Agile: Analysis of its Problems and their Solutions

Amandeep Singh
M.Tech (Computer Science)
Department of Computer Science, Punjabi University, Patiala

ABSTRACT
Although development teams and companies are adopting agile methodology very fast but there are some problems in and outside Agile Software Development Methodology which needs to be addressed. This manuscript analyzed various advantages and problems which are experienced during Agile development and Testing process. The possible solutions are also analyzed in this paper which can eliminate much of the problems, thereby increasing the efficiency and effectiveness of Agile Software Development Methodology, thus making it one step close toward the methodology of future. The comparison of agile with waterfall methodology is also discussed.

General Terms
Agile Software Development Methodology, Agile Manifesto.

Keywords
Agile Software Development Methodology, Communication Problems, Automation, Embedded Systems, Efficiency.

1. INTRODUCTION
The umbrella term 'Agile' [1] refers to the family of software development methods and techniques that are generally iterative and evolutionary in nature. This development is carried out in highly collaborative manner by self organizing teams so as to produce high quality software which provides value to their clients.
The techniques and methods used in Agile Software Development exits before the term 'Agile' was coined in context of software development. Although there are several advantages of agile software development but there are some challenges faced during its adoption. These problems are related to different areas of software development and testing.
In this paper we will discuss some of these problems and challenges and try to propose solution to them thus talking about the possibility of agile to be the methodology of future.

1.1 Agile Manifesto
The history starts from 1974, when E. A. Edmonds introduced adaptive software development process through his paper. Because of the problems and limitations like heavily regulated, regimented, micromanaged and heavy documentation involved in heavyweight methods lightweight software development methods (and now agile methods) evolved in the mid-1990s.
Lightweight methods that were implemented early include Scrum (1995), Crystal development techniques, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995). These are now typically referred to as agile methodologies, after the Agile Manifesto published in 2001.
In February 2001, 17 signatories or developers gathered at snowbird, Utah resort to discuss about the various lightweight Development processes. Through this meeting they agreed upon the common ground to form the 'Agile Manifesto', although each of them had their own different opinion on using the various agile methods and techniques.
Agile Manifesto [2] reads as follows:
"We are uncovering the better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

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

Principles behind the Agile Manifesto

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.

- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

2. WATERFALL MODEL VS AGILE MODEL

The differences [3] on basis of three points are given as:

2.1 Basic Difference

Waterfall model of software development is a sequential model of software development. As in the waterfall, the water falls from one altitude to the lower and then that lower to another lower altitude, in a similar way, the development cycle progresses sequentially, from one stage to the other stage one by one.
The different phases of waterfall model software development are as follows: requirement specification, analysis, design, coding, testing & debugging, installation and finally, maintenance. In this sequentially approach, the development team goes ahead to the next phase of development, only after they have completed the work in previous stage. Software development companies, adopting this model, spend a considerable amount of time in each stage of development, till all doubts are cleared and all requirements are met. The companies which adopt this model believe that considerable time spent in initial design effort corrects bugs in advance. Once the design stage is over, it's implemented exactly in the coding stage, with no changes done later on. The analysis, design and coding teams are separated and work on small parts in the whole development process. More stress is given on documentation of every software development phases.

As compared to the 'set-in-stone' approach of waterfall development models, the agile breed of models, give more emphasis on 'agility' and 'adaptability' in software development. Instead rigid development schedule, agile models involve multiple iterative development schedules that try to improve the quality of output with every iteration. Each iteration goes through the all the steps of design, coding and testing. The design is not set in stone and is kept open to last minute changes due to iterative implementation. There is cross functional team structure and self organizing. The design idea is never totally frozen, but it's allowed to evolve as new ideas come in with each release. Less importance is given to documentation and more stress is given to producing working software.

2.2 Efficiency

In this the two approaches are compared with respect to efficiency of development. Efficiency is decided by the quality of ultimate software product, the number of bugs and the time consumed during he development process. Generally it is said that agile models are more efficient than the waterfall model, due to their adaptability to the real world. The 'One Phase' and 'Rigid' development cycle of a waterfall model, makes it difficult to make last minute changes in requirements or design. While the agile methods, due to their iterative and adaptable nature, they can incorporate changes and release a product, in lesser amount of time as compared to waterfall development.

2.3 Fitness

Generally the waterfall model is fit for development of programs that don't require many changes in their design. In the projects where the designers of a software program can accurately predict the flaws that may arise, in advance, waterfall model is the first and appropriate choice. Despite all its flaws, a waterfall model design is easier to manage and handle.
Agile models are applicable in every area of software development. It depends a lot more on the team effort, communication with the customer, coordination between the team members. It is best fit for web based applications where its iterative nature helps in incorporating and correcting the various bugs that arise over time.
Basically, agile software development and waterfall model both have some advantages and disadvantages. So our mission should be to identify the disadvantages of agile and try to remove them either by using good practices of waterfall or using some new techniques.

3. PROBLEMS WITH AGILE AND THEIR PROPOSED SOLUTIONS

Although agile software development methodology is getting more and more famous still there are certain questions which need to be answered in order to achieve full performance of this methodology and make it compatible and efficient in all types of environments.

Some of these problems are:

1. Communication (Technical, between team)
2. Estimation
3. Automation and General testing
4. Communication (Customer)
5. Distributed Teams
6. Ambiguous requirements
7. Testing for Embedded Systems
8. Performance testing on an Agile project

3.1 Communication (between Team)

This problem is generally faced by the team members especially when the team is new to agile development or transitioning from some old software development to agile development. According to Michael puleio [4] in agile

methodology teams are self organizing and there are no specific roles of team members but when team is new to this methodology, developers don't know about testing and project management and testers also are dumb on development and project management. As a result of this clashes between team members over each others disciplines are often seen.

The solution to this problem of technical communication between team members is that before the project start everyone on the team must read books about the other disciplines. This reading of books and discussions will create a common understanding of basic principles. Thus common knowledge integrated with cross discipline work gives the common language as output. It results in more positive approach, no clashes between team members thus enjoyable work.

3.2 Estimation

This is also a big problem especially when tem is new to agile development. This occurs due to lack of experience of breaking down the big tasks into smaller ones. For example when a tester is new to agile testing, he will break the tasks in a manner in which he was breaking them while working under previous old development methodology. Same is the case with development. The developer (which is new to agile) will demand resources which are not the exact requirement.

The problem of estimation is tough, but it can be solved by good amount by using techniques like Wideband Delphi estimation help, listening to others and learning from your mistakes [4].

3.3 Automation and General Testing

Automation is one of the useful task in agile testing. Manual testing makes the agile process slow therefore it must not be done in bulk. Instead, for extreme programming and scrum like processes developers should take the initiative to build a simple tool for automation testing if the existing tools are not used. If many of the tools are present one must choose a tool that suits the quality standards for production code.

To code and test the feature in as single iteration only, we must write the functional tests first, and then develop the feature. This requires settlement between the team members of different disciplines [4].

3.4 Communication (Customer)

Customer communication is the backbone of agile software development. Agile manifesto's value also describes this point. Agile methodology says that customer communication and collaboration is of utmost importance so agile environment is highly customer commucative. In small scale projects customers generally remains onsite and thus must be consulted whenever the team feels necessary. But in large scale projects this may happen that customer remain onsite only for required period of time or not at the site at all. This creates the problem for agile development methodology.

A solution is ti incorporate a 'Customer Liaison' or you can say a middleman that connects customers to team. He has the domain knowledge of the project. The role of this middle man is to answer the questions which the development team asks to maintain the speed development. If he does not know the answer then he must know from whom or where to find that. Customer liaison serves as the strongest communication channel between the customer and the development team [5].

According to my approach there are number of technologies today which can be used to communicate with the off-site customer. The one technology called video conferencing can be used to make direct contact with the customer. He may be shown the working product, can be asked about his clarification easily through the use of technology. E-mails can also act as solution to this problem.

3.5 Distributed Teams

Generally agile development works well when the team is at one place that is all team members carry out development practices at one site only. But when team becomes distributed geographically then it starts creating problems. These problems are of communication between team members, coordination between them and others.

Distributed teams that are using TDD (Test Driven Development, an agile practice) can still gain the advantages of TDD by increased use of formal and informal communication, helped by change management and notification tools. Disciplined developers can also develop system level test cases by using test frameworks like XUnit family [8].

3.6 Ambiguous Requirements

When customer and developer fail to communicate and collaborate in initial stages then ambiguous requirements arises. This is potential problem for agile methods as requirements form the story cards.

Two solutions are there in this regard. One is to use two levels of requirements, first is high level user stories which includes features yet to be fully analyzed. Second levels of requirements are developed collaboratively with customer by clarifying and identifying the detailed work for next iterations. Second solution is to use the Bridge Person. This person will help both customer and developer by maintaining a lightweight documentation which contains key verbal points and whiteboard sketches that are collected in team meetings [5,6,7].

3.7 Testing for Embedded Systems

Agile testing generally unit testing can easily be applied to softwares but when embedded systems come into play then it becomes a difficult task. Embedded systems are those in hardware and software are integrated together. So test of software is bound with the hardware.

But the solution to it is given by Nancy Van Schooenderwoert, Ron Morsicato [9], they said that agile

methods work well on embedded systems also provided we use multiple test strategies. These strategies are like using the trouble log, hardware driver unit tests, domain level tests and others. They say that this resulted in a very low bug rate.

3.8 Performance testing on an Agile project

Jamie Dobson [10] adopted performance testing on a agile project in three levels namely Beta or Customer testing, Deployed testing, Developer testing. According to him, performance testing in an agile project theoretically easy but practically difficult to implement. Performance testing is difficult with constantly changing requirements or with moving aim. He also said that genuine performance testers can act as bottleneck for the process of agile methodology. Also real agile performance testing is still not possible according to him due to the expensive cost of the resources and setup needed for that. So the performance testing needs dedication, energy and design of good processes that suit the context.

4. CONCLUSION

Agile software development methodology contains the set of techniques which are very best for the software development. When the conditions are ideal or you can say all the required conditions imposed by agile methodology are met then it assures you to give the quality product.

Some of the facts [11] regarding Agile are:

- The major barrier for the Agile Development was the lack of ability to change the existing organizational culture and general resistance to change.
- 29% believe the barrier to be customer collaboration.
- 66% of the respondents said that Agile projects were faster to completion than previous non Agile ones.
- 87% of the respondents said that implementing Agile either improved or significantly improved their abilities to manage changing priorities.
- 32% of respondents said that they adopted Agile methods on outsourced projects.
- Lack of experience with Agile methods is the major reason of failure of Agile projects.
- 58% of the methodology employed in Agile projects is SCRUM.
- 63% of the companies have 1-2 sites using Agile.

So there are still some challenges like we explained which need to be addressed in order to increase its efficiency and effectiveness so that it can deliver quality product in any type of conditions and circumstances. Need is to provide solutions that make it so ‘agile’ that it can fit into any type of conditions easily and efficiently.

Of course, agile software development methodology is the methodology of future provided certain key issues must be addressed to move into the future.

5. ACKNOWLEDGMENTS

The author would like to thank the anonymous reviewers for their helpful suggestions for preparing the manuscript.

6. REFERENCES

[1] http://en.m.wikipedia.org/wiki/Agile_software_development

[2] http://agilemanifesto.org/

[3] http://naseemitgs.wikispaces.com/file/view/Waterfall+Model+Vs+Agile.docx

[4] Michael Puleio. 2006. How Not to do Agile Testing.

[5] Woi Hin, Kee. 2006. Future Implementation and Integration of Agile Methods in Software Development and Testing.

[6] Paul E.McMohan. 2005. Extending Agile Methods: A Distributed Project and Organizational Improvement Perspective.

[7] Paul E.McMohan. 2004. Bridging Agile and Traditional Development Methods: A Project Management Perspective

[8] Raghvinder S. Sangwan, Phillip A. Laplante. 2006. Test-Driven Development in Large Projects.

[9] Nancy Van Schooenderwoert, Ron Morsicato. 2004. Taming the Embedded Tiger- Agile Test Technique for Embedded Software.

[10] Jamie Dobson. 2007. Performance Testing on an Agile Project.

[11] http://www.versionone.com/pdf/2010_State_of_Agile_Development_Survey_Results.pdf