Strategic Testing for Integrated Software in Agile Development

Muhammad Kashif and Prof. Naeem Janjua
SZABIST
Karachi, Pakistan.

Abstract: *Applications are being developed by using different Agile Methodologies from past few years. In Agile Software Development, there is continuous interaction with Customer and frequent change in the requirements is expected. There are iterations with prioritized User Stories and Customer requires working quality product at the end of iteration. Due to continuous code integration and frequent builds, importance of Testing has been increased in Agile. This IS focuses on different Testing Strategies and Testing types that could be conducted in Agile. It focuses on major Testing Challenges being faced in Agile and possible solution of these Challenges. Manual Regression Testing Strategy is proposed for Agile if Automated Regression Testing is not possible.*

Keywords: *Agile Methodologies, Agile Software Development, Agile Testing Strategies, Agile Testing Challenges, Regression Testing.*

INTRODUCTION

Agile Methodologies are very popular these days in software development, which includes Extreme Programming (XP), Feature Driven Development, Dynamic System Development Method, Lean Development, SCRUM etc. Each of these methodologies has its own different benefits and challenges. In Agile, there is continuous change in requirements, continuous interaction with customer, frequent builds and we have to deliver working code at the end of iteration, so we have to ensure that proper tested and bugs free software is released to customer. For this purpose, different testing strategies are used in Agile, including different testing types, to deliver quality software to customer, which increases the importance of testing activities in Agile.

Aim of this IS is to study different Testing Strategies and challenges faced by adapting these strategies in Agile. On the basis of these challenges, solution of faced challenges will be proposed to improve the quality of software. Manual Regressing Testing Strategy will be proposed for Agile if Automated Regression Testing is not able to apply for any reason i.e. lack of skilled resources to create automated regression test cases.

Chapter 2 contains Literature Review; Chapter 3 discusses Agile Methodologies, focusing XP and SCRUM. In Chapter 4, different Testing Types are discussed, conducted in Agile. Chapter 5 and 6 discuss Challenges faced during Testing in Agile and Solution for these Testing Challenges respectively. Chapter 7 discusses Regression Testing and its types and in Chapter 8, Manual Regression Testing Strategy is proposed for Agile. Chapter 9 contains Conclusion.

LITERATURE REVIEW: TESTING STRATEGIES IN AGILE SOFTWARE DEVELOPMENT

This chapter covers the Literature Review of different testing strategies followed in Agile.

Testing Strategy using SCRUM

In [1], a testing strategy is specified for projects developed by using SCRUM. They used "Selenium Core and Selenium IDE" for creation of Automated HTML generated scripts. It is a web based tool used to record user's activities in browsers, generates scripts in HTML and then executes these scripts when required. They have used the open source tool named as TestLink to plan, execute and manage testing activities.

For bug reporting, they used open source tool named as Mantis. Once all reported bugs get fixed then Automated HTML based scripts are executed in Regression Testing to validate the fixed bugs. TestLink is used to generate variety of Reports after the completion of specific sprint.

Continuous Integration Testing Strategy

In [2], agile software testing practices are specified when continuous integration of code takes places. It tells that first of all Unit Tests should be created for the newly developed code in the Sprint Then Unit Tests are executed for each provided build.

Then sufficient Acceptance Tests are created for each sprint. Customer executes these Acceptance Tests and gives feedback for completeness of the specific iteration.

Testing Strategy for User Stories and Test Design Pairing Strategy

In [3], it is specified that as all team members should participate during Iteration Planning Meeting to identify user stories. In meeting, Test Strategy is also finalized that will be used in next iteration. It is also discussed that what types of testing will be performed in next iteration. Reason of identifying Test Strategy at this point is to calculate overall cost of Testing Activities. Test Design Pairing Strategy is also specified, which tells that Testers should work with Developers & Test Developers should work with Architect of Development Team to identify requirements of User Stories and to identify scope & test coverage.

AGILE SOFTWARE DEVELOPMENT

Agile Software Development processes have become very famous nowadays. In Agile, there is continuous interaction with Customer and there are small iterations, at the end of which, a working product is ready to deliver to Customer. "Agile processes have been proposed to expedite software production by decreasing documentation overhead and focusing on customer satisfaction through a continuous delivery of the system" [4]. Agile processes are very flexible because we get Customer Feedback very early.

"The goal of Agile Methods is to allow an organization to be able to deliver quickly and change quickly" [5]. Agile processes welcome the changes in requirements at any stage of development. "Teams using agile processes tend to make decisions more quickly than plan-driven teams, relying on more frequent communication to support this pace" [6]. User Stories are finalized by Customer for iteration. Frequent builds are produced due to continuous integration of code. Daily meetings are conducted within the team to get quick feedback for the status of each assigned task. Burn down chart is used to get the quick status of each task. Numerous Agile Software Development Methodologies are available to be used. Few of these Methodologies are XP, Lean Development, SCRUM, Crystal & Dynamic Systems Development Method (DSDM). In the following, mostly used Agile Methodologies will be discussed i.e. XP and SCRUM.

Extreme Programming

XP is used for small Projects. "Extreme Programming helps developers productively deliver high quality features to their customers on an on-going basis" [7]. Interaction with Customer is very high in XP and continuous communication takes place within team members and with Customer to raise issues upfront. "In XP, all developers work closely together so they can communicate informally rather than spending time documenting designs and decisions" [8]. Designing is performed by creating CRC cards. In Pair Programming, two developers write the same code to increase the performance of code. At the end of Iteration, Customer executes Acceptance Tests to validate the completeness of user stories.

Test Driven Development (TDD) takes place in XP. "Advocates for test-driven development claim that TDD produces code that's simpler, more cohesive, and less coupled than code developed in a more traditional test-last way" [9]. First developers create a test from User Story and mark it failed. Then developers write enough code to make the test pass. Then developers refactor the code to improve its quality. "To find errors in the start, TDD is the best approach in which we create the test before we start coding" [10].

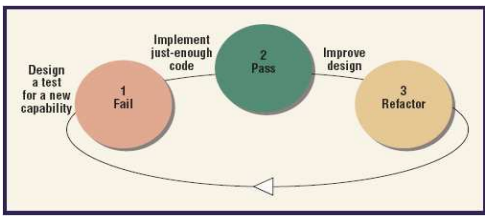

Fig. 01: Steps of TDD [11]

SCRUM

SCRUM is used for Large and distributed development. There is a Product Backlog in which User Stories are available. Selected Stories are picked up for Sprint Backlog during Sprint Planning Meeting. SCRUM Master is responsible for the completion of User Stories in Sprint Backlog. He conducts daily Standup SCRUM Meetings to get the status of tasks and to resolve the issues. Sprint Burndown chart tells how many tasks are remaining uncompleted in Sprint. At the end of Sprint, there is Sprint Retrospective Meeting to find what went wrong in last Sprint.

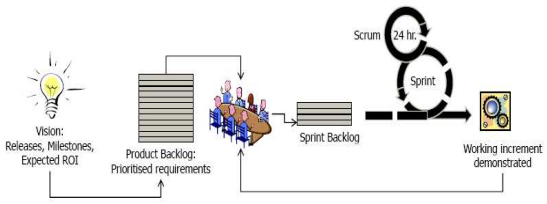

Fig. 02: SCRUM Methodology [12]

TESTING TYPES INVOLVED IN AGILE SOFTWARE DEVELOPMENT

Testing is very important part of software development. Mostly testing part is neglected in Agile because of the short time availability in the iteration. So there is a great risk to deliver buggy software to customer.

In [13], it is specified that in conventional software development, all involved people focus on quality of software. On the other hand in Agile, all people are involved in developing Tests. In Agile, there is continuous integration of code, daily builds and Customer requires a working and bug free product at the end of iteration, so importance of testing in Agile becomes increased. Following are the different Testing types that are performed in Agile:

Unit Testing

Unit testing is performed by developers by using different automated tools for unit testing. "JUnit is the most popular unit testing framework in the agile community" [14]. As much as unit testing is performed by developers on earlier stages will ultimately reduce the defects. After unit testing, refactoring of code is performed.

Regression Testing

Regression testing is performed when reported bugs in iteration get fixed and ready for verification. Along with failed test cases in build, part of already passed test cases is

also executed to check if already working code in previous builds working properly. Regression testing is performed at the end of build. If it is not possible to create automated regression tests then manual regression testing is performed. Manual Regression testing strategy is proposed at the end of this IS.

Exploratory Testing

Requirement changes frequently in Agile, due to which it is difficult to apply proper testing activities. Mostly very little time is provided to validate code in build because of unexpected delays of development. Also "some of the test cases took a lot of time to update when the application changed, resulting in the testers not using them as intended" [15]. So in these cases, exploratory testing is performed, in which tester uses his past experience to test functionality of provided build in short time by focusing more critical and risky modules to identify major issues as quickly as possible. In [16], it is specified that if testers will always follow the step by step instructions to execute the test cases then testers will lose their testing creativity.

Acceptance Testing

Acceptance Tests are created by customer from user stories. These tests are executed once all the stories in iteration get completed and full functional code is delivered to customer. Acceptance Tests could be automated if possible because customer has to execute these test cases number of times. But most of the time these test cases are manually created and executed by the client. "A user story is not considered complete until it has passed its acceptance tests" [17]. In case if bugs are required to be fixed in next iteration then another user story is created for each bug and prioritized for next iteration.

TESTING CHALLENGES IN AGILE

Importance of testing in Agile has been increased to develop a quality and Customer oriented working software. Due to continuous Testing activities in Agile, there are number of faced Testing Challenges. Few of them are as follows:

Inappropriate Testing Time Estimation

In [18], it is specified that Time estimation for different testing activities is not performed properly in Agile Software Development. Reason behind this problem is that it requires enough experience to break down big testing tasks into smaller ones first and then depending upon the total broken down sub testing tasks, testing time estimation is specified efficiently. Also duration of iterations in Agile is little i.e. few number of weeks, and most of the time, Development of the User Stories in particular Iteration does not get done on time, due to which the developed code is delivered to Testing Team very late and ultimately, already planned Testing Activities do not get completed on estimated time.

Lack of Automated Testing Activities

As Agile Software Development proceeds in small duration of iterations and there are number of builds produced within iteration, due to which, it requires a planned testing activities to check the correctness of integrated software. But most of the time it is observed that all most all the testing activities are performed manually, which requires lots of time to perform and complete these manual testing activities and ultimately, Testing timelines are not met and incomplete testing is performed for the provided builds. So confidence of testing team is not developed over the provided build.

Unavailability of Comprehensive Requirements

In [19], it is specified that Agile does not focus on producing detailed requirements, due to which most of the time Agile Testing Team does not understand requirements properly. Also Customer is not able to deliver all the requirements properly to the team. Due to these reasons, testing team suffers and they don't really know what to test and what to not test in current iteration.

Continuous Change in Requirements

In [20], it is specified that Requirements can be changed continuously at any point of time in Agile Software Development, so in this case, it is a great risk that testing activities are not performed properly. Frequent change in requirements need continuous updating the developed Test Cases to accommodate changes in the system for delivering a right product. But most of the time, this modification is not possible if changes are very frequent and if time provided for conducting testing activities is already very little due to frequent provided builds.

Unit Testing not Always Conducted

To cover as much as user stories in iteration, developers do not perform the unit testing every time for each of the developed module. This is mostly happens if inexperienced developers write the code or if review on their developed code is not performed. Due to lack of unit testing, most of the bugs are not identified initially and are identified in later stages of testing, so cost and time of the product is increased.

Regularly Delivering Running Code

Agile Software Development focuses on small iterations and requires frequent delivery of working and running code at the end of iteration, such that quality of delivered software should not be compromised. In this case it is a big challenge to deliver working and running code every time without any compromise of the quality of code when code is continuously integrating and frequent builds are provided to Testing Team.

Too Many Meetings

Agile Software Development focuses on less documentation, so variety of meetings are conducted for different purposes throughout the agile process i.e. Release Planning Meeting, Sprint Planning Meeting, Daily Standup

SCRUM Meeting, Sprint Review Meeting and Sprint Retrospective Meeting. Sometimes people don't get the actual benefit from these numbers of Meetings and more than expected time is consumed in the Meetings which decrease the usefulness of these Meetings.

Proposed Solution for Agile Testing Challenges

Due to Agile Testing significance, there are number of testing challenges which have already discussed in previous chapter. Following are the proposed solutions for each Testing Challenge to increase the quality and correctness of delivered product at the end of iteration:

Efficient Time Estimation for Testing Activities

Testing Team must know in advance that what type of Testing Activities required performing in iteration. It must also able to breakdown testing tasks into sub tasks. I.e. Regression Testing Task can be broken down into Test Environment Setup, Requirements Analysis for Test Creation, Test Cases Creation and Execution and Bug Reporting. Team must be efficient to estimate each sub task. Responsible person who is giving Testing Time Estimations should be experienced enough. Time estimation must contain time for risk factor if occurs i.e. Power Breakdown, Resource not available, Time consumed in discussions etc.

Enhance Exploratory Testing on Core Features

There is a need to enhance Exploratory Testing Activity in which Tester has some previous knowledge of application and on the basis of this knowledge; Tester knows the core features which could impact application the most. So Testers should be very clear about core features of application and impact of these core features on other modules of application to highlight major and critical bugs in short time period.

Automate Test Cases As Much As Possible

Hire skilled resources for Automated Testing. If it is not possible then Training sessions for Automated Testing should be conducted to improve the skills of already available testing resources. If it is not possible to automate the entire Testing Activities then perform partial Automated Testing Activities and perform remaining Testing Activities manually. But continuously improve the Automated Testing skills of the Testing Staff. Look for Automated Testing tools that are freely available. If a tool is expensive then purchase the license for limited users and separate the Manual and Automated Testing Teams within organization.

Updating Test Cases Whenever Requirement Changes

Continuous change in requirements in Agile can impact Agile Testing Activities in such a manner that whenever requirements change then Testing Team does not have enough time to update their Test Cases. So Testing Activities can not assure the quality and correctness of updated code. Solution for this challenge is that whenever requirement changes, update the relevant Test Cases as soon as possible before testing the updated code. While giving Time Estimation for Testing Activities, reasonable time should be allocated for updating Test Cases when Requirement changes.

Don't Forget to Perform Unit Testing

Sometimes Developers forget to perform Unit Testing for few of developed modules, due to which, bugs are identified in later stages of testing and cost of product is increased. So it is necessary to perform Unit Testing for developed logic in the module to decrease the number of identified bugs in later stages and ultimately cost of the application will be reduced because time for lots of rework will be saved in this case.

Effective Test Planning to Deliver Working Code

If Testing Team is big then separate Automated and Manual Testing Teams. If time is short for testing then focus on Exploratory testing to test core features of application, as well as verify all the Critical and Major bugs that have been fixed. Execute multiple Test Cases simultaneously and efficiently as Team already knows what Test Cases they have already written. Communicate all bugs immediately to development team when identified. Do not spend much time in reporting these bugs.

Make the Most from Effective Meetings

Make the meetings very effective by inviting only concerned people who really can get the most from these meetings. There should be a person in each meeting who must be responsible to execute the meeting properly. If no one will lead the meeting then Agenda of the meeting will never the fulfilled and meeting time will also be increased.

REGRESSION TESTING

In Regression testing, specific set of Test Cases is executed out of complete set for modified part of Project to verify that identified bugs have been fixed and no other issues have arisen in any other part of Project. "Regression testing can be performed after changes are made to the software, such as after nightly or regular builds or before a new version of the software is released" [21]. Regression Testing might be an expensive activity because if specific Test Cases are not selected efficiently then cost of Project is increased. Regression testing is also performed if new functionality is added with existing one. "For each modification, two aspects of verifications are necessary: the modified part functions correctly according to the new requirements; and the modification does not impose any adverse effect on the unmodified functions" [22].

Difference between Retesting and Regression Testing

Generally people are confused for the concepts of Retesting and Regression testing. Both are totally different.

Retesting

It is the type of testing in which same build is tested again without any change in the build.

Regression Testing

It is the type of testing in which different build is tested again with a change in the build.

Types of Regression Testing

Types of Regression testing are as follows:

Selective Regression Testing

If there is minor change in the system and its impact on overall system is not too much then Selective Regression testing is performed. Less cost is required in this type of Regression testing.

Basic Regression Testing

If there is a change comparatively greater than minor change but there is no change in the Design of system then Basic Regression Testing is performed. In this case, greater number of test cases is executed along with little more cost.

Full Regression Testing

If major design of system changes then all the test cases created for the system are executed in Regression testing to avoid any risk. This type of Regression testing is called Full Regression testing which requires more cost as compared to previous two types of regression testing.

Automated Testing better than Manual Testing?

There is misconception that automated testing is better than manual testing because manual testing is most time consuming and requires more resources and time. No doubt that automated testing requires less resources and time but actual fact is that, manual testing always has an edge over automated testing because while performing manual testing, tester keeps an eye on other scenarios as well, which cannot do an automated tool. Anyhow there are specific scenarios where automated testing might help us but apart from it, manual testing has its own worth.

PROPOSED MANUAL REGRESSION TESTING STRATEGY IN AGILE TESTING

Manual regression testing strategy is proposed for Agile by considering an example of Mobile Application which is developed for Smart devices i.e. iPhone, Blackberry, Android etc. This Mobile application is used to connect people on different Social and Location based Networks. All smart devices on which this application is running, can communicate with a server to send and receive data. In the following, it is specified that how to breakdown application into different parts and then identifying particular Test Cases from each part to execute in Regression Testing:

Moduling

First of all identify the main modules of application. As we discussed above that all devices on which this application is running, communicate with server, named as Host, to send and receive data. So we can say that there are two main modules of this application which are Host (Server) & All Devices / Platforms. Suppose that there are 300 Test cases created for this complete application and out of which, 50 Test cases are created for Host module (10 Test Cases for each device on Host side) and rest of 250 Test cases are created for All Platforms module.

Grouping

Once modules are identified then divide the application in groups. Suppose that application consists of following groups:

- Home
- Friends
- Nearby
- Instant Messaging (IM)
- Settings

Categorizing

To categorize application, identify how many types of platforms are available to create Test Cases. These platforms are as follows:

- iPhone
- Blackberry
- Android
- Windows Mobile
- J2ME

As discussed in Moduling part that out of Total 300 TCs for Application, 250 TCs are related to all Platforms. Suppose that 50 TCs each are created for all 5 platforms. In these 50 TCs for each platform, suppose that 10 TCs are created for each group i.e. Home, Friends etc.

Prioritizing

TCs for Host and all platforms have assigned a priority number ranging from 1 to 5 depending upon the importance of each TC. Suppose that these Priorities are as follows:

- 1 → Highest
- 2 → High
- 3 → Normal
- 4 → Low
- 5 → Lowest

Suppose we have 60 TCs in total for iPhone (10 for Host related to iPhone, in which 2 have Highest Priority, 1 has High and 2 have Normal Priority, and 50 TCs for iPhone platform, in which 5 have Highest, 10 have High and 5 have Normal Priority) then out of 60 TCs, select only those TCs which have Highest, High and Normal Priority. In our case, we have to select 25 out of 60 TCs.

Pre & Post Requisites of Test Cases

Create a chart diagram for all TCs in application. This chart diagram shows pre-requisite and post-requisite TCs for each TC in application due to which dependency of each TC can be identified. In our case, we had selected 25 TCs out of 60 for iPhone Platform in Prioritizing section, so along with these 25 selected Test Cases, also select Pre-requisite and Post-requisite TCs of these 25 TCs. Consider following Chart Diagram to show how to identify Pre and Post-requisite TCs:

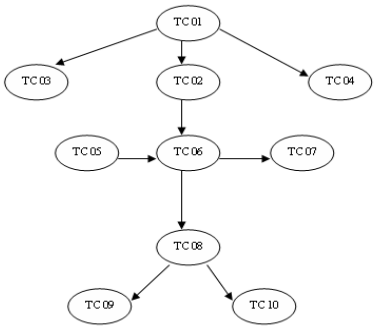

Fig. 03: Pre & Post Requisite of each TC [Self]

Suppose if TC 06 and TC 09 were failed in previous iteration and both TCs have priority i.e. Highest and Normal respectively, then after fixes, when Regression testing is performed then along with both of these failed Test Cases, Pre and Post-requisite for these two Test Cases will also be selected and executed in Regression testing. For TC 06, Pre and Post Requisite TCs are TC 02, TC 05, TC 07 and TC 08. Similarly for TC 09 its pre-requisite TC 08 is already executed, so only TC 09 will be executed. In this way, Test Cases from complete application are selected and executed in Manual Regression Testing of Agile Software Development.

CONCLUSION

In this IS, different Testing Strategies are discussed that are followed in Agile. Then XP and SCRUM are discussed as main Agile Software Development Methodologies. Different Testing Types used in Agile is also part of this IS along with Testing challenges and their solutions. Particularly Regression Testing and its types in Agile are discussed in this report. Finally a proposed manual Regression Testing Strategy is specified for if Automated Regression Testing is not possible to implement.

REFERENCES

[1] Eliane Collins et al, "Iterative Software Testing Process for Scrum and Waterfall Projects with Open Source Testing Tools Experience", Nokia Institute of Technology – INdT, Manaus – Amazonas Brazil, 2010.

[2] Sean Stolberg. "Enabling Agile Testing Through Continuous Integration". Pacific Northwest National Laboratory. Agile Conference 2009.

[3] Susan D. Shaye. "Transitioning a Team to Agile Test Methods". Agile Conference 2008.

[4] Jo˜ao W. Cangussu et al. "A Control Approach for Agile Processes", Department of Computer Science, University of Texas at Dallas. Proceedings of the 29th Annual International Computer Software and Applications Conference, 2005.

[5] Lan Cao, et al. "How Extreme does Extreme Programming Have to be? Adapting XP Practices to Large-scale Projects", Proceedings of the 37th Hawaii International Conference on System Sciences – 2004.

[6] Mike Cohn, Doris Ford. "Introducing an Agile Process to an Organization", Published by the IEEE Computer Society, 2003.

[7] Paul King, ASERT and Craig Smith, Suncorp. "Technical lessons learned turning the agile dials to eleven!", Agile Conference 2008.

[8] Frank Maurer and Sebastien Martel. "Extreme Programming Rapid Development for Web-Based Applications", University of Calgary, IEEE, 2002.

[9] David S. Janzen, et al. "Does Test-Driven Development Really Improve Software Design Quality?", California Polytechnic State University, San Luis Obispo, 2008.

[10] Taha Karamat, Atif Neuman Jamil. "Reducing Test Cost and Improving Documentation In TDD", Proceedings of the Seventh ACIS International Conference on Software Engineering, 2006.

[11] Ron Jeffries, Grigori Melnik. "TDD: The Art of Fearless Programming", Published by the IEEE Computer Society, 2007.

[12] John Fodeh, "Testing in the Agile World", Solution Architect, Global Testing Practice, Hewlett-Packard Development Company, L.P. 2008.

[13] David Talby et al. "Agile Software Testing in a Large-Scale Project". Published by IEEE Computer Society, 2006.

[14] Chih-Wei Ho et al. "On Agile Performance Requirements Specification and Testing", Department of Computer Science, North Carolina State University. Proceedings of AGILE Conference, 2006.

[15] Erik Karlsson and Fredrik Martensson."Test processes for a Scrum team", Master's Thesis at the department of Computer Science, LTH, Lunds University, 2009.

[16] Kealy Opelt and Tracy Beeson. "Agile Teams Require Agile QA: How to make it work, an experience report". Menlo Innovations LLC. Agile Conference 2008.

[17] Acceptance Tests, URL: http://www.extremeprogramming.org/rules/functionaltests.html, Last Visited on Feb, 2011.

[18] Michael Puleio. "How not to do Agile Testing". Microsoft Corporation. Proceedings of AGILE Conference 2006.

[19] Cecile Davis and Leo van der Aalst. "Testing in Agile Software Development Environments with TMap NEXT". Sogeti Nederland B.V., based in Vianen, the Netherlands, 2010.

[20] "Agile Testing Challenges". RBCS Time Tested, USA, 2008.

[21] Mary Jean Harrold. "Reduce, Reuse, Recycle, Recover: Techniques for Improved Regression Testing". College of Computing, Georgia Institute of Technology, Atlanta. Software Maintenance, 2009. ICSM 2009. IEEE International Conference from 20-26 Sept. 2009.

[22] Michael Ruth. "A Safe Regression Test Selection Technique for Web Services". Department of Computer Science, University of New Orleans, Second

International Conference on Internet and Web Applications and Services (ICIW'07), 2007.