ISSN:0975-9646

Agile Testing

Sakshi Sachdeva[1], Purnima Khurana[*]

[*]*Assistant Professor ,*
[1,2]*Department of Computer Science PGDAV College(M), University of Delhi, Delhi, India*

***Abstract*— A software testing practice that follows the principles of agile software development is called Agile Testing. Agile is an iterative development methodology, where requirements evolve through collaboration between the customer and self-organizing teams and agile aligns development with customer needs. Agile testing saves time and money as it provides regular feedback from the end user. Also, daily meetings can help to determine the issues well in advance.**

***Keywords*— Agile, Testing, Stakeholders, Customer, Software**

I. Introduction

The technology & software development industries adopt the latest technology trends and development practices but their opinion often depends on how good or bad the new technology or software development practices are. Agile software development processes and methods is probably the most potent example of where the industries practitioners range from supportive and embracing of the concepts, through to those who simply see another example of an excuse for bad practice or make statements to their peers “that it will never work in our organization”.

It is an observation that much of the industry practitioners who feel agile to be a bad practice appears to be driven by a lack of understanding of what it actually means to be “agile” or more importantly, what “agile” means within the context of the individual, i.e. what role they perform, and their organizations software development context, i.e. the type of software development they perform.

This paper will discuss some of the aspects of what agile software development is or more importantly, what “agile testing” means within the context of the individual, i.e. what role agile testers perform, and their organizations software development context, i.e. the type of software development they perform.

An iterative and incremental (evolutionary) approach to software development in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams, with "just enough" ceremony that produces high-quality solutions in a cost effective and timely manner which meets the changing needs of its stakeholders. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible response to change.

Agile testing is a software testing practice that follows the principles of agile software development. Agile development recognizes that testing is not a separate phase, but an integral part of software development, along with coding and delivers the business value desired by the customer at frequent intervals, working at a sustainable pace.

Agile team members use a "whole-team" approach to "baking quality in" to the software product. Agile testers use their expertise in eliciting examples of desired behavior from customers, collaborating with the development team to convert those into executable specifications that guide coders. Testing and coding are done incrementally and iteratively, building up each feature until it reaches a milestone so as to release to production. Agile testing covers all types of testing. The Agile Testing Quadrants provide a schematic classification to help teams identify and plan the testing needed [1],[2].

II. Basic Practises In Agile Methodology

A. *Practice 1: The Daily Stand-up Meeting*

One of the most common agile practices and the easiest to implement is the daily status meetings or "stand-ups". In a brief session, team members report to each other current status, work to be completed before the next meeting and what their roadblocks are. These meetings can range from daily, weekly or monthly.

Each of the individual skills teams, i.e. the testing team, will often increase the frequency of team meetings as they approach the start of execution of the designed tests. These daily meetings are critical to understand what is happening within the testing cycle, especially with respect to any issues that are slowing or preventing test execution from progressing.

It is a collective meeting where team members offer to help others and prioritize the teams work. It is not a meeting where the Manager tells them what they must do. Also significant roadblocks are discussed at this meeting for example a particular defect is holding up a lot of test cases.

It is also a great opportunity to talk about process improvements and continuous changes. Often these suggestions result as one of the team has been trying a new procedure which when discussed is frequently used by the rest of the team.

It is important that this meeting focuses on the critical issues and not become a “complaints” meeting. The manager acts as the facilitator to ensure that the three basic questions used in an agile daily stand-up meeting are explicitly discussed.
1)"What have you completed yesterday?”
2) “What are you doing today?”
3) “What road blocks have you struck?”

The same rules apply to these meetings as they do to daily stand-up, that is:
1) The meeting is time-boxed - generally no more than 15 minutes.

2) Keep the focus of the meeting on the above mentioned questions only.
3) The manager is the facilitator to assist the team in developing a collective responses and solutions to the issues and roadblocks raised.
4) The manager, is responsible for eliminating the road blocks identified.

B. Practice 2: Face to Face Communication

All projects need effective lines of communication, both internally within the team and externally, outside of the team. The most effective team communication is via face to face discussions and meetings. Some examples are walkthroughs, defect management meeting and workshops.

No matter what development disciplines are required, each agile team contains a customer representative. This person is appointed by stakeholders to act on their behalf and makes a personal commitment to being available for developers to answer mid-iteration questions. At the end of each iteration, stakeholders and the customer representative review progress and re-evaluate priorities with a view to optimize the return on investment (ROI) and ensure alignment with customer needs and company goals.

C. Practice 3: The Business must be Involved on a Daily Basis

Subject Matter Experts (SME) and Business Analysts (BA) joining the testing team at specific periods within the project also promotes better communication. The following are examples of where the involvement of the SME or BA can be extremely beneficial:

1) Reviewing of defects with the testing team before they are formally raised
2) Working with the testing team during test case preparation to answer questions immediately and ensuring that the test cases contain the correct information through informal reviews and that the test cases provide full coverage of critical Business processes and functions. They can also help in prioritizing the test cases.
3) During test case execution to answer questions where the test case does not match the software delivered. They may also supply information on how the existing system works which may not have been detailed within the requirements were documented, but may be required at the time of testing.

Developing relationships between the user and the testing staff so that they could sit together for a couple of days to observe how users work can provide a significant improvement of the understanding of how the users actually use the system. The benefits that this contact with the users can provide is extremely beneficial early in the project as the testers understand the usage of the system. This approach if continued throughout the project can result in assistance being provided with test case preparation, execution and defect detection. It is not unusual to see some of the users join the testing team either as an SME or during test execution. They involvement of the users is a principle concept of Agile.

D. Practice 4: Deliver Working Software Frequently

Most agile development methods break tasks into small increments with minimal planning without involving long-term planning. Iterations are short time frames (time boxes) that typically last from one to four weeks.Each iteration involves a cross-functional team working in all functions: planning, requirements analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is demonstrated to stakeholders.

Project teams actually use iterative development, i.e. time boxing, within every project. The following are examples of where time-boxing and iterative processes are followed:

1) Towards the end of test execution for managing late change requests and the defect fix and test cycle.
2) Determining which defects and change requests are delivered in which build to the test environment.
3) Often project deadlines are fixed and the resources are fixed, so particularly nearing the end of execution when there is little value in introducing new members as the lead in time for familiarisation with the tests and system outweighs the execution the individual could undertake. This means that the original scope of the project may need to be cut and functionality moved to subsequent releases.

This incremental and iterative technique minimizes overall risk and accommodates changes in the project quickly and easily. Iteration might not add enough functionality to warrant a market release, but the goal is to have an available release (with minimal bugs) at the end of each iteration. Multiple iterations might be required to release a product with new or enhanced features [1],[2].

III. HOW AGILE IS DIFFERENT ?

There are some aspects of agile software development which are very different to those which are more commonly used and thus make traditional testing professionals move to agile development. Some of those aspects are as follows:

A. Greater Collaboration

Agile developers and testers work closely together, promoting face to face communication over passing documentations back and forth to each other. They get to realize that documentation is the least effective manner of communication between people.

B. Shorter Work Cycle

Due to the adoption of test-driven development (TDD) approaches, greater collaboration, and less reliance on temporary documentation, the time between specifying a requirement in detail and validating it is now reduced to minutes, not months or years.

C. Agile Developers Welcome Change

Agile developers treat requirements like a prioritized stack which is allowed to change throughout the lifecycle. A changed requirement is an advantage if you are able to implement it.

D. Greater Flexibility is Required in Testing

Agile development team does not hand off a "complete specification" to the testers which they can test again. Here the requirements evolve throughout the lifetime of the project. Hence testers need to be flexible in their testing approach.

E. Greater Discipline is Required of IT

It's very easy to say that you're going to work closely with your stakeholders, respect their decisions, produce potentially shippable software on a regular basis, and write a single test before writing enough production code to fulfill that test (and so on) but a lot harder to actually implement such things. Agile development requires far greater discipline and hard work than does traditional development.

F. Greater Accountability of Stakeholders

One of the implications of adopting the practices of active stakeholder participation, prioritized requirements, and producing working software on a regular basis is that stakeholders are now accountable for the decisions that they make.

G. Greater Range of Skills are Required

In Agile Methodology, it is not enough to be just a tester, or just a programmer, or just an analyst. Agilists are moving away from the Tayloristic approaches of traditional development and are moving towards a highly iterative and collaborative approach which requires generalizing specialists [3].

IV. WHAT'S AN AGILE TESTER ?

An agile tester is a team member who drives agile testing. He is a professional tester who embraces change, collaborates well with both technical and business people, and understands the concept of using tests to document requirements and drive development. Agile testers tend to have good technical skills, know how to collaborate with others to automate tests, and are also experienced exploratory testers. They're try to learn what customers do so that they can better understand the customers' software requirements.

A developer who is involved into testing beyond unit testing. An exploratory tester, accustomed to working in an agile manner, is attracted to the idea of an agile team. Skills are important, but attitude counts more. Testers tend to see the big picture. They look at the application more from a user or customer point of view, gather and share information, work with the product owner to help them express their requirements correctly and adequately. This helps the customer or the product owner to get a product with all the functionality or features they require.

An agile tester is not a quality police officer, whose task is to protect the customers from inadequate code. The characteristics that make someone succeed as a tester on an agile team are probably the same characteristics that make a highly valued tester on any team.

Agile testers, and maybe any tester with the right skills and mind-set, are continually looking for ways the team can do a better job of producing high-quality software. They enjoy learning new skills and taking on new challenges, and they don't limit themselves to solving only testing issues. Agile testers help the developer and customer teams address any kind of issue that might arise. This is made possible by attending local user group meetings or roundtables to find out what other teams are doing. It also means trying out new tools to help the team do a better job of specifying, executing, and automating customer requirements as tests.

Creativity, openness to ideas, willingness to take on any task or role, focus on the customer, and a constant view of the big picture are just some components of the agile testing mind-set. Good testers have an instinct and understanding for where and how software might fail, and how to help team to reduce chances of failure.

Testers might have special expertise and experience in testing, but a good agile tester isn't afraid to jump into a design discussion with suggestions that will help testability or create a more elegant solution. An agile tester has a mind-set that is results-oriented, craftsman-like, collaborative, eager to learn, and passionate about delivering business value in a timely manner [5].

V. AGILE PRINCIPLES AND VALUES

Individuals can have a big impact on a project's success. But a team is more than just its individual members. Agile values and principles emphasize to focus on the people involved in a project and how they interact and communicate with each other within a team. Agile values and principles helps enhances team morale and better velocity than a team with poor functioning of talented individuals.

The Agile list of principles define how we approach software development. Our list of agile "testing" principles is partially derived from those principles. We've also incorporated guidelines and principles that have worked in many teams. Your team's own values and principles will guide you as you choose practices and make decisions about how you want to work.

The principles we think are important for an agile tester are :

- Provide continuous feedback.
- Deliver value to the customer.
- Enable face-to-face communication.
- Have courage.
- Keep it simple.
- Practice continuous improvement.
- Respond to change.
- Self-organize.
- Focus on people.
- Enjoy [5].

VI. What Do These Principles Bring To The Team ?

Together, they bring business value. In agile development, it's the responsibility of the whole team to deliver high-quality software due to which the customers are delighted, makes the business more profitable and all this in turn, brings new advantages for the business.

Team members play many key roles, and agile development tends to avoid classifying people by specialty. Even with short iterations and frequent releases, it's easy to develop a gap between what the customer team expects and what the team delivers. Using tests to drive development helps to prevent this, but you still need the right tests.

Agile testers not only think about the system from the viewpoint of stakeholders who will live with the solution but they are also well versed with the technical constraints and implementation details that face the development team. Programmers focus on making things work. If they're coding to the correct requirements, customers will be happy. Unfortunately, customers aren't generally good at expressing their requirements correctly and completely. Driving development with the wrong tests won't deliver the desired outcome. Agile testers ask questions to both customers and developers early in the cycle and often, and help shape the answers into the right tests.

During story estimating and planning sessions, agile testers look at each feature from multiple perspectives: business, end user, production support, and programmer. They consider the problems faced by the business and how the software might address them. They raise questions that may become assumptions made by the customer and developer teams. At the start of each iteration , they help to make sure the customer provides clear requirements and examples, and they help the development team turn those requirements into tests. The tests drive development, and test results provide feedback on the team's progress. Testers help to raise issues so that no testing is overlooked; it's more than functional testing. Customers don't always know that they should mention their performance and reliability needs or security concerns, but testers remember to ask about those. Testers also keep the testing approach and tools as simple as possible. By the end of the iteration, testers verify that the minimum testing was completed.

Agile testers take a much more integrated, team-oriented approach than testers on traditional waterfall projects. They adapt their skills and experience to the team and project. A tester who sits and waits for work to come to her, or expects to spend more time planning than doing, is likely to cling to skills she learned on traditional projects and won't last long on an agile team [5].

VII. Agile Testing Quadrants

Models are useful for software development as well as for our daily lives. As new models are dreamed up, they're out in the world for other people to use, adapt, and evolve, or possibly try to destroy. In Agile Testing: A Practical Guide for Testers and Agile Teams, Janet Gregory shared the version of the Agile testing quadrants (2008) based on Brian Marick's Agile testing matrix, which has been used extensively and has been found useful too.

In recent years, many people have adapted the quadrants to fit their teams' individual needs. That's all good; techniques and models must evolve as we learn more. However, because models are inherently flawed, we often see misinterpretation of quadrants model. That is bad because it leads to incorrect conclusions and an idea that there is one "real" model that fits everyone's needs. Let's look at ways the quadrants model has been improved and adapted over the years and try to correct misunderstandings about how to use the quadrants.

The Four Quadrants

The quadrants are numbered Q1–Q4 in order to refer to them conveniently. This numbering doesn't imply that teams should do these categories of tests in that order. Here's how we specify the quadrants (see Figure 1):

Q1 -- technology-facing tests that guide development
Q2 -- business-facing tests that guide development
Q3 -- business-facing tests that evaluate the product
Q4 -- technology-facing tests that evaluate the product

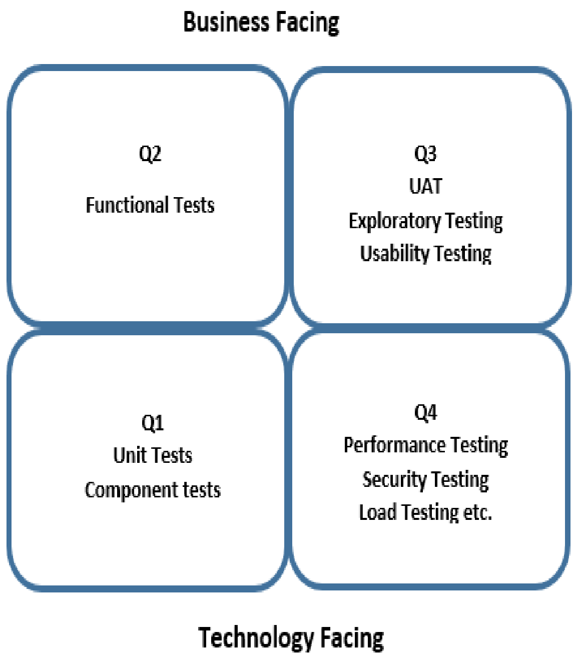

Fig. 1 Agile Testing Quadrants

The categories aren't meant to be hard and fast rules. Many types of tests cross over the quadrant boundaries. For example, if you're doing exploratory testing with paper prototypes to learn more about what the customer might want, that's a Q2 activity, whereas exploratory testing a delivered feature or part of a feature to critique is Q3.

Q1: quadrant one tests are basically technology facing as it involves unit tests and component tests. As it involves unit testing of individual components it helps isolating problems. Component testing help test whether the components work together correctly. Here the developer writes the unit testing and perform these tests too which helps find the actual problem areas and in turn improves the internal quality of each small component of the system. So, the goal of Q1 testing is quality.

Q2: Quadrant two involves functional tests. This quadrant is business facing as it basically tests how the system behaves. The testing is performed to test the functionality of the system. This type of testing helps customer gain confidence of the system. As this involves tests that take input and provide output, it helps gain knowledge about the external quality of the system.

Q3: Quadrant three includes exploratory testing, user acceptance testing, usability testing and alpha, beta testing. These are done by recreating actual user experiences. These can be performed as an informal demo with the customer. These help build confidence of the system and also provide a quick feedback. Quadrant three tests require good skills, experience, intuition, critical thinking. The customer feedback and the results of this testing is to be discussed and shared with the programmers.

Q4: Quadrant four involves performance testing, security testing, reliability testing, scalability testing, and stability, maintainability and compatibility testing. This type of testing depends on priorities [4].

VIII. The Purpose Of The Quadrants

The Agile testing quadrants aren't a recipe or prescription. There's no one "true" version of them. They simply provide a taxonomy of types of testing that might be done on any given software project.

The teams must think about testing first as they plan at the release, feature, and story levels. This must extend to the people doing the various testing activities. Talking through the quadrants provokes conversations about testing, which often extends to architecture and code design changes to make testing easier. Each team can adapt the quadrants to suit their context.

Models are useful to help us think through product features, assess risks, collaborate with customers, and verify internal (as per the delivery team) and external (as defined by the customer team) quality. They're never a perfect answer, and they aren't "one size fits all." There are many other models to help teams ensure that all necessary testing activities are planned and executed in a timely manner and in a good way [4],[5].

Conclusion

An agile testing mind-set is customer-focused, results-oriented, craftsman-like, collaborative, creative, eager to learn, and passionate about delivering business value in a timely manner. Agile testers apply agile values and principles such as feedback, communication, courage, simplicity, enjoyment, and delivering value in order to help the team identify and deliver the customer requirements for each story.

Agile testers add value to their teams and their organizations with their unique viewpoint and team-oriented approach.

References

[1] Leanne Howard, "Agile – Why the fear?" November 12, 2009. [Online]. Available: http://agiletesting.com.au/files/2009/11/agile_why_the_fear.pdf.

[2] "Agile Testing,", Dec, 2013. [Online]. Available: http://en.m.wikipedia.org/wiki/Agile_testing.

[3] Scott Ambler, "Agile Testing and Quality Strategies: Discipline Over Rhetoric" , [Online]. Available: http://www.ambysoft.com/essays/agileTesting.html

[4] Lisa Crispin, "Using (and Abusing) the Agile Testing Quadrants, Part I, November 20, 2014. [Online]. Available:http://www.cutter.com/content-and-analysis/resource-centers/agile-project-management/sample-our-research/apm141120.html.

[5] Nitin Bharti, "Ten Principles for Agile Testers", July 26, 2010. [Online]. Available:http://agile.dzone.com/articles/agile-testing-principles