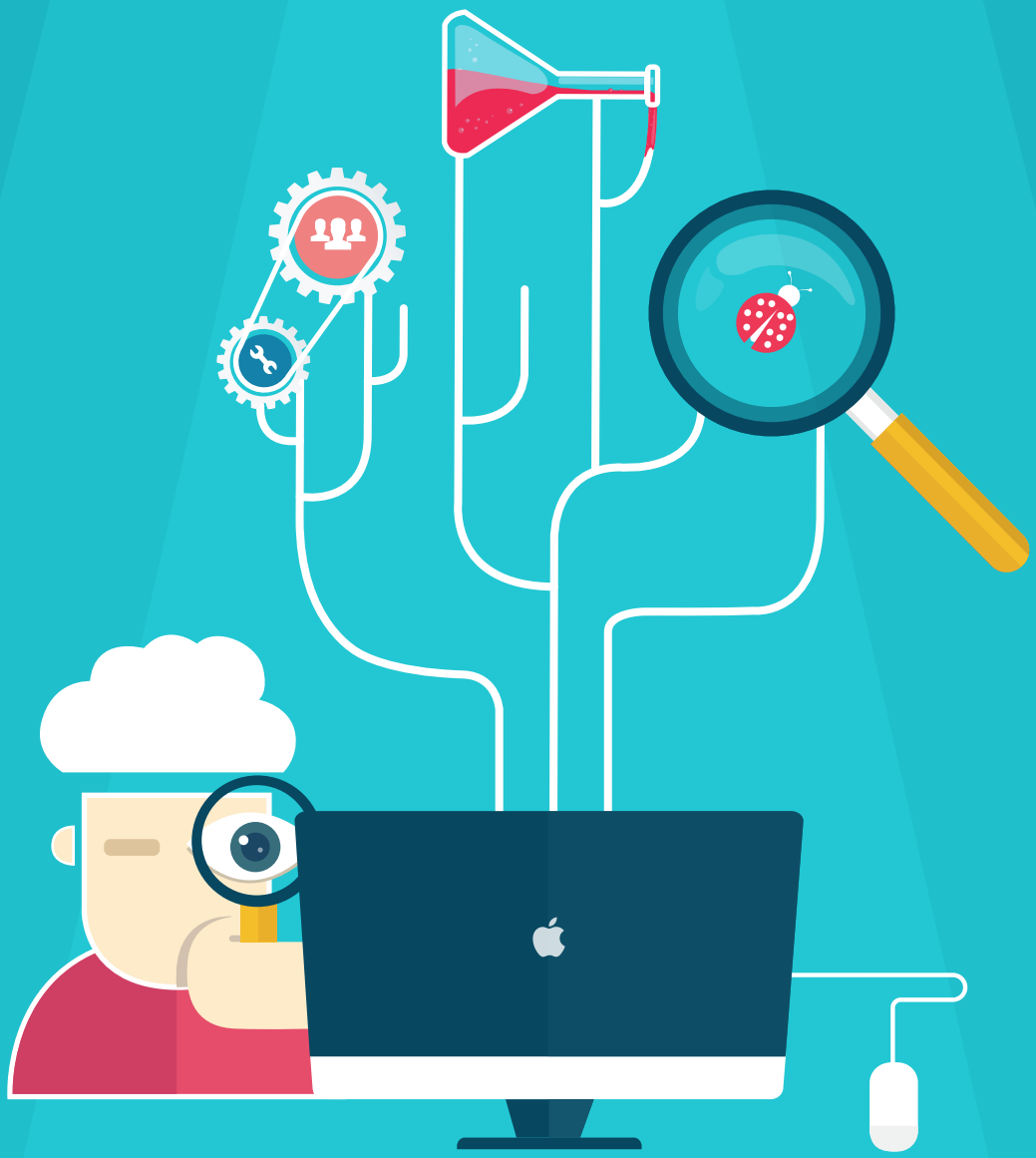


Best Practices in Agile Testing

Many businesses are moving from the waterfall model to the Agile model of software development. However, many of them, attracted to Agile by the potential advantages it offers, fail to grasp or implement the concept properly and end up not gaining any significant benefits from the process.

INDEX

Suyati℠
dream. dare. do

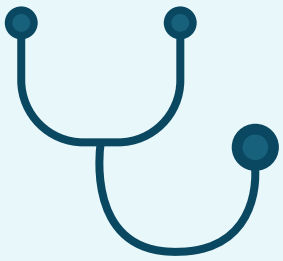
"Intro"duction

"**Agile**" is an abstract concept, and "Agile Software Development" has no precise definition. However, the "Agile Manifesto" offers a good framework of key values and principles on what exactly can be constructed as "agile." In a broad way, Agile software development process is an iterative and incremental approach to software development, ideally performed using an effective governing framework by self-organizing teams in a highly collaborative way, to generate high-quality solutions in a cost effective and timely manner.

The major motivation behind taking the Agile route to software development is its facilitating an iterative approach where development takes place in stages - with one working phase released before moving on to the next phase. This allows not just flexibility to add or change requirements mid-way into the development project, valuable in today's highly fluid business environment, but also predictability in delivery schedules and costs. Another key motivation is the high degree of collaboration between the client and project team, providing more opportunities for the team to truly understand the client's vision.

Agile Testing

A big misconception associated with the Agile development method is that it does not place much importance to testing. The traditional waterfall model of software development has testing as a separate activity at the end of the development process. In Agile, there is no such distinct process. Rather, testing is ingrained into the development process. This offers significant advantages but also poses several challenges.



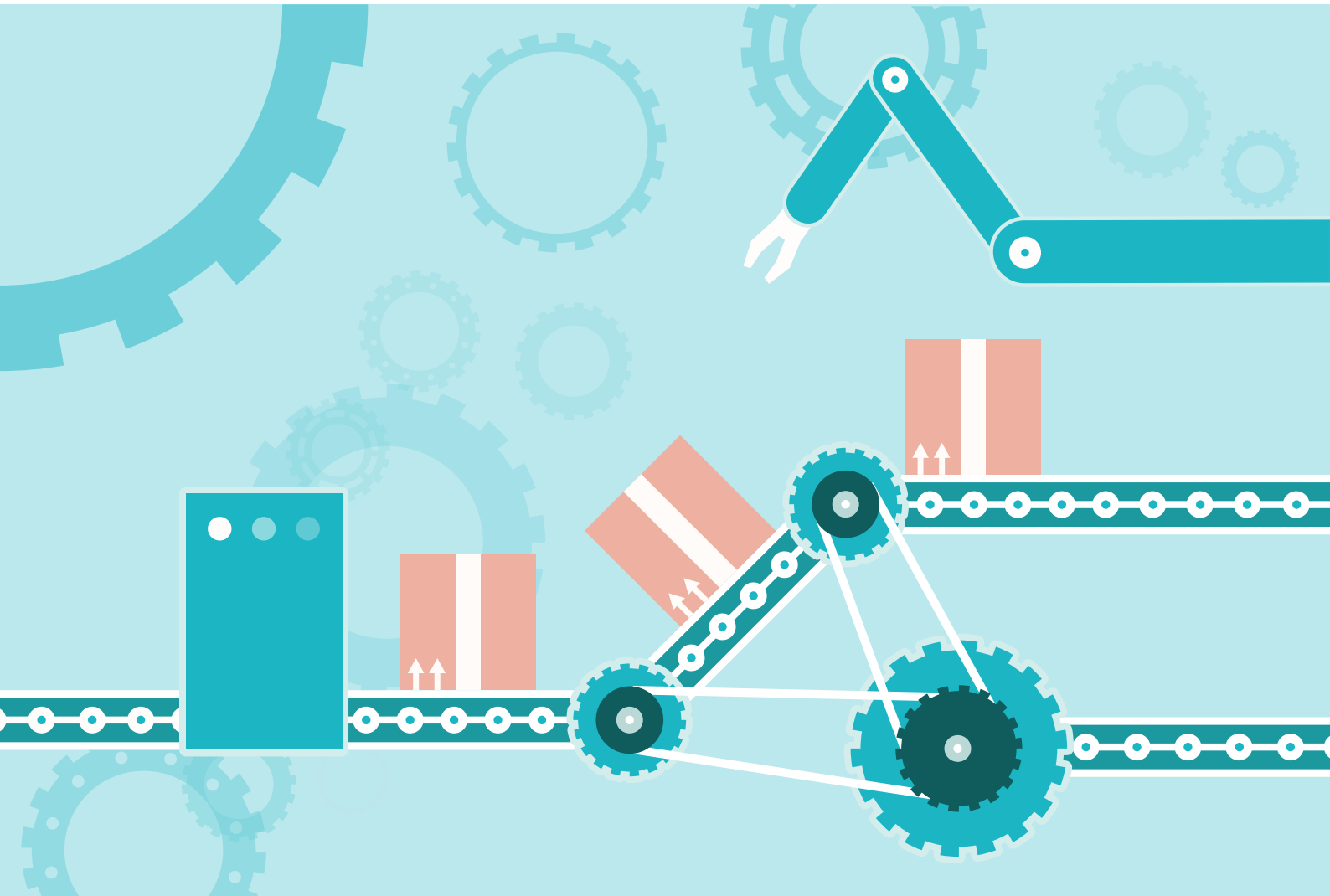
The ultimate aim of any agile process is to deliver on time, in scope, on budget, and at the right level of quality, and agile tests see to these ends. However, the specific scope of testing varies depending on the stage. At the inception stage, agile testers review various models, reconcile stakeholder goals and objectives, and set up appropriate tools. During every construction iteration that produces workable software, testers work side by side with developers in a single team, taking a proactive approach to trying to avoid problems or glitches rather than fixing it at the end.

A standard testing phase in the waterfall model takes anywhere between three to six months after the development team completes coding. In agile, testing takes place simultaneously with coding, eliminating the dedicated test cycle and in the process saving at least 25% of the cycle time. While there are no rigid rules on how to go about such integrated agile testing, adopting these time tested best practices would help the development team to get the most out of the process and resolve the key challenges.

Testers in the traditional waterfall approach of software development primarily make sure that the code is bug free, and act as gate-keepers of quality. In Agile, where there is no distinct testing phase, the entire development team shares these responsibilities. This, however, pose a challenge related to role clarity - what exactly are the "testers" supposed to do? The best approach for the tester in such a scenario is to act more as a facilitator, by learning new things, seeking to improve upon the previous, and using the acquired insights to help the team succeed.

"Adopt a Facilitative Approach"

Overall, agile focuses on customer usage over technical implementation and on uncovering flaws rather than confirm completeness. The testing needs to confirm to such an approach as well.



"Place the Customer First"

While agile is all about a facilitative approach and flexibility to make changes, this comes with a significant risk of ambiguity. Adopting an approach of writing test cases helps to resolve such ambiguity in a big way.

In a typical Agile development approach, programmers write automated unit test cases. The basis of such test cases is breaking down the required features of the iteration into test requirements. The "testers" in the development team write test cases in a way that "developers" in the team may design the code in a way that no defects are reported. In such an approach, the best practice is for testers to place the customers first - or understand the actual needs of the end-customers before writing test cases. This will allow developers to design the code in a way that the end customer does not report any defect. Thus, the best agile way is a proactive approach of pre-empting any defects or bugs, rather than trying to correct defects and bugs after developers have finished writing the code.

This approach of placing the customer first and understanding their requirements will not work unless the development team sets up a robust feedback and communication mechanism with their targeted customers upfront. The "testers" in the development team would also do well to hone up their communication skills, to effectively communicate with the customers.

Keep Track OF Requirement Changes

A big advantage of the incremental approach adopted by agile is the ability to make changes almost at will, even when the development process is underway. While this helps businesses in a big way, it poses a major challenge for the development team in general, and testers in particular.

Success depends largely on those responsible for testing keeps themselves up-to-date on the latest requirements, and this requires it instituting a robust communication mechanism. A daily meeting, even when different members of the team are spread over different time-zones would help in a big way.

Testers also need to consider the big picture while co-opting such requirement changes. Conventional "waterfall" testers are primarily concerned with the integrity of the code and what it achieves, and are not too concerned with the purpose or need of such a code. Agile testers, however, have an additional responsibility of considering the big picture, or how a specific iteration that is being reviewed fits into the grand scheme of things. Specifically, testers need to envision the impact that any feature may have on other parts of the system, and thereby pre-empt issues.

Perform Regression Testing

One risk with Agile's incremental approach is the possibility of existing features that work properly being broken on addition of new features. To mitigate such risks, it is important for agile teams to not confine the testing to the newly added features, but also undertake regression testing to ensure that the existing features still work. Failure to do so, or taking shortcuts can affect quality greatly and even sap the vitality of the code.

However, completing both regression testing and testing of new features in a short incremental cycle remains a big challenge. The problem is compounded by the regression test suite increasing with each cycle. The best of intentions notwithstanding, it is practically impossible to keep pace with regression testing if it is performed manually. To overcome the challenge, testers need the skills to automate routine tasks, learn new scripts, and adopt newer tools.

Undertake PERFOMANCE and Load TESTING as well

Many agile teams, owing to paucity of time or pressure to release the iteration fast, focus only on functional testing and ignore performance and load testing. This is a big mistake, and opens up the risk of the software being afflicted with performance and load problems post-release.

Here again, automation would make such tasks feasible, but the tester needs to be aware of what to test in the first place. The best Agile tester has skills and knowledge of different types of testing - functional, exploratory, end to end, usability, load, and performance, and apply them as needed.

USE the Right Tools

It is a good idea to use various tools to improve the quality of code, and also make the task of reviewing the code easy.

- Version Control tools such as ClearCase and others make explicit the sections of code that have changed, and which requires review. Such tools also control the versions of the test automation scripts.
- Integrated development environment (IDE) tools such as Eclipse, NetBeans, and IntelliJ IDEA, highlights errors while writing code, facilitating the writing of error-free code faster.
- Build tools such as Ant and Maven offer easy ways to report and document build results, and easily integrate it with automation and test tools.
- Code coverage tools such as Cobetura for Java makes explicit the degree to which the source code of a program has been tested.
- Unit level tools such as JUnit for Java and NUnit for .Net automate unit testing.
- Behavior Driven Development (BDD) such as easyb and JBehave for Java, and NBehave and NSpec for .Net facilitate writing test specifications which customers can read easily. It also finds use to automate testing.
- API layer functional test tools such as Fit and FitNesse tools find use as API layer functional test tools. These tools promote collaboration between different team members to come up with the correct tests.

Consider Automated Test Driven Development

Agile's "Test First Development" or "Test-Driven Development" (TDD) approach makes it necessary to write a failing test before writing the code. With this approach the developer not only gets a good set of working automated tests that ensure a high level of coverage and regression protection, but it also leads to better quality code and a simpler, cleaner design. The test cases define clearly the input data, expected results, and required behavior of the system. The development feature is not completely done until all the laid-down acceptance tests have passed.

While TTD is almost synonymous with Agile, many developers fall into the trap of considering it as a replacement for thinking through an appropriate testing strategy. Testers ideally need to be technically aware of what they are testing and understand the impact on automation as well as the functional implications. TTD is only one of the several means to achieve a goal, and not necessarily the best means.

The Agile development model works as a good source of competitive advantage over other conventional methods - but only when applied the right way.

About SUYATI

Suyati is a young, upwardly mobile company focused on delivering niche IT services to support myriad Digital Engagement strategies. Our expertise includes integration and delivery of CRM, CMS, and e-commerce solutions, in addition to mobile app development for iOS, Android and Windows.

We offer several ready-to-use cloud-based solutions. Voraka is a unique cloud-based writer-management tool, making it easy for content curators to manage writers and their deliverables. SoCXO is another cloud-based solution, offering enterprises a brand advocacy platform, to help them identify brand advocates and enabling such advocates to share content and become storytellers over social media.

www.suyati.com services@suyati.com

Reference:

1. http://swreflections.blogspot.in/2013/05/7-agile-best-practices-that-you-dont.html
2. http://www.capgemini.com/resource-file-access/resource/pdf/ maximizing_the_value_of_good_testing_practice_in_an_agile_environment_final.pdf
3. http://www.globaltelecomsbusiness.com/pdf/AMDOCS%20WHITEPAPER_Agile%20Testing%20whitepaper.pdf
4. http://www.seguetech.com/blog/2013/04/12/8-benefits-of-agile-software-development