The Case for Agile Testing

By applying principles of Agile software development, QA teams can help bake in code quality as applications are being developed, ensuring more stable and business-aligned outcomes.

Executive Summary

For a variety of reasons (time, money, skills, etc.), the testing phase of software development sometimes gets short shrift from developers and IT managers. Yet testing is the only way to determine the quality of software and whether an application will function properly in real time. It plays an important role in the software development lifecycle (SDLC). Without an effective testing strategy, companies sometimes blindly take on significant risks that may go well beyond simply having poorly functioning software. Organizational ramifications can be dire, including the risk of:

- Customer dissatisfaction.
- Brand injury.
- Competitive threat.
- Product revenue loss.

At the development level, an oversight in testing can trigger countless technical and project outcome challenges:

- Cost overruns.
- Poor resource estimations.
- Schedule overruns.
- Overlooked functionalities and scenarios triggered by poor developer/tester communication, resulting in compromised quality.

Agile testing mitigates these risks and ensures that IT organizations achieve the technical and business goals of the development process. In this white paper, we provide an overview of how Agile testing works, why it is important, and the challenges it poses.

Agile Testing Defined

Agile testing is a software testing practice that follows the principles of Agile software development. Agile development integrates testing into the development process, rather than keeping it as a separate and distinct SDLC phase. Agile testing involves a cross-functional Agile team actively relying on the special expertise contributed by testers.

Agile teams use a “whole-team” approach to “bake in quality” to the software product. Because testing occurs in real time, this approach allows the team to collaborate actively with the development team, helping them identify and transfer issues into executable specifications that guide coding. Before the start of testing, the development team’s detailed design documents must be approved so that test case creation can start. Both coding and testing are performed incrementally and iteratively in Sprints (or iterations), building each feature until it delivers enough stability and adds quality to the product (see Figure 1).
Sprinting Ahead

In the Agile methodology, a Sprint is a set period of time during which specific work must be completed and made ready for review.

Each Sprint begins with a planning meeting. During the meeting, the product owner (who requested the work) and the development and testing teams agree upon exactly which activities will be accomplished during the Sprint. The development team has the final say when it comes to determining how much work can realistically be accomplished during the Sprint, and the product owner has the final say on what criteria need to be met for the work to be approved and accepted.

The duration of a Sprint is determined by the Scrum master, the team’s facilitator. Once the team reaches a consensus on how many days a Sprint should last, all future Sprints should be the same. Traditionally, a Sprint lasts 30 days.

After a Sprint begins, the product owner must step back and let the development team do its work. During the Sprint, the team holds a daily stand-up meeting to discuss progress and brainstorm solutions to challenges. The project owner may attend these meetings as an observer but is not allowed to participate unless it is to answer questions. The project owner may not make requests for changes during a Sprint and only the Scrum master or project manager has the power to interrupt or stop the Sprint.

At the end of the Sprint, the team presents its completed work to the project owner and the project owner uses the criteria established at the Sprint planning meeting to either accept or reject the work.

The Mechanics of Agile Testing

The combined team, including both development and testing, takes responsibility for analyzing the business specifications. Together, they define the Sprint goal.

The QA team defines the testing scope (i.e., test plan). That is then validated and approved by the entire team and the client.

Simultaneously, while the development team starts the implementation of modules (in the very first Sprint), the QA team begins work on the test case design. These are properly documented either in a testing tool or in an Excel spreadsheet that is handed over to the development team and project sponsor from the business side to review. This is to ensure that test coverage is as complete as possible.

Once the test case review and any modifications are completed for a particular Sprint, the QA team then begins testing on the QA environment. Defects found during testing are logged properly in a defect tracking tool. Depending on the severity and priority of defects, fixing them can be delayed but then is taken care of in upcoming
Sprints. At the end of each Sprint, the team determines, along with the project sponsor, which defects are to be fixed in the current iteration. This iteration continues until all planned Sprints are completed.

QA, along with the development team and business organization, defines which main flows (test cases) will be automated. When code is ready to test (after the end of each Sprint), QA works with development to execute test cases on the development environment, in order to identify the early stage defects so developers can fix them during the next round, on a priority basis. This process is then repeated throughout the development process. Automated test cases are run daily throughout the SDLC.

White-board/stand-up meetings are conducted daily involving members of all teams associated with product development, support and testing. This helps to resolve the issues faced by team members and provides a clear picture of progress in both the coding and testing areas. Agile promotes the introduction of requirements at all stages/iterations of the SDLC; however, the testing team determines when to end this process to ensure product stability (see Figure 2).

**Manual Testing: Regression**

The manual testing team plays an important role in determining product quality. Once the majority of coding is completed (i.e., in the last planned Sprint), the manual regression test cycle starts. This is key since it helps to determine the stability of the application before it is pushed to preproduction deployment. Code and defect fixes continue during this phase, but this is to add value to the product. Preproduction deployment is dependent on the approval of the test manager’s testing report; deployment only occurs if the report depicts green status for all application modules.

Although a dedicated testing team exists in the Agile environment, unit testing by an individual developer is beneficial to find early defects and minimize the rework of both the testing and development team.

**Automated Testing: Regression**

Automation is a critical component of Agile testing. It would otherwise be impossible to keep pace with the Agile development schedule. Automation is also used to run regression testing. The combined team (developers, product owners and testers) usually predetermines, at the start...
of the project, which parts of the software will be tested using automation.

Continuous integration/builds, unit, functional and integration test execution as well as continuous or automated deployment are common areas where automation may work better than traditional tests.

The entire project team agrees up front about which of the main flows will be automated. Automated tests consist of unit tests, capable of verifying even the most minute segment of software. Automation is required primarily to determine the stability of modules developed in each Sprint. Any defect found is reported and fixed, based on its priority.

Furthermore, it is possible to execute the test set multiple times per day, per hour or even more frequently if needed.

**Best Practices in Agile Testing**

Effective Agile projects generally address the following important elements:

- Promote the importance of testers and encourage continuous feedback sharing with the programmers and the customer.
- Testers actively participate in meetings to define the main business flows.
- Testers work on short iteration activities alongside developers.
- Encourage traceability between the requirements, test cases and bugs.
- Testers contribute to user story improvements.
- Leverage the specialized skills of test-driven development, including unit testing, continuous integration and unit level.
- Leverage automation testing as a key way to do regression testing.

**Agile Testing Advantages**

The benefits of Agile testing include:

- Testing requirements are discussed and refined as a team (during stand-ups/Scrums) allowing the combined team to better address the business/technical aspects of the requirement. This enables overall alignment and prevents misunderstandings.
- The Agile process often requires having entry and exit criteria for Stories (a compression of
things to do in a particular release/iteration). Agile testing ensures that each requirement is well defined and measurable, allowing QA specialists to determine whether the requirement was actually completed or not.

- QA participates in the big-picture requirements-writing stage, thus ensuring that testing estimates aren’t overlooked.

- Automated tests are fully leveraged to implement regression testing.

- Quality becomes the combined team’s responsibility, rather than just solely that of the testing team. The entire team agrees to the testing strategies, test cases and defects prioritization plan.

**Agile Testing Disadvantages and Mitigations**

Agile testing proves to be the best testing methodology only if the requirements are clear to the project sponsors. If the big-picture requirements are unclear, the details can become muddied. For new products, the software architecture normally takes a path based on the initial requirements. If the requirements frequently change (as allowed for in Agile), the following scenarios can occur:

- **The team struggles to adapt to changes** because significant effort has already gone into the initial requirements development and testing process.

- **Challenges in estimations and sizing requirements.** Sometimes QA gets short shrift since it’s logically the last task in completing the user Story. Therefore, any delay in the prior development task risks impacting QA timelines.

- **QA is sometimes prevented from executing a test case for the whole iteration,** leaving the team struggling to finish the task.

- **Not asking the right questions.** It is very dangerous for QA not to ask questions, especially at the point where the user Story is picked up for implementation. Daily team meetings can avoid this problem.

- **Addition of new user Stories into the current iteration.** QA should be included in the addition of the new user Story, to build up appropriate commitments and estimations in order to avoid misalignment and protracted timeframes.

**Looking Forward: Getting Started with Agile**

Kicking off an Agile testing project is easy. Where to dive in depends on organizational maturity and skills, among other factors. Regardless of how your organization stacks up, the following guidelines can be applied:

- **Tackle things one day at a time.** A good first step might be to start having a daily stand-up meeting.

- **Attack it incrementally.** Now that the team is communicating a little better, start breaking things into iterations. Your goal is to develop each feature, unit test it, QA it and accept it within the two weeks.

- **Plan just as much as you need to.** The Scrum master, product owner and each developer, QA and writer should attend (along with usability, architects, etc.). The goal is to enumerate what will be accomplished in the iteration as well as what the acceptance criteria are.

- **Review what you’ve accomplished.** End the iteration with a demo and a retrospective.

- **Integrate frequently.** Continuous integration will likely be key. Integrate at least daily. Stay as releasable as possible.

- **Focus on what is most important.** Another key area (which takes a while to master) is backlog, or the ranked list of features/Stories that the team will address. These are ranked by priority (most important first). This is key—even if in some cases the order between two items is arbitrary. At least once, the organization should iterate and reevaluate the rankings. Adding, removing and shuffling based on knowledge gained is a good thing.

**Footnote**

1 www.agilealliance.org/the-alliance/the-agile-manifesto.
References


About the Author

Naziya Iqbal Sayed is an Associate within Cognizant’s Quality Excellence and Assurance Business Unit. Over the last three-plus years, Naziya has provided quality analysis for banking and financial services clients. She earned a B.E. in information technology from Dr.B.A.M. University, Aurangabad, Maharashtra. Naziya can be reached at Naziya.Sayed@cognizant.com.

About Cognizant

Cognizant (NASDAQ: CTSH) is a leading provider of information technology, consulting, and business process outsourcing services, dedicated to helping the world’s leading companies build stronger businesses. Headquartered in Teaneck, New Jersey (U.S.), Cognizant combines a passion for client satisfaction, technology innovation, deep industry and business process expertise, and a global, collaborative workforce that embodies the future of work. With over 75 development and delivery centers worldwide and approximately 178,600 employees as of March 31, 2014, Cognizant is a member of the NASDAQ-100, the S&P 500, the Forbes Global 2000, and the Fortune 500 and is ranked among the top performing and fastest growing companies in the world. Visit us online at [www.cognizant.com](http://www.cognizant.com) or follow us on Twitter: Cognizant.