ISSN 2249-6343
International Journal of Computer Technology and Electronics Engineering (IJCTEE)
Volume 2, Issue 4, August 2012

Impact of Agile Methodology on Software Development Process

Gaurav Kumar, Pradeep Kumar Bhatia

Abstract **– Agile methodology that utilizes iterative development and prototyping are widely used in variety of industry projects as a light weight development method which can satisfy to the changes of requirements. Short iterations are used that are required for efficient product delivery. Traditional software development processes are not much efficient to manage the rapid change in requirements. Despite the advantages of Agile, criticism on agile methodology states that it fails to pay attention to architectural and design issues and therefore is bound to produce small design-decisions. Here, in this paper we identify the impacts that agile methodology has on software development processes with respect to quality within the organizational, methodical, and cultural framework.**

Index Terms– **Agile Alliance, Agile Methodology, Crystal Method, Extreme Programming, Feature Driven Development, Scrum, Test Driven Development.**

I. Introduction

Agile Methodologies are a group of software development methods that are based on iterative and incremental development. The four major characteristics that are fundamental to all agile methodologies are: adaptive planning, iterative & evolutionary development, rapid and flexible response to change and promote communication [1, 2]. Its main emphasis is in obeying the principles of "Light but sufficient" and being people-oriented and communication-centered. As it is named as lightweight process, it is more suitable for the development of small projects [3]. Agile software development takes the view that production teams should start with simple and predictable approximations to the final requirement and then continue to increment the detail of these requirements throughout the life of the development. This incremental requirements refinement further refines the design, coding and testing at all stages of production activity. In this way, the requirements work product is as accurate and useful as the final software itself [4].

Manuscript received August 7, 2012.

Gaurav Kumar*, Department of Information Technology, Panipat Institute of Engineering & Technology, Samalkha, Panipat, Haryana, India (e-mail: er.gkgupta@gmail.com)*

Pradeep Kumar Bhatia*, Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, Haryana, India (e-mail: pkbhatia.gju@gmail.com)*

The principle of agile software development proposes [5] that "at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly". In other terms it may be said that agile methodology addresses exactly the challenges of an unpredictable, disordered business and technology environment [7]. Agile methodologies are used to achieve higher quality software in a shorter period of time, self organizing teams, customer collaboration, less documentation and reduced time to market [8, 9]. Agile methodology includes a family of lightweight methods that include Scrum, Crystal Clear, Extreme Programming (XP), Adaptive Software Development (ASD), Feature Driven Development (FDD), and Dynamic Systems Development Method (DSDM) Crystal, Lean Software Development etc. [10]. Agile methods break tasks into small increments with minimal planning called Iterations. Iterations are short time frames that runs from one to four weeks. Each iteration involves a team working through a full software development cycle, including planning, requirements analysis, design, coding, unit testing, and acceptance testing. This minimizes overall risk and allows the project to adapt to changes quickly. Most of the agile implementations use a formal daily face-to-face communication among team members. In this brief communication, team members report to each other what they did the previous day, what they intend to do today, and what are the hurdles they faced When customer or domain expert works directly with the development team everyone learns something new about the problem [15, 16,17].

II. Background

The Agile Manifesto defines the approach of agile software development. In 2001, some of the manifesto's members set up the Agile Software Development Alliance or Agile Alliance as a nonprofit organization that refined the philosophies captured in their manifesto into a collection of twelve principles. The twelve principles behind agile manifesto are [16, 22].

- The 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.

ISSN 2249-6343
International Journal of Computer Technology and Electronics Engineering (IJCTEE)
Volume 2, Issue 4, August 2012

- 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 behavior accordingly.

III. Popular Agile Methodologies

Considering the business view, Agility is the ability of an enterprise to act in advance for the changes happening in the environment in order to maximize the benefits. Customer feedbacks are welcomed by an agile enterprise and try to incorporate those feedbacks in the product [27]. Some of the popular agile methodologies are briefly described as under:

A. Extreme Programming

The goal of Extreme programming (XP) is to improve software quality and responsiveness to changing customer requirements. Extreme Programming improves a software project in five essential ways: First is *Planning* in which the project is divided into iterations. At each iteration, Iteration planning is started, User Stories are written, Release planning is done to create release schedule to make frequent small releases. Second is *Managing in which* team is given a dedicated open work space. A stand up meeting starts every day in which project velocity is measured and if there is some inflation, fixes the problem. Third is *Coding* in which As per the standards, Code is written, test driven development code is used. All production code is pair programmed; only one pair integrates code at a time, continuous integration is made. Fourth is *Designing*. Here choose a system metaphor, no functionality is added early, Refactor the design whenever and wherever possible. Fifth is *Testing* in which all code should be unit tested and should pass before it can be released, When a bug is found tests are created, acceptance tests are run often and the score is published.

B. Scrum

Scrum is an agile process most commonly used for product development, especially software development. Scrum is a general-purpose project management framework that is applicable to any project with aggressive deadlines with complex requirements and a degree of uniqueness.

In Scrum, projects progress via a series of iterations called sprints. Each sprint is typically 2-4 weeks long. A typical scrum team has between five and nine people, but Scrum projects can easily scale into the hundreds. The team does not include any of the traditional software engineering roles such as programmer, designer, tester, or architect. The *product owner* is the project's main stakeholder and represents users, customers and others in the process. The *ScrumMaster* is responsible for making sure the team is as productive as possible.

C. Feature Driven Development(FDD)

FDD is a client-centric, architecture-centric, and pragmatic software process [8, 28]. There are five main activities in FDD that are performed iteratively. *Firstly* an overall model is developed in which the initial result will be a high-level object model and notes. At the start of a project the goal is to identify and understand the fundamentals of the domain that the system is addressing, and throughout the project this model will be refined out to reflect what is to be built. Secondly, a features list is developed; grouping of them is done into related sets. Third is *plan by feature*; as the end result are a development, the identification of class owners and the identification of feature set owners is done. Fourth is Design by Feature includes detailed modeling. Fifth is *build by feature* that includes programming, testing, and packaging of the system.

D. Crystal Method

Crystal Methods is an agile software development methodology developed by Alistair Cockburn in which people are more emphasized in software development rather than tools or processes [8, 9]. Crystal methods are a toolkit of methodology elements to suit individual projects. Large or safety critical projects require more methodology elements than small non-critical projects. With Crystal Methods, organizations only develop and use as much methodology as their business needs demand. Basically this is used for small teams and small projects that are not life critical.

IV. Literature Review

Various studies and surveys have been made that shows agile methodology's popularity based on characteristic of requirements, small or big organization, and experience of the project team. Agile methods have proven their effectiveness and are transforming the software industry. Some of the review findings are presented here.

As demonstrated by Andrew et al [1] using a survey based approach, agile methodology is favorable due to improved communication between team members, quick releases and flexibility of designs. Scrum methodology is the most popular; and test driven development and pair programming are the least used practices.

Anfan Zuo et al [3] apply formal methods into agile software development. They applied rCOS an object oriented approach to agile methodology to improve accuracy of the system analysis and facilitating system development with object-oriented ideas.

ISSN 2249-6343
International Journal of Computer Technology and Electronics Engineering (IJCTEE)
Volume 2, Issue 4, August 2012

Jeffrey et al [8] shows that XP is used heavily in the organization. After that in order of decreasing usage, Scrum is used, then feature driven development, dynamic development methodology, adaptive software development, and then other agile methodologies which are modified as per company are used.

As further demonstrated by A. Ahmed et al [9], scrum is used most commonly, 50% of the projects are done with active stakeholder participation. 66.7% participants were agreeing that productivity has improved and quality is improved by 50%.

Based on a study, Pirjo et al [12] shows that agile methods are good for some programming environments, but not for all. Projects that involve large teams, well-defined requirements, clients needing high assurance and large code-bases, the traditional plan-oriented project profile works well. Therefore, Agile methods produces best results in case of when the team is small, the requirements are not yet well defined, the project code base is small and the customer is interested in seeing significant progress. However, as a software project transitions from a small prototype to a large stable system with a large team, with promises to keep and dates to meet, then agile methods alone is not sufficient then some additional mechanism is needed.

O. Solo et al [13] show that adoption of agile processes increases with company size. Also the adoption of agile methods in large and distributed environments is being addressed more frequently. 54% of the participants used XP and the 5 factors in XP used are open office space, coding standards, 40h week, continuous integration and collective code ownership. 27% used Scrum practices and the remaining used other practices of Agile.

Tore et al [14] report that XP is seemed difficult to introduce in large, complex organizations but easier in other organization types. Pair programming is inefficient and XP works best with experienced development teams. Also there is lack of attention to design and architectural issues.

Behrouz Far [18] mapped the software reliability engineering into an agile development process. As per the study, test driven development seems to be incompatible with the reliability model.

Subhas et al [20] proposed a framework for identifying the important changes required and challenges involved for adopting agile software development practices in traditional development organizations.

Markus et al [23] highlights the negative impact of change in requirements on customer satisfaction. The main contribution of their paper pertains to the interaction effects between change in requirements and agile methods on customer satisfaction. They found that work climate, final product adaptability and willingness to adapt to change have a positive moderating effect on the relationship between change in requirements and customer satisfaction.

Rober Imreh et al [29] concludes that agile software development has a significant impact on quality. They identified some of the major quality impacts of Agile Software Development and various approaches of remediation have been recommended within the organizational, methodical and cultural framework and industry best practices.

The study performed by Sharifah Syed et al [30] shows that agile methodology is more people-oriented than process oriented in a more volatile environment. Excepting the satisfaction of the developers this is helpful only when the requirements are uncertain or volatile.

V. BENEFITS OF AGILE IN SOFTWARE DEVELOPMENT PROCESS

The key benefits of agile methodology in software development processes due to which agile methodology should be adopted while developing software are shown in the figure1 and explained in detail thereafter.

A. Handling Change of Requirements

Planning phase is dramatically improved. First, because customers are directly involved in the development process, that is, customers control the processes of projects through on-site interaction, requirements truly reflect the current needs of the end users.

B. Fault Detection

As testing is performed during each iteration, faults are detected earlier and can be fixed before it increases in severity than with a plan-driven process model. Also, continuous testing allows continuous testing feedback, which further improves code developed in future iterations.

C. Increased Performance

Daily standup meetings provide an opportunity to exchange valuable information and to fine tune improvements continuously. The ability to discuss complex projects through simple stories and simple design encourages teamwork. Better communication leads to increased knowledge sharing, self organizing teams, team morale as employees begin to trust and gain the trust of their team members. This increases team productivity and generates better performance in terms of good Return on Investment than the sum of all individual output.

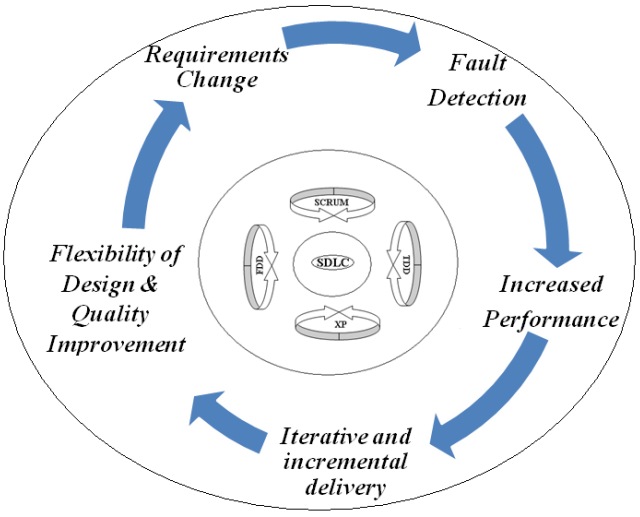

Figure1: Agile Software Development Methodologies with Benefits

ISSN 2249-6343
International Journal of Computer Technology and Electronics Engineering (IJCTEE)
Volume 2, Issue 4, August 2012

D. Iterative and incremental delivery

Project delivery is divided into small functional releases or increments to manage risk and to get early feedback from customers and end users. These small releases are delivered on a schedule using iterations that typically last between one and four weeks each. Plans, requirements, design, code and tests are created initially and updated incrementally as needed to adapt to project changes. Software functionality progress can be checked and monitored much more frequently rather than at end of long milestones.

E. Flexibility of Design

Flexibility defines ability to change directions quickly. As handling change in requirements is the main feature of agile methodology, design has to be made flexible that can handle changes easily. Flexibility is based on the development process used for the project.

F. Improvement in Quality

Test-driven development and refactoring is used. Refactoring leads to higher code reuse and better quality. All aspects of software are improved, from design and architecture to performance of the products of each sprint. Improved communication leads to faster turnaround time for blocking bugs.

VI. Limitation of Agile Methodology

Nothing in this world is without any shortcoming or limitation. Here are some of the limitations and shortcomings of agile methodologies we got based on the literature.

- Main emphasis is on development rather than design and user. It basically focuses on processes for getting requirements and developing code and does not focus on product design.
- High testing lead times and low test coverage.
- Many teams requiring high coordination and communication from project managers.
- Does not scale well to large projects, as numerous iterations are needed to complete the desired functionality.
- Too much time may be devoted to any single, small feature.
- On a large scale project, opportunity cost to employ agile methods may be too high for a foregone production on more profitable and lean projects.
- Management Overhead is increased because a successful application of an agile methodology relies heavily on strong teamwork, the project manager must remain involved in the dynamics of the team.

VII. Conclusion

Agile software development stresses in - evolving requirements accomplished by direct user involvement in the development process, rapid iterations, small and frequent releases. The improvements in software development process include more stable requirements, earlier fault detection, less lead times for testing, increased communication, and increased adaptive capacity. Different methodologies require different changes to the management and software development cultures [26].There are number of factors that can directly and indirectly influence the development projects in agile framework. Adopting agile development methodologies has a positive impact on both the productivity and the quality. Hence, development team and customer both are satisfied with its implementation in software development processes.

References

[1] Andrew Begel, Nachiappan Nagappan, "Usage and Perceptions of Agile Software Development in an Industrial Context:An Exploratory Study", First International symposium on empirical software engineering and measurement, pp. 255-264, 2007.
[2] Peter Maher, "Weaving Agile Software Development Techniques into a Traditional Computer Science Curriculum", Proc. of 6th IEEE International Conference on Information Technology: New Generation, pp. 1687-1688, 2009.
[3] Anfan Zuo, Jing Yang, Xiaowen Chen, "Research of Agile Software Development Based on Formal Methods", International Conference on Multimedia Information Networking and Security, pp. 762-766, 2010.
[4] Michael J Rees, "A Feasible User Story Tool for Agile Software Development", Proc. Of 9th Asia-Pacific Software Engineering Conference (APSEC' 02), 2002.
[5] Outi Salo, Pekka Abrahamsson, "Integrating Agile Software Development and Software Process Improvement: a Longitudinal Case Study", pp. 193-202, 2005.
[6] Xiaofeng Wang, "The Combination of Agile and Lean in Software Development: An Experience Report Analysis", IEEE Agile Conference, pp. 1-9, 2011.
[7] Richard Mordinyi, Eva Kuhn, Alexander Schatten, "Towards an Architectural Framework for Agile Software Development", 17th IEEE International Conference and workshops on Engineering of Computer Based Systems, pp. 276- 280, 2010.
[8] Jeffrey A. Livermore, "Factors that impact implementing an Agile Software Development Methodology", pp. 82-85, IEEE 2007.
[9] A. Ahmed, S. Ahmad, Dr. N. Ehsan, E. Mirza, S.Z. Sarwar, "Agile Software Development:Impact on Productivity and Quality" , pp. 287-290, IEEE 2010.
[10] Ying Wang, Dayong Sang, Wujie Xie, "Analysis on Agile Software Development Methods from the View of Informationalization Supply Chain Management", 3rd International Symposium on Intelligent Information Technology Application Workshops", pp. 219-222, 2009.
[11] Tobin J. Lehman, Akhilesh Sharma, "Software Development as a Service: Agile Experience", Annual SRII Global Conference, pp. 749-758, IEEE 2011.
[12] Pirjo Nakki, Kaisa Koskela, Minna Pikkarainen, "Practical model for user-driven innovation in agile software development", Proc. Of 17th International Conference on Concurrent Enterprising, pp. 1-8, 2011.
[13] O. Salo, P. Abrahamsson, "Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum", IET Software, pp. 58-64, Vol. 2, No. 1, Feb. 2008.
[14] Tore Dyba, Torgeir Dingsoyr, "What do we know about Agile Software Development", IEEE Software, pp. 6-9, Sep./Oct. 2009
[15] Agile Alliance, http://www.agilealliance.org/
[16] Agile Manifesto and Agile Principles, http://agilemanifesto.org/
[17] "Agile Software Development" Wikipedia, http://en.wikipedia.org/wiki/Agile_software_development
[18] Behrouz Far, "Software Reliability Engineering for Agile Software Development", pp. 694-697, IEEE 2007.
[19] Kazuhiro Matsuo, Shota Anzawa, "Project Practices of Agile Software Development for Under Graduate Development", 40th ASEE/IEEE Frontiers in Education Conference, Session S2D1-S2D2, Oct. 2010.
[20] Subhas C Misra, Uma Kumar, Vinod Kumar, Gerald Grant, "The Organizational Changes Required and the Challenges Involved in Adopting Agile Methodologies in Traditional Software Development Organizations", pp. 25-28, IEEE 2006.

[21] Vladan Devedzic and Sasa R. Milenkovic, "Teaching Agile Software Development: A Case Study", IEEE transactions on education, vol. 54, no. 2, pp. 273-278, May 2011.
[22] Osama Sohaib, Khalid Khan, "Integrating Usability Engineering and Agile Software Development: A Literature Review", International Conference On Computer Design And Appliations (ICCDA 2010), Vol. 2, pp. 32-38, IEEE 2010.
[23] Markus Kohlbacher, Ernst Stelzmann, Sabine Maierhofer, "Do Agile Software Development Practices Increase Customer Satisfaction in Systems Engineering Projects?", IEEE International Systems Conference (SysCon), pp. 168 - 172 IEEE 2011.
[24] Jiangping Wan, Weiping Luo, Xiaoyao Wan, "Case Study on Critical Success Factors of Agile Software Process Improvement", pp. 628-631, IEEE 2011.
[25] The Rules of Extreme Programming, http://www.extremeprogramming.org/rules.html
[26] Kristin Fergis, "The Impact of an Agile Methodology on Software Development Costs", Project Report, EAS 499 Senior Capstone Project, April, 2012.
[27] Nayan Jyoti Kar, "Adopting Agile Methodologies of Software Development", SETLabs Briefing, Infosys Technologies, vol. 4 no. 1, pp. 1-9, July-Sep. 2006.
[28] Feature Driven Development and Agile Modelling, http://www.agilemodeling.com/essays/fdd.htm
[29] Robert Imreh, Mahesh S. Raisinghani, "Impact of Agile Software Development on Quality within Information Technology Organizations", Journal of Emerging Trends in Computing and Information Sciences, Vol. 2, No. 10, pp. 460-475, October 2011.
[30] Sharifah Syed-Abdullah & Mike Holcombe & Marian Gheorge, "The Impact of an Agile Methodology on the Well Being of Development Teams", Empir Software Eng, pp. 143-167, Springer 2006.

Gaurav Kumar is a Research scholar (Software Engineering) at Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, Haryana, India. He has done M. Tech (Information Technology) and B.E. (Computer Science & Engineering). He is Gate Qualified in 2007. He has 4 years teaching experience and 2 years industrial experience.

Dr. Pradeep Kumar Bhatia is Associate Professor at Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, Haryana, India. He has guided many research scholars and M.Tech scholars. His areas of specialization are Software Engineering and Computer Graphics. He has done many projects sanctioned from UGC.He has more than 18 years teaching experience.