Agile QA Process: Principles, Steps, and Best Practices

The agile QA (Quality Assurance) process is a set of practices and methodologies aimed at ensuring that software developed within an Agile framework meets the desired quality standards. It aligns with Agile development principles, emphasizing collaboration, flexibility, and continuous improvement.

Why make the switch to agile QA?

Agile is one of the most popular methodologies in software development as agile projects tend to achieve superior outcomes, including higher adaptability, customer satisfaction, efficiency, quality, and team collaboration compared to traditional approaches in project management.

If agile development is already in place, it would also benefit your QA team to use the benefits provided by the existing agile environment to make your processes more efficient. When a QA team integrates with an existing agile development setup, they can benefit from its iterative nature by aligning testing within development cycles. This integration fosters collaboration, adaptability, and a focus on customer needs, ultimately making testing processes more efficient and effective.

But how is agile QA different from traditional QA methodologies?

Using the waterfall method, testing typically comes later in the development process, causing a significant delay before teams can start testing. As a result, when testing finally begins, teams often face a tough choice: either prolong the release date to ensure thorough testing or rush through testing to meet deadlines, risking the product's quality.

In agile QA, the QA team joins the Software Development Lifecycle (SDLC) from the start, a departure from traditional development methods called continuous testing. This early involvement enables swift incorporation of stakeholder feedback, allowing immediate adjustments.

Similar to how coding demands code reviews, continuous testing is crucial for ongoing issue identification in the product. Test automation can also help make the feedback process faster, especially when aiming to run repetitive and time-consuming tests like regression testing and functional testing.

Principles of agile QA

Here are some essential principles of agile QA:

1. Test early and often

The "shift left" approach, involving QA early in development and fostering mutual understanding between QA and development teams, cultivates a shared drive for higher product quality. Testing should occur continuously, accompanying every code enhancement, fix, and UI update rather than solely when introducing new functions. This frequent testing promotes ongoing quality assurance throughout the development cycle.

2. Automate what you can, but don't automate everything

Automation is essential to agile QA, but it still has costs and should not be done willy-nilly with no strategy. It's important to automate parts of testing that take time and are tedious, but that doesn't

mean that manual testing should be removed from the process entirely. Manual testing is still needed in cases like exploratory testing, which requires human thinking and curiosity to ensure that no edge cases are missed.

3. Provide continuous feedback and open communication

Create open communication channels to foster ongoing dialogue and nurture a culture that values sharing feedback and diverse opinions.

Consider conducting product demos with stakeholders to create a feedback-rich environment that supports continuous improvement and ensures that the product evolves in line with stakeholder expectations.

Create transparency within the testing process, fostering an environment where the team feels comfortable providing honest and constructive feedback. Emphasize that such feedback is valued and that the team operates within a safe space to share opinions. Encourage education over blame, promoting a culture where learning and improvement precede assigning fault.

4. Establish a culture of accountability and shared ownership

The project's success shouldn't rely solely on one individual; rather, each team member holds accountability for the project as a whole. This collective responsibility ensures shared ownership and commitment, fostering a collaborative effort towards achieving project goals.

5. Focus on the end-users

Ultimately, the product's success hinges on delivering value to the customer. Remember, it's the end-users—not your team—who will use the product. When determining testing approaches, prioritize user experience and product usability. Centering testing strategies around the end-user's needs ensures a product that truly meets their expectations and requirements.

6. Respond to change

Empower the team to adapt and remain flexible in response to unexpected changes. Cultivate a mindset that embraces agility, enabling the team to pivot swiftly and effectively navigate any alterations or challenges that arise during the project.

7. Self-organize

Enable the team to self-manage by assigning and tracking their own tasks and progress. This autonomy ensures that the team takes ownership of their work, promoting accountability and efficiency as they strive to deliver valuable and top-quality software.

Key steps in an agile QA process

Here are some key points to keep in mind when executing the agile QA process in correlation to the SDLC:

Planning

In the planning stage of the development cycle, engaging the QA team early is crucial. This involvement enables them to brainstorm possible risks of features and proactively plan what tests could be executed in the test execution cycle.

Creating well-documented, reliable, agile test cases is essential. Early collaboration empowers the QA team to plan effectively, anticipate challenges, and devise risk mitigation strategies.

Execution

Instead of acting as adversaries on separate teams, developers and testers must collaborate in finding and resolving bugs. In some cases, pairing up developers and testers could be conducive to making the agile QA process better in the sense that both sides could share their knowledge with each other in order to develop high-quality software. Additionally, after the test execution cycle is done, prompt reporting of discovered bugs and thorough analysis using proper QA metrics is essential.

Continuous improvement

As the project progresses, the QA team also needs to be able to adapt to constant changes. In order to do this, having a regular review of the QA process and holding retrospectives to reflect on each sprint and take corrective actions is helpful. Here are some questions to consider during retrospectives:

1. What went well during this sprint?
2. What challenges did we encounter?
3. How effective was our collaboration as a team?
4. Did we meet our sprint goals? If not, why?
5. What improvements can we make for the next sprint?
6. Were there any blockers or bottlenecks we faced?
7. Did our processes and strategies work effectively?
8. How can we better support each other as a team?

Communication and collaboration

Maintaining open communication between the QA team, development team, and stakeholders is crucial. It's essential to track their feedback and consider valuable insights. If needed, be ready to adapt requirements to ensure they align with solving the end user's problems and meeting the

ultimate project goals. This adaptability fosters a more user-centric and effective development process.

Agile QA methodologies

There are several agile QA methodologies that the testing team could utilize depending on their needs. Among the most popular are the following:

1. **Test-driven development (TDD)**: Here, code is written after unit test cases are created and then optimized later on.
2. **Acceptance test-driven development (ATDD)**: Follows a process where code creation occurs after developing acceptance tests, which are aligned closely with the project requirements. Unlike unit test cases in Test-Driven Development (TDD), where tests are more focused on code functionality, ATDD prioritizes creating tests based explicitly on acceptance criteria and later optimizing the code to meet these predefined criteria.
3. **Behavior-driven development (BDD)**: This type of methodology involves running tests to ensure that the system behavior meets the requirements every time

Measuring agile QA success

Quantitative and qualitative QA metrics offer ways to gauge the effectiveness of the QA process. Organizations may opt for different metrics based on their specific situation or strategies. These metrics serve as benchmarks to assess the performance and success of QA practices, allowing for tailored evaluations aligned with organizational goals and objectives.

The following table lists QA metrics that can be used to measure agile QA success:

QA Metrics	**Formula/What they measure**
Test Effort	The formula for calculating Test Effort can vary based on how you define and measure effort. Here are a few ways to represent it: **Hours spent:** Test Effort = Total hours spent on testing tasks by the QA team **Percentage of total project effort:** Test Effort = (Hours spent on testing / Total project hours) * 100 **Test case coverage vs. Effort:** Test Effort = (Number of test cases created + Number of test cases executed) / Effort expended in creating and executing tests **Defect density vs. Effort:** Test Effort = Number of defects found / Effort invested in testing

	Automation ratio: Test Effort = (Hours spent on manual testing – Hours spent on automated testing) / Total hours spent on testing.
Test Effectiveness	(Bugs detected in 1 test / Total bugs found in tests + after release) x 100
Test Coverage	(Number of tests run / Number of tests to be run) x 100
Requirements Coverage	(Number of requirements covered by tests / Total requirements) x 100
Defect Density	Number of Defects / Unit of Measurement refers to the specific parameter or metric used to quantify the size, volume, or scope of the software being measured. Examples include: The number of lines of code in the software, The number of individual test cases designed for the software, or the number of distinct modules or sections in the software.
Defect Distribution	Identify components with the highest bug density
Defect Turnaround Time	Time from bug discovery to resolution
Customer Satisfaction	Measured by a set of key performance indicators (KPIs)
Defect Leaks	Bugs found in production or UAT / Bugs found in testing

While this is not an exhaustive list, it is meant to help teams ensure that their current QA processes are effective for their organization.

Best practices for implementing an agile QA process

The following is a list of best practices to be considered when implementing an agile QA process:

- Implement test automation for repetitive tests that are tedious and time-consuming when done manually
- Ensure collaboration between development teams and QA teams by keeping communication lines open and building trust and transparency within the team

- Make use of acceptance criteria as the standard when providing feedback
- Make use of continuous integration/continuous delivery (CI/CD) pipelines and other devops tools in order to help with iterative development and continuous testing
- Make use of agile QA metrics to provide value to the product
- Conduct product demos with the stakeholders in order to get feedback on how to improve product quality

A test management tool like TestRail can provide advantages for agile QA:

- **Test case management:** TestRail allows easy creation, organization, and management of test cases. For agile teams, this means efficiently outlining test scenarios and ensuring comprehensive coverage across iterations.
- **Visibility and collaboration:** TestRail offers a centralized platform for teams to collaborate, ensuring visibility into test execution, results, and progress. This facilitates seamless communication among cross-functional agile teams.
- **Traceability**: TestRail enables linking test cases to user stories or requirements, ensuring traceability. This helps agile teams understand the test coverage for each feature or requirement.
- **Test execution and reporting**: TestRail supports test execution, allowing teams to run and report tests efficiently. Agile teams can generate comprehensive reports, providing insights into test results and progress.
- **Adaptability**: TestRail's flexibility allows agile teams to adapt to changing requirements easily. It accommodates alterations in test cases or plans as iterations evolve.
- **Integration with agile tools**: TestRail integrates with various agile project management tools like Jira, Trello, or Asana to streamline workflows. These integrations ensure smoother synchronization between test management and agile development activities.

Bottom line

Agile QA processes accelerate, strengthen, and enhance QA practices within an organization. They align with agile methodology striving for rapid, iterative delivery to receive swift feedback. The focus is on consistently delivering valuable product increments in each iteration.

While any process has its challenges, the advantages of agile QA outweigh the risks. Considering the potential benefits, implementing agile QA could be a valuable step for organizations seeking to enhance the quality of their products or services.

To learn more about how TestRail can help you successfully implement an agile testing strategy, check out TestRail Academy and take free multimedia courses on topics like agile testing, the fundamentals of testing, test automation, and more!

Agile QA FAQs

Agile QA Roles

In Agile QA, the development and QA teams collaborate closely rather than operating in isolation. Here's a breakdown of the distinct roles within the integrated team:

QA managers

Instead of simply managing the QA team as a team lead, QA managers are treated as subject matter experts who provide testing guidelines, standards, and methodologies.

QA lead

With the help of other members of the QA team, the QA lead has to decide on what test management tools to use, the testing approach, the testing strategies, and the best way to have a productive workflow for the testing team.

Developers

Having developers participate in testing is critical to agile QA for several reasons:

- **Faster bug identification:** Developers understand code deeply, spotting issues quickly.
- **Improved communication:** Bridges the gap between teams and clarifies requirements.
- **Immediate corrections:** Developers address misunderstandings promptly.
- **Shared quality responsibility:** Developers take ownership of product quality.
- **Optimized testing:** Developers focus on targeted and efficient tests.
- **Quick feedback loops:** Early detection of issues in the development cycle.
- **Enhanced test automation:** Developer involvement improves automated testing practices. The development team could work on unit and integration tests by writing automation code for white-box testing.

Automation testers

These testers will write coding scripts for automation tests like regression tests, end-to-end tests, visual tests, etc. Automation testers could also coach the developers on how to approach unit testing and integration tests.

Manual testers

One team member could be both an automation tester and a manual tester (e.g., QA engineers) but manual testing requires a more hands-on approach than automation testing.

While automation testing requires writing code, manual testing only requires human thinking and curiosity—and could be used for scenarios like exploratory testing and acceptance testing.

Product owner

Product owners represent the stakeholders and communicate their requirements to the team. They prioritize features, manage the product backlog, and ensure the developed software meets the business needs.

Product owners are integral in guiding, validating, and accepting the output of user acceptance testing (UAT), ensuring that the developed product meets the envisioned business outcomes and fulfills stakeholder expectations.

Common challenges in agile QA

In agile QA, several common challenges can arise:

Being averse to changes

Teams unfamiliar with Agile might resist change due to the flexibility these methods demand. Early adoption of an Agile mindset is beneficial, emphasizing the ultimate goal of delivering a valuable product over rigid adherence to specific processes. This mindset shift can ease the transition and prioritize the end goal of value creation.

Budgeting risks

Resource allocation could be at risk when adapting to new features and changing priorities. Make sure to adopt flexible budgeting approaches in order to minimize the impact on the project.

Scheduling risks

In Agile, test execution cycles are brief, leaving less time to create detailed test plans and conduct extensive testing. Schedules can change suddenly, requiring the team to be ready to manage risks effectively and adapt quickly.

Possibility of scope creep

While it's tempting to create a product that caters to every customer's needs, there's a risk of the product becoming something the primary user doesn't require. It's crucial to define the project's scope—what's included (in-scope) and what's not (out-of-scope)—to prevent scope creep. This clarity ensures the product aligns closely with the intended user's needs without unnecessary additions.

Lack of documentation

Agile methodology prioritizes functional products over extensive documentation, potentially leading to a scarcity of documentation within the team. This shortage might elevate the risk of scope creep and other challenges. However, it's vital not to overlook documentation entirely. Instead, emphasize creating documentation that adds tangible value, focusing on essential details crucial for understanding and maintaining the product.

Benefits of agile QA

Here is a list of some of the benefits of implementing an agile QA process:

- **Rapid bug detection:** Agile teams emphasize frequent, iterative releases for early and frequent testing. Agile QA aligns with this approach by aiming to test these releases as soon and as often as possible. This strategy enables the development team to swiftly address any bugs discovered, releasing fixes without delay.

- **Fast feedback loop:** There's no waiting around for dependencies. Agile QA eliminates the blocker of having to wait for the software development process to be finished. Once a deliverable can be tested, the QA team gets to testing immediately.
- **Highly collaborative:** With early involvement in the process, the QA team not only identifies issues but also offers insights for enhancing the product or refining the development process. This proactive engagement allows them to provide valuable suggestions for improvements.
- **Flexible:** As priorities change within the organization, agile QA teams can shift their focus and adapt to new problems quickly.
- **Optimized resources:** If there is potential for bottlenecks or blockers, Agile QA teams are able to reallocate or re-prioritize their resources.
- **Centralized software testing tools and processes:** Centralizing the software testing process and tools doesn't imply rigidity; rather, it signifies having established standards and tools ready for immediate testing implementation. While there's an existing framework in place, it doesn't hinder the ability to adapt or evolve the testing process as needed.
- **Lower costs**: Involving the QA team as early as possible in the process assures that potentially high-cost bugs are detected and fixed early on.