVIEW

If you're trying to make a seamless transition from waterfall to Agile in your organization, Kanban may be the best choice, thanks to its ability to incorporate agility into your process without turning it upside down. It can be a good starting point for further transition into more collaborative methodologies like scrum and a bridge to create the close developer/tester interactions needed for TDD, ATDD, and BDD methodologies.

Kanban drives visibility into production and helps to identify bottlenecks in the process since it creates and opens up each phase of the process. This visibility enables you to quickly identify which steps in your process are hurting efficiency, giving you the tools to drive greater agility and throughput.

WATERFALL VS. KANBAN

Kanban requires planning on a feature level, meaning that team members must be ready to perform their duty at any given time (i.e. plan tests in the morning, write them at lunch, run them in the afternoon).

Kanban requires many frequent handoffs between team members with the goal of minimizing the time these handoffs take, so it is critical to keep status up to date and provide visibility to the rest of the team into progress.

BEST PRACTICES

- ✓ A user story is received from a customer or sales representative and converted into acceptance criteria to minimize potential miscommunication.
- ✓ Code is created against the acceptance criteria, approved by the team and tested in one or many sandbox-like environments.
- ✓ Upon completion, code is tested in a production-like environment, staged into a build and deployed out into production.

KEY TEAM MEMBERS

- Product Owner
- Project Manager
- Developers
- Automation Engineers
- Testers

TEST DRIVEN DEVELOPMENT (TDD)

With Test Driven Development — the quintessential Agile testing style — code is developed against automated unit tests rather than time-consuming and often misunderstood requirements.

OVERVIEW

While some organizations may struggle with the concept of this "backwards" methodology, this approach delivers one of the most collaborative and efficient product development processes possible.

In TDD, code is written to make the automated unit test pass and the test (rather than the requirement) drives documentation. Why is this important? In a typical requirements-driven process, a business analyst defines a requirement and hands it off to developers and testers for interpretation. Expectations are typically only aligned when the test is run against the code, often to find that the requirement wasn't fully specified or was misunderstood by some or all parties. With TDD, not only is the additional requirements documentation removed, but testers and other team members are involved up front so that collaboration can drive a more complete definition of the unit test driving the feature development.

WATERFALL VS. TDD

Waterfall "layers" that allow organizations to perform less-than-perfect requirements gathering are removed, forcing teams to be extremely accurate up front. This requires additional initial effort but results in a product that's more closely aligned to the needs of the customer.

TDD is often paired with a CI/CD deployment methodology that is rarely leveraged in a waterfall development environment.

TDD is best run with automated unit testing instead of a manual approach, requiring developers that have the necessary skills as well as a framework to develop and launch the tests.

BEST PRACTICES

- ✓ Automated (preferably) or manual unit tests are designed and written in advance.
- ✓ Once a test is complete, the developer is tasked with writing code against it until it passes, and when it passes only refactor code. With this method of leading with the test in mind, documentation is approximately a third of what it would be in a waterfall approach, and developer scope creep is minimized.
- ✓ Though there are many potential benefits of aligning with TDD, it's not for everyone. But for firms that are nimble — or want to become nimble — the hardwired agility built into a TDD approach is often too attractive to ignore.

KEY TEAM MEMBERS

- Product Owner / Business Analyst
- Project Manager
- Developers
- DevOps Manager / Release Manager
- Testers

BEHAVIORAL-DRIVEN DEVELOPMENT (BDD)

As a subset to Test-Driven Development (TDD), Behavioral-Driven Development (BDD) uses an initial requirement that's driven from end-user behavior, rather than a technically facing unit test.

OVERVIEW

As with most development processes, BDD starts with a functional specification typically written in a Given/When/Then format. With this process, however, consolidated specification functions as a guide for the developer, tester and product owner based on the behaviors the system should exhibit. This provides a nearly fool-proof guide for the automation engineer in generating the automated test the developer will code against.

This automated test is used as the benchmark to determine if the feature is complete, as it is in TDD. Typically, a test fails several times before passing and the feature is considered complete, at which point the developer is only able to refactor the code and not to add any additional functionality.

The efficiency driven from this smart automation strategy is one of the most intriguing aspects of BDD — and one of its key differentiators from other Agile methodologies.

WATERFALL VS. BDD

BDD relies on technical testing talent with knowledge of behavior-driven automation frameworks, neither of which are standard to most waterfall organizations.

BDD requires a different specification format (Gherkin, typically) than what is common is a standard Waterfall Business Requirements Document.

BDD is often paired with CI/CD development pipelines that are not common in waterfall development.

BEST PRACTICES

- ✓ The BDD process is kept lean by streamlining documentation, but it relies heavily on the collaboration of the product owner, developer and tester as a cohesive team often coined as the “three amigos.”
- ✓ Cucumber is often used as the leading automation tool. User behavior is initially described in plain text, step definitions are added and features are tested.
- ✓ Testing platforms are designed for simplicity and reuse of assets, giving testers the ability to leverage as many of the previous automation assets created as possible.

KEY TEAM MEMBERS

- Product Owner / Business Analyst
- Project Manager
- Developers
- DevOps Manager / Release Manager
- Automation Engineer / Testers

QASymphony

ACCEPTANCE TEST DRIVEN DEVELOPMENT (ATDD)

Similar to Test Driven Development (TDD), ATDD is a very Agile approach where tests are designed in advance and code is written to comply with the test. But these tests are more customer-facing acceptance tests than technically facing unit tests.

OVERVIEW

If you want a customer to purchase your product, it's not just functionality that matters — how end users perceive your product is as important as the features themselves. Empowering customer perception to drive product performance is the thinking behind Acceptance Test Driven Development (ATDD), eliminating the requirement-driven process waterfall. Instead, customer input is gathered up front and distilled into Acceptance Criteria, which are then translated into manual or automated Acceptance Tests that are developed against.

While it may seem challenging to define the application behavior in the early stages, building the product with the end user's needs in mind will result in a higher adoption rate than software developed under contrary methods. Additionally, it removes one more additional layer of potential misunderstanding – the disconnect between how the development team may interpret the use of the product vs. how the end user actually uses the product, reducing risk of delivering underadopted features or needing to rework features in later releases.

WATERFALL VS. ATDD

ATDD requires customer-facing roles (customer support, sales, etc.) to be involved up front in refining the acceptance criteria.

ATDD is most efficient when automating acceptance tests up front (though not required) and these skills are missing on some waterfall teams.

Tight collaboration between all facets of the engineering organization are required in ATDD, which is unusual for many waterfall organizations.

BEST PRACTICES

✓ ATDD tests against pre-determined acceptance criteria driven by customer demand, requiring close interaction with the customer base.

✓ The most important question that needs to be answered is "Will customers use the system if it does 'X'?", with the next important question being how can we validate the system does "X"?

✓ Team members who are close to the customer — such as account / support managers or even a focus group — can help develop the acceptance criteria that drives the acceptance test creation.

KEY TEAM MEMBERS

- Customer / Customer Advocate
- Developer
- Product Owner / Business Analyst
- Automation Engineer / Testers
- Project Manager

WHAT'S NEXT FOR AGILE TESTING?

The role of the tester is changing. Testers are not just focusing on rote test case creation and test execution anymore. They're tasked with providing holistic testing strategies, understanding automation, and building a greater base of knowledge around both development and product usage.

As more businesses use Agile development and testing automation, testers must adapt and become quality champions with a broader skillset and field of vision.

3 Things You Need to Know About the Future of Agile

01 — COMMUNICATE

Testers must become highly collaborative, integrated members of their team and act as champions of quality within their organization. They cannot expect their job to solely revolve around test case creation and execution, and should not expect any typical day in their new role.

02 — DIVERSIFY

Functional testers should become subject matter experts in the various testing methods beyond manual scripted execution (automated and exploratory) and fully understand all facets of a testing strategy. Agile teams may go through different cycles of needing one, two, or all three types of testing and being at least familiar with all three makes it much easier to contribute value throughout the application's lifetime.

03 — BUSINESS VALUE

One of the key tenets of Agile is making sure the customer gets the most value out of the application being built, and testers play as much of a part in that story as any other Agile team member. Leveraging exploratory testing is a great way to enter the head of a customer/end user and start to quickly understand their key desires and concerns with the application.

10 THINGS YOU NEED TO CONSIDER WHEN TRANSITIONING TO AGILE

1. EVALUATE YOUR SKILLSETS

Make sure that your team has the right mix of skills to help you succeed in a more agile landscape, and don't overlook the value of "soft" skills as well as the more technical ones.

2. DON'T OVERLOOK THE AGILE MINDSET

More than anything, shifting to agile is often about resetting expectations around roles, teams, and how they work.

3. IT WON'T HAPPEN OVERNIGHT

You can't just flip the switch and be agile - any good transition plan takes into account the time the transition will take to complete

4. GOING AGILE IS A LIFESTYLE CHANGE

You must factor in continued commitment to agility, otherwise you will be at risk of your change unraveling

5. AGILITY MUST BE A TOP DOWN AND A BOTTOMS UP DECISION

The day to day agile practitioners need to be bought in on the positive benefit of the transition and be champions within teams of the proposed change

6. BE READY TO LET YOUR OLD INVESTMENTS GO

Don't let the sunk cost fallacy trick you into believing you should slow down or risk your agile transition to get more out of old investments you made in waterfall processes and technology

7. TRACK YOUR RESULTS

Don't just trust that Agile will save your organization money - develop ways of tracking the bottom line impact of the transition so you can know when to invest more in particular agile change initiatives

8. LEAN ON (AND CONTRIBUTE TO) THE AGILE ECOSYSTEM

The software development community is full of 1,000s of agile thought leaders that are willing and able to support you in your transformation. Don't be a hero and feel free to rely on the community, and make sure to invest back in it once you gain some lessons learned

9. BE WILLING TO MAKE MISTAKES

As with any important business process change, there are sure to be mistakes along the way. Agile is all about educated experimentation, so be willing to make mistakes as long as you can identify when to change course, make those adjustments quickly, and take away lessons learned for the next time

10. THERE'S NO DEFINITION OF AGILE

Don't believe that Agile can only be implemented in one way. Agile is an approach to which there are many different methodologies, take the best of the ones that work for you and get rid of the ones that don't. Don't follow an Agile process just because a successful competitor or admired company does - do what works the best for your business and drives your bottom line results.

THE QASYMPHONY MISSION

TO PROVIDE TESTERS WITH THE TOOLS THEY NEED TO BE SUCCESSFUL IN AN AGILE ORGANIZATION AND ENABLE COMPANIES TO CREATE BETTER SOFTWARE.

With our robust suite of testing tools, we're the only provider of truly enterprise-level, end-to-end Agile solutions. Companies like Adobe, Barclays, BetterCloud, Cardlytics, Salesforce Marketing Cloud, AirWatch, Nordstrom and Vonage trust QASymphony to help them work smarter.

Gartner 2015 CoolVendor

QASymphony was named a Cool Vendor in Application Development by Gartner in 2015 and is headquartered in Atlanta, GA.

Learn more at www.qasymphony.com or call **844-798-4386**

Sign-up for a **FREE TRIAL** | Request a **FREE PERSONAL DEMO**