IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
PP 57-62

Study and Analysis of Agile Methodology

R.Janarthanan Mca., M.Phil[1], Dr.A.Hema Mca.,M.Phil., Ph.D., Pgdbi[2].
Ph.D Scholar, Department of Computer ApplicationsKongunadu Arts and Science College (Autonomous)
Associate Professor and HOD, Department of Computer Applications, Kongunadu Arts and Science College (Autonomous)

Abstract: *Testing is very important activity in software development process. Effective testing produces high quality software. Testing is done effectively resulting in quality end product, which meets customer requirements. Agile is an iterative development methodology, where both the development and testing activities are connected. Iteration is process of delivering a small release of software. Agile Testing starts at the beginning of the project with rapid integration between development and testing. This paper deals with agile methodology, principles and agile testing methods. The objective of this paper is to analyze and understand the agile methodology that is to be used in the testing process.*

Keywords: *Software development, Software Quality, Customer.*

I. INTRODUCTION

Agile software development is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. Agile methodologies share a set of core ideas which are prescribed from method to method in subtly different ways; iterative and incremental delivery of working code, frequent collaboration with stakeholders, closely working, self-organizing teams, and the ability to embrace change late in the project.

1.1. Twelve principles of Agile Methodology:

1) Customer Satisfaction
Highest priority is given to satisfy the requirements of customers through early and continuous delivery of valuable software.

2) Welcome Change
Changes are inevitable during software development. Ever-changing requirements should be welcome, even late in the development phase. Agile processes should work to increase customers' competitive advantage

3) Deliver Working Software
Deliver a working software frequently, ranging from a few weeks to a few months, considering shorter time-scale.

4) Collaboration
Business people and developers must work together during the entire life of a project.

5) Motivation
Projects should be built around motivated individuals. Provide an environment to support individual team members and trust them so as to make them feel responsible to get the job done.

6) Face-to-face Conversation
Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.

7) Measure the Progress of the Working Software
Working software is the key and it should be the primary measure of progress.

8) Maintain Constant Pace
Agile processes aim towards sustainable development. The business, the developers, and the users should be able to maintain a constant pace with the project.

9) Monitoring
Pay regular attention to technical excellence and good design to enhance agility.

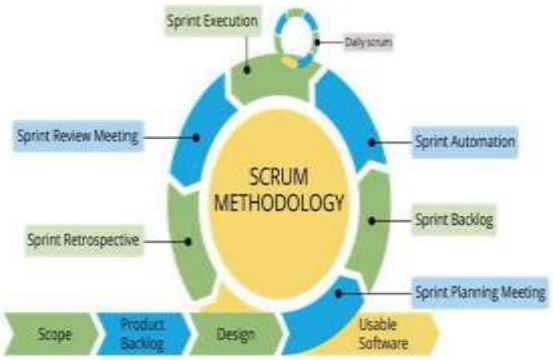
10).Simplicity
Keep things simple and use simple terms to measure the work that is not completed.

11).Self-organized Teams
An agile team should be self-organized and should not depend heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.

12).Review the Work Regularly
Review the work done at regular intervals so that the team can reflect on how to become more effective and adjust its behavior accordingly.

II. AGILE METHODOLOGY

1) Scrum



The Scrum framework should be simple. It is less a traditional methodology and more a framework for learning. The framework, as stated, is part of the Agile software development, and consists of a product owner, Scrum master and team.

Product Backlog: The product owner will make a list of work that needs to be done, and they will they place it in order according to priority. This is building your project backlog.

Sprint Planning: Using the product backlog, teams start with the highest priority items and determines how to achieve this objective.

The Sprint: A short duration to complete objectives, usually two to four weeks, but with daily Scrum meetings to make sure things are progressing as needed.

Scrum Master: The Scrum Master is just that, an expert on Scrum, who is overseeing the project throughout and offering advice and direction based on their experience and skills. The Scrum Master should only be working on one project at a time, to provide it their full attention, and focus on improving the team's effectiveness.

Completed Sprint: The sprint is complete only when the work is ready to be delivered to the customer or shown to the stakeholder.

Review: You want to look back on the sprint and see what worked and what didn't. You can then take the information and apply it to future sprints to replicate the positive and reduce the negatives.

Repeat: Once through this cycle, it starts over again by going back to the backlog and taking the next ready item at the top of the priority list. Then you just follow the above steps, improving the process through the prior experience, continuing to refine the work to make it as efficient as possible.

2) Kanban

Kanban is one of the Lean tools designed to reduce the idle time in a production **process**.

Kanban is a very simple Agile based methodology rooted in manufacturing (it was developed by Toyota to help increase productivity in factories).

Kanban is not time-based. Rather, it is based solely on priority. When a developer is ready for the next task, he/she pulls it from the to-do list. Since there are fewer planning meetings, this approach means the team needs to be extremely close.

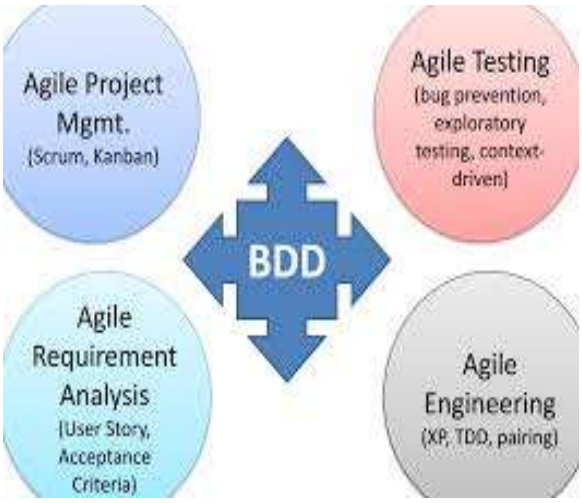
Kanban offers a simple transition for the right teams. Kanban it's important to remember that this methodology offers the quickest way to bring code to production, but the code is likely to have some technical debt. Kanban is best suited for small teams or teams that don't produce features for the public and/or promise certain dates for releases.

Key team members who should be involved in a Kanban environment include:
1). Product Owner
2).Project Manager
3).Developers
4).Automation Engineers
5).Testers

III. AGILE TESTING METHODS:

1) Behavior Driven Development (BDD)

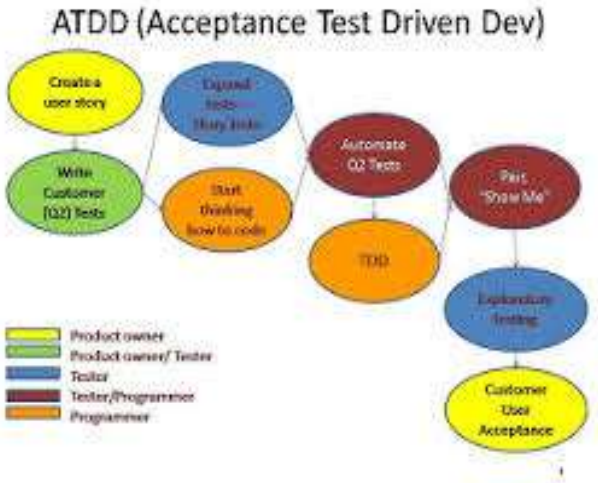
In Agile environments, BDD plays a vital role because it strongly encourages the use of Agile methodologies during the development and testing. BDD brings customers, end-users, BAs, QAs, and SEs of the software product into one table for effective sharing of knowledge on the system and its testing requirements.



Many people have heard of or used Test Driven Development (TDD). BDD starts with an initial requirement based on end user behavior and calls for tests that are "human readable" and can even replace some requirements documentation.

BDD starts with a functional specification using the Gherkin Given/When/Then syntax. This specification then guides developers, testers and product owners who move across features.
BDD requires a "smart" automation strategy that drives a high level of efficiency. This strategy sets BDD apart from other Agile methodologies.
BDD is extremely different from standard Waterfall testing because the former requires test cases to be written early against requirements and calls for those tests to be executed toward the end of the development cycle. With BDD in an Agile environment though, tests are not based on requirements and testing happens in conjunction with development of the features.
BDD approach, on the other hand, lends itself to the business owners writing the tests. This switch reduces communication (or miscommunication) between the business analysts, developers and testers.
1). Product Owner/Business Analyst
2). Project Manager
3). Developers
4). Automation Engineer/Testers

2) Acceptance Test Driven Development (ATDD)



1). ATDD closes the loop between product and dev teams. By nature, this method involves bringing in multiple teams to help make the project a success The beauty of this type of collaboration up front is that the development team can go into the project with a clear picture of the end user's needs in mind.

2).ATDD increases efficiency in the development process. Starting with a clear understanding of a specific requirement can speed up the development process significantly.

3).ATDD promotes a shared understanding of "complete." When the whole team is involved in creating the requirements for the end goal, there is a lot more clarity on what the finished project should look like. No time is wasted on miscommunication, unmet expectations, or last minutes changes due to lack of understanding about the end user.

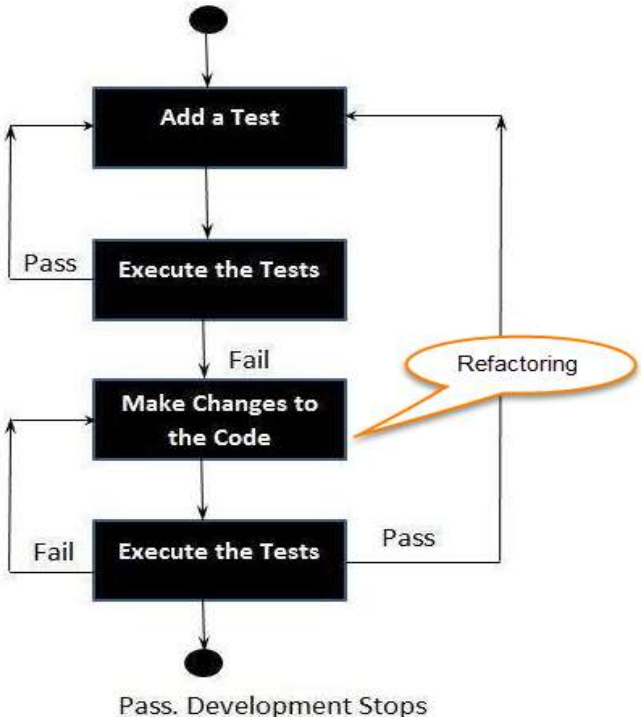
ATDD collects input from customers, uses that input to develop acceptance criteria, translates that criteria into manual or automated acceptance tests and then develops code against those tests. Like TDD and BDD, ATDD is a test-first methodology, not a requirements driven process.ATDD helps eliminate potential areas for misunderstanding by removing the need for developers to interpret how the product will be used. ATDD goes one step further than TDD and BDD though because it goes directly to the source (aka the customer) to understand how the product will be used.

3). Test Driven Development

TDD can be defined as a programming practice that instructs developers to write new code only if an automated test has failed. This avoids duplication of code. TDD means "Test Driven Development". The primary goal of TDD is to make the code clearer, simple and bug-free. Test-Driven Development starts with designing and developing tests for every small functionality of an application.

TDD test involves,

1. Add a test.
2. Run all tests and see if any new test fails.
3. Write some code.
4. Run tests and Refactor code.
5. Repeat.



IV. AGILE DELIVERY PROCESS:

Once you determine which testing methodology is right for your organization, you're not quite done yet. You still need to align testing with the delivery. To achieve this goal we recommend a three-pronged approach:

1) Get involved in the development process as early as possible

The sooner testers can get involved, the better. Ideally, testers should be present from day one. That's because giving testers a seat at the table every step of the way provides a higher level of insight into requirements and goals, encourages collaboration and helps hammer home the need to conduct frequent (if not continuous) testing.

2) Test frequently, but thoughtfully

As more and more teams adopt Agile methodologies, efficiency is everything. This need for speed has led teams to embrace DevOps and continuous integration as well in order to keep things moving, and that requires testing more frequently. But in the midst of an efficiency and frequency focused shakeup, testers need to remain thoughtful so as not to create more overhead and run unnecessary tests that actually slow down the process.

3) Hit the ground running with test creation

Keeping in mind the need for speed in today's Agile, DevOps driven world, testers need to hit the ground running when it comes to getting tests created. Specifically, the more testers can reduce the time from requirements gathering to test creation, the better. Having a seat at the table for all conversations from the very beginning should help in this regard.

V. CONCLUSION

Even though Agile has already made significant role in the the software development lifecycle, there's still a long way to go, especially among testing teams. To perform testing effectively and efficiently agile methodologies will require testers to go beyond test creation and execution and begin to focus on code delivery

and integration. At the same time, testers will need to own their testing skills, become more involved in the entire software development process and continue to develop a collaborative relationship with developers.
In the future, three key values will become particularly important for testers working in Agile environments:

1) Communication between testers is top priority.
2) Skill diversity according to the short time.
3) Delivering the product satisfying the customer expectations.

BIBLIOGRAPHY

[1]. https://agile testing.org/
[2]. https://www.guru99.com/agile-scrum-extreme-testing.html.
[3]. https://www.tutorialspoint.com/software_testing_dictionary/agile_testing.html
[4]. https://en.wikipedia.org/wiki/Agile_testing [5].https://www.agilealliance.org/
[5]. http://www.origsoft.com/whitepapers/software-testingglossary/glossary_of_terms.pdf
[6]. https://searchsoftwarequality.techtarget.com
[7]. https://www.cigniti.com