Global Journal of Computer Science and Technology: B
Cloud and Distributed
Volume 14 Issue 1 Version 1.0 Year 2014
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Agile Software Development and Testing: Approach and Challenges in Advanced Distributed Systems

By P. Rajasekhar & Dr. R. Mahammad Shafi
Jawaharlal Nehru Technological University, India

Abstract- More and more companies are adopting Agile methods as a flexible way to introduce new software products. An important part of any software project is testing. Agile testing may have similar aims as traditional software testing, but the structure of the team is different, testers need to support quality infusion through entire team. Test automation and selection of test tool can help project teams deliver more effectively, and in shorter timescales. The challenges in testing of cloud are visible also in the tools for automatic test case execution. This paper addresses some of these challenges and also highlights every aspect of software testing process in Agile development.

Keywords: *agile methods, unit testing, cloud testing.*

GJCST-B Classification: *K.6.3*

Strictly as per the compliance and regulations of:

Agile Software Development and Testing: Approach and Challenges in Advanced Distributed Systems

P. Rajasekhar [α] & Dr. R. Mahammad Shafi [σ]

Abstract- More and more companies are adopting Agile methods as a flexible way to introduce new software products. An important part of any software project is testing. Agile testing may have similar aims as traditional software testing, but the structure of the team is different, testers need to support quality infusion through entire team. Test automation and selection of test tool can help project teams deliver more effectively, and in shorter timescales. The challenges in testing of cloud are visible also in the tools for automatic test case execution. This paper addresses some of these challenges and also highlights every aspect of software testing process in Agile development.

Keywords: *agile methods, unit testing, cloud testing.*

I. Introduction

In general Software engineering involves the design, develop, maintenance and documentation of software systems. Software engineering can be viewed as an approach that combines computer science and the customer to develop a set of tools and techniques to solve problems [2]. As we notice the methodologies continually looking for ways building delivery models which offer their customers workable, innovative solutions which provide competitive advantage. During early 1990s, the methodologies war was between object-oriented (OO) design and traditional development models. Developing applications in complex environments presents challenges that never before faced by the management gurus. There are complexity results from the inherent complexity nature of applications themselves. For example internet based applications are highly distributed- may contain components written in a variety of languages deployed across machines with different architectures and operating systems. In its early days the proponents of OO argued that its widespread adoption would allow for greater flexibility in software development for distributed networked applications than earlier structured techniques. Many researchers and professionals involved in promoting the OOP techniques. However Object-Oriented Programming (OOP) technology is not a software development model; OOP does enhance the effectiveness of earlier software development models. As Software development involved more critical, dynamic and customer centric projects, new challenges are emerged which effects the growth of companies. These difficulties include more customer involvement, changing nature of customer requirements, projects with tight deadlines and over budgets. With the existence of such problems, the OOP technology cannot satisfy the objectives of software development companies. A number of IT professionals started to work individually on new approaches to develop software. Currently Agile is one of the highly practiced methodologies. Trends in distributed software systems have gained increasing importance in recent years have been for software to move to much larger venues in the web, cloud and mobile applications. One way to handle software development in an ever changing environment is by imbibing Agile methodologies as part of development process.

The Agile software development methods evolved in the mid-1990s as a reaction against heavyweight waterfall model of development. The early implementations of agile methods include Extreme programming (Beck), Scrum (Schwaber), Crystal (Alistair Cockburn), Feature Driven Development, Agile Modeling. These are now typically referred to as agile methodologies, after the Agile Manifesto Published in 2001. The formulation of Agile Manifesto occurred when a group of IT professionals with similar ideas and objectives met in Salt Lake City, Utah, USA, and issued so called the Agile Manifesto [9].

Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have came to value:

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

That is, while there is value in the items on the right, we value the items on the left more.

Author α: Master Degree in Computer Science and Engineering from Jawaharlal Nehru Technological University.
e-mail: prajasekhar456@gmail.com
Author σ: Doctoral Degree in Computer Science and Engineering from University of Allahabad.

II. End-to-End SDLC with Agile Methodology

There are a number of different Agile methodologies (XP, Scrum etc.,), each with its own combination of practices. All these methods follow the core principles of Agile manifesto with common goal and values. Agile methods emphasize working software as a primary measure of progress. In his book, Extreme Programming Explained [1], Beck proposes twelve different ideas that are key to his form of agile development. These key factors provide agility to a project aiming at speed, communication, group effort, customer feedbacks. Agile software development process is a conceptual framework for software engineering where the entire project is divided into smaller pieces or in iterations, each of which handled separately [3].Software development during one unit of time is referred to as an iteration, each defining its own set of tasks(features) that combine to make requirements.

In Agile development user stories describes the system requirements. Using these high level user stories, team will create a useful map of the full system that is valuable for telling big stories about the end to end use of the entire system .Then the system is divided into different iterations during the release planning step. Iterations are short time frames that typically last from one to four weeks, each iteration involves a cross functional team working in all areas of development: design, coding, testing. An iteration might not add enough functionality to warrant a market release, but the goal is to have an available release with minimal bugs. When the iteration is developed and tested, the system is sent to the customer for feedback. The customer provides his/her feedback in the form of stories and again the same steps are followed later. When the required levels of functionalities are delivered then customer stops writing stories and development stopped. Multiple iterations might be required to release a product or new features(Figure-1). Every release should be as small as possible, containing the most valuable business requirements (Beck, Kent 2000).

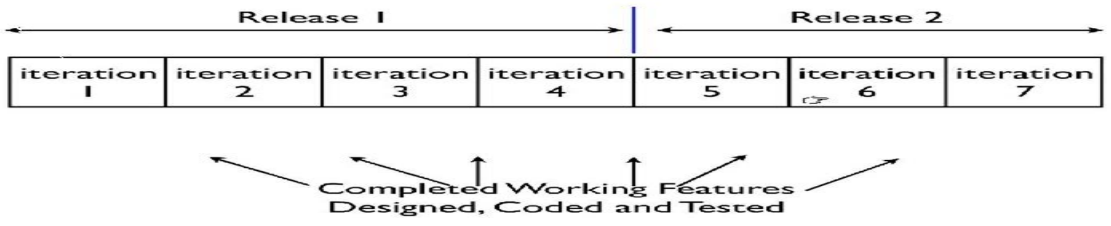

Figure 1 : Releasing Planning with Multiple Iterations

III. Software Testing and Quality Assurance in Agile Development

In Agile methods there is no specific testing phase, instead integrates testing into development process [5].Programmers do some integration testing and also unit testing while developing the iteration. The incremental feature acceptance (acceptance test) is usually done by the customer. This minimizes overall risk and allows the project to adapt to changes quickly [4]. A highly practiced QA activity in Agile method rely on customer feedback. According to Agile manifesto people are more important than process and tools; customer interaction at every level is important aspect of an Agile process. The following (Figure-2) explains the testing activities in Agile method during one iteration.

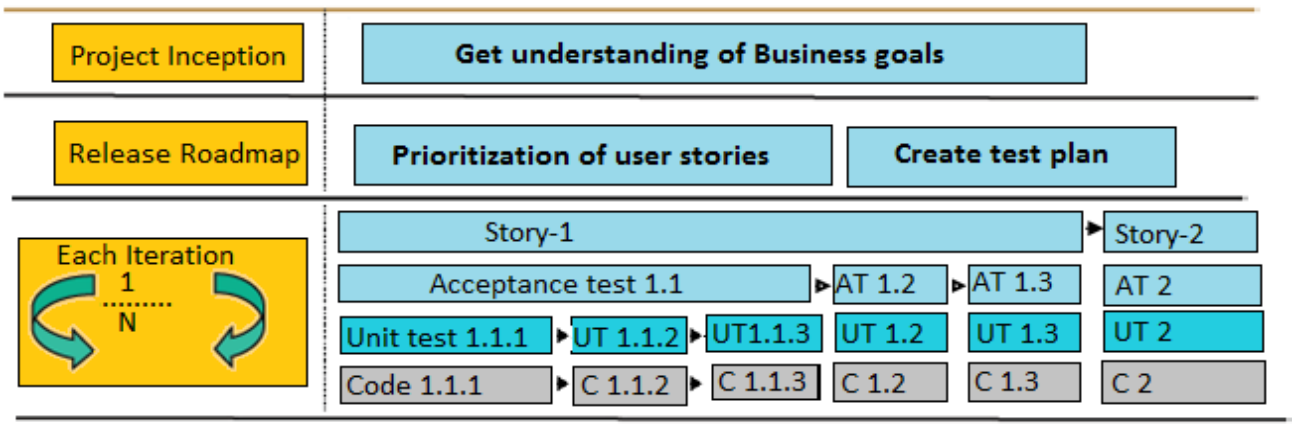

Figure 2 : Testing activities during one iteration

Software quality assurance is the systematic activities providing evidence of the fitness for use of the total software product. Different Agile methods follow different strategies in practice. These practices are broadly summarized as follows:

a) *Unit testing:* One typical way of software quality assurance is developer testing: developers test their code as they write, often in the form of unit testing. Unit testing is a good way to communicate over issues in the code under development without waiting for other units to be available, provides fault detection at a lower cost comparing to do so at a later stage [6].

b) *Automation:* Manually writing test cases can be tedious and rigorous documentation (keep track of testing results) must be maintained. The execution of the manual tests and the examination of the results can be error-prone and time consuming. When schedule pressures rise, manual testing often gets forgotten. Since delivering projects efficiently without unit testing is nearly impossible, developers could employ automated tools for unit testing. Projects based on Agile have multiple releases which each need repeated testing of units under development during integration. Apart from automating unit tests, one can introduce the automation in some or part of the acceptance testing and integration testing. So, automating tests can be very beneficial and is emphasized in agile development.

c) *Test Driven Development (TDD):* Another important quality assurance activity in some Agile methods is early testing. Test-driven development (TDD) is a type of unit test in which programmers write test cases and actual tests before they start programming (Beck 1999, Extreme Programming 2002a). It contains short iterations and test cases are developed first then code is written to make pass those test cases. TDD uses automated tests that can then be used as regression tests whenever a new build is done. One difference, however, between TDD and traditional unit testing is that the test programs are written before the application code is written.

IV. Agile Testing and Testing Challenges

In the previous section we described some of the useful testing and quality assurance activities in Agile methods. It's a common misconception that Agile projects don't need a rigorous approach to testing. But if we compare Agile with other traditional approaches, then we will come to know that from testing perspective Agile methods have lacked in different important aspects of software testing process[7]. In Agile development there is no specific role defined for tester and often testers are treated as junior developers. A good tester has many distinguishing feature that make difference with developer. In [7] the authors say that Agile development can be benefited through a team of professional testers.

Agile Testing is a software testing practice that follows the principles of Agile software development. Agile Testing involves a cross-functional Agile team actively relying on the special expertise contributed by testers. This allows the combined team to better meet the project's defined business, software usability, quality, and timeline objectives.

Unit tests are foundations of any Agile projects, effective unit test can be done with automation. The more the team can automate the testing, the faster they can move on to the next developments. But testing in an Agile way is not without its challenges, are left in utilizing the automated tools in some applications. Testing in complex distributed environment like cloud testing poses significant challenges for a tester to perform unit testing with an automated test generation tool. Moving the application to the cloud will in some cases present some differences in how to implement a certain test or test case depending on the cloud environment. Often, a test case requires the system under test to have a certain internal state as a starting point of the test case. A big challenge for the tester is to achieve the required state prior to test case execution. Another problem with Agile methodologies is that testing is not done in a fixed pattern at the end of the development phase, but instead after each package or integration of packages. Hence a project based on Agile methodology, planned tests must be creative in order to allow adaptation to the changes as well as the schedule.

V. Conclusion

Studying the role of testing and identifying a set of issues when building and testing complex distributed database systems in a cloud, the focus of our thesis, is the least researched area. Very few papers have been published that focus directly on software testing of cloud applications. The main result revealed by our primary study is that for developing and providing a system aimed for the cloud is effective with Agile development process. But it is critical that an Agile tester who is expected to test the quality and performance of cloud applications has a good understanding of what makes a Cloud Computing application and distributed architecture, as well as a good understanding of the tools available and their strengths and weakness for testing different types of cloud applications. To support our work an additional research goal was made to evaluate the applicability and support of various techniques and open source tools for advanced distributed database testing like in a cloud environment.

References Références Referencias

1. Beck, Kent. Extreme Programming Explained. Addison-Wesley, 2000.
2. Pfleeger, S.L., and Atlee, J.M., Software Engineering: Theory and Practice, Prentice Hall, 2005.
3. Conchango, "The Agile revolution: Reduce project failure rates through adoption of Agile", 2005, URL: www.conchango.com .
4. Cohn, M. and Ford, D. "Introducing an Agile Process to an Organization", Volume 36, Issue 6, Pages: 74-78, June 2003, ISSN:0018-9162.
5. Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J., "Agile Software Development Methods: Review and Analysis", VTT Publications, Pages 478, 2002.
6. Paul Ammann and Jeff Offutt. Introduction to Software Testing. Cambridge Univ Press, 2008.
7. Juha Itkonen , Kristian Rautianinen, and Casper Lassenius, "Towards Understanding Quality Assurance in Agile Software Development", International Conference on Agility Management (ICAM 2005).
8. www.agilealliance.org/guide/unittest.html.
9. www.agilemanifesto.com