Master Thesis
Computer Science
Thesis no: MCS-2008-25
June 2008

SOFTWARE TESTING PROCESS IN AGILE DEVELOPMENT

Ahsan Nawaz
&
Kashif Masood Malik

Department of
Computer Science
School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the Department of Interaction and System Design, School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:

Authors:
Ahsan Nawaz
ahsan174@hotmail.com

Kashif Masood Malik
kash_leo@hotmail.com

University advisor:
Guohua Bai
gba@bth.se
Department of Interaction and System Design

Department of
Interaction and System Design
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet: www.bth.se/tek
Phone: +46 457 38 50 00
Fax: + 46 457 102 45

ABSTRACT

Software testing is the most important process to verify the quality of a product. Software testing in Agile development is very complex and controversial issue in literature and industry. Different people have different views about software testing in Agile methods, because most of Agile methods do not focus much on software testing activities. Agile strongly focus on the close customer collaboration, short iterations and frequent deliveries. But when it comes to software testing, then it is challenging, as Agile do not include many destructive testing practices, which are normally required for a quality product. This thesis covers the area of software testing process in Agile development. Agile development processes could be more beneficial and refined by adding testing practices and for this purpose; we proposed a concept of an independent integrated software testing team. This research also identifies the practices of Agile development in industry and the critical issues in industry while practicing Agile development. The issues of automated and manual testing, good practices in automation, and how to manage independent testing teams in Agile development are also high lightened. This report highlights every aspect of software testing process in Agile development. This research is based on literature reviews and an industrial survey.

Keywords: Software testing, Agile development process, Quality assurance.

ACKNOWLEDGEMENT

In the name of Allah who is the most gracious and merciful. We are thankful to our creator who blessed us with abilities to complete this thesis.

We are very thankful to our supervisor Guohua Bai for his guidance and patience at every step of this thesis. Without his support and invaluable feedback, we could not be able to complete it. We acknowledge his contributions to enhance our knowledge on the subject.

We are also thankful to our friends especially Waqas Mahmaood and Tariq Asghar for sparing their time to review our thesis, and their moral support. We are also thankful to Shahid Mujtaba and Shoaib Ahmed for helping us during the conducted industry survey.

We cannot forget to thank our parents and siblings, who always pray for our success. Their love always remains the key source of our motivation. We dedicate our thesis to our respective families.

TABLE OF CONTENTS

Abstract ... **iii**
Acknowledgement ... **v**
Table of Contents ... **vi**
Table of Figures ... **1**
Introduction ... **2**
Chapter 1: Background ... **3**
1.1 Agile Software Development ... 4
1.1.1 Principles of Agile ... *5*
1.2 Agile Vs. Conventional Way of Development ... 6
Chapter 2: Problem Definition/Goals ... **10**
2.1 Software Testing Process in Agile ... 10
2.2 Challenges in Software Testing Process in Agile ... 11
2.3 A Separate Testing Team ... 12
2.4 Automated Testing ... 13
2.5 Goals ... 13
Chapter 3: Methodology ... **14**
3.1 The Literature Review ... 15
3.2 Industry Survey ... 16
Chapter 4: Agile Software Development ... **18**
4.1 Software Development Methodologies in Agile ... 19
4.1.1 Scrum ... *19*
4.1.2 eXtreme Programming (Xp) ... *21*
4.1.3 Feature Driven Development (FDD) ... *24*
4.1.4 Crystal Clear Methodology ... *26*
4.2 Agile Development Practices in Industry ... 30
4.2.1 Management of interactions with customer in industry ... *31*
4.2.2 Critical Issues in Agile Development ... *32*
4.2.3 Critical Issues in Industry, While Practicing Agile Development ... *33*
Chapter 5: Software Testing Process in Agile Development ... **35**
5.1 Quality Assurance activities of Agile in Industry ... 35
5.2 Automated and Manual Testing ... 36
5.2.1 Automated Testing ... *36*
5.2.2 Manual Testing ... *36*
5.3 When to Automate Test? ... 37
5.4 Automated testing in Agile ... 38
5.5 Manual and Automated Testing in Industry ... 39
5.6 Good practices in Automated Testing ... 40
5.7 Separate Testing Team ... 43
5.8 Why Do We Need Separate Testing Team? ... 44
5.9 Practitioners View about Agile Working ... 46
5.10 Industry Practitioners' View about Having Separate Testing Team ... 47
5.11 Roles and Responsibilities of Developers and Testers in Industry ... 48
Chapter: 6 Results ... **49**
6.1 Communication Model for Effective Working of Independent Software Testing Team in Agile Development ... 51
Chapter 7: Conclusion ... **54**
Chapter 8: Future Work ... **55**
References ... **56**
Appendix ... **60**

TABLE OF FIGURES

Figure 1: waterfall process for software development [35] 3
Figure 2: The Agile working 5
Figure 3: Waterfall vs. Agile [34] 7
Figure 4: Work Flow 15
Figure 5: The Scrum process [37] 20
Figure 6: Xp process [37] 23
Figure 7: Feature Driven Development Life Cycle [45] 25
Figure 8: The Design by Feature and Build by Feature process of FDD [13] 26
Figure 9: Dimensions of Crystal methodologies [13] 27
Figure 10: Test's lifespan [50] 38
Figure 11: A Communication Model for Effective Working of Independent Software Testing Team in Agile Development 51

INTRODUCTION

"Software Testing is the process of executing a program or system with the intent of finding errors. Or, it involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results" [1]. Practices of software testing activities give confidence to the companies that the software meets the requirements and will perform all the required tasks. Testing is an essential activity in software engineering that is used to validate whether the system behaves as intended and identifies the malfunctions. It is widely used in industry for quality assurance, as it provides a realistic feedback of the behavior of the system [3]. Nowadays the software systems are being used to perform critical tasks where the margin of error is really low. So, these systems should be error free and contain a high quality. Software testing is an important process that can help to achieve software quality assurance. Companies are spending huge amount of money on testing activities. Research shows that more than 50% of the total cost of development is devoted to software testing [2]. Currently Agile is one of the highly practiced methodologies. According to 'Agile adoption rate' survey, Feb 2008, by Scott W. Ambler; 69% organizations are using one or more Agile projects, and Agile success rates: 82% for co-located teams, 72% for non co-located, 60% for significantly distributed. Agile is an evolutionary approach to software development which is performed in a highly collaborative manner by self-organizing teams that produces high quality software in a cost effective and timely way which also meets the changing needs of its stakeholders [5]. The software is delivered to the customer very quickly; customer checks it for errors and sends some new changes and requirements to include before the last iteration. So, user is provided with a chance to test the product and provide the team with feedback about the working and the functionality of the system [10]. Agile development approach believes in the involvement and frequent communication between the developer team and stakeholders, and regular delivery of functionality. According to Agile development, people are more important than processes and tools; and the customer must be involved in the entire process [4]. Most of Agile methods do not focus much on testing.

Agile methods rely strongly on customer or user collaboration and do not include many destructive testing practices. Some of the methods, e.g. Xp, provide a very rigorous set of constructive developer practices that aim to produce good enough quality without other testing than user acceptance tests that are the responsibility of the customer. However, other Agile methods do not provide such a complete set of practices and recognize the need for specific testing practices also on the integration, system and acceptance test levels. Heartbeat QA practices could be enhanced, e.g. by introducing the role of an independent tester who tests each completed feature in collaboration with the developer. This provides instant feedback on the achieved quality, which is based on the independent destructive tests and not only the developer's own constructive practices. [17] Only some of the Agile methodologies contain some practices for software testing and most of the activities are done by the software developer. So, the question arises that are these activities are enough to get a quality product? The aim of this research is to highlight the importance of software testing in Agile development process by having an independent software testing team.

The structure of paper is in the following way: First chapter describes the background and related work. Second chapter explains the challenges and goals of this research. The third chapter describes the methodology followed to conduct this research. Fourth chapter describes the software development methodologies in Agile, Agile development practices in industry, and some critical issues in industry while practicing Agile methodologies. Fifth chapter is the main discussion section. This section describes different issues like: software testing process in Agile development, why separate testing team is important? Automated and manual testing, Industry practices of Agile development, survey results, and critical issues faced by practitioners in industry. In Sixth chapter results are given on the basis of literature review and the conducted survey. Chapter 7 concludes the paper and contains suggested future work. At the end of the report, references and appendix are given.

CHAPTER 1: BACKGROUND

Software development is really a complex task. There are so many extensive methodologies that have been developed to give a step by step guidance to the organizations, while developing a system. Waterfall approach is a conventional way of developing software. It is a traditional way of developing software. In this approach some predefined steps are being followed like; requirements, design, coding, testing, and maintenance. The requirements for the system are fully defined at the start of the process, then a design is fully created, then coding and testing of the system occurs. Each of these activities leads to the next.

This kind of heavyweight methodology contains a huge documentation for every step. The project team is expected to follow the structured plan, requirements are gathered at the start and then work separately from the customer, and finally deliver the complete piece of software that meets the initial expectations by keeping in mind the cost and the total time frame. Here lies the problem, because the Initial expectations can be change dramatically at any phase of the development. Change can occur when the business requirements or market conditions are changed may be it is because of the change in the direction of the company. As the current market is not a static one and same is the case with software projects.

So, in case of a business change, the final product contains almost 65% of delivered functionalities that are rarely or never used. Waterfall process does not allow any kind of overlapping or influencing of one phase on another. For example, if there are some issues that are arising in design process and they require changes then the development team cannot get them aboard. There is no way from where the progress of the project can be measured, the customer is provided with the final product and may be he/she will highlight some problem that requires a huge change in the overall design of the system [4].

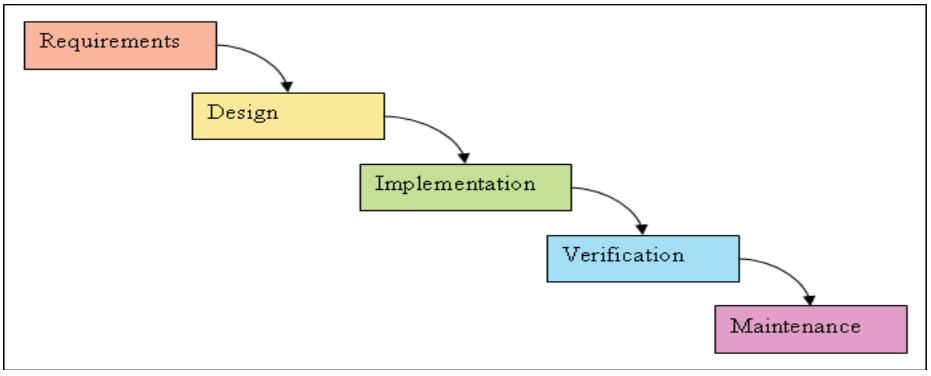

Figure 1: waterfall process for software development [35]

Figure 1 shows the waterfall process for software development. It shows the flow of different phases during development. At first the requirements for the system are gathered and they develop a requirements specification document. All of the requirements are gathered at the start of the system in this phase; on the basis of those requirements the design for the system is developed. Then the implementation of the actual system take place and the main application is developed. When the system is fully developed and the developers feel that it is ready for testing, the system is sent to the testing department to check that either the system is according to the requirements of the customer or not. The testing team tests the system and if there are some bugs or problems in the developed application, then they send it back to the development team for removing those bugs. They conduct different types of tests and when they feel that the system is according to the requirements, the system is delivered to the customer. The customer checks the system and if during running the system, the customer feels any problem it is reported to the maintenance team.

This approach is one of the well known and practiced approaches in software development industry. Most of the companies are still following this development process and some of them are following it with some desired changes. This approach for software development has shortcomings; some of them have been described at the start of this chapter. Waterfall development includes a huge documentation and most part of it is never consulted in the future. So there is no usage of those bulky records. This development process does not have any flexible process; phases cannot overlap into one another. There is no room for any future changes in the requirements, because the requirements from the customer are gathered at the start of the process in requirements phase and customer is provided with a final product. During the development process no interaction or communication with the customer is made, so sometimes when customer experiences the product it isn't according to his/her requirements. In the result of such attitude, customer sends it back to redo some of the things and which can cost extra amount of money and time. So, the projects can go out of time and money and sometimes they end in a disaster.

1.1 Agile Software Development

In response to these conventional and rigid methodologies, iterative development processes were born which are called as Agile. Agile allows change in the requirements throughout the development cycle and stresses on the close coordination between developer and customer. The central idea is the close involvement and a frequent communication between the development team and stakeholders and delivery of functionality on a regular base. Agile is a flexible approach to development. In the manifesto, Agile development gives preference to the left side over the right side elements. It values the items on the left more. Agile Manifesto is given below [14]:

Table 1. Agile Manifesto

Right Side	Left Side
Individuals and interactions	processes and tools
Working software	comprehensive documentation
Customer collaboration	contract negotiation
Responding to change	following a plan

According to [15], Agile gives preferences to the individuals and the interactions among them over processes and tools. Agile methodologies are formed on a concept that the individuals working in the organization are the most important part of the project. There should be proper communication between the team members. Because, if the communication among the team members will be regular then they will be able to overcome some of the important problems and there will be more chances for individuals to learn from the experiences of their senior members. Just because of the close coordination among them they can make more efficient systems and can share their issues with one another. They believe in a piece of working software instead of a comprehensive documentation. The working system will be more beneficial for customer as compare to that bunch of documentation in order to provide development team with feedback. They think that customer collaboration is more important than contract negotiation because close coordination of customer is also a quality assurance and defect detection activity. In Agile, customer actively participates in the whole development process and guides the team about the system's requirements. Agile methodologies prefer a quick response to change over following some predefined plan. Because today's market is dynamic, it is not a stagnant market, so processes and projects should be flexible to accommodate a change.

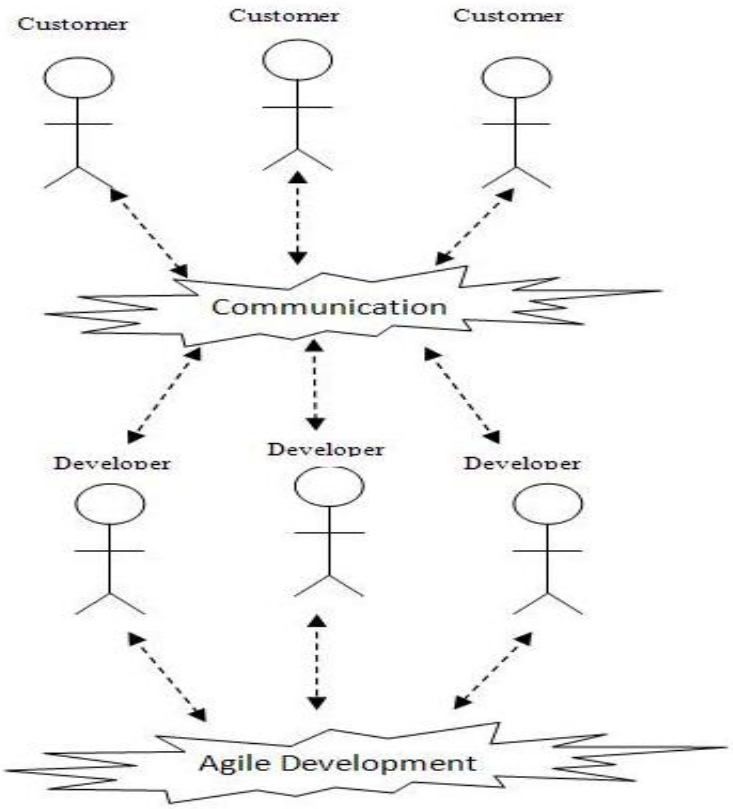

Figure 2: The Agile working

Figure 2 shows the working of Agile process. In Agile the developers work very closely to the customers and the most important part of the process is customer. Developers interact directly with customers to get feedback and to deliver a working product. Developer discusses each and everything with customer and prioritizes his/her work. The main focus of Agile is the satisfaction of the customer through a quick and continuous delivery of useful and working pieces of software". It also believes that business requirements can change at any time in the development process and people are more important part of the development than processes and tools; and the customer must be a part of the entire process. Close coordination of customer helps to cope with the changes at any stage of the process and to ensure that changes in requirements can be applied at the earliest stage. Agile does not contains huge documentation of the requirements before starting design, coding and testing. Each project is divided in smaller pieces or in iterations, each of which is handled separately [4].

1.1.1 Principles of Agile

Agile practitioners keep in mind different ways or basic set of rules during any of Agile methodology. According to [14], following are the principles of Agile development:

1. Satisfy the customer through the early and quick delivery.
2. Welcome change in requirements even in the late in the project.
3. Keep delivery cycle short (weeks).
4. Business and development people should work together.
5. Build project around some motivated people.
6. Place emphasis on face to face communication.
7. Working software is primary measure of progress.
8. Promote substantial development pace.
9. Continuous attention to the good design and technical excellence.

10. Simplicity is essential.
11. Best result come from self organizing team.
12. Teams discuss regularly that where and how to improve.

There are various Agile methods like; Scrum, Crystal Clear, Extreme Programming, Adaptive Software Development, Feature Driven Development and DSDM (Dynamic Systems Development Method). The most common of them are XP and Scrum. XP or Extreme Programming has four key values including simplicity, communication, courage and feedback. It has small releases pair programming and delivering of business values. The Dynamic Systems Development Method also calls for the involvement of the customer, scheduled delivery of project chunks and iterative development cycles.

Scrum begins with the making of a product backlog that is a list of customer requirements. Then they prioritize each element of the backlog to know that which item should be given the higher attention and work on the highest first. Here, each iteration is a *Sprint* that consists of a period of one month's duration. There is proper planning at the start of each sprint and every day starts with a 15 minutes meeting, where they discuss individuals work and to discuss their daily task. The frequent communication helps the development process easily adopt the changes in priorities and content. Team presents the functionality to the product owner for review at the end of each sprint.

[16] Says that Agile methods such as Extreme Programming, Crystal, and Scrum etc. have got a great attention recently. Agile methods focus on early and continuous delivery of software always welcome the changing requirements and give value to early feedback from customers. Agile methods cut out inefficiency, bureaucratic system and anything that is unable to add any additional value to a software product. Some key practices of agile methods are: scheduling according to prioritization, delivery of software in increments, regular feedback from expert customers, special stress laid upon face-to-face communication, pair programming, test-driven development, automated regression testing, regular integration, self organizing teams and periodic tuning of the methods. A working piece of software is the primary measure of success [16].

Due to all these traits, Agile development has become very popular. Nowadays companies are trying to adopt the Agile methodologies like; Scrum, XP, FDD etc. The early experiences for implementation of Agile show that it has been really successful and result oriented. According to [14], two known companies have experienced Agile development and their results are excellent. They have achieved better quality, reduction in lead time, cheaper systems and last but not least they achieved customer satisfaction. All of these factors are the core of any successful development process and in the current scenarios the companies are always looking for some flexible and cheaper processes to develop their applications. A lot of companies are coming towards Agile and a recent survey shows that the success rate of Agile is 71.5% [5]. In Agile methodologies teams work in a form of small groups and develop the system within those teams by dividing the whole system in iterations. Currently the companies with smaller and medium sizes have experienced with Agile development, and declared it as a great process for development. According to [13], Agile methodologies like SCRUM, XP are most suitable for companies with small or medium sizes because it may be hard for the management to handle larger teams with bigger numbers of individuals. Agile can be useful for the companies to achieve different factors like: Communication, estimation skills, iteration planning and responsibility [36].

1.2 Agile Vs. Conventional Way of Development

The conventional way of development is a heavyweight methodology and it contains a huge documentation for every step. This requires a lot of time and effort. It has its own advantages but the problem is that most of the time in an organization no one will read the whole bulk of documents. So in a way the effort is useless. The project team is expected to follow the structured plan, requirements

are gathered at the start and then work is separated from the customer. Finally, deliver the complete piece of software that meets the initial expectations by keeping in mind the cost and the total time frame. The actual problem lies here because the initial expectations can change dramatically at any phase of the development. Change can be a result of a business requirement or market conditions may be it is because of the change in the direction of the company. As the current market is not a static one and same is the case with software projects.

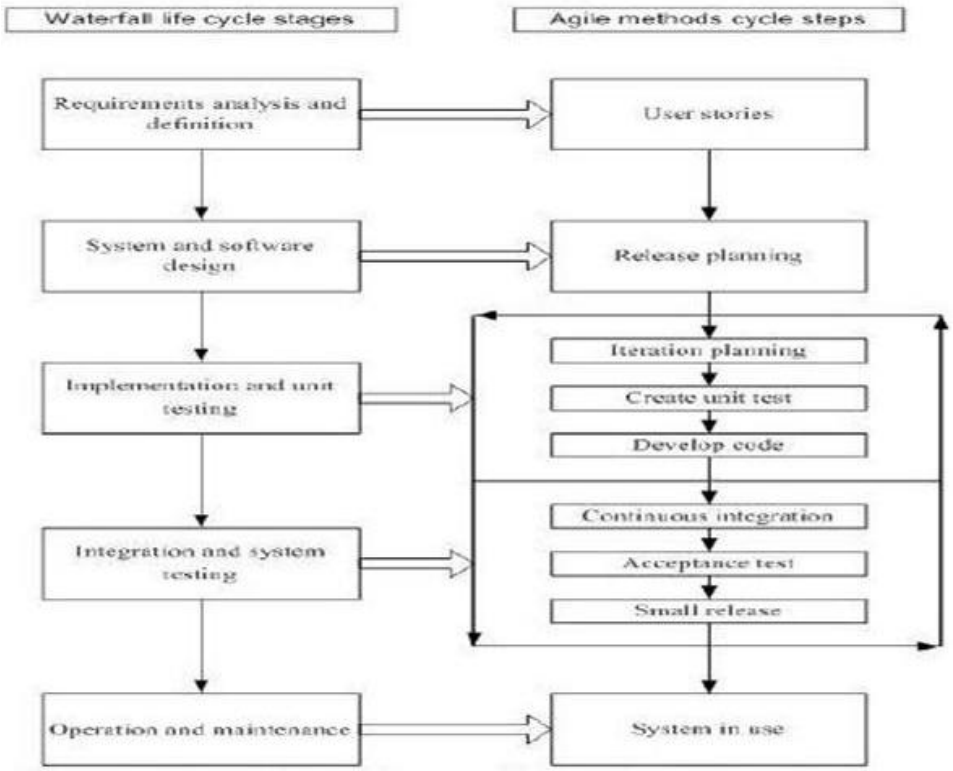

Figure 3: Waterfall vs. Agile [34]

Figure 3, shows a comparison between the different activities that are conducted in waterfall and Agile development. Even it is a well known figure that is used in different articles for comparing activities of Agile development with waterfall development process, but we feel that there is no point in which we can compare the activities in such a way. It can give a confusing impression to a new reader. Both of these processes have different nature and type. We took it in another way to show our reader that how different activities are actually done at different levels in both processes. It shows that in conventional way of development some predefined steps are being followed like: requirements, design, coding, testing, and maintenance. The system is developed as a whole in waterfall development and delivered to the customer. Where as in Agile development, customer writes user stories about the requirements of the system. Then the system is divided into different iterations during the release planning step. Then the iteration is developed and tested. After that 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.

In case of a business change, the final product contains almost 65 percent of delivered functionalities that are rarely or never used [46]. A survey by [14] shows that 60% of the delivered functionalities are never used or very rarely used. The reason behind this fact is that, all of the requirements are gathered at the start of the project during the requirements and after the requirements phase the process for changing requirements is sealed. Customer is provided with a final product and sometimes what team is providing him/her is not what customer actually wanted. Waterfall process does not allow any kind

of overlapping or influencing of one phase on other. For example, if there are some issues that are arising in design process and they require changes, then the development team cannot get them aboard. There is no way from where the progress of the project can be measured, the customer is provided with the final product and may be he/she will highlight some problem that requires a huge change in the overall design of the system [4]. Usually In conventional way of development, there is no proper or regular communication with customer. Sometimes there is a need to change in the last phases of the process and if they try to implement it, the whole system ends in a disaster. Typically these systems go out of time and money.

In the conventional way, the project manager makes all the decisions regarding the software build, brand issues, change management business processes and testing. But how one person can be so sure about everything, and individuals should be given a chance to share their innovations. In Agile, projects still face failures but mostly errors and problems are caught at the early stages of the process and the continuous iteration systems helps to achieve the task. Daily meeting help to highlight the problems and the flaws are much more likely to be spotted, as each iteration would result in a working application for the customer to test, means another set of eyes to spot problems. So the next cycle could refocus its development activity based on the learning from previous problems and also making the required changes that are occurred in business requirements. Due to a growing complexity of software projects and the rigidity of the waterfall framework, more and more companies are adopting to agile methods.

The fact is that, nowadays the most important in software development issues are; time-to-market and the flexibility. Conventional way focuses on detecting errors early, because they assume that making changing in existing software is bit expensive. In Agile software development the strategy is: organization of a team in a way to facilitate the design of changeable software. Close and continuous relations to the stakeholders and short feedback iterations enable the team to develop software faster and to react on a change in requirement more flexibly [11].

Currently Agile is one of the highly practiced methodologies. According to survey by Scott W. Ambler; the success rate of agile software development is 71.5%. Agile is an evolutionary approach to software development which is performed in a highly collaborative manner by self-organizing teams that produces high quality software in a cost effective and timely way which also meets the changing needs of its stakeholders [5]. Agile projects deliver an executable product as early as possible. In Agile development companies deliver quick releases, and make them available for the user to try. The software is delivered to the customer very quickly and the customer checks it for errors, and sends some new changes and requirement to include before the last iteration. So, user is provided with a chance to test the product and provide the team with feedback, about the working and the functionality of the system [10].

The processes in Agile do not have separate coding and testing phases just like in conventional way of development. In Agile the code is written during iteration and is tested also at the end of the same iteration. Programmers do some integration testing and also unit testing while developing the iteration. The testers and programmers work in a close coordination to reduce the chances of error occurrence [12]. In Agile when software is developed, in the mean time they check it for errors. It is the responsibility of the developers to conduct the unit testing and the rest of the testing of the system is done by the customer (i.e. Acceptance testing[1]). When the user examines the system and does the acceptance testing, he/she provides the team with feedback. So, on customer's feedback changes are made in the system. For example, in XP or SCRUM methodologies, testing is an integral part of the system development. An entire system is divided into some iteration, and testing of the system is done during each iteration. The customer remains on site all the time during the development of the system, and continuously checks the developers and provides them feedback about the system development.

[1] Developers write the unit tests to check if the code is doing the right things. Acceptance tests are written by customer to check if the system is according to their requirements.

So it increases communication and feedback. This is also a kind of quality assurance activity in Agile. Testing is an integrated part of the system development in Agile [13].

We want to look into the strategy for having a separate testing team. Because we feel that this process can be made more effective by accompanying a separate testing team. We feel that a tester is always worthwhile for any development process, because an independent tester will test the system with intent of finding errors. Developers can use automated software testing to automate some activities in software testing to reduce the human effort in software testing. The main activities of software testing are generating test inputs, running test inputs, and verifying test executions. The goal of automated test execution is to minimize the amount of manual work involved in test execution and gain higher coverage with more test cases. Automated testing will generally avoid the errors that humans make when they get tired after multiple repetitions. The test tool will not skip any tests by mistake. It can also record the results of the test accurately. The results can be automatically fed into a database that may provide useful statistics on how well the software development process is going on [8] [9]. Nowadays companies are using automated tools to reduce the burden of the work and to assure reusability. We will highlight the current testing practices in agile development, and will discuss different dimensions of using Automated and manual software testing in agile development. One important thing to identify is, the responsibilities of testers and developers regarding testing.

CHAPTER 2: PROBLEM DEFINITION/GOALS

Nowadays the software systems are being used to perform critical tasks and margins of errors are really low. So, these systems should be error free and contain a high quality. Software testing is an important process that can help to achieve software quality assurance. "Software Testing is the process of executing a program or system with the intent of finding errors. Or, it involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results" [1]. The main advantage of software testing is that the software which is being developed can be executed in its real environment. The practicing of software testing activities give confidence to the companies that the software meets the requirements and will perform all the required tasks. Testing is an essential activity in software engineering that is used to validate whether the system behaves as intended and identify the malfunctions. It is widely used in industry for quality assurance, as it provides a realistic feedback of the behavior of the system [3]. Companies are spending huge amount of money on testing activities. Research shows that more than 50% of the total cost of development is devoted to software testing [2].

Agile development believes on the involvement and frequent communication between the developer and stakeholder, and regular delivery of functionality. Agile alliance says, "to satisfy the customer through early and continuous delivery of valuable software". According to Agile development, people are a more important than processes and tools; and the customer must be involved in the entire process [4]. So, Agile is people centric rather then process centric. As Agile development believes that if the software developers are working closely with stakeholder then there is no need of separate testing team. Agile methods alter the landscape for quality assurance by assigning responsibility for quality to the developers and customers, also assigning new roles to the QA professional. There is a lot more work to be done to figure out that how this new landscape will work? How it will work in our organization, and to determine the appropriate role for the testers in your organization? [7]. we feel that there is a need of a separate testing team in Agile development. We think that a professional tester is always worthwhile for the process. We have focused on the following research questions in our study.

- What are the current software testing practices of Agile development in industry?
- What can be the affects of separate software testing team in Agile development?
- How and when the manual and automated software testing can be used to attain better results in Agile development?

2.1 Software Testing Process in Agile

With the increase in the criticality of the software systems, the need for quality products has increased. Customers are always looking for quality products. Companies are investing a lot of money to achieve quality in software products. Software testing is a quality assurance activity. It is an important part of any project. Agile software development focuses on individuals and interaction, strong collaboration with customers, and finally with short and frequent deliveries of valuable working software. If we look at these activities, they are useful from testing and quality assurance point of view, but if we compare Agile with other conventional methods, then we will come to know that from testing perspective, Agile methods have lacked in different important aspects of software testing process [17].

If we look into the quality assurance practices which are being followed in four Agile methods, then we would come to know that these methods have greater emphasis on building constructive quality practices. The software testing process has a destructive attitude whereas fewer Agile method practices attribute this behavior. These methods are based on iterative and incremental development (IID) that

uses short, often time-boxed development cycles. Customer satisfaction is the main task of Agile methods. There is less emphasis on tools, processes, documentation, and following a specific plan, which are traditionally most important in achieving quality assurance and testing practices. [18]

Quality assurance is one of the most important and crucial aspect in today's software system, due to the criticality of the systems. Quality assurance consists of all activities and practices which are used to ensure software product quality as a part of the development process. Software Testing is a quality assurance activity, it is a process to detect the differences between developed product and required conditions, and to evaluate that either the application has all of the desired features or not [17].

Software testing is a process which is used to check whether the developed product is according to the customer's requirements or not, and whether all of the features in the application are working properly or not. Testing approaches traditionally include a number of activities that are conducted at the end of the development project [19, 20]. Sometimes in conventional way of development, testing phase has an unpredictable length and effort requirements. Because, it can be a recursive process of check and verification, where the developer sends the application to the tester and he tests it for errors and sends it back and so on. Clearly, time-boxed Agile projects require a different approach to quality assurance and testing. However, most Agile methods do not say a lot about testing a system. Several of its methods include a number of good practices for developers during the iterations, including automated and unit testing. Agile is such a popular development method nowadays but still only few methods give any guidance for conducting higher levels of testing than unit and integration testing [21].

[17] conducted a survey and, they came up with their results after working closely with 11 small companies for 4 years. They have observed that currently quality assurance is rising as a critical issue that is poorly understood in the companies. All of the sample companies have been working and practicing with Agile methods in their software development processes. These practices provided them with solid low-level developer practices, but the testing of the software products remained challenging. The testing activities during the iterative development are much more then the activities at the end of the iteration. Besides its proper working there can be some other functionality that must be checked at the end of the iteration or release.

There are only one or two practices which are defined for ensuring software quality of the developed software increment on the iteration completion time. And the practices on the iteration time horizon are rather defined insignificantly as compared to the heartbeat practices. Methods, e.g. Xp, purely rely on strong heartbeat practices and leaves only progress tracking using a burn down chart at the end of the iteration [17, 27]. The other methods that have less effective heartbeat practices feel to evaluate the system and the achieved quality at the end of the iteration by system testing, but they do not concentrate on providing concrete guidance on how to perform it. For example, in FDD the only advice given to accomplish it is to decide which builds and how often the system is handed over to customer [26].

DSDM method has a stronger approach for quality assurance at the iteration time but do not have enough guidelines for the activities at heartbeat time boundaries. This approach is almost like a small waterfall process inside each time box [17]. In SCRUM the process has a great enforcement on quality assurance activities, including following some good design, coding standard and also testing activities. Only unit testing and integration testing is done by the developers and at the end of the iteration, acceptance testing is done by customer. So SCRUM also lacks in testing specific testing activities.

2.2 Challenges in Software Testing Process in Agile

Agile manifesto is the set of rules or principles for Agile software development. These principles consist of the ideas that are basic guidelines and are common for all Agile development methods. If we take a look at the software testing process in Agile methods, we see that it's really different from the

testing process in the conventional way of development; and from the traditional point of view the basic rules in Agile manifesto has some bigger challenges. First of all, the ultimate priority of Agile development is to deliver a working piece of software to customers early and continuously with a rapid releasing cycle. For testing process it is a challenge because, if the release cycles are rapid then it will put fixed deadlines for testing activities and that does not allow maximizing the testing time if more defects are found than estimated [17]. They cannot send them back to fix, because the testing time is pre planed and they need to deliver a product on a set date.

Secondly, Agile demands that changing requirements should be welcomed even in later stages of the development. Testing activities are traditionally been based on specifications that are completed in a phase of development and then they can be used as a basis for test design and other quality assurance activities. And if they will change the requirements and which will eventually change these documents then it will challenge the traditional way of doing testing. The face to face conversation of developer and the customer also creates some challenges for testing activities [17]. Because these activities relies on the documentation, and this documentation also have some information about the test results. If they have a close collaboration then that information will be in the minds of the customers and the developers. So, it can create some challenges for testing in Agile development. They say that, the only measure of progress is the working software. So, it means that testing cannot be a whole independent phase at the end of the iteration because it is supposed to provide results as soon as possible, that either the developed product is a working software or not. Another challenge for testing is the rule of simplicity. Simplicity means, the amount of work which is not done should be increased in the process. So, this rule clearly can eliminate the testing activities from the process, because the testing activities do not directly add any value to the code or any features. Apparently they look useless [17].

2.3 A Separate Testing Team

Traditional quality assurance in software development has not been followed in Agile software development. The traditional way of testing consists of some basic accepted rules, and these rules have not been focused a lot in Agile development. [17] Has stated some of those rules of traditional software testing and their negation in Agile development. As Agile development is a people centric process, not a process centric. They say that if the developer and stakeholder work closely and have a proper communication then there is no need for a separate tester. We will look into the strategy for having a separate testing team for Agile development. Because, research shows that if a developer tests his own program then he will left bugs in the software, but a tester will always test the system with an intention of finding more and more errors; so, he will test the software more effectively. We think that more complex tests should be done by independent testers. We think that a separate testing team can result in more refined Agile products. As Independency is one of the fundamental principles of testing. The testing process should be independent. The programmers should avoid from testing their own programs and that a programming organization should not test its own programs [22]. In Agile methods, the emphasis is given to the developer level activities, including unit and integration level testing by automated tests written by the developer themselves. This can create some problems, because it is really hard for someone to see problems in his own code, and the most important thing is that the testing by developers does not reveal any misunderstanding on the specification or requirements [17]. In extreme programming (Xp), there is a role for a separate tester [28, 17], but the tester is still a part of the development team. In the Crystal Clear the tester keeps on rotating among developers and the main responsibility of tester is reporting bugs [17].

In Agile development, testing is done by developers and customers, developers conduct unit and integration testing while customers do the acceptance testing. But a good software testing requires professional skills, because it is a creative and intellectually challenging task. A professional tester is required to do this professional task, so that the task can be performed effectively and efficiently [19, 23].

In Agile methods, testing is usually done as a task that developers do as a part of the development task or as a customer's task. Customer is very closely involved in everyday development in some of the Agile methods and takes the responsibility for acceptance testing. But if a customer has expertise and skills which are necessary for a tester, and has the capabilities to act as a tester, only then you can think of assigning him the task to test the system. It means that, how someone can test an application when he does not have any skills and knowledge about testing software? Dynamic Systems Development Method (DSDM) has recognized the need for basic skills for a tester and [30] recommends that at least there should be one developer or tester in each team who has received training in testing [30]. Section 5.2 has discussed the issue of separate testing team in detail.

2.4 Automated Testing

Now a day most people prefer to have automated testing but still there is need of manual testing to get rid of more bugs. The main thing about selection between manual and automated is that when to automate test? According to [31], the main rule of thumb is always 'use common sense'. If you have some test which has repetition then automation is best to run these tests. It gives the ability to main stream scenario and run automation against code that frequently changes to catch regressions in a timely manner. The cost of automation is more especially when writing the tests or configuring the automate framework. The visual reference cannot be automated e.g. if the font color or size can't be defined via code then it is manual test. If a test case runs only twice then it should be manual to save the cost. Through it, the ad-hoc or random testing is possible for tester. More bugs can be found via ad-hoc technique as compare to automation. Manual testing is more time consuming, but at the same time it is more helpful too. After every new build code tester must rerun all required tests again, this will be a huge at the end [31].

2.5 Goals

Our goals are to identify the critical issues of software testing, that the companies are facing in Agile development. We looked into their working and saw that how they organize their testing processes and will highlight the current tools and practices of software testing in Agile development. We discussed the role and importance of a separate testing team in Agile development. If we manage to discover a role for a separate testing team then next question will be that who will do what? So, we also highlighted this issue and tried to identify the roles of software tester and developer in Agile development. As we discussed earlier, automated testing is becoming more popular and companies are adopting this convenient way of testing software. The thing is that they cannot say software is bug free by just doing automated testing, because manual testing has also its importance. The question is that how it can help for a better and fast testing process? So, we also identified that when to automate test in Agile development.

CHAPTER 3: METHODOLOGY

A research methodology defines what the activity of research is, how to proceed, how to measure progress, and what constitutes success. There are many approaches to conduct a research. In which Qualitative research, Quantitative research, and Mixed research methods approaches are more popular.

Qualitative research is often contrasted with quantitative research. Qualitative studies are tools used in understanding and describing the world of human experience. The qualitative paradigm aims to understand the social world from the viewpoint of respondents, through detailed descriptions of their cognitive and symbolic actions, and through the richness of meaning associated with observable behavior. [33]

Quantitative research is often contrasted with qualitative research. A quantitative methodology involves mathematical modeling, empirical/statistical survey and experiment. Experiments can be on simulation or system, and there are two types of system experiment i.e. proof of concept and comparison. In the collection and analysis of data, quantitative research is concerned with numbers and measurement, rather than words.

"Mixed method research in which the researcher uses qualitative research paradigm for one phase and the quantitative research paradigm for a different phase of the study". [32] Mixed method is combination of quantitative and qualitative methods. It tries to mix the best of qualitative and quantitative research into research design. Mixed research takes an eclectic, pragmatic, and commonsense approach, suggesting that the researcher mix quantitative and qualitative in a way that works best for the given research question. Mixed research uses both deductive and inductive methods, obtains both quantitative and qualitative data, attempts to corroborate and complement findings, and takes a balanced approach to research.

A mixed research methodology was used to achieve our goals. We have conducted both qualitative and quantitative studies. We divided our study into two parts; the first one is the literature review, and in this part we have studied the literature about our research questions. The second part is an industry survey, in this part we have conducted a survey in which we visited different companies and interviewed some open ended question about their practices of software testing. We conducted the survey to investigate the software testing in Agile development, in terms of both management and technical perspective and see that how companies are actually practicing testing process in Agile development. We consulted the literature to highlight the research work by the experts, to extract and summarize the latest research on Agile development, and to support our findings. The following Figure shows the flow of our work, and how we managed our methodology.

Figure 4 shows the process of achieving the results. First we have identified the problem, and done some study to understand the domain knowledge and also set our scope. Then we planned our whole work along with the time. The next step was to select a methodology to conduct the research. We have selected a mixed methodology for achievement of our goals. First part was literature review, in which we read that what literature says about our area of research. The other part is an industry survey, which contains some interviews from the individuals to know about the actual practices in the software market, and to use the results in support of our idea. In chapter 4: Software Testing in Agile Development, we have combined the literature review and the industry literature to answer the research questions. We have briefly discussed our findings from industry survey and from the literature.

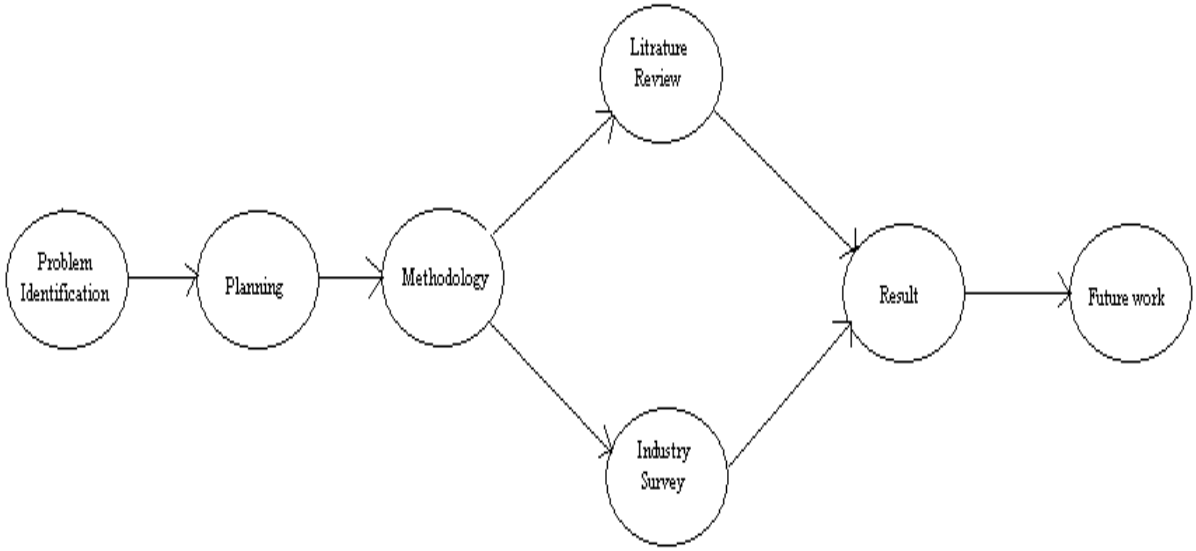

Figure 4: Work Flow

3.1 The Literature Review

Literature always plays an important role to know the state of art or to know that other researchers say about any specific topic. To know the existing material and to know the answers of some of our questions a deep literature study was done. We looked on different search engines and resources for any written materials regarding our research area. To find different articles and books we used, IEEE Explorer and ACM Digital Library, Google search engine, Engineering village, ISI web of knowledge, InterScience, Science Direct, ebrary. These are well reputed search engines and they are known as richest databases as for as the scientific articles or books are concerned. Our main focus was to find out researchers and practitioners work on Agile development, Quality assurance practices in Agile methodologies, Software testing process and its practices. We also consulted some of the web sites, depending on our need and also the author. Our focus was to find the latest articles, books, conferences, web sites and journals related to our area. To get more practical knowledge, we also looked for some articles and white papers by companies, about their experiences with Agile development. We tried to find all resources that can support our idea. We have looked into the current practices and issues during software testing process in Agile development. We tried to look for the references which were published after 1990 to date. During our search we used different keywords to find the best suitable resource for information.

We collected material from literature on "software testing and Agile development" considering the following factors:

- The quality of articles
- Information resources
- Careful extraction of data for refining information gathered
- Careful Analysis and synthesis of data
- Carefully suggesting recommendations for future work

3.2 Industry Survey

To initiate new research or to present a new idea, it is very important to know that, what the current practices in the industry are. We surveyed different organizations to know their current software testing processes in Agile development. We have contacted a number of companies with different sizes that are following Agile methodologies. In the survey we tried to highlight some critical issues, practices and tools (automated or manual) in the market during software testing in Agile development. We tried to approach a management person and preferably a test manager to provide us response against our questionnaire. We interviewed them with open ended questions. We contacted them face to face, but whenever it was not feasible for us to travel then we use email and telephone call to communicate to get their views. The focus of our questions was to know the Agile working in the organization. Some of our main questions were:

- How do they manage their flow in Agile development?
- How are they practicing quality assurance activities with respect to Agile development?
- How are they conducting manual testing and automated testing?
- How are they practicing software testing activities in their development process with respect to Agile development?
- What are the critical issues they are facing while practicing Agile development?

During discussion we also discussed our idea with them and to get some response. Most of the time during interviews, our intentions and questions were focused on the idea of independent testing team. We also tried to know that what the management says about the use of automated and manual testing, and how they assign the role and responsibilities of automated and manual testing to individuals. We gather some feedback from them, and discussed their practical experiences in chapter 4 and 5. At the end we showed results based on our theoretical work (literature review) and empirical study (survey).

During industry survey we choose twelve companies with different sizes and working in Agile development. These companies are working in Sweden, Pakistan, and United Kingdom. We first contacted them through email to get some appointment for interview. The final five replied us in response. We have mentioned these companies as company A, B, C, D, and E. Company A is a Swedish software technology and design company offering products and services that differentiate and enhance the user experience of portable devices. It's headquartered in Malmo, Sweden, and has local offices in Korea and USA. Their business focus is on graphical user interfaces (GUI), enabling clients to push the visual limits in both mass-market and feature devices. They work with four of the five leading companies in mobile device space today. Publicly announced clients include SonyEricsson, Samsung, Telia Sonera and Orange. Two of their representatives provide us answers and views against our questions. One of them was working in research and development department and other was a senior developer. We have contacted them through phone and email. Company B is a Swedish consulting company and software house focusing on the areas of Mobile Multimedia and Software Business Management. It has its branches in bigger cities of Sweden: like Malmo, Karlskrona and Stockholm. Their Malmo office is responsible for application development. It is a medium size company. They are working in Agile development and help their customers to adopt Agile. We have contacted their information help desk and got feedback in response. It was through email. Company C is a big size software house and consultant in Pakistan. They have their branches in UK and USA. They are working in Agile development and also provide consultancy in networking problems. We have contacted the project manager through email. Company D is also working in Pakistan and it is a small software house. They develop different applications for computer systems. They have customer from different countries in the world. They have local offices in two cities of Pakistan: Islamabad, Karachi. We have conducted an interview with Scrum master through phone call. Company E is a world leading provider of telecommunications equipment and related services to mobile and fixed network operators globally. Over 1,000 networks in more than 175 countries utilize their network equipment and 40 percent of all mobile calls are made through their systems. They are one of the few companies worldwide that can offer end-to-end solutions for all major mobile communication

standards. They have a joint venture with Sony Ericsson Mobile Communications; through this they offer a range of mobile devices, including those supporting multimedia applications and other services allowing richer communication. Their headquarter is in Stockholm, Sweden. We have conducted a face to face interview and contacted them through email.

CHAPTER 4: AGILE SOFTWARE DEVELOPMENT

It is widely reported that 80% of all software projects fail, and last year, the IEEE conservatively estimated that over $60 billion dollars was spent on failed software projects. When asked why their projects failed, managers and employees cited a wide range of issues. But these 5 reasons surfaced over and over again, as the main reasons why their projects failed:

Lack of end-user (customer) involvement, Poor requirements, Unrealistic schedules, Lack of change management, Lack of testing, Inflexible and bloated processes [60].

Agile also cannot find the solutions of all expected problem in software development but the solution of all these five reasons can be found in Agile development. Let's take a look at these five reasons.

Customers in Agile are considered as boss, and a member of development team. Customer do the acceptance testing so it can be assure that the project is going in right way. Customer also approves the requirements of the project. In this way Agile overcome the lack of user involvement problem.

To overcome with poor requirements; Agile insists to write acceptance testing before writing code. After gathering the requirements, they are defined as features. By doing acceptance testing before coding, customers have to think about what they have asked to be delivered. In this way the requirements gathering way is changed and also the team can improve quality and schedule the project.

Agile insists that change can be occurring at any level in project except the time of delivery product. The customer is sitting with the development team and customer can change, add or remove some features from project. So having Agile development, it is understood that change is part of Agile development.

Testing in Agile is different than conventional way of development. In Agile, developers need to write test cases first and then the code. So the code should be written according to test cases. Whenever the code is changed the tests are run automatically. This way of testing ensures the quality of product from the start of the project.

Project management is also part of Agile process. For example in scrum team have a schedule to deliver the project, they will have to follow the burn down charts and chart of test pass or fail. In this way they will are managing their project automatically.

There is a special intention towards development in Agile. The new team to Agile development first notice the short cycle times, this short cycle starts from idea and end on a final product. While in conventional way of development, this may take long time. In Agile development lifecycle, it takes couple of weeks.

In short life cycles as in Agile, the work is divided in many small tasks, and each task passes through design, development and testing in very short time. This short cycle change all the processes. In this kind of development, development teams work together in same place, they use lighter weight ways for the development and they use automated testing to save the cost and time.

Having Agile process, development teams have joint intellectual efforts, development and delivery of software at high frequency. Feedback from customer is also faster and in this way they can make corrections and changes accordingly. In this way the return on software can be increased and the risk can be decreased. There are also financial reasons to move towards Agile.

4.1 Software Development Methodologies in Agile

The different methodologies of Agile development are given below:

- Scrum
- Extreme programming (Xp)
- Feature Driven Development (FDD)
- Crystal Clear Methodology (CC)
- Dynamic Systems Development Method (DSDM)
- Adaptive Software Development

But first four methods are more popular and are being used in industry now days. Therefore we will discuss the practices of Scrum, Xp, FDD and CC in this chapter. Most of the companies are working with Scrum methodology of Agile development in combination with Xp, so we felt that it will be not feasible for us to explain all of the practices of all methodologies. That is why we selected some good practices of these four methodologies and tried to explain them as briefly as we could.

4.1.1 Scrum

The 'SCRUM' term is derived from the strategy in the game of rugby. In the game it means, getting an out of play ball back into the game with a team work. The Scrum approach has been to develop to manage the process for system development. This approach does not concentrate on some specific software development techniques for implementation phase. It basically guides the management that how team members can function in orders to achieve the system flexibility in an environment where the requirements are constantly changing [37].The basic idea of Scrum is that; there are several technical and environmental variables that are likely to change during the process. These variables include requirements, resources, time and technology. This makes the system development an unpredictable and difficult process, and if the development team wants to achieve a useful system at the end then the process should have a flexibility to accommodate changes. Scrum can help organizations to achieve better engineering activities like software testing, as it has some frequent activities for management to check the system regularly for any kind of deficiencies or impediments.

The process of Scrum includes different activities for system development. There are some simple steps involved in this methodology. Scrum involves a Rapid prototyping, means that the requirements are gathered incomplete at the start of the system. In this methodology the team simply takes some overview of the desired application from the customer. The initial requirements are not only incomplete but they can also be changed during the process. Scrum contains both managerial and also development processes [38]. So it is helpful for development team and also for management to handle the whole process in a better way. When the requirements are gathered, then the planning takes place. And a simple design for the system is developed. It is not that kind of a conventional way of planning and designing. It is a short meeting that takes place and they finalize the whole planning and designing. After the planning and design phase, the whole project is divided into small iterations that are called sprints. According to [37], before each sprint the teams plan sprints, identify the backlog items and assign it to different teams. Each sprint consists of some backlog items. A backlog is the amount or the work to be done [37].

Each sprint has a backlog, and each sprint implements a fixed number of backlog items of a project. The team members identify the total number of backlog items to be implemented, and after each sprint they also review it. After the completion of each sprint the team members have a meeting and they discuss that which backlog items they have completed and what is need to be done. They also discuss that how they have completed their previous tasks and what were the drawbacks in their previous sprint. At the end they integrate their work and send to it the customer for acceptance testing. The customer has already done with acceptance test cases, so he tests the system and provides his feedback to the management. If there any bugs or they have any further implementation remaining then these

items are added to the product backlog items to be done in the next iteration. Product backlog is the complete list of requirements that includes the total bugs, requests from the customer, usability and performance improvements [37]. If the customer feels that this delivered product has enough to meet their requirements then the further development is closed. When enough backlogs are implemented, the customer believes that the release is worthy; the team closes the development [37].

During scrum, the teams have daily meetings of 15 minutes. During the meeting they remain stand just to make it short. They discuss that what each of the team member will do on that day and also if someone is facing any problem during implementation then he/she can also discuss it with other senior team members. They also discuss the blocks that must be cleared. During the sprints, teams develop, wrap, review and adjust each backlog item. [37] Says that, during development the teams implement backlog item, write code, test and documents the changes. In wrapping, in order to demonstrate the changes, they create the executables. They add new backlog items, possible risks and features during the review and to update the change they finally consolidate the data from review.

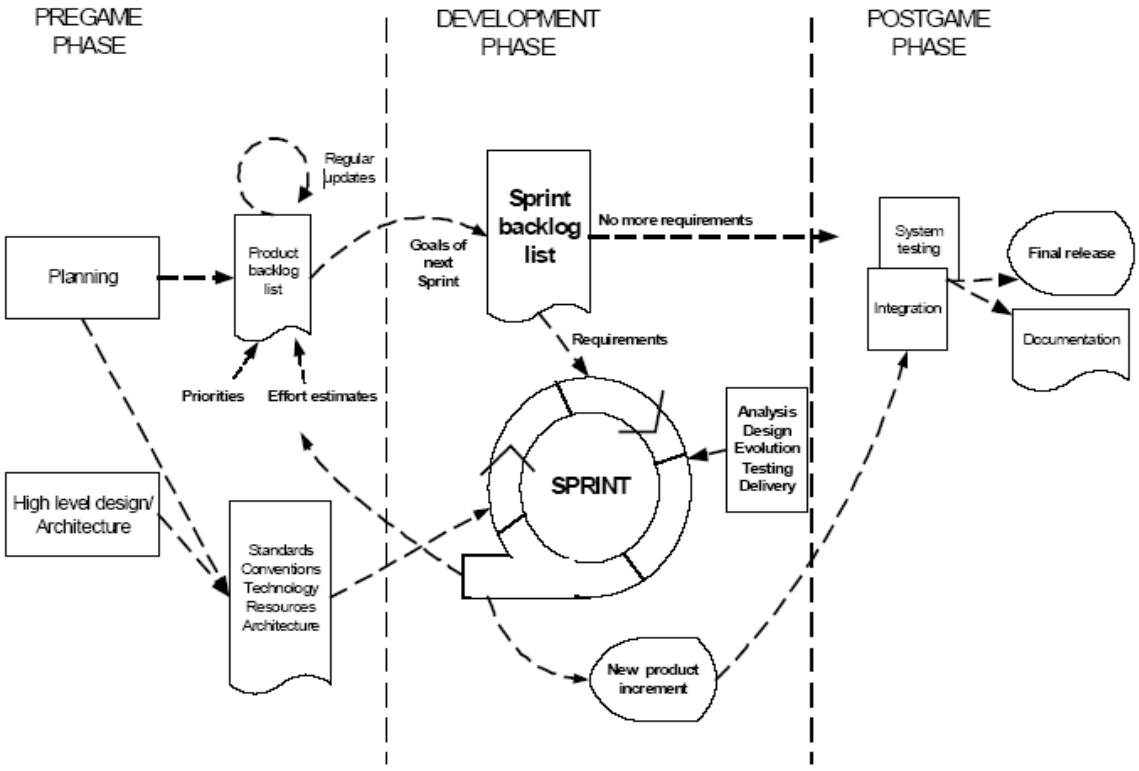

Figure 5: The Scrum process [37]

Figure 5 shows that, in Scrum include three different phases, the pregame phase, development and the postgame phase. In the pregame the requirements are gathered from the customer, and planning for the system is done. Planning of the system includes; standards, conventions, technology, resources and architecture. A huge attention is given to the high level design and architecture in Scrum. On the basis of planning a product backlog list is maintained. This list is regularly updated. All the tasks are prioritized and on the basis of prioritization level they are sending to the sprint backlog items. In the development phase the team has a sprint meeting in which they decide the roles and responsibilities. There is also a daily meeting for a close coordination and to discuss any issues in the development. Development phase is a close phase, means no further requirements can be entered in this phase when they are in the middle of development. Every sprint has analysis, design, evaluation, testing and delivery phases. After each sprint the team has a sprint review meeting to analyze themselves. When the development team has developed all of the sprint backlog items, the system is integrated and sends to the customer for acceptance testing. If the customer finds any bug in the system, it is again included

in the product backlog list for next iteration. In this way the system is developed in the form of small iterations and feedback from the customer is received regularly.

i. 4.1.1.1 Quality Assurance Activities in Scrum

Scrum is a management methodology. It has some important rules and practices for management, and management can get help to organize and better handle their processes by using this methodology. So, it is not an engineering process with some defined quality assurance activities. Management can introduce any activities by their own to get a quality product. Dr. Ralph van Roosmalen says that, Scrum is a framework for project management and it does not contain any help for testing or development practices. Mostly companies working in Scrum use its combination with Xp. In this way Scrum assist in the project management and the practices of Xp are used to guide development. It is hard for a traditional tester to set himself in the Scrum project as for as the testing is concerned. There is nothing about testing in Scrum and Xp does have something about testing but we cannot call it a guidebook for a tester. Most common quality assurance activities that are being practiced in companies working in Scrum are:

- Unit testing
- Continuous integration
- Regular sprint meeting
- Regular daily meeting
- Strict to coding and design standard
- Acceptance testing
- Test automation
- Exploratory Testing

Iterative lifecycle and frequent communication are the important aspects of SCRUM for a tester. There two require some adjustments from the tester's side, and they can keep the some things in mind. Testing of the product is done during the iteration and not at the end of the development life cycle. It is the duty of the tester to decide that what to test when the product is still unfinished. Scrum is all about working as a team, and collaboration is the basic idea. So, for a tester it is highly recommended that he/she must work closely with other team members rather then working in isolation. Testers should be given an active participation in the daily meeting and a tester should be present at daily status meetings that and it is maximum of 15 minutes long. For a tester perspective it is worthy and quality oriented if he/she works with other testers and figures out that what to test, instead of testing from the requirements documents.

4.1.2 eXtreme Programming (Xp)

Extreme programming (Xp) has been introduced as a result to the long development cycles of tradition and conventional development models [39]. It is one of the known methodologies of Agile development. It is a collection of different practices, rules and ides that are inherited from some previous methodologies. The main characteristics of Xp are short iterations with rapid feedback and small releases, the regular participation of customer, continuous testing and integration activities, collective ownership of the code, not a huge amount of documentation and pair programming [37]. In Xp the physical environment is really important, and Xp is more effective in the companies with smaller and medium sizes. According to beck, the team size should be three to twenty members.

In Xp there are three to ten programmers in a team, and they work in one room with facing monitors outward in a circle. There are also small iterations in Xp. The iteration period is three week long, and the release period is 2 to 5 iterations. Requirements are gathered from the customer. Index cards are used to gather requirements. Customer writes stories on simple index card. Developers and customers have regular meetings, in which they discuss different aspects of the system requirements. Customers

and programmers negotiate on what they will do in next iteration, and customer also prioritizes, alters and minimizes the scope of the work. The programmer estimates the time it will take to complete each iteration. He/she writes the task for each story on wall or on a white board. Developer discusses each story with customer, and writes it on story card to know everyone that what's going on.

During the development phase, programmer work in pairs. They follow strict coding and design standards, and whenever they make any changes; they test and integrate. They develop tiny increments of fifteen minutes to few hours. While the developers are busy with developing the application and implementing the story cars, the customer customers are writing acceptance tests, and also visit programmers. During the Xp, there regular meeting of developers and these meeting are also the same in Scrum, fifteen minutes long and standing during the whole time. There are meetings at the end of each iteration. While the developers are busy in programming, there are experts who are consistently available at the site for any kind of help and other advantages. The planning and development phase takes two to four weeks. Each release is one to four months long.

User writes the requirements in the form of stories on story cards, on the basis of these user stories; the team plan and design the system. Then release planning takes place, and system is divided into different iterations. After each iteration, the system is send to the customer for acceptance testing. Any feedback or further amendments are added to the next iterations. During the development the customer keeps on writing stories, which are added in release planning phase. Integrate often is an important practice of Xp. Project velocity means the amount of time that is completed and the amount of work to be done. Pair programming is one of the known practices of Xp, in which two programmers work combine by sitting in front of one monitor. They share ideas and add their own innovation into the design. Refactoring is also an important practice that means restructuring the whole system by removing the duplications, improving communication and making the system more and more flexible [37]. In whole of the Xp process, there is a close coordination with the customer. The developers argue and negotiate with the customer on requirements. The developer estimates the total time of implementing whole of the story cards, and on the basis of that customer decides the delivery time.

According to [37], companies have reported a great increase in their total outcomes. A company used Xp in web development project has reported a 66.3% increase in the total lines of codes in their total project. Practitioners say that the use of pair programming is really useful in companies during development. It can increase the total outcome of the developers. They can work in a better way by maintaining a better design and coding standard. If any of them sees anything noticeable, then he/she can change it to make it better. They can cross check each other's work too. This practice has made Xp really popular.

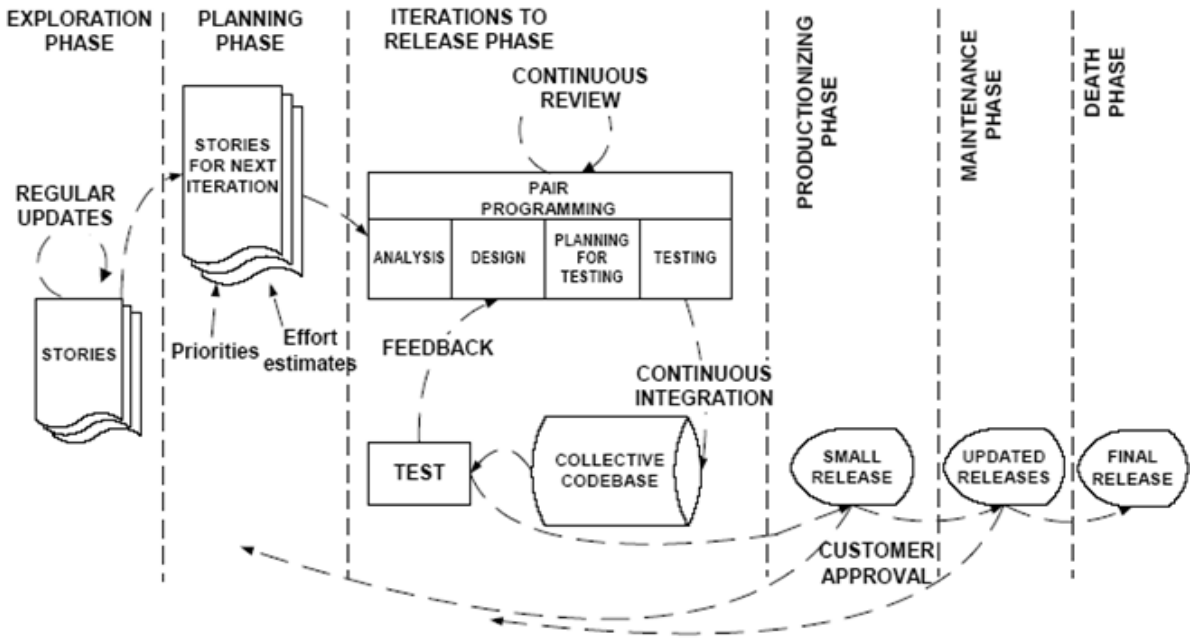

Figure 6: Xp process [37]

Figure 6 shows that Xp process contains six different phases. In the exploration phase requirements are gathered. Customer writes the stories on index cards. He writes everything he wants to be implemented. The project team tries to make them ready in the mean time. They select tools, technology and practices for the project. The user stories are regularly updated. In the planning phase, the project team along with the customers sets the priority to the different tasks. Then the development team estimates the total time each story card will accommodate. Both the parties make an agreement on the first delivery of the small systems' part. The first delivery does not exceed two months time period. The iteration to release phase consists of several small iterations before the final release. Te whole system is divided into small iterations. Each iteration normally takes two to four weeks. The first iteration contains the whole architecture of the system. Customer decides that which functions to be included in which iteration and customer also conducts the functional testing at the end of the each iteration. In each iteration there are certain things that include like; analysis, design, planning for testing and testing of the application. The next one is the product ionizing phase, in which a lot more testing and check of the system is down before its release. There can be some more changes in the design and new changes can be a part of the system if it is the releasing iteration. The system is send for customer's approval, if customer has identified any problems, then the system is sent to the maintenance department and if he needs some more changes in the application then these requirements are sent back to the user stories to be included in the next iteration. Then the system is integrated and finally released in the death phase. During the iteration they integrate their work regularly and eliminate the bugs. They also test each other's work and on the basis of the feedback they update their code. This practice makes the code more refined.

ii. 4.1.2.1 Quality assurance activities in Xp

Xp contains a lot of quality assurance practices during its closed iteration. Xp is a set of rules or practices. When the project team plans different iterations of the system, then system is handed over to the development team. The development team plans each iteration and when they start working on any iteration no more requirements can be added or come inside the closed iteration box. The developers try their best to achieve the refined and quality product. The practices performed in Xp include:

- Test driven development
- Continuous integration
- Pair programming
- Acceptance testing
- Collective code ownership
- Coding standards
- Simple design and continuous refactoring
- On- site customer
- Face to face communication

At the end of the iteration they evaluate the customer's acceptance results. In Xp there are no such quality assurance activities at the time of the release. We have discussed most of the concepts mentioned above in our previous chapters. Test driven development is an important practice in Agile development to achieve a quick and quality product. According to [17], Xp contains a lot of rigorous quality assurance activities for the developers and that also helps to achieve a quality, in case of other development methods of Agile, they leave testing of the product up to the management of the organization to decide that how to test and how to combine different testing activities of traditional ways with Agile ways [43].

4.1.3 Feature Driven Development (FDD)

"Feature-Driven Development (FDD) is a client-centric, architecture-centric, and pragmatic software process. The term "client" in FDD is used to represent what Agile Modeling (AM) refers to as project stakeholders or eXtreme Programming (Xp) calls customers. FDD was first introduced to the world in 1999 via the book Java Modeling in Color with UML, a combination of the software process followed by Jeff DeLuca's company and Peter Coad's concept of features. FDD was first applied on a 15 month, 50-person project for a large Singapore bank in 1997". [45]

Feature driven development is essentially a software management method instead a software engineering life cycle development. FDD is less Agile because it uses many established software development methods. FDD involves planning and up-front modeling and design.

iii. 4.1.3.1 FDD life cycle

There are following five steps of FDD life cycle: [44]

i. Shape Modeling
ii. Build Feature List
iii. Plan by feature
iv. Design by feature
v. Build by feature

Step 1-3 represents the setup time for a FDD project while step 4 and 5 represents the process time.

In FDD testing is done through inspection, FDD takes unit testing build by features; it doesn't define the level of formality for unit testing, it leaves that to chief programmer or team lead to do what is suitable according to project and situation. Xp's unit testing techniques can also be used in FDD. Working of FDD is shown in the following figure:

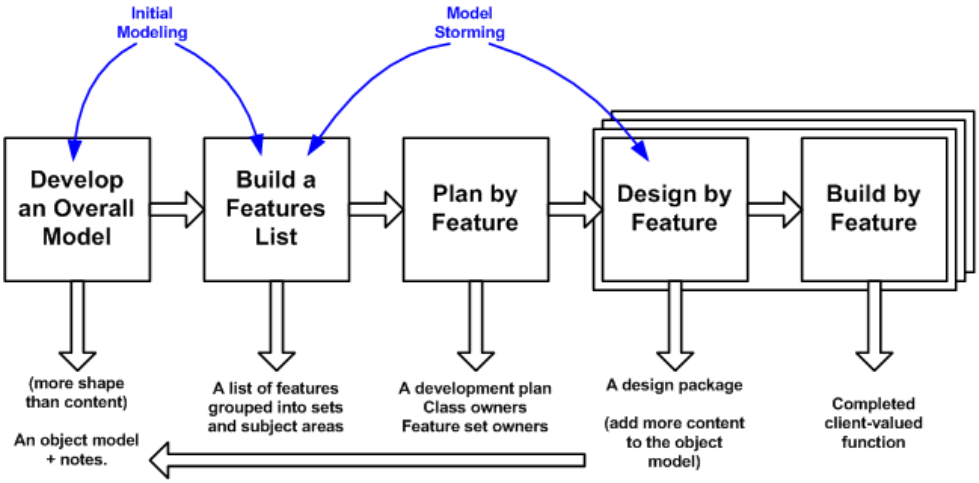

Figure 7: Feature Driven Development Life Cycle [45]

According to [44], there are six primary roles in FDD project: Project manager, Chief Architect, Development Manager, Chief Programmer, Class Owner, and Domain Expert. The concept of class owner is where FDD differs with Xp; in Xp it's a collective ownership the idea in which any developer can update any artifact. While FDD assigns classes to developers and when it need to change in any class then only owner of that class can make change in class i.e. individual code ownership. FDD follows the Agile modeling but the only difference is the class ownership, FDD also insist of regular builds, same as in Xp. AM's practices and methods can be clearly applied on FDD, but mismatch only at two points, class ownership and AM's practice of collective ownership.
There are following five steps of FDD process according to Palmer [49].

iv. Develop an Overall Model

When the development of an overall model begins, the domain experts are already aware of the scope, context and requirements of the system to be built [49]. A walk through is presented in this step, in which the team members are informed of the high-level description of system. The overall domain is divided into further two parts and a detailed walkthrough is held for each domain. After walkthrough the team members work in small groups to produce object models. After producing object models the members discuss and decide the appropriate object models for each domain. Simultaneously, an overall model shape is constructed for the system. [49]

v. Build Feature List

The first step provides a good base to build a good features list for the development of system. In this features list, development team presents the client valued functions included in the system. These presented features are major feature sets.

vi. Plan by Features

In this step, high level plans are being made, in these plans all the features are arranged in sequence by their priority and importance. Furthermore, the classes identified in the first step are assigned to class owner (Individual developer).

vii. Design and Build by Features

This is final step of FDD process; it's an iterative process in which selected features are developed. In this step one iteration takes maximum two weeks. There can be multiple teams that are working on

different features and developing their own feature sets accordingly. This iterative process includes many tasks like: design inspection, coding, unit and integration testing, and code inspection. After every successful iteration, the designing and building of new feature set started. This process is shown in detail in the following figure 8.

viii. 4.1.3.2 Quality assurance activities in Feature Driven Development

In feature driven development most of the quality assurance activities are done during the iteration. This method does not contain a lot of testing activities, and it focuses on the use of the established quality assurance practices with the Agile development method [43]. These activities include:

- Unit testing
- Regular builds
- Design inspection
- Code inspection
- Individual code ownership

At the end of the iteration usually the team does a separate system testing. There are no special activities at the time of release.

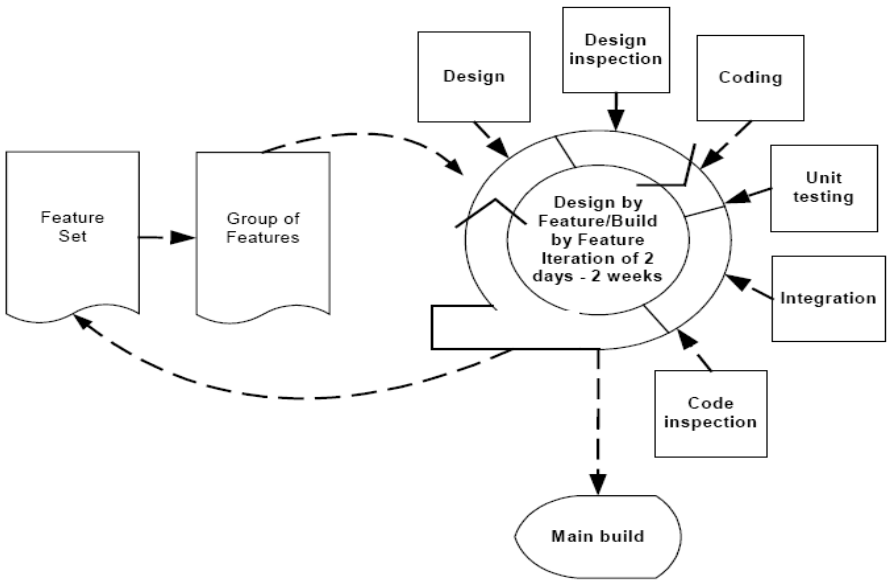

Figure 8: The Design by Feature and Build by Feature process of FDD [13]

4.1.4 Crystal Clear Methodology

Crystal Clear is the smallest of a series of methodologies for software development, all created by Alistair Cockburn. An Agile methodology which is used with small teams i.e. 2-8 persons. According to Alistair *Cockburn*, it attempts to describe a full methodology of the lightest, most habitable kind that will still produce good results. It is prioritized for *project safety* (delivering a system in adequate time / budget per the sponsor's priorities) and for that to be *effective* and *habitable* (meaning the people can live with it and will actually use it).

Review and analysis by [13] crystal family of methodologies includes different methodologies. Each number of crystal family is marked with a color indicating the 'heaviness' of the methodology, i.e. the darker the color the heavier the methodology. Crystal clear suggests choosing the appropriate color of methodology for a project based on size and its criticality as shown in figure 9. Larger projects require

more coordination and heavier methodologies than the smaller ones. If the system will be more critical then it needs more rigors.

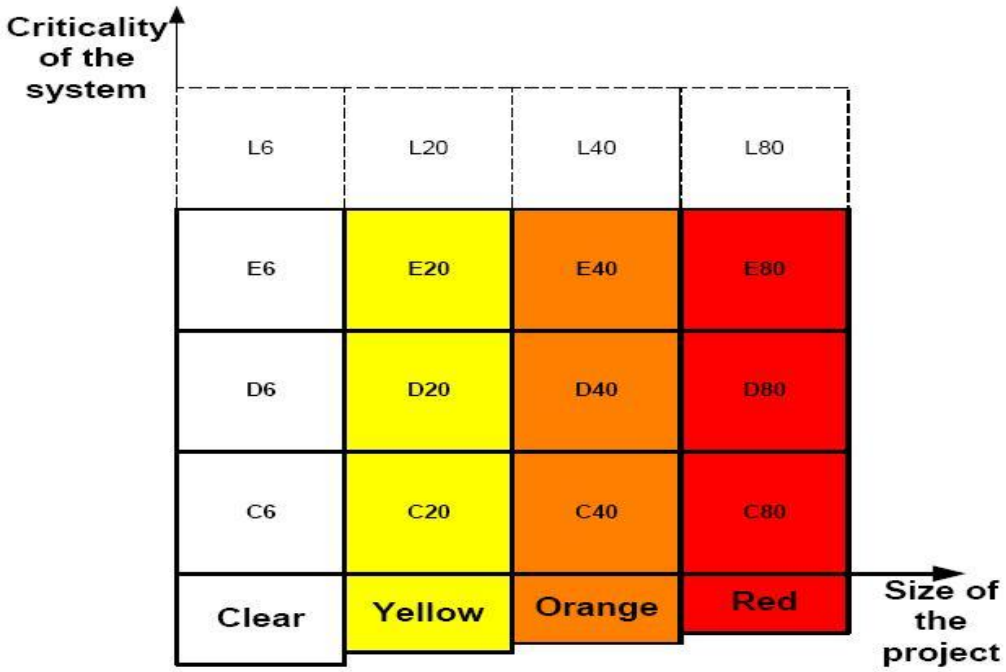

Figure 9: Dimensions of Crystal methodologies [13]

According to [13], there are some common features, rules, and values to all methods in crystal family. In all the projects always incremental development cycles are used with the maximum length of four months, but preferably between one to three months [48]. But the emphasis is on cooperation of people and communication. According to *Cockburn,* crystal methodologies do not limit any development practices, tool or work products, while also allowing the adoption of, for example Xp and Scrum practices.

In Crystal Clear the main requiring separate persons are [48]: sponsor, senior designer-programmer, designer-programmer and user. These embody multiple sub roles. For example, the designer-programmer consists of business class designer, programmer, software documenter and unit tester [47]. The sub roles are: coordinator, business expert and requirements gatherer [48]. The business expert represents a business view in the project, possessing knowledge about the specific business context, He should be able to take care of the business plan, paying attention to what is stable and what is changing [47].

A team using Crystal Clear should be located in a shared office-space due to the limitations in its communication structure [48]. Crystal Clear suggests the following policy standard [48]:

- Incremental delivery on regular basis
- Progress tracking by milestone based on software deliveries and major decisions rather than written documents.
- Direct user involvement
- Automated regression testing of functionality
- Two user viewing per release
- Workshops for product and methodology-tuning at the beginning and the middle of each increment (delivery within a two to three months time frame).

The policy standards of methodology are mandatory but can, however, be replaced with equivalent practices of other methodologies such as Xp and Scrum [48]. There are following safety properties established for crystal clear:

i. Frequent Delivery
ii. Reflective Improvement
iii. Osmotic Communication
iv. Personal Safety
v. Focus
vi. Easy Access to Expert Users
vii. A Technical Environment with Automated Tests, Configuration Management, and Frequent Integration

According to [46], first three properties are required in crystal clear while the other four takes it further into safety zone.

i. Frequent Delivery

Frequent delivery is to deliver a software to user not only iterations. Frequent delivery is not the property for crystal clear or Agile methods only, this is very important for any development process. There are a lot of advantages of frequent deliveries; one can get quick feedback and check the progress of team, users can check with the development that their requirements are fulfilling, developers can work according to deadlines.

Typically it takes about two months for each delivery but if a team is working on a web, then they may deliver weekly. There may be some negative effects of frequent delivery as mentioned in [46] that, if a team deliver frequently, the user will get annoyed with them. If they deliver not frequently, the team can miss the real problem and cannot get proper feedback. Alistair Cockburn mentioned a best strategy in [46] that, in this situation is to find a friendly user who doesn't mind trying out the software, either as a courtesy or out of curiosity. Deploy to that one workstation. This allows the team to practice deployment and get useful feedback from at least one user. If you cannot find a friendly user to deliver to, at least perform a full integration and test as though you were going to. This leaves only deployment with a potential flaw.

ii. Reflective Improvement

Reflect and improvement can be done if a team will get together, discuss the issues wither each other, suggest that what might work better, and make those changes, simply we can say its reflect and improve. After every few weeks, the people get together in a workshop or retrospective to discuss the things working on.

Alistair Cockburn explained an example in [46] that, the project that gave me the surprise was Project Ingrid. At the end of the first iteration — which was supposed to be four months long, but they had extended — they were far behind schedule, demoralized, and with what they recognized as an unacceptable design. It was what they did next that surprised me: They released 23 of the 24 client-side programmers to go back to their old jobs, hired 23 new people, changed the team and management structures, paid for several weeks of programmer training, and started over, requiring the new group to redo the work of the first team and make additional progress.

At the end of the second iteration, they were behind again schedule but had a design that would hold and the team structure and programmers were functioning. They held another reflection workshop, made additional changes, and continued. When I interviewed them, they were in their fourth iteration, ahead of schedule and content with their design and their work practices. The people on the team are

the best equipped to say what is best suited to their situation, which is why Crystal Clear leaves so many details unstated, but for the team to finalize. The Reflective Improvement mechanism allows them to make those adjustments.

iii. Osmotic Communication

Osmotic communication can be defined as: A close communication between a development team sitting in same place, where everyone can hear others. Alistair Cockburn says: Osmotic Communication means that information flows into the background hearing of members of the team, so that they pick up relevant information as though by osmosis. This is normally accomplished by seating them in the same room. Then, when one person asks a question, others in the room can either tune in or tune out, contributing to the discussion or continuing with their work. An example of osmotic communication is given in [46] as:

We had four people doing pair programming. The boss walked in and asked my partner a question. I started answering it, but gave the wrong name of a module. Nancy, programming with Neil, corrected me, without Neil ever noticing that she had spoken or that a question had been asked. It is fact that when people are working in same place then question answering is natural but it is disturbing as well. But the advantage of this communication is that people can get rapid feedback having osmotic communication. This closed communication suitable in small projects having small teams.

iv. Personal Safety

This is not required in crystal clear but it is very important for any project, in personal safety, anyone is able to speak to manager or other team members that the design is not good, schedule is not feasible, or to tell others for improvement. This is very positive factor to come over with weaknesses of team and to make team strong by pointing others with their weaknesses.

Personal Safety is an early step toward Trust. Trust, which involves giving someone else power over oneself, with accompanying risk of personal damage, is the extent to which one is comfortable with handing that person the power. Some people trust others by default, and wait to be hurt before withdrawing the trust. Others are disinclined to trust others, and wait until they see evidence that they won't be hurt before they give the trust. Presence of trust is positively correlated with team performance [46].

v. Focus

Focus means, to know what you are going to do, and then complete that task. Alistair Cockburn says that: Just knowing what is important isn't enough. Developers regularly report that meetings, requests to give demos, and demands to fix run-time bugs keep them from completing their work. It quite typically takes a person about 20 minutes and considerable mental energy to regain their train of thought after one of these interruptions. When the interruptions happen three or four times a day, it is not uncommon for the person to simply idle between interruptions, feeling that it is not worth the energy to get deeply into a train of thought when the next distraction will just show up in the middle of it.

vi. Easy Access to Expert Users

Easy access to expert users provides many advantages to development teams, like rapid feedback, frequent deliveries, and updated requirements without any delay. There are some methods to access the users, according to [46] there are three best user access methods as:
Weekly or semi-weekly user meetings with additional phone calls; you may find that the user loads the team with information in the first weeks. Over time, the developers need less time from the user(s), as they develop code. Eventually, a natural rhythm forms, as the user provides new requirements

information and also reviews draft software. This natural rhythm might involve one, two or three hours a week per expert user. If you add a few phone calls during the week, then questions get answered quickly enough to keep the development team from going off in a false direction.
One or more experienced users directly on the development team; this is only rarely possible, but don't discount it. I periodically find a team located inside the user community, or is in some way collocated with an expert user.
Send the developers to become trainee users for a period; odd though this may sound, some development teams send the developers to either shadow users or become apprentice users themselves. While I don't have a very large base of stories to draw from, I have not yet heard a negative story related to using this strategy. The developers return with an appreciation and respect for the users, and an appreciation for the way their new software can change the working lives of the users.

vii. A Technical Environment with Automated Tests, Configuration Management, and Frequent Integration

Technical environment depends upon; Automated tests, Configuration management, and Frequent integration. Alistair Cockburn says that: One lead designer reported to me that he was unable to convince anyone on his team to run the build more than three times a week. While he did not find this comfortable, it worked for that project. The team used one-month long iterations, had Osmotic Communications, Reflective Improvement, Configuration Management and some Automated Testing in place. Having those properties in place made the frequency of their Frequent Integration less critical. So these three factors play main role to develop a good environment for a development team to make things less critical.

4.1.4.1 Quality Assurance activities in Crystal clear

During the iteration the development team itself performs some quality assurance activities on the product. These activities include:

- Automated tests and frequent integration
- Side by side programming
- Osmotic communication
- Easy access to expert users

Crystal clear only contains quality assurance activities during the iteration. There are no QA activities at the end of the iteration and also at the release time.

4.2 Agile Development Practices in Industry

We have conducted an industry survey and tried to get feedback from the practitioners. We also investigated that what are the practices of Agile development in industry. We tried to highlight that how the companies are actually practicing Agile? How they are conducting quality assurance activities and especially software testing activities in their development processes? How they are managing the manual and automated testing and the roles and responsibilities of testers and developers in their organization. Section 3.2 describes all the details of industry survey.

During the survey we came to knew that most of the companies are practicing SCRUM as a development methodology. The reason they have given is that it is really useful as for as the management perspective is concerned. Most of our sample companies are not following Agile development completely. They are practicing the best suitable contents of Scrum for their company. Some of the companies are using Scrum for management and with that they are still practicing conventional way of development. They still have proper phases of requirement, design and testing.

They are using the meetings, sprints and review concepts for their benefits. Some of the bigger companies are practicing Agile in their own ways. They are trying to follow whole of its practices but they feel that it is not feasible for them. Their representative said "we do not follow any rules about how, we just do it our way, and whatever is interesting at the moment". D company said that they need to deliver product with tight schedule that's why they are using tailored approach of Agile methodologies. A company is using Scrum+ the practices of Xp. We found it quite interesting to be mentioned. The list of different common activities of Scrum and Xp are given below:

i. Xp

- collective code ownership
- pair programming
- integrate continually
- frequent releases
- working in sustainable pace

ii. Scrum is Implemented at one and half project, but not for all feedback

- small teams
- w sprints
- sprint planning meeting
- Short iteration
- daily scrum standup
- sprint review meeting
- product backlog
- sprint backlog
- time estimation
- burndown chart

4.2.1 Management of interactions with customer in industry

The close coordination of customer is really important in Agile development. In Agile the customer provides regular feedback to the project team. In Xp, the customer remains on the site for the full time to monitor the code and design standards and also to give suggestions. But, in companies it is really hard for companies to assign someone to regularly present on site because usually they are busy and customer cannot remain on site. We asked companies that how they manage the interactions and communications with the customers? Company A says that, if it is a large project, there is normally an appointed project manager on our side that handles the communication with the customer. And for other projects, they have a support team and account managers. Company B says that, they don't have a direct interaction with the customer; it's their management who interacts with the customer and provides them feedback in case of meetings and story cards. Company C says that, they arrange meeting of business analysts and development leads with customer on twice a week basis. In absence of customer, the business analyst acts role of customer. The company D says that the requirements are finalized by business analysts, demos are conducted by development managers and team leads. So, they as a developer do not have direct interaction with customer. It's their team leads who communicates with the customer and finalize the requirements. According to company E, The communication with customers is handled by the test leader and not the testers. Form the above answers we analyzed that they do have a communication with the customer. It's the management who makes the decision and interacts with the customer. But, they can interact with customer whenever they feel any problem during the implementation of the system.

4.2.2 Critical Issues in Agile Development

There are always two types of issues, one general issue and secondly specific issues. General issues are common in most of organizations while some specific issues which related to some specific organization and these may not be same for other organizations. So, first we will highlight some general issues which we came to know from literature review and then we will include some specific issues faced by different organizations, which we came to know after the survey we conducted for our research.
Some general issues are as follow:

i. Training Requirement

An important issue is how to introduce Agile Methods in an organization and how much formal training is required before a team can start using it. [59]

ii. Culture, People, Communication

[59] Has discussed that, three most important success factors as: culture, people, and communication. To be Agile is a cultural thing. If the culture is not right, then the organization cannot be Agile. In addition, teams need some amount of local control; they must have the ability to adapt working practices as they feel appropriate. The culture must also be supportive of negotiation as negotiation is a big part of the Agile culture.

As discussed above, it is important to have competent team members. Organizations using Agile use fewer, but more competent people. These people must be trusted, and the organization must be willing to live with the decisions developers make, not consistently second-guess their decisions.

Organizations that want to be Agile need to have an environment that facilitates rapid communication between team members. Examples are physically co-located teams and pair programming.

iii. Feedback

Other than these issues, there are some more general issues in which feedback is most important; we can call it stakeholder's participation, because Agile is based on fast feedback from customer, and expects that customer will be available for quick feedback. But sometimes the stakeholder is not available to give the feedback or sometimes stakeholder annoyed to be contacted again and again. So, this issue also affects the success rate of the product.

iv. Requirement

Sometimes People aren't good to identifying what they need? And they change their minds. This can also occur because of the non technicality of the customer. Once customer gets the feedback from the development team, the requirements also get change by customer. Also some times business environment change or market strategy change and developers need to make changes accordingly, which is also a hurdle in the success of a project.

v. Management

If the team had good senior management support throughout the effort as it was clear to them that the team was delivering. The team's senior manager focused on technical issues such as scheduling and estimating and the junior manager focused on softer issues such as ensuring the team had the resources they needed, including training, and that they weren't being delayed by politics. [45]

vi. Team Size

Team Size in Agile development is also an issue. If the selection of team and team size is made good then the project will get better success, but if the decision is bad about choosing appropriate team according to project, then later in the team may face many problems to handle that project.

vii. Multiple roles for an individual

According to [45], many organizations make changes in Agile method techniques to use according to their needs because they find those techniques not suitable for them. When making changes in Agile methods, people should take care of some very important issues which may affect their organization and the success rate. Especially when assigning the roles then avoid one individual covering more than one role at the same time. Each role comes with different set of responsibilities. This separation is not always possible; dealing with conflicts is left to the common sense of the people involved [45].

viii. Ethical Issues

Some ethical/personal issues are also very important for any organization. In these issues trust, courage, respect, openness, and commitment. If there is lack of trust on the team members then they cannot perform well. Team members must be encouraged for their performance to make them more confident. There should be respect of all ideas from team members, if a person don't feel respect or identity then it will affect psychologically and performance will be decreased. If general openness of project and discussion of issues is encouraged during daily meetings then team members will bring problems and in this way the problem can be measured. By daily meetings one can identify the morale of team, if the morale is low then there should be something wrong and the project manager should identify the problem and deal with it.

According to survey of Scott Ambler following are some important points asked by professionals.

87.3% believed that delivery high quality is more important than delivery on time. 87.3% believe that meeting actual needs is more important than building the system to specification. 79.6% believe that providing the ROI is more important than delivering under budget. 75.8% believe that having healthy workplace is more important than delivery on time and on budget. 61.3% believe that delivering when the system is ready to be shipped is more important than delivering on schedule. So the quality, scope, money/budget, and schedule are also important factors to be considered for success.

ix. Testing

There can be several problems for a tester in Agile development environment. If the ester has been working in a conventional way of development and now he/she is switching towards Agile development then it will be even harder for him/her to settle down. The testers find it hard to write test cases, because here they do not have requirements document. They will have requirements on story cards and by looking at them and design they will write test cases. So, some time testers find it harder. They also need to change a little of their attitude by making it destructive to constructive, and that is complete reversal of the testing in conventional way of development. There is no proper planning for testing Agile development as there is no time for that, so that can create some problems.

4.2.3 Critical Issues in Industry, While Practicing Agile Development

During our survey we asked companies about the critical issues they are facing while practicing Agile development. We just wanted to know for future researchers to work on those areas. Company A has reported that, mostly the balance between ensuring the highest possible development speed vs. being able to promise customers fixed dates for features and bug fixes, and providing them with support. Many of customers use old waterfall models for their development, and pressure to adopt their sup-optimal ways of working. The product backlog is not easy to manage, especially to keep them in strictly ordered list. To produce forecast for software releases is not easy without calculation of development speed, and it will be possible by using burn down charts. There is problem to follow and manage sprint retrospective meetings for each sprint.

Company B has reported that, it is really hard to run simultaneous project at same time, because Scrum gets things done and it also takes full eight hours of developer's day, and then other project suffers. So time management is an issue when simultaneous projects are running same time. For a new

department, especially when switching from conventional way to Agile, then developers don't feel easy to use practices. It's just like reinventing the wheel. When bigger tasks are taken on, they are very hard to estimate and that way time and quality suffers.

Company C says that, setting up the environment and the support a team needs is not trivial. It takes time for developers to work in group.

Company D says that, most of problems occur when achieving timelines. Company E says that, the most critical issue is time. If designers aren't ready with the functionality in time, the time to test shrinks. Other issue is if the code/functionality contains several faults then you need to test it more to make sure that all faults are corrected and then other functionality perhaps won't be tested as much as needed.

From the feedback by companies we analyzed that mostly the problems are regarding the proper management of the Agile development process. For example some of the companies have reported problems with their product backlog item. It means that they are facing difficulty to get a fix listed backlog item, which is an Agile practice. Some have problems form the customer's side which is quite natural. But educating a customer of convincing him/her is also a difficult task. If the customer is insisting on using the conventional way of development then it can be so hard for the company to settle down the things. Time estimation problem has been reported by more then one companies. And especially when they are working with the bigger project it is even harder. The company E, which has reported a critical issue with testing and time management for testing, is a bigger company in software industry. It means that bigger companies do have some problems in managing the testing practices in Agile development. We feel that time management is really important aspect in Agile practices, and companies need to plan for testing and should give proper time to the testing of a system in a sprint.

CHAPTER 5: SOFTWARE TESTING PROCESS IN AGILE DEVELOPMENT

According to Edsger Dijkstra, "*Program testing can be used to show the presence of bugs, but never to show their absence!*"

Software testing is a process, or a series of processes, designed to make sure computer code does what it was designed to do and that it does not do anything unintended. Software should be predictable and consistent, offering no surprises to users. [51]

Complete software testing is impossible, not only theoretical but in practical as well. There are always two ways to test any software; automated testing or manual testing. Both have their own advantages and disadvantages. Manual testing requires a good knowledge of testing engineer, where the automated testing can be performed for a large number of tests in short time. Mostly people prefer to have automated testing, but even then there is still need of manual testing to get rid of more bugs. There are many factors involved to get better results, for example, when to automate the test and when to manual? In software industry both manual and automated testing are in use but most of people prefer to use automated testing to save the time and cost. Test automation or manual testing depends upon the scenario and its importance, if the scenario is not so important then manual is more suitable.

5.1 Quality Assurance activities of Agile in Industry

We asked the companies that how they are practicing quality assurance activities in their Agile development process? Different companies came up with some different answers but almost related to each other. Company A is practicing continuous integration using the Bamboo build/test server from Atlassian. For each check-in, the code is built and tested for a number of configurations and target environments. If an error is found in the test suite, the screen in the development room shows a red warning, and emails are sent out to the relevant people. Then the developers can fix this problem. Their main focus for quality assurance in the sprint is on the unit testing and regularly integration testing. The developers and testers are adding new test cases to the test suite that is run on the continuous integration server as part of daily development and test activities. They are also adding performance and stress tests to the test suite. And at the end of the each sprint they have a separate regression testing. Company B says that to assure the quality in a developed code they do informal testing regularly and meet every day for the scrum meeting. Users and "customers" (concept designers) also sit with developers during sprint meeting. Developers meet with "customers" whenever they find any difficulty or misunderstanding. They have a regular and continuous interaction with customer. Usually they directly meet or send an email to the "customers" daily, without taking any appointments. In company C, the developer commits work after proper unit testing, then work done a developer is tested by team lead, and it is sent to QA team. Means the work is first tested or reviewed by the team lead and then is send to the testing team for testing. Company D is following a tailored approach for development and same is used here for testing. The quality assurance works closely with the development team before delivering it for testing in order to ensure timely delivery to the market. Testing is done parallel to the development and final testing is done after the completion of the product. From above results we analyzed that in industry most of the companies are assuring their quality inside a sprint by unit testing and automation testing and also at the end of the sprint by manual testing (regression testing). While the company E responded as: Continuous build, so everything checked-in into Clear Case is buildable (and hopefully working). Also nightly build and automatically installation with regression test run. All of the companies are using testing teams manually to assure their quality of product. Every company has a testing department. Before, release testing of the system is done to assure the quality.

5.2 Automated and Manual Testing

5.2.1 Automated Testing

Automated testing can be defined as a testing in which no human intervention is involved. Test automation depends upon the importance of scenario, if the scenario is not so important then manual is suitable. And also another reason to automate the test is: when there are hundreds of lines of code to test or there is repetition then automated testing is more suitable to save the time. There are some pros and cons of automated testing. Some of pros are as:

- Automation is best to run repeatedly tests.
- It gives the ability to main stream scenario and run automation against code that frequently changes to catch regressions in a timely manner.
- Automated tests can be run at the same time on different machines, whereas the manual tests would have to be run sequentially.
- Cost is low when long term testing.

Following can be the cons of automated testing:

- The cost of automation is more, especially when you writing the tests or configuring the automate framework.
- The visual reference cannot be automated e.g. if the font color or size can't be defined via code then it is manual test.
- If the tool has limitations then those tests are manual.
- The big con is that it does not find a new bug.
- The cost of test automation is high when its short term testing.

Other than these there are many factors involved in testing. It mostly depends upon the project type and company's management, who decide that when to automate the test. Test automation depends upon scenarios, in some cases manual testing is not possible, for example when there are a lot of repetition, there is thousand lines of codes or when the testing is long term then only automated testing is suitable to save the time and cost.

5.2.2 Manual Testing

Manual testing is vice versa of automated testing, manual testing is a testing in which human interventions are involved. In this testing, test engineers test the code their self. For manual testing a good knowledge of software tester is required and a software tester possess a certain set of qualities i.e. patient, observant, speculative, creative, innovative, open-minded, resourceful, un-opinionated, and skillful. Having all these qualities a software tester can find more bugs, but complete testing is not possible. Manual testing is suitable if a program is not so important. Some of pros and cons of manual testing are:

- If a test case runs only twice then it should be manual to save the cost.
- Ad-hoc or random testing is possible for tester.
- More bugs can be found via ad-hoc technique as compare to automation.
- There are no limitations in manual testing.
- You can find new bugs using manual testing.
- Manual testing is more time consuming.
- After every new build code tester must rerun all required tests again, this will be a huge at the end.

- If testing is long term then manual test is time consuming as well as higher cost.
- To report the first bug to programmer, manual testing takes less time than automated.

There would be more depending upon the nature of program and scenarios. Same effort is required each time and manual test is not reusable. An important factor that is natural in human that, no one likes to fill in the same form again and again that's why manual testing is boring as well, and also it takes more effort and time than automated testing. But even then manual testing is necessary to find out more bugs that are not possible through automated testing. Statistics shown that only 5-20 percent of bugs are found through automated testing while 80% of bugs are found with manual testing, where some of the 5-20% bugs are found during the creation of test cases, it means human contribution is more important to discover the bugs.

5.3 When to Automate Test?

Automated testing is one of biggest investment that a software company makes in testing. The maintenance of test scripts is more difficult and time consuming than maintaining manual test cases. So, now the question is: How to recognize that when we should automate the test and when to run manually? The answer of this question depends upon many factors, like; different scenarios, company's management, how important the program is, time dead line, cost budget, and also regressions in program. All these factors are very important to recognize that whether automate the test or run manually.

Functional testing can often be automated and mostly it is part of regression testing or smoke testing.

If we talk about GUI, then the basic user interface testing can be automated but when there are frequent changes in user interface then automated testing may cause the high cost.
If many workflows involve human intervention then one should not automate the test.
According to [50], rerunning a test against a new release to ensure that behavior remains unbroken or to confirm that a bug fix did indeed fix the underlying problem is a perfect fit for automated testing.

Programs in which there are huge sequences of data, inputs to system or transactions in a random order are good to automate. For load testing automation is a best way, especially when we have to check that what a system would respond against hundreds of users.

Other then all these, there are some scenarios discussed by 'Brian Marick' are as follow:

- Both automation and manual testing are plausible. That's not always the case. For example, load testing often requires the creation of heavy user workloads. Even if it were possible to arrange for 300 testers to use the product simultaneously, it's surely not cost-effective. Load tests need to be automated.

- There is no mandate to automate. Management accepts the notion that some of your tests will be automated and some will be manual.

- You first design the test and then decide whether it should be automated. In reality, it's common for the needs of automation to influence the design.

- You have a certain amount of time to finish your testing. You should do the best testing possible in that time. The argument also applies in the less common situation of deciding on the tests first, then on how much time is required.

- If the product is being tested through a GUI (graphical user interface), and your automation style is to write scripts (essentially simple programs) that drive the GUI, an automated test may be several times as expensive as a manual test.

- If you're testing a compiler, automation might be only a little more expensive than manual testing, because most of the effort will go into writing test programs for the compiler to compile.

These are some possible scenarios by a practitioner, but there can be still many scenarios in which, whether automated testing fits or manual. In the following figure another example of failure or life time of automated testing is shown:

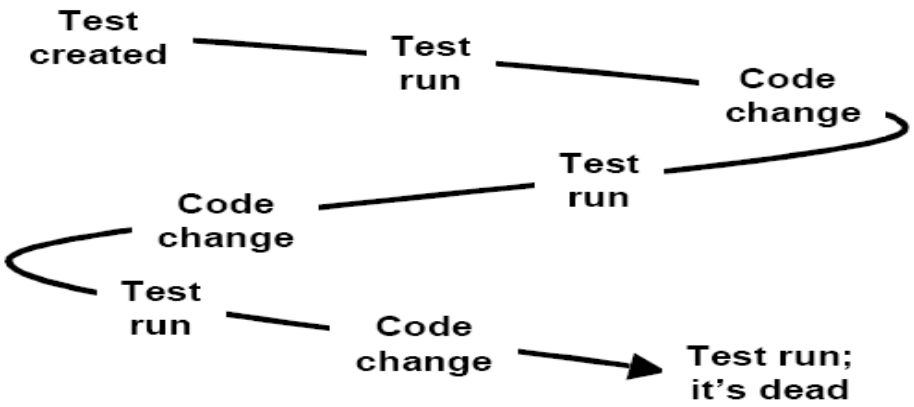

Figure 10: Test's lifespan [50]

In this figure, lifespan of test is shown, at some points the product will be changed and the requirements will be also changed, then test will be failed cause of those changes. But an automated test suit can explore the whole product in shirt time while manual test takes long time to revise. The cost of automating a test is best measured by the number of manual tests [50].

5.4 Automated testing in Agile

The difference between an Agile and waterfall method is the short feedback loop. Automation of tests can help Agile methods to shorten the feedback loop. In traditional processes the automated testing is replacing the manual test procedures. In the traditional way of development, the tests are normally written by the testers. Testers test the system and send the feedback without looking and taking in to account the current changes and updates into the code. These tests involve testing the functionality of the whole application. So in this way the developer gets the late and incomplete feedback from the tester. Test automation is a good thing, which can make the testing little easier. It can save money and make the process more effective. In Agile methods the test automation is used to execute the unit testing, which is helpful to test the smallest pieces of the system. This thing can be done very quickly and regularly. It can help the developers to execute the same tests again and again and test the system after each update in the code, and also helpful to make the feedback process short.

The role of automation is really appreciable in Agile development, and it still has an important place in the world of Agile Development. Automation has several advantages like: repeatability, consistency, and better coverage by driving data. The automation cannot eliminate all of the bugs in the system of its own, for the purpose manual testing is also required. During the development there will be some time when it becomes a regression testing, where the system is checked for errors and then send back for redo and so on.

There can be some issues regarding implementing automation in Agile development. As the time period of Agile projects or iterations are less then one month and there is not a lot of planning involved in this process. As mostly Agile projects are made on some loose design and it also developed on the basis of some system metaphors. This can create some issues, because automation is a complex task and it requires some planning. It requires some time to develop these tests. But there is a solution or recommendation for this problem. Automated system tester should develop the automated testing metaphors in the mean time when the project team is developing the system design. This will give them a design approach to test the system. As the teams do some planning, before each iteration then there should be some extra time designated to the automation testing planning and their infrastructure development.

[40] Has shared his experiences while adopting Agile in their company. They faced different problem and challenges at the start. One of the biggest challenge was the software testing. According to him, selection of a proper automation tool is always a difficult task. Despite of the fact that they never wanted to use any manual testing, they have used a combination of both, manual testing and automated testing. The testing was never completely automated. If the automation of acceptance testing will be part of a sprint definition and the "done" criteria, the team must develop a test strategy and automation parallel or in advance of development. This can keep the feedback loop in sync with development the testing activities do not take any time. If the feedback is not in sync with development then it can create some confusion that either the task completed is "done" or not [40].

The philosophy of Agile development is to react to the changes in the requirements so quickly. The testers should also adopt the same philosophy while developing automated test scripts. Automated testers can quickly do the execution by using the external and function for allowing quick changes to use script for future usage. In the Agile development the teams have a close coordination among them, and they communicate with each other regularly. The automated testers should also communicate and interact with the teams to discuss the design principles and they also should decide the scripts to be integrated for regression testing. The regression testing should be there to refine the system.

Automation test cases are helpful to capture and store the assets (test documents, test cases etc.), by allowing them to regularly getting a satisfaction and saving some time for the company. Agile development is all about working piece of software, individuals, close collaboration of customer and quick respond to change. Agile has been working for long time and its practitioners are saying that we don't need testing specialists but now they say that may be they need them [42]. The automation still has some usage in Agile development and test planning tools can be used with Agile development to make it work in a better way.

5.5 Manual and Automated Testing in Industry

From getting their feed back against our question that how they are managing manual and automated testing in their development process?, we analyzed that most of our sample companies are using both of manual testing and automated testing. Some of them are using automated tools for that and relying more on the automation for catching bugs during the sprints. Company A is using the Bamboo build/test server from Atlassian for continuous integration. They also manually test the product before each official release. The customer teams also verify that the updates work on the customer's handsets before sending them new versions. They do have a regression testing before the final release to assure that no more errors are present in the system.

Company B is not using any automated tools and according to them, they are relying more on manual testing. They hand out prototype builds to a group of internal or external testers. As bugs occur, they report them; the developers try to fix most of them right away, and if there are some harder ones that

are difficult to handle by that time, then they are put on the list for next sprint. New feature requests from stakeholders are taken care of directly, or put on the list in the product backlog.

Company C is using automated and manual testing for quality assurance. They are using tools like JUnit, for automated testing. They use it while developer is testing his/her own work before sending it to the team lead. The team lead then review his/her work and sends to the testing department for more detailed manual testing. Company D is practicing both automated and manual testing is done, load and stress testing is done using custom made tools specific to the product and also by using Mercury load testing tools. Functional testing is normally done manually.

Company E is using both manual and automated testing techniques. All tests, manual and automated, are documented and stored. The automated tests are stored in test suites and run both as regression tests and functional tests. But the emphasis is on automated tests.

From the above feedback we analyzed that most of the companies in industry are utilizing both of testing ways like: manual and automated testing. The only difference is the use of automation tools. The sample companies were the well known and bigger companies in the industry. When asked about why to use both testing ways? They answered that they cannot rely on one kind of testing, because automated testing is not enough to get a quality product and if they do only manual testing then it can take a lot of time and effort. In the result of that the project can go out of time and budget. So, that is why they use automated testing to test the application as soon as possible but still there remain some complex problems that need to be addressed through manual testing. During the sprint the developer writes test cases to test his/her own code.

5.6 Good practices in Automated Testing

As we have mentioned above that automation tools are really helpful for an organization to test and refine their application in a more suitable and rapid way. It is also a tricky thing to decide that which tests need to be automated and which of them should be conducted manually. According to literature there are some practices that should be considered before conducting automated testing. Selecting the proper tool and using it properly is also important, but these practices can also make some different as for as the effectiveness of testing is concerned. There are three main steps involved in automation. First is to prepare for the automation, and then selecting the team and tool and the finally implementing the automated testing. We found some really interesting practices while performing all the above three steps.

While preparing for automation of tests, it is really important for companies to set a realistic outcome or expectation from the automation results. As it is highly impractical to automate the whole bunch of test cases, so, determining some test cases that can provide the best outcomes is important and also a good practice. The purpose of automation is proper usage of the tester's time; it is not used to eliminate them. The test cases that are run frequently, just like regression tests and the tests that require a huge amount of data as an input are the good choices for automation. If there is a test, which require some human intervention, then that test is not a suitable choice for automation. There are certain other cases, in which a test case should be automated and automation can be beneficial. Those cases can be:

- The test is repetitive, and the developer needs to test the system after every change.
- If the test is helpful to evaluate some high risk conditions.
- If it is hard to perform the test manually or it costs enough.
- If it requires different data values to perform the same action.
- If the test is the base for any configuration.

If we automate the repetitive and boring tasks then it can help the testers to do some interesting and in depth stuff. If there is an area in your application that is modified or changed daily then it is really hard to find that what is a passing result and you need to keep a close eye on those cases.

Prepare the applications for automated testing is also a good practice. There are different properties of automation tools to locate the control of the application. It is a possibility that the script uses the label of the button to locate the object, but if the user changes the button name from submit to OK then it will be impossible for the script to locate the button. So, if the developer makes any changes in the application then he/she should immediately inform to the tester, so that the tester can make the necessary changes in the script. Another good practice in this regard is that, the developer gives a unique name to the controls, because it will help tester and the script to identify the control more easily. It will also reduce the chances that the tool needs to rely on the location coordinates to find the control. There should be proper search criteria. If the tester finds that controls do not have unique search criteria then he/she must contact the development team in order to determine a set of different control properties that can combine to create solid search criteria. There should be a defined standard display resolution. Sometimes the tools use different resolution values to locate a control. So by changing the display setting can affect the script execution. The display setting should be a standard one. If the testing activities require different display settings then try to create different subroutines for different settings, so that it can avoid any effects in the system testing.

When the companies are preparing their automation tools and teams, there can be some good practices that can be helpful in order to perform effective automation. The managers should identify the strengths and weaknesses of their team in good way, and should manage them in a way that they know that who is good at doing what. They should divide their efforts by assigning the work to the individuals who are better at performing those tasks then their fellow employees. For example there are some people in the company who are good at script writing instead of test cases and vice versa. Then create an automation test plan. A plan is also worthy for starting any process and to decide that which test should be automated. Automation plan allows identifying the set of tests to automate initially, and also serves as a guide for future tests.
Try to identify your goals for the automation. The following can be some possible goals:

• Speeding up the testing process to allow quick releases
• To allow the testing of the application to occur more frequently
• Improving the test coverage
• To ensure the consistency in the application
• To improve the reliability of testing process

In the planning, try to document the different types of tests or processes to be automated. It includes following:

• Regression testing – These tests are run for every build
• Data-driven testing – They require different sets of data over many iterations of the test
• Configuration tests – These are run on many different platforms
• Backend/Non-GUI testing – These tests verify the data which is collected by the application

We need to define some measurements for success of automation. In the automation plan, you need to mention that how you will analyze if your automation is effective. We need to set some measures to conclude your automation success or failure results. Some potential measures for successful automation can be like: in a less time, the same number of features is tested; more features are tested in the same time and the cost of testing is lesser then the previous testing cost.

While developing the test cases, try to avoid tests that are only performed a few times. The creation and debugging of automated scripts is time consuming thing. The automated tests should be repeatable, and when you document them, you should also write the expected outcomes, prerequisites, and the goal of the test and also the determination of failure. It should be ensured that each and every test case has proper purpose identifiable results. It is helpful to begin by automating previously

documented, manual test cases. When developing tests try to identify some checkpoints, you can use to verify the test results. Decide by yourself that what changes you can make in the application to get a desired outcome.

The selection of a proper tool for automation is also really important in automation. When you are selecting a test automation tool, you need to check that either this tool supports the technology that is used in your application. Either the tool is technologically updated? It is really hard for company to train its employees for a new tool and the cost on the training can be underestimated. So try to analyze the strengths and weaknesses of your team and see that if the tool is matching with your employees skills. Check that if this tool requires users to be programmers or it allows testers at different levels?

While implementing automated testing, you should document the automated tests. It is a good practice. Writing test plan is a good practice, but in addition to this you must develop an automation testing standards document. Create a standard for recording the scripts, and before recording, establish some naming conventions for scripts and variables. Create a document that contains a list of all automated tests and the purpose for each test. Using the comments is a healthy practice in order to explain the logic of the test.

PLANON is a big company working in the field of software development. They have a policy for automation testing. Their automation approach looks quite interesting. According to them for each Planon software product there exist an automation regression tests and it at least regression test covers the high risk areas of product. The testers discuss test automation with developers. It is the duty of developers to build unit tests and testers review them or highlight any area that is left uncover on the basis of risk and available capacity. Developer and tester needs to have a proper communication to get a better test automation approach. They are conducting manual testing before the every official release, to make sure that program is working accordingly and the results of program are same as required. Also customer teams verify updates work on the customer's handsets before sending them new versions.

The automated tests should be reusable and maintainable. When you are writing the test cases, try to create small scripts rather then combining them to make complex tests, which are difficult to understand, update and debug and also very long. Try to make keep them reusable, understandable, maintainable, and modular.

The script can be make reusable by keeping it small and focused on a single task. For example if you are testing an application and you write a script for testing a small functionality, then you also use that in the future for testing the same functionality on some other form. Using external variables and external data can be helpful in order to make scripts reusable. If the values for input fields are not hard coded in the script, you can use the same script to test different conditions.

If you maintain a document that contains the detailed descriptions of your script, then it can be helpful to make your script understandable. If you write the description, functionality and purpose of the script then it can help to understand that what types of scripts are available and how to combine them into larger tests. Using of comment can be really helpful in order to make the script understandable. The general rule for adding comments is to use them in variables, functions, subroutines, and branching structures just to give a clear picture that how the internal components work.

If you use external data it will help to make scripts reusable and it will also make them easy to maintain. An automation tool that includes a scripting test based interface or which allows you to edit a script in a text editor can reduce maintenance time.

Modular is also an important property as for as the scripting is concerned. If you create scripts to perform specific tasks then it will give you the advantage of combining them to test the complex applications and also to troubleshoot them easily. So, the above best practices form the basis for test

automation effectiveness. If companies implement them, then they can help them to avoid the common mistakes and improve company's testing process regardless of the automation tool they use.

5.7 Separate Testing Team

Testing is an important activity in any software development process. We have discussed in previous chapters, the working and practices of Agile development methodologies. Our previous sections describe that how quality assurance is done in Agile development. We have done a literature review and also conducted an industry survey to know the actual working of software testing process in Agile development. When we talk about introducing a separate testing team, we do not mean to have a team of "monkey testers", who test the system recessively and work as a separate department. We say that a separate team should be part of the project team, that can test can system independently with and intent of finding errors and with the typical tester's mind, but the team should work integrated with the developers and other project team. They should be actively participating in any of the project meeting. But when we talk about testing a system, they should test it with a tester's mind with some Agile thinking.

The focus of Agile software development is on the interactions between the individuals, close collaboration of customer, short development cycles and the frequent deliveries of the working piece of software. If we see it from the perspective of software testing, then it has some challenging things, and Agile methods have some lacking aspects if we compare it with traditional way of testing. There are some really important aspects of software testing that are overviewed in Agile development, and these practices are the fundamental to a successful quality assurance. The Agile working says that, if the developer and customer have a close coordination with one another then there is no need for a separate testing team.

Basically the processes of Agile software development are based on iterative and incremental development, which uses development cycles which are short and often time-boxed. The satisfaction of the customer is at the highest priority in Agile development, by providing him/her with short and regular deliveries. As the fundamentals of Agile development basis on close interaction of the customer and quick responding to change and less emphasis is given to the processes, tools, documentation, and following a plan. These activities are the traditional way of getting quality product and among the best quality assurance and testing practices.

Quality assurance is really an important and a crucial task of most efforts of software development and testing is one of them. In the traditional way of development, testing has a separate phase at the end of the development, and that includes checking the system for bugs and also integration of different parts of the system. Sometimes this can take a long time and often project goes out of time frame. Defiantly in Agile this process should be different but quality is such an important factor and we cannot sacrifice quality for time constraint. Because in Agile time is a crucial factor, and there is no such time for proper planning for a phase and for a process like testing there require a proper planning. Most of the Agile methodologies do not say anything about the testing. Most of them include unit testing, automation testing and acceptance testing at the end of the process. But still only a few methods give any information more then unit and integration testing [21]. So there is need to introduce some practices for software testing of the product in Agile development.

According to Scott Ambler, the qualities of Agile products are because of the different practices like: collaborative working, incremental development, development of the products by using refactoring, test driven development, and modeling and through better communication.

We have discusses earlier the challenges with software testing in Agile development, and it also raises a question that are the quality assurance activities are properly handled in Agile projects? Some authors [17, 53, 54] also have mentioned some problems with Agile quality assurance activities.

5.8 Why Do We Need Separate Testing Team?

In software development there are different roles but in this research we focused on the role of testers. The testers play a key role in development; developers may lack some skills in development which testers provide. But in Agile development the common mistake is to see testers as junior developers. A good tester has many distinguishing feature that make difference with developer. By understanding only this difference, project managers can make a good and successful team.

Which projects may not need independent test staff? The answer depends on the size and context of the project, the risks, the development methodology, the skill and experience of the developers, and other factors. For instance, if the project is a short-term, small, low risk project, with highly experienced programmers utilizing thorough unit testing or test-first development, then test engineers may not be required for the project to succeed. [57]

It would be frustrated for testers which realize that testing is an easy task. Testing requires patience and flexibility. To understand the testing, a person should know the details and understand the big picture of project. The skills and attitude of tester and developer is always different, although a developer can test well or a tester can write a decent code; but the skills that the job requires are always different and opposite. The famous quote about testing is: "*You make it, we break it*".

Software testing provides information about the status of software which is under development and it helps to take decisions. The strength of testers is often their ability to be generalists; they need to have a broad understanding of many areas. This contrasts with developers, who are often required to be specialists' in particular technical areas (networking protocols, for example, or database internals or display libraries). Testers, in contrast, need to be able to get up to speed quickly on any product that they test. They often have little time to learn a new product or feature. On the other hand, developers need to have a thorough understanding of the libraries or protocols they'll be using before they can work on something new. Otherwise, they may break something. They're used to being allowed to gain mastery before being expected to deliver. [55]

A tester need to have the knowledge that a user have, this helps tester use the product as user wants, instead a developer might like the product to be. It is very difficult for a developer to test a product as user's perspective. Good testers are non skilled, they need to know the customer's domain and requirements or they pretend that they don't know things that they are doing. It is a different way to acquiring knowledge.

According to [55], comparison of characteristics of a good tester and a good developer are shown as below:

Table 2. Comparison of good Tester and good Developer

Good Testers	Good Developer
Get up to speed quickly	Thorough understanding
Domain Knowledge	Knowledge of product internals
Ignorance is important	Expertise is important
Model user behavior	Model system design
Focus on what can go wrong	Focus on how it can work
Focus on severity of problem	Focus on interest in problem
Empirical	Theoretical
What's observed	How it's designed
Skeptics	Believers
Tolerate tedium	Automate tedium
Comfortable with conflict	Avoid conflict
Report problems	Understand problem

All above mentioned characteristics clearly differentiate the tester and developer's role. A good developer thinks theoretically, while a good tester think empirically. Developers take much more time to identifying a problem than testers. So, tester should not be judged according to developer's criteria. Good developers and testers need both thinking, but there are times when each discipline stresses different talents and calls for different emphases. [55]

It's hard for developers to learn everything they need through conversation, many factors involved in this like: Customer don't have enough time to detailed conversations.

Developers may understand that they achieved the goal so it's done, but a tester can point out many technical mistakes, and a tester can think in user's perspective.

Experts are characteristically bad at explaining why they do what they do. Their knowledge is tacit, and it's hard for them to make it explicit. It's the tester's responsibility to draw them out. Fortunately, many testers are quick studies of a domain - they've had to be. It's also the testers' responsibility to think of important ideas the business experts and programmers might overlook. For example, both business experts and programmers tend to be focused on achieving return on investment, not on loss. So they concentrate more on what wonderful things a new feature could do, less on what it shouldn't do if people make mistakes (error handling) or intentionally misuse it (security). The tester should fill that gap; make sure the tests describe enough of the whole range of possible uses of the feature. [56]

In Agile development, the developers develop an application, and they conduct an integration and unit testing. Then the working is delivered to the customer for testing. Software testing requires a lot of specific skills and challenges, because it is such a creative and intellectually challenging task. A professional tester is required to do this professional task, so that the task can be performed effectively and efficiently [19, 23]. In Agile methods, testing is usually done as a task that developers do as a part of the development tasks or as a customer's task. Customer is very closely involved in everyday development in some of the Agile methods and takes the responsibility for acceptance testing. This is a problem and can create some major problems too [29, 17]. If your customer has expertise and skills which are necessary for a tester, and has the capabilities to act as a tester, only then you can think of assigning him the task to test the system. It means that, how someone can test an application when he does not have any skills and knowledge about testing software? Dynamic Systems Development Method (DSDM) method has recognized the need for basic skills for a tester and [30] recommends that at least there should be one developer or tester in each team who has received training in testing [30].

Another problem in software testing is so called oracle. The principles which are used for finding the correct result of the test are referred by oracle. It is an important thing for a professional tester to notice the defects that the tests show by thoroughly inspecting the results [19, 17]. A lot of responsibilities of testing are placed on developers in Agile methods. They conduct automated testing and write the tests as well. From a professional tester's point of view it is not obvious that the results will be reveling defect and all of the problems. Automation testing is good; but it does not have the capability and effectiveness to highlight all of the problems.

There are different software development techniques in Agile. Test-Driven Development (TDD) is one of them. It contains short iterations and test cases are developed first then code is written to make pass those test cases. The code is continuously tested until it passes the test cases. These test cases cover the desired improvements, new functionalities and the software is refactoring to accommodate all the required changes. These tests help to provide a rapid feedback after each change. Practitioners stress on the fact that the test-driven development is a method for software designing not to test method. Test driven development also takes time away from the core development and sometimes the code is really hard to test. Test driven development is also not so effective in applications with complex graphical user interface. So we cannot rely on the test driven development to test a system up to quality.

Need of separate testing team is because of its destructive nature and this fact makes software testing more effective. The purpose of testing is to find errors in the program and to highlight some of the bugs that can be the reason of creation of major future problems in the application. According to [22], testing is a process of executing a program with intent of finding errors. Test cases are written for valid, invalid, expected and unexpected conditions, to check the behavior of the application in any case. In Agile methods the testing is done by developers who test their own applications. So, in such case the destructive attitude of a tester is really hard to find. Developers mostly try to test their program to make it run able but developer will not test to make it not working. The focus of Agile methods is much constructive, and they practice constructive testing practices for achieving quality into the product. If we look at the Agile literature, we will see that the working of the methods are in a way that the features and the functions of the application should work instead of finding and highlighting errors in the application so that it cannot work. This can create some problems, because even all of the unit tests are passed the system can still crash. Developers will look for the little errors and problems that can create trouble in the proper running of the application but may be they will skip some bugs that can still manage to crash the system. Professional testers do a lot of testing like: performance testing, reliability testing, functional testing and test other qualities on the system, test case design and test environment setup. These tasks are not directly connected to the development tasks. [23, 26] says that, If you want a program to work, you would like to see a working program, by making you miss failures.

Software testing not only finds the errors but it also tells us that at what level we have achieved the quality. Testing provides information about defects and problems in an application and also evaluates the achieved quality. Metrics are required to evaluate product quality e.g., the total number of found faults, fault classification, and the test coverage. Agile methods contain testing activities and also their proper implementation during the development process. But, this does not give any information about the level of achieved quality of the product, or evaluation of the product. For example, developers write the automated unit tests and always keep running and passing which is a good practice in order to achieve quality. It can be easily tracked by the amount of test cases or the amount of covered methods. But until then it does not give us any information about the achieved quality. For that purpose we need some metrics e.g., faults, test coverage, and quality of the tests [17].

In Xp, DSDM, and FDD the emphasis is given on the importance of the quality assurance activities at the low level, where developer is responsible for most of the QA activities, but it is really unclear that how the software testing activities are related with the other development tasks [17].

[17] Also states some important aspects of software testing process in Agile development. The authors feel that this process has some huge lacking and need to be overcome by more of testing and quality assurance activities. The testing activities in Agile processes are not enough to get a quality product. So they propose a role of independent tester's role at iteration time [17]. [58] stresses on having an integrated teams in Agile development, where the testers do work collectively with developers but with a tester's thinking.

On the basis of discussion in above sections we think that Agile methods should be accompanied with a separate testing team, and they can get more advantages by introducing such team. [17] Also says that Agile development can be benefited through a team of professional testers.

5.9 Practitioners View about Agile Working

During our survey we imposed different questions to companies regarding the implementation of Agile development in industry. We asked them that what they think about the actual working of Agile development where most of the quality assurance is done by the developer and there isn't any tester's specific position; do they feel that those activities are enough to achieve quality in software products?

The answers were quite interesting. Company A says that, it depends; they feel that it is a good idea to have people who focus on ways to test the code base as a whole but the best case is to have a number of test engineers that can do both manual and automated testing. Mostly come up with tools that test the code in many different ways. A close relationship between the developers and the testers is important. The testers should be involved from the planning/requirements phase and start thinking about testing new functionality early. They should also be familiar with testing trends and have a grasp of what is new and interesting when it comes to test theory and tools. And in their organization they don't have very much use of classical "monkey testers", i.e. people that do simple manual testing following a strict test specifications. Company B says that, "No matter the size of the product or the company, I think it's like this: to get a high quality product is mainly a concern of the sense of quality in the head of each developer. If issues of tests, security, safety, stability, etc, and methods (very important) of handling these issues, are fresh in the mind of each developer, the quality come with that. I believe that each developer has to be a tester, when working Agile. A developer that is not also a tester is a bad developer. It's as sad and true as it sounds. To answer the question, Yes, I think the Agile test methods would suffice in small size project, having the right mind of developing. But still, quality fallbacks can of course remain, Agile projects or not, so there is anyway always a need of extensive testing by external expert testers". Company C says that, before writing code they write test cases and after coding they do unit testing. Testing is not quality assurance, so a separate quality assurance is required. Company D says that, as the developers who develop the application cannot test the application effectively by thinking out of the box, therefore testing is done in parallel with development process to overcome the issues related to delivery of a quality product. Company E replied that, Agile working, TDD (test driven development), is a great way to ensure that the product delivered fulfills the requirements stated. Tester and designers, both should be considered as developers. They do what they need to do to ensure the delivery.

From these answers we analyzed that even if the developers develop an application and test it by themselves; even then there will remain some complexities and shortcoming to be tested by the professional testers. All of the sample companies have agreed that there is still a need to have some independent test team; which can test the system with intent of finding errors and can provide more professional feedback. Every sample company has a separate testing department that tests the system before its release. They say that the Agile working can be successful in a small sized company where developer can develop a system and also can test it for his sake, but for a large company, to get a quality product it is really necessary to get a separate testing team.

5.10 Industry Practitioners' View about Having Separate Testing Team

We asked a question to industry practitioners that if the developer and stakeholder have a close coordination then there is no need for a separate testing team. We just wanted to get their feedback that either this rule is actually applicable in industry or not. We feel that for a bigger company it was really impossible to manage it in this way. The company A says that, in some cases this might work in smaller teams, or if the customer has a good test team themselves that can do testing on behalf of the development team. The development team needs to test the code extensively as part of the SCRUM tasks. And a task is not done until the related code is designed, written, reviewed, tested and documented. Company B says that, they have to have close coordination, otherwise it's not Agile. Separate testing is always useful but not always vital. It depends on the product and size of organization. Company C says that, there is much need of a separate testing team to assure bug free code. And company D says that "It is not possible at least for our company, as we are dealing with health care and financial solutions and major and even then minor issues cannot be tolerated". Company E believes that, it is necessary to have a separate testing team for testing. But at the end if needed then developers should also test with testers to make sure the delivery on time.

So, from the above discussion and feedback we analyzed that Agile rule of not having a separate testing team may be work for a really small company, but for a bigger company, who wants to achieve

quality product, it is really impossible and unprofessional not to have a separate test team. As one of the companies said that they cannot tolerate a minor error so they want a testing team. It means that for developers it is impossible to test a system up to quality level. They still need a separate testing team.

5.11 Roles and Responsibilities of Developers and Testers in Industry

In an Agile development process the interesting thing for us was to know the roles of software developers and testers. During the survey we asked to companies that how they are doing their quality assurance activities and what are the roles and responsibilities of software developers and testers in their organization? Company A says that with the primary responsibilities of developing code for an application, the developers write many of their own automated test cases and add them to the test suite as they implement in their process. The latest version of the test suite is run by each developer as he/she develops code. When a bug is fixed, a test case is often added to make sure that the error does not happen again. The testers do manual regression testing before each official release. They also write automated test cases for the test suite. The testers also help the customer with the development of acceptance testing. Sometimes they come to help developers whenever it is required. Company B says that they do not have any formal process or responsibilities for developers and testers. They utilize their duties wherever needed and whenever needed. In company C, the testers have a duty to provide acceptance test to developers, developers write unit test according to these. Testers test their code accordingly. Company D says that, the Unit testing and code review is done by development team, functional, load/stress testing is done by testing team. Company E says that, in the end of the iteration some of the developers can help to test if needed. They also conduct a lot of integration test during development to facilitate function test. It means developers and testers have their own separate roles, but developers can also test if needed.

After analysis we came to know that most of the sample companies are having different roles and responsibilities for testers and developers. The unit testing and automated testing is done by the developer to better test his code. The rest of the complex manual testing is done by the testers. They also use testers to assist developers and customers. It feels like that they have teams which are so integrated and working together.

CHAPTER: 6 RESULTS

Agile development is most practiced methodology now days in the industry. Our previous chapter describes the working and practices of Agile methodologies in theory and in industry. As testing is the most important practice in acquiring quality assurance in a software product, and a huge percentage of the cost and resources of the whole project is spend on testing process. Most of Agile methodologies do not say much about testing process in during the development except Xp, which has some known practices of quality assurance during sprints, to be done by a developer. We have discussed all of the aspects of software testing process in Agile development. We did a literature review and industry survey and analyzed different results.

We have discussed some of the general critical issues while practicing Agile in previous chapters. There are also some companies specific issues that we came to knew while received feedback by companies. We have analyzed that mostly the problems are regarding the proper management of the Agile development process. For example some of the companies have reported problems with their product backlog item. It means that they are facing difficulty to get a fix listed backlog item, which is an Agile practice. Some have problems from the customer's side which is quite natural. But educating a customer of convincing him/her is also a difficult task. If the customer is insisting on using the conventional way of development then it can be so hard for the company to settle down the things. Time estimation problem has been reported by more then one companies. And especially when they are working with the bigger project it is even harder. From here we can say that Agile is good for the projects with small or medium sizes. As companies are not following fully Agile, by just practicing most of its management techniques, so they have not reported any technical or development problem, except the problem of shrinking of time to test the application in case if the designers do not ready in time with the functionality, the testing of a code that has a lot of faults, and with team members while switching from conventional to Agile. We feel that there is need to do some more research on some of Agile areas like, management of project backlog item, problems while switching from conventional to Agile development, communication and interaction, proper assignment of roles and responsibilities and time estimation of the bigger projects. We feel that as for as the project team is concerned, the team management can conduct some training sessions to educate them and train them. This can help them while working within integrated teams. There is also need to educate the customer to tell him/her that why you are using Agile development, because sometimes the customer hesitates to work with an Agile team, because he/she is busy and cannot be continuously on site or in contact. The company E, which has reported a critical issue with testing and time management for testing, is a bigger company in software industry. It means that bigger companies do have some problems in managing the testing practices in Agile development. We feel that time management is really important aspect in Agile practices, and companies need to plan for testing and should give proper time to the testing of a system in a sprint. Some of the above areas need to be explored.

We suggest that a company should utilize a combination of both manual testing and automated testing to get a quality product. There is no doubt that automated testing is quick, efficient and mostly easy to use, but there are some faults which are not high lightened by automated testing. To overcome those faults you need to conduct a manual testing. From the response we received from different companies who are working with Agile development, we analyzed that most of the companies in industry are utilizing both of testing ways like: manual and automated testing. The only difference is the use of automation tools. The sample companies were the well known and bigger companies in the industry. When asked about why to use both testing ways? They answered that they cannot rely on one kind of testing, because automated testing is not enough to get a quality product and if they do only manual testing then it can take a lot of time and effort. In the result of that the project can go out of time and budget. So, that is why they use automated testing to test the application as soon as possible but still there remain some complex problems that need to be addressed through manual testing. During the sprint the developer writes test cases to test his/her own code and then send s the code to the tester for more detailed manual testing. The companies have a process of manually testing the system and especially before the release. They say that they cannot afford even a minor error, and the system

should error free, that is why they conduct manual testing to assure the quality. We have also discussed different aspects in the previous chapters, by using which the management can conduct the testing process in better way. There are different practices they can adopt to make their automation process more effective. They can get the duties of testers to cover all areas of the code.

We have discussed the working of Agile in our previous chapter, and it shows that during its methodologies most of the quality assurance work is done by the developers and agilest believe that the activities done by the developer during the development are enough to get a quality product. Our hypothesis was that, even then those activities are good enough to run that application but we cannot say that; the application has got quality. We came out with a result that there is always a room for a testing team in Agile development to get a quality product. From the literature and the survey, we analyzed that those activities are not enough to get a quality product in case of a bigger company. This rule may work in case of smaller companies but in case of medium and bigger companies you must need a testing team to test the product. From these answers we analyzed that even if the developers develop and application and test it by themselves, even then there will remain some complexities and shortcoming to be tested by the professional testers. All of the sample companies have agreed that there is still a need to have some independent test team. That team can test the system with intent of finding errors and can provide more professional feedback. Every of our sample companies have a separate testing department that tests the system before its release. They say that the Agile working can be successful in a small sized company where developer can develop a system and also can test it for his sake, but for a large company, to get a quality product it is really necessary to get a separate testing team. And also all of our sample companies are using testing teams to assure their quality of product manually. Every company has a testing department. Before the release the testing of the system is done to assure the quality.

So, from the above discussion and survey feedback we analyzed that Agile rule of not having a separate testing team may be work for a really small company, but for a bigger company, who wants to achieve quality product, it is really impossible and unprofessional not to have a separate test team. As one of the companies said that they cannot tolerate a minor error so they want a testing team. It means that for developers it is impossible to test a system up to quality level. They still need a separate testing team. As all of the companies we contacted were using their separate testing team. Developers are still doing the practices of Xp, like unit testing and automation but there are still teams who do some complex testing practices. By accompanying a separate testing team, Agile development can result in more effective and refined products. A software tester is always worthwhile for any development process. A tester will test the system with intent of finding errors and his attitude is always destructive, and this fact can make a system more refined and bug free.

We suggest that, there should be an independent testing team in Agile development methodologies. Here when we say independent, then it does not mean a team that is working as a third party just like a “monkey testers”, we mean that the team should test the system independently with an attitude of tester and should work integrated. They should be some independent thinkers, or conventional testers in new collaborative Agile environment. It is really important to know that what are the roles and responsibilities of software tester and developer in Agile development. From the survey we came to know that most of the sample companies are having different roles and responsibilities for testers and developers. The unit testing and automated testing should be done by the developer to better test his code. The rest of the complex manual testing is done by the testers. They also use testers to assist developers and customers. It feels like that they have teams which are so integrated and work combinable. They have flexible tasks for both of the teams. If the project manager wants to switch someone to other team, he/she can do it. The primary roles are defined but in case of assistance someone can be switched around.

6.1 Communication Model for Effective Working of Independent Software Testing Team in Agile Development

If we introduce a separate testing team then, the important thing is to identify, that how the project management can handle the communication and interactions of software tester with developers and customers. Communication and integration are the soul of Agile development, so we suggested that the testing teams should work very closely with other teams. Assigning the roles and responsibilities of the software testers and developers with respect to Agile development is also of important concern.

We have proposed a communication model for the effective working of an independent testing team. This model will help the practitioners to know that how they can manage the communication and interaction of software testers with the developers and customers in an organization. It will be beneficial for the organizations to follow the same process for managing their testing teams in a better way.

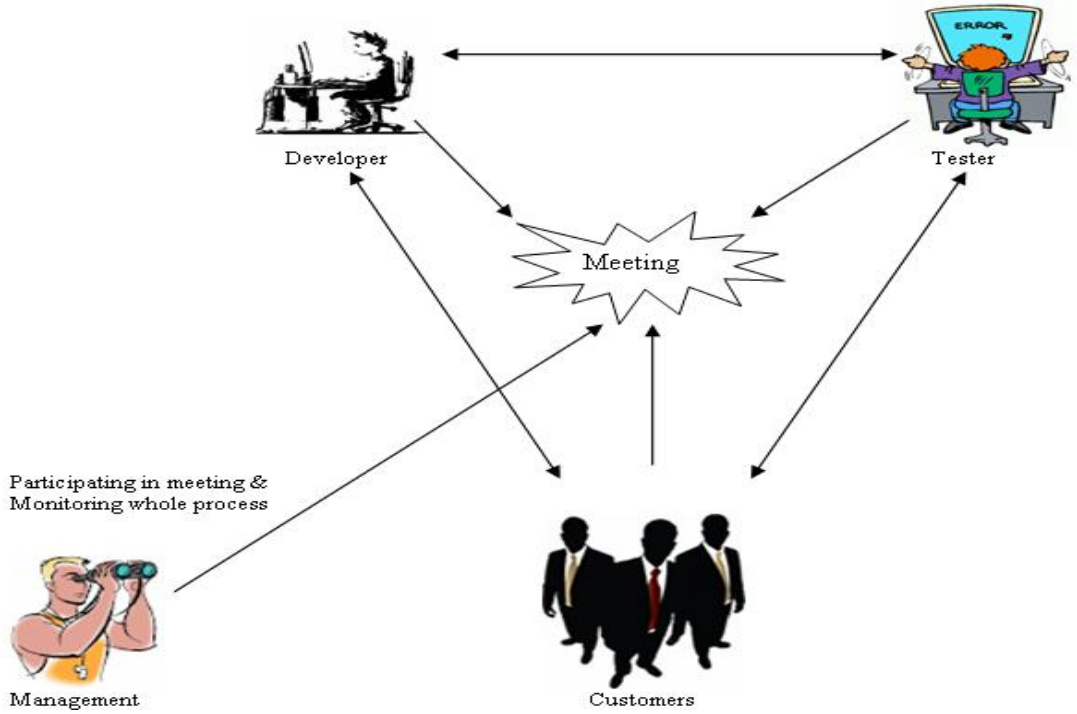

Figure 11: A Communication Model for Effective Working of Independent Software Testing Team in Agile Development

Figure 11 shows a communication model for the effective working of independent software testing team in Agile development. Here are four different actors in the figure. The developer, tester, customer and the management, and it can be a project manager or a scrum master or any one who is heading the project. When we say developer and tester in figure, it means the development department representative, testing department representative and customer representative. There are so many other important members of a project team like: designers etc., but we are focusing on the interaction of the testing team with the above mentioned actors. We suggest that all of these actors should work very closely and must participate in the meetings. There should be regular meeting for quick and regular feedback. The meetings can be of different types, like the internal meeting of the project team and the feedback meeting with the customer. The project team meetings can be a sprint meeting, a daily meeting, a weekly meeting, a monthly meeting, or a review meeting. If it is a project team's internal meeting then project team including developer, tester, designers, project manager and other project team fellows should be present. It is not necessary that every tester or developer should be present at meeting, because from the industry survey we learnt that in most of the bigger companies; only the team leads attend the meeting. For example test manager and development manager can take part if it

is a meeting with the customer, and that is the known practice in industry. Mostly in bigger companies team leads conduct meeting with the customer. Sometimes the customer is really busy and does not have time for regular meetings; in that case he/she conducts meetings only with management. The management then let the other project team know through writing the requirements on the story cards. In the figure each and everyone has a two way communication with each other. The management can directly communicate to anyone at any time. The management is responsible for providing necessary resources, tools, training and assistance to the project team. The management also monitors the whole process. We have described some useful practices in earlier chapters, for management to manage their teams in a good way. Those practices are also applicable in this model.

We feel that the testing team should be a part of all of the planning and designing phase so that they should know the system and also can work as a team. The testers should be a part of all the sprint team meeting and also daily team meetings. We think that in the project meetings testers should be present along with developer and customer. Testers should have an access to all of the resources like story cards etc. The developers should do the unit testing and rest of the testing should be done by the testers. Yes the testers should change a little of their attitude by making themselves flexible by working with the developers whenever needed. The management can also ask them to do the pair testing along with developers. Pair testing is a concept which means that testers normally sit with developers and test the system. In this way tester and developer can learn a lot and can find the errors and bugs in a better and efficient way. We feel that testing is a duty of the testers and it should be done by them not by the developers, but one thing should be in mind that the process of testing should not be recursive; where testers test the system, find error and send it back for fixing and developers fix it and send it again to tester and the process goes on and on. This sort of testing can make the system go out of time and also out of budget. Yes, regression testing is good but access of everything is bad. The tester can also assist the developer in developing the unit tests. He/she can also help customer in creating acceptance tests. They can also write automated test cases for the test suite. The tester can be placed with the developer at one monitor to do a code review or to do some other duties.

The testers are integrated into the project teams and they are always up to date. They can easily communicate with the developers, designers, technical writers and also with the management. In such a team the primary role of the tester should be to test the application and to find errors. He/she can ask question regarding the developed code and discuss it with developer. If the tester feels that there should be some changes then he/she can forward it to the test manager. He can bring people closer to each other by being a quality analyzer. He should give and impact that he is the only one who is conscious about the testing and wants the system to be bug free. Tester should decide the testing time and effort. The capacity or the development of test cases is up to infinite but the tester should be really careful with selection of test cases. He should prioritize the test cases and their selection is really tricky. That's why we are stressing that the tester should be in each and every project meeting so that he/she can know that, what are the priorities of the project team? There should be proper planning for testing. The testing should be assigned some time during the sprint and after the sprint. The tester writes the test cases from story cards or design, tests the application, and provides his/her feedback to the developer.

While testing the system, the tester should keep the project priorities in the mind and should keep track of all the problems. The tester can review the unit tests, and check their results. The tester can do the testing for scrum one time in a week. They can also investigate the product backlog and at the end of the system they can send bugs to fix for the next sprint

James Bach says that, the tester should provide something more to the development team then the automation tools and on site customer. And if he/she is not doing so able to provide more information o the uncover errors in unaddressed areas, and then the tester needs to change his/her style of doing things; because, in Agile development the team will expect more from tester's side.

Now working in such environment will also change the attitudes of the testers. If a company is switching from conventional way of development to the Agile development then there will be a lot to

do by a conventional tester. He/she needs to learn a new vocabulary with some typical practices, tools and roles. The practices of testing will be started from the day one. The testing process can long from two to four weeks in different iterations. In Agile there will be no requirements specification documents, from where he/she can write the test cases. The tester needs to be sharpening enough to understand the design and develop tester cases. The requirements for the system can be finding on the story cards. They will need to have knowledge of tools such as test driven development, automation tools, continuous integration tools and refactoring. In Agile the performance of an individual is really important, and in a perfect Agile environment developers are so obsessed with the conventional testing. So, testers need to discuss it with developers. There will be same working area and they need to work closely with the whole project team. Automation testing can occur in a huge amount in Agile and testers need to be tool oriented.

The communication of developer with customer is also important. The primary job of developer is to do the coding for the application. He also develops the unit tests for testing. Developer also does the automation testing and can get assistance of tester while developing the automation tests. After the development, he/she sends it to the tester. The developer gets the requirements from the story cards, customer or from designers. He/she estimates the total time for development of a task and for the iteration. The customer regularly provides feedback for the application. If the customer is on site then he/she will also give suggestion to make the system more user friendly, design and standards. In this way the communication will be continuous. Customer does the acceptance testing and provides his/her feedback. Customer also does meetings with developer to decide the items for the next iterations. If the customer is not on site and developer faces any hurdle then developer can contact to the customer through email or phone.

In this model a lot of things depend on management. They should overcome the hurdles between the project team. There should be regular meetings to know that what everyone is working. There should be review meetings within and after the sprint. In the meetings the manager should ask everyone about his/her task and discuss if someone is facing any problem during implementation. In this way the junior team members can learn a lot from the experiences of the senior team members. These meeting also increase a sense of responsibility in team members. They must know the capabilities of individuals and, should regularly switch their team members in order to make any process faster or efficient. We are agreed with [52], that the managers should coach the testers. They should review the test approach of every team. There should be clear vision for testers and also for the management as for as testing for the system is concerned. There should be proper development and maintenance of a test policy, test strategy and a test project plan. By having different meeting of project teams along with testers can help to increase the knowledge of testers [52].

CHAPTER 7: CONCLUSION

After this study, we think that Agile development processes should be accompanied with a separate but integrated testing team. The roles and responsibilities of software testers and developers should be assigned. In the industry every company has its own independent software testing team. Sometimes if it's company's own product then they also call external testers to test the application. Those external testers can be some expert users, or a sample test group. The testers should work very closely with software developers and other project team. Testers should be part of every project meeting, so that they can know that what the intentions of the project team are. The idea of not having a testing team can be useful in a small organization with a small project size, but in bigger organizations, it is hard to achieve quality product without a separate testing team. We concluded that there should be a combination of both manual and automated testing. Automated testing is effective but still there is need for manual testing. Automated testing can be done by the developer during the sprint and before tester can test the application before the final release. Pair testing is a useful activity and need to be practiced in Agile projects. There should be proper time slot for test planning in Agile processes. Management plays an important role in Agile methodologies. We feel that as for as the project team is concerned, the team management can conduct some training sessions to educate them and train them. This can help them while working within integrated teams. There is also need to educate the customer to tell him/her that why you are using Agile development, because sometimes the customer hesitates to work with an Agile team, because he/she is busy and can not be continuously on site or in contact.

CHAPTER 8: FUTURE WORK

We feel that in agile development, there are still some areas that need some work to be done. According to our survey results we analyzed that companies are still facing some problems in implementing Agile development in industry. We feel that there is need to do more research on some of the areas in Agile development. This can help the industry practitioners for the better management of their processes. While practicing Agile development there are still unclear issues like: proper management of project backlog item, teams facing problems while switching from conventional to Agile development, the communication and interaction among the teams, proper assignment of roles and responsibilities, time estimation of the bigger projects and managing the testing practices in Agile development. We recommend the future researcher to explore some of the above areas to resolve these problems.

Our proposed model is not a mature one, because we focused on the tester and testing process in Agile development. We just tried to go an overview that how a testing team can work in a better way in Agile development. It is in an implementable form as far as a smaller or a medium company is concerned, but in case of a bigger company we feel that there are still some others actors left that need to be included in this model. We suggest that this model should be worked on and implement in the industry.

REFERENCES

[1] Hetzel, William C., "The Complete Guide to Software Testing", 2nd edition. Publication info: Wellesley, Mass.: QED Information Sciences, 1988. ISBN: 0894352423.

[2] Harrold, M. J. 2000. "Testing: a roadmap", In *Proceedings of the Conference on the Future of Software Engineering*, ICSE '00. ACM, June 04 - 11, 2000, ISBN: 1-58113-253-0

[3] Bertolino, Antonia, "Software Testing Research: Achievements, Challenges, Dreams," *Future of Software Engineering, 2007. FOSE '07* , pp.85-103, May 2007, IEEE.

[4] Conchango, "The Agile revolution: Reduce project failure rates through adoption of Agile", 2005, URL: www.conchango.com (last checked: May 02, 2008).

[5] Scott W. Ambler, "Agile Modeling: Effective Practices for Extreme Programming and the Unified Process", Publisher: John Wiley & Sons, April 2002, ISBN#: 0471202827.

[6] Peter Schuh, "Integrating Agile Development in the Real World", Publisher: Charles River Media, December 2004, Pages: 364, ISBN-10: 1584503645.

[7] Koch, Alan. "Agile Software Development: Evaluating the Methods for Your Organization", Publisher: Artech House, October 2004, ISBN-10: 1580538428.

[8] Xie, T. "Improving Effectiveness of Automated Software Testing in the Absence of Specifications", In *Proceedings of the 22nd IEEE international Conference on Software Maintenance* (September 24 - 27, 2006). ICSM. IEEE Computer Society, Washington, DC, 355-359.

[9] Sridhar, V. "Implementing Automated Testing", Hershey, PA, USA: Idea Group Inc., 2003.

[10] Ian Somerville, "Software Engineering: 7th edition", Pearson, Addison Wesley, 2004, ISBN: 0321210263.

[11] Coldewey, J. , "Agile software development - introduction and overview", Wirtschaftsinformatik, v 44, n 3, June 2002, Pages: 237-48, Publisher: Friedr. Vieweg & Sohn Verlagsgesellschaft.

[12] Cohn, M. and Ford, D. "Introducing an Agile Process to an Organization", Volume 36 , Issue 6, Pages: 74-78, June 2003, ISSN:0018-9162.

[13] Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J., "Agile Software Development Methods: Review and Analysis", VTT Publications, Pages 478, 2002.

[14] Abrahamsson, P., "Agile software development: Introduction, currant status and future", VTT Publications, November 2005.

[15] Jim highsmith, "Agile Project Management: Creating Innovative Products", Publisher: Addison-Wesley Professional, April, 2004, ISBN-10: 0321219775.

[16] Tichy, W. F., "Agile Development: Evaluation and Experience", In *Proceedings of the 26th international Conference on Software Engineering*, May 2004, IEEE Computer Society, ISSN: 0270-5257.

[17] Juha Itkonen, Kristian Rautiainen, and Casper Lassenius, "Towards Understanding Quality Assurance in Agile Software Development", International Conference on Agility Management (ICAM 2005), Helsinki, July 2005.

[18] Fowler, M., J. Highsmith, “The Agile Manifesto” Software Development Magazine. August 2001, URL: http://www.sdmagazine.com/documents/s=844/sdm0108a/0108a.htm (last checked: April 01, 2008).

[19] Myers, Glenford J. “Art of Software Testing”, Hoboken, NJ, USA: John Wiley & Sons, Incorporated, pages: 234, June 2004, ISBN-10: 0471469122.

[20] Royce, W. W., “Managing the development of large software systems: concepts and techniques”, In Proceedings of the 9th international Conference on Software Engineering (Monterey, California, United States). International Conference on Software Engineering, IEEE Computer Society Press, Year of Publication: 1987, ISBN: 0-89791-216-0.

[21] Abrahamsson, P., Warsta, J., Siponen, M. T., and Ronkainen, J., “New directions on Agile methods: a comparative analysis”, In Proceedings of the 25th international Conference on Software Engineering, International Conference on Software Engineering. IEEE Computer Society, Pages: 244 – 254, Year of Publication: 2003, ISBN ~ ISSN: 0270-5257.
.
[22] Kaner, C., Bach, J., and Pettichord, B., “Lessons Learned in Software Testing”, Pages: 320, Year of Publication: 2001, John Wiley & Sons, ISBN: 0471081124

[23] Kaner, C., Falk, J. L., and Nguyen, H. Q., “Testing Computer Software”, Second Edition, Pages: 480, John Wiley & Sons, Year of Publication: 1999, ISBN: 0471358460.
.

[24] Lippert, M., Becker-Pechau, P., Breitling, H., Koch, J., Kornstädt, A., Roock, S., Schmolitzky, A., Wolf, H., and Züllighoven, H., “Developing Complex Projects Using Xp with Extensions”, Volume 36, Issue 6, June 2003, Pages: 67 – 73, Year of Publication: 2003, ISSN: 0018-9162.

[25] Theunissen, W. H., Kourie, D. G., and Watson, B. W., “Standards and Agile software development”, In Proceedings of the 2003 Annual Research Conference of the South African institute of Computer Scientists and information Technologists on Enablement Through Technology, September 2003, ACM International Conference Proceeding Series, vol. 47, Pages: 178 – 188, ISBN:1-58113-774-5.

[26] Palmer, S. R. and Felsing, M., “A Practical Guide to Feature-Driven Development”. 1st Ed. Publisher: Pearson Education, Pages: 299, Year of Publication: 2001, ISBN: 0130676152.

[27] Beck, K., “Extreme Programming Explained”, Publisher: Addison-Wesley Longman Publishing Co., Pages: 190, Year of Publication: 1999, ISBN: 0-201-61641-6.

[28] Crispin, L. and Rosenthal, K. S., “*Testing Extreme Programming”*, Publisher: Pearson Education, Pages: 306, Year of Publication: 2002, ISBN: 0321113551.

[29] Pyhäjärvi, M., K. Rautiainen, "Integrating Testing and Implementation into Development," Engineering Management Journal, Vol: 16 (1): 33-39, 2004.

[30] Stapleton, J., "DSDM: Dynamic Systems Development Method," *Technology of Object-Oriented Languages and Systems, 1999*, pp.406-406, Jul 1999, ISBN: 0-7695-0275-x.

[31] Bret Pettichord, “Seven Steps to Test Automation Success”, 2001, URL: http://www.io.com/~wazmo/papers/ (last checked: April 17, 2008).

[32] 'Burke Johnson', 'Larry Christian', "Educational Research Quantitative, Qualitative, and Mixed Approaches", URL: http://www.southalabama.edu/coe/bset/johnson/dr_johnson/2textbook.htm, (last checked: April 03, 2008).

[33] Myers, M., "Qualitative research and the generalizability question: Standing firm with Proteus", Volume 4, Numbers 3/4, March, 2000, URL: http://www.nova.edu/ssss/QR/QR4-3/myers.html, (last checked: March 16, 2008).

[34] Huo, M., Verner, J., Zhu, L., and Babar, M. A., "Software Quality and Agile Methods", In *Proceedings of the 28th Annual international Computer Software and Applications Conference, Volume 01*, IEEE Computer Society, Pages: 520 – 525, Year of Publication: 2004, ISBN: 0730-3157.

[35] Serena, "An introduction to Agile software development", Serena Software Inc., June 2007, URL: www.serena.com/docs/repository/solutions/intro-to-Agile-devel.pdf, (last checked: May 01, 2008)

[36] Highsmith, J., "Agile Project Management: Creating Innovative Products", Publisher: Addison Wesley Longman Publishing Co., Year of Publication: 2004, ISBN: 0321219775.

[37] Schwaber, K. and Beedle, M., "Agile Software Development with Scrum", 1st Edition, Publisher: Prentice Hall PTR, Pages: 158, Year of Publication: 2001, ISBN: 0130676349.

[38] Highsmith, J., "Agile Project Management: Creating Innovative Products", Publisher: Addison Wesley Longman Publishing Co., Year of Publication: 2004, ISBN: 0321219775.

[39] Beck, K., "Embracing Change with Extreme Programming", Volume 32, Issue 10, October 1999, Pages: 70 – 77, ISSN: 0018-9162.

[40] Puleio, M., "How Not to Do Agile Testing", In Proceedings of the Conference on AGILE 2006, IEEE Computer Society, Pages: 305 – 314, Year of Publication: 2006, ISBN: 0-7695-2562-8.

[41] Jared M. Spool and Joshua Porter, "Agile Development Processes: An Interview with Jeff Patton", Sep 2006, URL: http://www.uie.com/articles/patton _interview/ (last visited: May 08, 2008).

[42] Ken Arneson, "Automates testing in Agile environment", Checkpoint Technologies, Inc. URL: www.tampabayqa.com/ Images/Agile.ppt, (last visited: May 08, 2008).

[43] Palmer, S. R. and Felsing, M., "A Practical Guide to Feature-Driven Development". 1st Ed. Publisher: Pearson Education, Pages: 299, Year of Publication: 2001, ISBN: 0130676152.

[44] Stephen Palmer, "The Coad Letter: Modeling and Design Edition, Issue 70, Feature Driven Development and Extreme Programming" URL: http://dn.codegear.com/article/29684 (Last checked: May 11, 2008)

[45] Scott W. Ambler "Agile Modeling: Effective Practices for EXtreme Programming and the Unified Process", Publisher: John Wiley & Sons, April 2002, ISBN-13: 978-0471202820

[46] Alistair Cockburn "Crystal Clear: A Human-Powered Methodology for Small Teams", Publisher: Addison-Wesley Professional; 1st edition, October 2004, ISBN-13: 978-0201699470.

[47]. Cockburn A. "Surviving Object-Oriented Projects: A Manager's Guide", Publisher: Addison Wesley Longman, 1998, pages: 272, ISBN-10: 0201498340.

[48] Cockburn A. “Agile Software Development”, Publisher: Addison Wesley Longman, 2002, pages 304, ISBN-10: 0201699699.

[49] Palmer, S. R. and Felsing, M., “A Practical Guide to Feature-Driven Development”. 1st Ed. Publisher: Pearson Education, Pages: 299, Year of Publication: 2001, ISBN: 0130676152.

[50] Brian Marick, “When Should a Test Be Automated?” Testing Foundations, URL: www.testing.com (last checked: May 11th, 2008)

[51] Myers, Glenford J. “Art of Software Testing”, Hoboken, NJ, USA: John Wiley & Sons, Incorporated, pages: 234, June 2004, ISBN-10: 0471469122.

[52] Ralph Van Roosmalen, “Testing and Scrum”, Planon, Scrum Gathering, October 2007, URL: http://www.scrumalliance.org/resources/270 (last checked: April 25, 2008).

[53] Stephens, M., D. Rosenberg, “Extreme Programming Refactored: The Case Against Xp”, Pages: 432, Publisher: Apress, August 2003, ISBN-10: 1590590961.

[54] Theunissen, W.H.M., D.G. Kourie, B.W. Watson, "Standards and Agile Software Development", Proceedings of SAICSIT, pp. 178-188, 2003.

[55] Bret Pettichord, “Testers and Developers think differently”, STQE Magazine, January 2000 URL: http://www.io.com/~wazmo/qa/ (last checked: May 03, 2008).

[56] Brian Marick “A Roadmap for Testing on an Agile Project”, June 2004, URL: http://www.testing. Com /writings/2004-roadmap.html, (last visited: May 13, 2008).

[57] Rick Hower, “Software QA and Testing”, URL: http://www.softwareqatest.com (last checked: May 13, 2008).

[58] Jonathan kohl, “Software Investigation: Collaborative software testing”, Feb 2005, URL: http://www.kohl.ca/blog/ archives/000081.html, (last checked: 08/05/02).

[59] Mikael Lindvall1, Patricia Costa1, Kathleen Dangle1, Forrest Shull1, Roseanne Tesoriero1, “Empirical Findings in Agile Methods”, Fraunhofer Center for Experimental Software Engineering, Maryland, Volume: 2418/2002, Publisher: Springer Berlin / Heidelberg, ISBN: 978-3-540-44024-6.

[60] Object Mentor, “Why Use Agile”, URL: http://www.objectmentor.com/omSolutions/Agile_ why.html (last checked: May 15, 2008).

APPENDIX

Following is a set of questionnaire that we used to conduct an industry survey. We sent this questionnaire to companies that are practicing in Agile development, to get their feedback. We have tried to cover every aspect of our research area in these questions to know all industry practices with respect to Agile development and practitioner's views.

1. What are the practices of Agile development in your organization?

2. How are you following Agile Development? (Are you following exact ways or practices of Agile development or implementing it with some desired changes)?

3. How do you practice quality assurance activities, with respect to Agile development?

4. How are you conducting manual testing and automated testing?

5. How do you practice software testing activities in your development process, with respect to Agile development?

6. What are the critical issues you are facing during Agile development?

7. In your organization, what are the roles of developers and testers as for as the software testing is concerned with respect to Agile development?

8. How are you managing the interactions and communications with customer?

9. What are your views about actual Agile working, where most of the testing activities are done by software developers? Do you think these practices are enough to get a quality product in a bigger company?

10. Do you think that, if the customer and developer have a close coordination, then there is no need for a separate testing team in Agile? Your views?