Agile Test Management

GUIDE

QMETRY

(408) 727-1101 info@qmetry.com www.qmetry.com

Table of Content

(408) 727-1101 info@qmetry.com www.qmetry.com

Agile Testing Guide

Agile testing is the software testing methodology that stems from the Agile software development principles. The essence of Agile testing practice is that it incorporates testing into the dev process, rather than keeping it a separate SDLC phase. In this guide, we will explore how applying the core principles of Agile software development, testers and QA teams can help improve the code quality ensuring faster releases and be more aligned to business goals.

But first, the case for Agile testing. In the race for digital transformation, the one constant that remains is delivering quality digital experiences at speed. To deliver this quality at speed paradigm, the software development and testing ecosystem embraced the Agile methodology.

Since then, Agile practices have become an important part of software development. The basic premise of Agile is higher productivity, collaboration, faster time to market and lower risk than conventional development approaches.

There are 4 foundational values defined in the **Agile Manifesto, supported by 12 principles that govern the practice**. In a nutshell, the practice emphasizes early delivery, continuous and incremental improvement, collaboration and quick responsiveness to change.

But over the years Agile practices evolved. There was a wider move towards DevOps practices. With that, there seemed a gap in testing and QA processes. Testing was still stuck in waterfall mode or not completely agile.

This led to some of the following questions. How does testing fit into the Agile framework? How to set up an Agile testing team? What are some different ways to adopt Agile for your testing teams?

(408) 727-1101 info@qmetry.com www.qmetry.com

Where Agile Testing fits in

Ideally, Agile testing begins at the start of the project with high and constant collaboration between dev and test teams. Agile testing may not follow the usual sequence but is continuous. The idea is to have a cross-functional Agile team that works as a single unit towards the common aim of achieving quality.

So, the central idea is to use an iterative testing approach to deliver quick results, shorten feedback loops and simplify testing though the quality life cycle. Agile testing needs collaboration, flexibility & adaptivity. Agile success is linked to the teams' ability to show their work regularly and collect feedback.

This fast pace of development and testing means testers must follow some ground rules:

- Prioritizing what to test. The most critical aspects are tested first, since everything can't be tested at once.
- Relying on test automation to increase coverage and productivity and achieve continuous testing.
- Continuous improvement by using data and past trends.
- Use exploratory testing to shorten the time from code delivery to testing and focus on a working code or working software rather than documentation.
- Communication and collaboration with developers to keep up with the rapid changes that occur in the lifecycle of a release.

(408) 727-1101 info@qmetry.com www.qmetry.com

Setting up an Agile Testing Team

Setting up a successful Agile team for QA requires adoption of continuous quality through collaboration, breaking of silos and automation.

There are other two tasks that need to occur in parallel: testing of new stories within sprints and regression testing. Regression testing pack needs to be implemented to get fast and valid feedback on the health of application in the face of ongoing changes. Deployment or build pipelines are vital to the success of your Agile practice. These decide how a story travels from product backlog to live production.

For successful Agile QA that enables frequent release of quality code, all stakeholders must abide by the deployment pipeline guidelines. Develop the pipeline based on best practices and include activities that are part of every stage.

Here are a few tips before you set up your team:

1. Appoint a leader
2. Assign responsibility and ownership to each individual
3. Establish testing best practices and development standards
4. Train extensively and frequently
5. Create an effective communication channel
6. Share demos and sprint reviews
7. Invest in the right tools and toolchain

An effective Agile tester doesn't simply ensure code success in developer environment but also in other environments after integration. Continuous Integration allows detecting and solving build problems early in the process so you can fix them without losing crucial project time.

Other than great communication and team skills, Agile testers should have requisite technical skills for code reviews, creation of user stories, requirements understanding and collaborating with developers on unit testing. It also involves a fair knowledge of automation at the time of unit testing and integration.

Over and above, agile testing teams will incorporate a wider range of testing tools and technologies into their checks from design to delivery and deployment. Since agile testers also work on several stories/features simultaneously, they need tools to organize and manage these multiple requirements.

(408) 727-1101 info@qmetry.com www.qmetry.com

Various Agile Terminologies and Approaches

Agile practices are all about fluidity and flexibility. As iterations progress, the requirements evolve; each iteration leads to an integrated working increment and is ready for user acceptance testing. The feedback received is then used as an input for future iterations.

Continuous Integration & Continuous Quality

Continuous Integration is necessary for Agile success because it helps you gain faster feedback and faster results. If a build is integrated and then it fails, you know exactly where the fault lies in the iteration. Frequent integrations allow you to be prepared for a release when required. Testing is an essential link for all phases of development to ensure continuous quality. There are several Agile methodologies that support this construct.

Scrum

Scrum is one of the most well-known Agile testing methodologies used by the vast majority. It is highly iterative and focuses on identifying the main features and objectives before each sprint. This is done to reduce business risks but provide value. Scrum is highly collaborative and demands frequent stand-ups and retrospectives to keep the communication flowing. For teams transitioning from waterfall methodology, Scrum is probably the better bet because it is time-based and you can plan out releases in advance.

Kanban

Kanban literally 'visual signal' from Japanese, has its origin in the manufacturing industry. It is a visual method of managing the work with a focus on continual delivery. A good example of Kanban is a Trello board that is split into different lists that visually present the status, capacity and stages of task completion. As opposed to Scrum's time-based iterations, Kanban is based on priority. Developers, testers pull the next tasks from the to-do lists as they are ready. The tasks are displayed on a Kanban board for all team members to access and prioritize work from the queue. It relies heavily on:

- Work in Progress (WIP) Limit
- Lead Time

However, Kanban is probably better suited for smaller teams or projects that are not highly time sensitive.

Agile Testing Methodologies

Test-Driven Development (TDD)

Test-Driven Development (TDD) is a coding practice that helps developers write new code once an automated test fails. The main goal of TDD is to bring more clarity, simplicity and accuracy to tests. It begins by designing and developing tests for every small function of an application before the actual coding begins.

Acceptance Test Driven Development (ATDD)

This takes TDD one step further where the entire team of stakeholders discusses acceptance criteria much early in the development process. ATDD is quite ideal for agile testing because it helps close the loops between product and dev teams, increases collaboration and brings efficiency.

(408) 727-1101 info@qmetry.com www.qmetry.com

Behavior Driven Development (BDD)

BDD uses the same principles as TDD but it goes for higher level tests at the business level instead of unit tests. BDD takes the ideal or accepted behavior and asks for tests that even non-developers can understand. It uses the 'outside in' approach – implementing only those behaviors that contribute to these business outcomes to minimize redundancy. BDD increases the scope of the feedback loop to include business stakeholders and end users who may not have software development knowledge.

Exploratory Testing

Exploratory testing is the simultaneous process of test design and execution as opposed to scripted testing (with preset procedures and processes). Exploratory tests don't follow a precise script decided in advance.

In other words, exploratory testing allows testers with the freedom to test the code in an exploratory and somewhat chaotic way. Exploratory tests are then complementary to automated tests, because they aim to find possible issues that are beyond the scope of automated tests.

Session based Testing

Session based testing is exploratory testing but with more structure and organization. Where exploratory testing is completely unscripted, there is a question of accountability and expertise of the tester involved.

Session based testing relieves some of these concerns by keeping the essence of exploratory testing with more time-boxed, uninterrupted sessions, testing against a charter and requiring testers to report on the testing that occurred in each session.

Eventually, these sessions are completed with a debrief between testers and managers to analyze the outcomes.

(408) 727-1101 info@qmetry.com www.qmetry.com

Importance of a Test Plan

An important facet of Agile testing is a Test Plan. Because unlike waterfall, the Agile practice needs test plans to be written and updated for each release. The test plan includes the types of testing done in a particular iteration.

What does a typical Agile test plan include?

Requirements and Testing scope

The key task here is to understand features and user stories — limited only to that iteration, defining the test strategy, scope and estimating the time to be spent.

Story or feature verification phase

This stage involves test execution for stories and features, bearing in mind that some of these might not be fully functional. The aim is to ensure all features work well independently and with other components.

Prepare to rescope

The very nature of agile development is fluid and change-friendly. The test plan is a living entity that gets updated often to reflect the various changes.

(408) 727-1101 info@qmetry.com www.qmetry.com

Levels and types of testing based on the feature complexity

It is necessary to have a QA build strategy and types of testing and test tools figured out. When should automated tests be developed? Who should undertake unit testing? What tools will be required for performance testing and automated testing? This planning is essential for Agile testing success.

Identifying risks and mitigation strategies

Test plan should prepare and plan for risks. For instance, a complex feature may require multiple execution paths and the time-frame may be inadequate to test the feature. The mitigation strategy in this case would allow allocating several resources for a short period of time to pick up the most likely execution paths and avert risks.

System verification and acceptance

Once most features are built, integration testing is conducted for the system. This ensures that new feature sets and various last-minute changes don't disrupt the system. Targeted regression testing is also crucial, as is planning time for defect validation. After the bugs are fixed during the regression phase, the system is more stable and a release candidate is created. Smoke testing is necessary prior to release to ensure that all critical defects are fixed and the system is whole.

Feedback and retrospective

Test plan should prepare and plan for risks. For instance, a complex feature may require multiple execution paths and the time-frame may be inadequate to test the feature. The mitigation strategy in this case would allow allocating several resources for a short period of time to pick up the most likely execution paths and avert risks.

Technical Debt

It is a fact that the increased velocity of testing opens your release to more crashes if you don't address quality. Technical debt should always be a part of the plan. What works best is to periodically address this debt as part of an iteration. This can include many things like documenting code, correcting requirements post-facto, fixing automation scripts etc.

(408) 727-1101 info@qmetry.com www.qmetry.com

Best practices for aligning Testing with Delivery

Agile transformation shouldn't stop at the Development Testing Divide. To ensure that testing is not treated as a separate phase from development and that it doesn't get pushed to the end of the window, make sure each iteration is not like a mini-waterfall.

The ideal Agile practice intertwines development and testing so that testing informs and directs the development rather than merely critiquing it. In the Agile world, testing is not the responsibility of a select few.

1. Using the whole team approach to include testers and QA managers as full members of the Agile Development team is necessary for agile success.
2. Testers need a seat at the table early on the development cycle for higher involvement and a better understanding of requirements and goals. Leverage the QA team as active contributors to the planning and requirement analysis.
3. Use the power of test automation to implement regression testing.
4. Ensure traceability within the requirements, test cases and bugs.
5. Use specialized skills of TDD, including continuous integration and unit level. Adopting these practices into the day-to-day will foster the quality of deliverables and reduce rework.
6. Reduce the time curve from requirements gathering to test creation. This is necessary to gather the momentum required for testing in the DevOps world.
7. Implement continuous testing.

(408) 727-1101 info@qmetry.com www.qmetry.com

Common Challenges in Agile Testing

Implementation challenges owing to the fast-paced release cycles

Shorter sprints in the era of DevOps require faster integration and better collaboration. There are several gaps between conventional test management and the Agile approach. Outdated tools and processes often weigh down the agile delivery cycles.

Continuous feedback

High frequency of releases to production requires a consistent quality of software throughout development. Testing early and testing often is the linchpin of Agile success. To get more value, and maintain the continuous feedback loop, collaboration and communication are necessary. This can be a challenge for larger teams or multi-location distribution teams.

Transparency and clarity of common purpose can enable continuous testing. For instance, writing unambiguous user stories, building consistent and relevant end-to-end test scenarios using trends, data and analytics.

Also make sure that each story has the right acceptance criteria and that the story context is unambiguous and well-understood by all team members at the outset. It is important to start authoring tests as early as possible, such that the feature can be tested as soon as it is available.

(408) 727-1101 info@qmetry.com www.qmetry.com

Test Automation

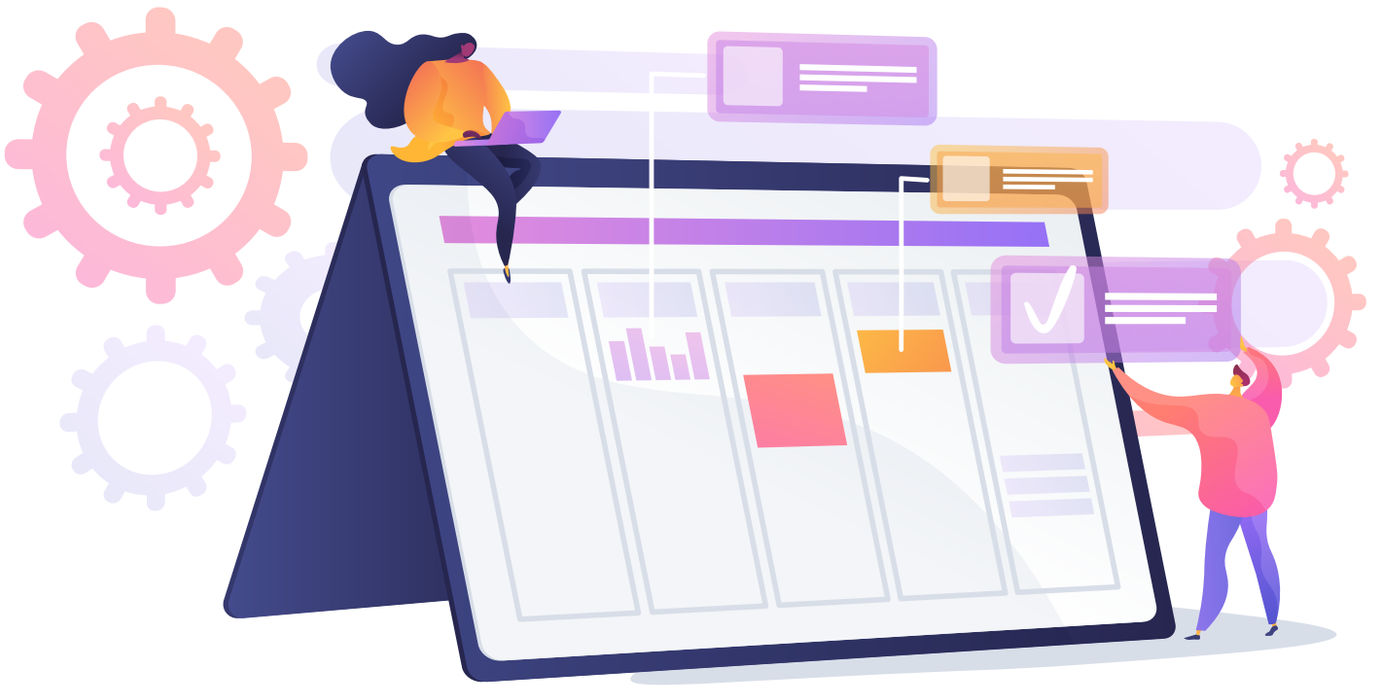
Continuous testing requires automated testing to shorten the feedback loop and enhance the test coverage. Knowing when to automate and how to automate is how the battle won.

Technical competence can be a challenge here for integration testing and API testing as well as scripting UI automation checks. If testers are from a non-agile background, they may struggle to keep pace with the continuous delivery cycles.

DevTestOps

The constantly blurring lines between development and operations, have changed the way products are developed and tested. There is a higher premium on speed and quality and to balance the speed/quality equation is easier said than done.

Many teams that adopt agile in theory, still allow for a considerable testing delay and leave less time for QA and improvement. This causes buggy releases and poor product performances, eventually making customers unhappy. This requires a fresh approach to agile testing that emphasises on automation, frequent test runs, better feedback mechanism, continuous integration and bi-directional sharing.



(408) 727-1101 info@qmetry.com www.qmetry.com

New Trends in Agile Test Management

Contemporary agile testing requires more synergy between tools and data. Contemporary practices require tools and solutions that allow easy plugging of automation, generation of API Keys that enable your tool to integrate the automation results.

Agile testing teams need to be able to create or import requirements with test coverage. This is informed decision making that lets you make go/no-go decisions. Modularity and link-ability of the toolset is also important.

Test management tools should enhance reusability and visibility providing a complete view of the test progress, with adequate reporting and smarter quality insights.

Agile testing solutions have now evolved not only to deliver continuous testing, but more intelligent testing powered by ML and AI. One of the smarter and more efficient ways to test involves the use of predictive and prescriptive QA. Tools led by AI and ML can harness the power of data to anticipate defects even earlier, optimize the test processes, predict outcomes and prescribe recommendations to best the current model.

(408) 727-1101 info@qmetry.com www.qmetry.com

The importance of choosing the right Agile testing tools

Modern tools like QMetry Test Management enhance reusability, traceability and are custom-made for Agile teams. Empowered by these, you can create test suites by release and tracking incremental gains in performance, build by build.

When choosing your Agile test management tool, look for solutions that are also available on the cloud. Tools that provide permissions and workflow, enable setting up roles and projects inside test management, provide complete visibility and integrate with your project management, automation suites and CI/CD tools. Features like cross project reporting, dashboard gadgets, ability to sync defects and test results in real time can be a real game changer for your efficiency.

**If you'd like to see effective test management in action,
then try our free trial version today.**

Start Free Trial

QMETRY

(408) 727-1101 info@qmetry.com www.qmetry.com