



Technical White Paper

The Agile Advantage

Khursheed Abdi

Testing Practice Manager
Sonata Software Limited

STATEMENT OF CONFIDENTIALITY

Information included in this document, in its entirety, is considered both confidential and proprietary to Sonata Software and may not be copied or disclosed to any other party without its prior written consent.

Abstract

This white paper discusses the significance of Agile Testing, principles underlying Agile Approaches and the various processes involved in Agile Testing. It also draws a comparison between the Traditional and Agile Approaches of software development and testing.

For an easier understanding of the subject, it also includes a case study on Agile Testing.

About the Author

Khursheed Abdi is a Testing Practice Manager at Sonata Software. He has been working with Sonata for more than 5 years and has more than 8 years of experience in the area of Software Testing.

Khursheed also has good exposure to implementation of Testing Process and Methodology for various projects at Sonata.

Table of Contents

- 1. Why Agile?1
- 2. Agile Vs. Traditional Approaches1
- 3. Principles underlying Agile Approaches.....2
- 4. Agile Testing.....3

1. Why Agile?

Agile: “Active,” “Alert” and “Lively”

Intense competition necessitates organizations to respond to the changing customer needs and competitive moves rapidly. In an era when “change remains the only constant,” application development teams are under constant pressure to adapt to these frequent changes in business requirements. At the same time, there is also a need to deliver projects within shrinking timelines and with reduced risk, as also to demonstrate business value (RoI) early in the cycle. The new mantra is “Time-to-Benefits” as opposed to “Time-to-Market.”

However, traditional development approaches like the Waterfall Model fall short of expectations in such a scenario. This is because in these approaches, progress is generally measured in terms of deliverable artifacts such as requirement specifications, design documents, test plans, code reviews, etc. A major drawback of the Waterfall Model is that it divides a project into distinct stages, which may render it ineffective if the requirements of the project are not well understood or tend to change during the course of the project.

Agile Processes, in contrast, assume that all requirements cannot and will not be specified at the start of the project. They, therefore, include measures to address a high rate of change in requirements as well as to respond to those changes in short iterations. The USP of Agile Approaches is that they focus on producing completely developed and tested subsets of features at regular short intervals. They also help development teams obtain the smallest workable piece of functionality to deliver business value to the customers early and continually improve it while adding more functionalities throughout the lifecycle of the project.

2. Agile Vs. Traditional Approaches

Parameter	Agile	Traditional
Customer	Part of the engineering and change agent. Determines the scope and velocity of demand fulfillment	Contract-led; brings about change through a detailed process
Developers	Agile, knowledgeable and collaborative	Plan-driven; have adequate skills and access to external knowledge
Requirements	Largely emerging and rapidly changing	Knowable early and largely stable
Architecture	Designed for current requirements	Designed for current as well as foreseeable requirements
Testing	Testing drives the development	Validation at the end of development
Refactoring	Inexpensive	Expensive

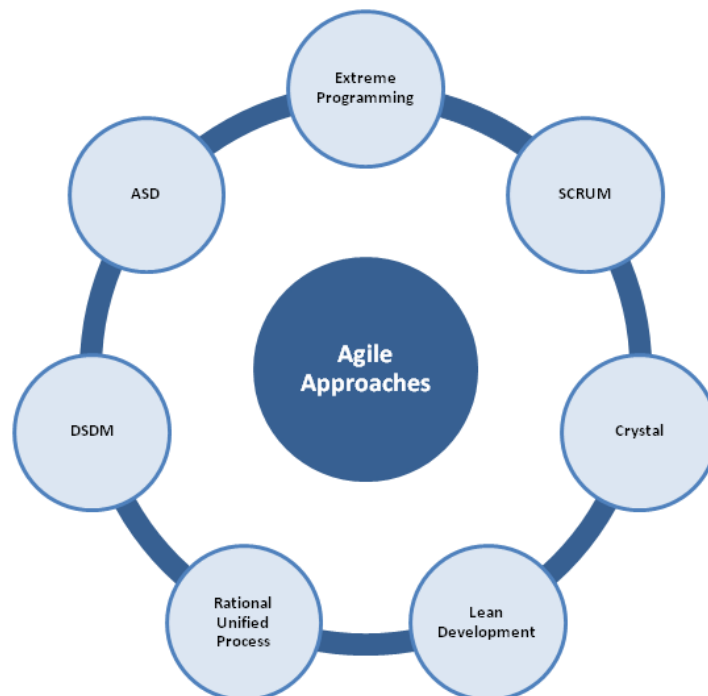
Parameter	Agile	Traditional
Documentation	Light-weight documents; software is a key project artifact	Detailed documentation and sign offs
Size	Smaller teams and projects	Large teams and projects
Primary Objective	Rapid value	High assurance

3. Principles underlying Agile Approaches

Some of the principles underlying Agile Approaches are:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Agile Approaches include various methodologies such as Extreme Programming (XP), SCRUM, Crystal, Lean Development, Rational Unified Process, DSDM, ASD, etc.



4. Agile Testing

Agile Testing is a kind of software testing process that abides by the principles of the Agile Manifesto, giving prime importance to satisfying the customer through early and continuous delivery of useful software. Agile Testing involves testing from the customer's standpoint at the earliest and as often as possible once the code achieves stability through module/unit-level testing.

Agile Testing involves the following processes:

- Test-driven development (TDD)
- Automation to the maximum extent possible
- Continuous integration
- Exploratory testing

Test-driven development

Test-driven development is a software development technique that consists of short iterations. As part of this technique, the development team writes new test cases for all new functionalities first and then implements the production code indispensable for passing the tests. The final step of test-driven development is re-factoring of the code to improve its structure without modifying its functionality.

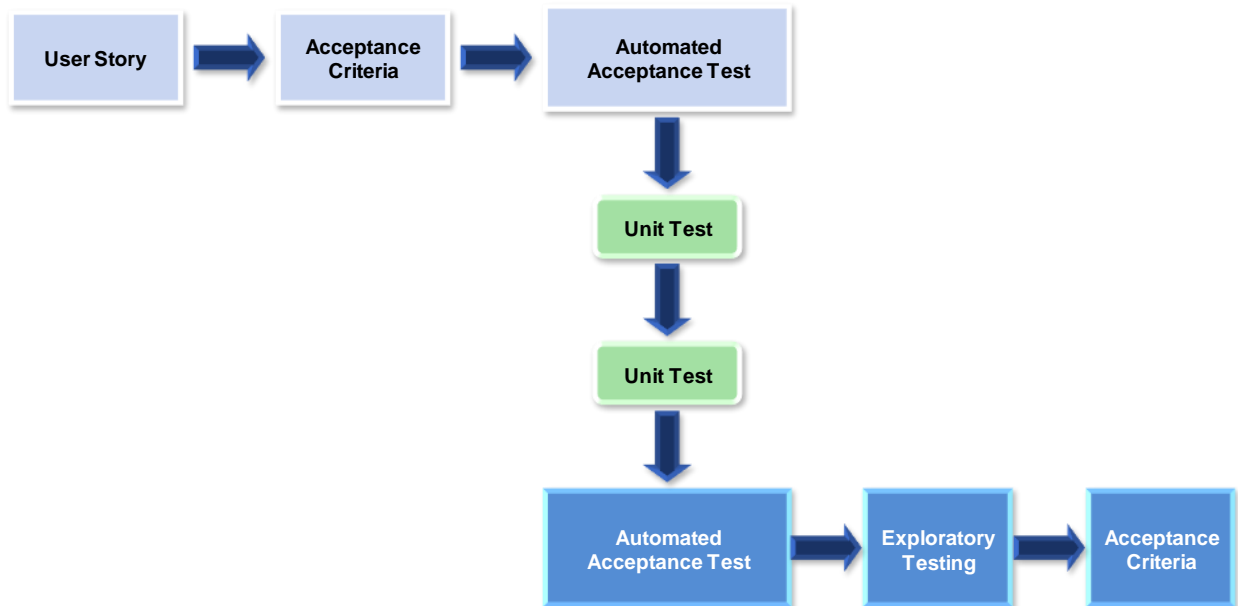
The key steps involved in test-driven development are described below:

- As soon as the functionality requirements/changes are captured and understood through use cases or story-boarding, unit test cases are written to ensure that the focus is on meeting requirements before coding begins.
- The tests are then run and a failure indicates that the test harness is functioning properly.
- Just enough code is written to pass the tests. This ensures that the developers are focused on delivering precisely the required functionality and that the code so delivered meets all user requirements. Automated tests are run to validate this aspect.
- The code is re-factored to improve its structure and make it more elegant, without altering its functionality.

Continuous Integration

Continuous integration refers to a software development process, in which members of a team integrate their work often (at least once every day), resulting in numerous integrations each day.

Whenever a change is checked in by a team member, an automated build process is initiated, including integration testing of the build. This is done to keep the risk of conflicts and failure during integration very low.



Case Study

Sonata uses Agile approach to expedite delayed rollout of Inventory Management System for one of the telecom companies in Europe

The customer -- one of the telecom companies in Europe -- had implemented a Inventory Management System using an Operational Support System (OSS) created by one of the leading telecom OSS solution providers. The Inventory Management System provided a simplified view of the customer's deployed and planned network inventory as well as an understanding of how the inventory was utilized across multiple services. It also consolidated data from multiple sources into a single database and therefore, hosted all the current inventory information. The Inventory Management System also interacted with other external components involved in the order processing cycle.

The first four releases of the Inventory Management System were executed using the Waterfall Model and all of them were delayed, leading to deferred rollout of the application. The delays and the pressure to get the application into production left little time for testing, that too very late in the project cycle.

Sonata offered a solution, based on the Agile methodology SCRUM, for the subsequent releases of the Inventory Management System. This approach was followed for Releases 5 and 6 of the application and delivered good results.