Agile Software Development Issues: A Review

Laxmi Shankar Awasthi[1], Anand Kumar Rai[2] and Karuna Shankar Awasthi[1]*

[1]Deptt. of Computer Science, Lucknow Public College of Professional Studies, Lucknow.

[2]Deptt. of Computer Science, Mumtaz Post Graduate College, Lucknow.

[*]Corresponding Author:ka7052@gmail.com

Abstract: The agile software development method is the promising methodology in contemporary software requirement changing environment. In this paper six research papers were reviewed thoroughly concerning the agile software. In these papers few agile issues has been raised for the quality and cost effective software product. Agile software development methodology draw the attention of software development people and software engineering Researchers during the last decade but systematic research and published fruitful results still remains quite limited. Every agile practice has its own expansion cycle that results in scientific, managerial and environmental changes in the software industries. This paper explains the existing problems with agile software project. Nowadays agile practices are becoming more and more leading in the software industry. Agile processes are not always favorable, they have some boundaries as well, and this paper also discusses six recent research papers (from 2009 to 2015) problems and reveals some agile issues.

Keywords:
Agile principal, Agile Philosophy, Agile Manifesto.

1.0 Introduction

Agile software development (ASD) is nowadays a promising software engineering approach for different kind of software project development. Agile principles at first advocated by a group of 17 software practitioners[4]. The doctrine they introduced leading to the emergence of the ASD philosophy are based on their previous success and failure experiences with various software development projects . These practitioners had their own philosophies lying on how they approached software development. However, all of them advocated close teamwork between software development and business teams besides feed storage development by software teams, face-to-face communication besides over importance on written documentation in projects, recurrent delivery of portions of working software project besides final delivery of the complete product at the end. accepting changing requirements by customers besides defining a predetermined set of requirements, and adaptive organizational capability of teams according to changing business requirements. These principles inspire the idea of ASD (Fowler and High smith, 2001). On the contrary, the traditional software development projects earlier to agile were more paying concentration on following well-defined plans and detailed documentations. This article may help the academicians, researchers and software practitioners interested in the area of Agile to gain an overall understanding of the area as well as this article will help to the software engineering organizations.

2.0- History and evolution of ASD

We sum up some of important phases of the progression of different software development methods. After this we elaborate how agile came into existence. The software development methods came into existence successively namely code-and-fix method, the stage wise method (Bengington, 1956), the evolutionary development method (McCracken and Jackson, 1982), the transform method (Balzer , 1983), the waterfall method (Royce, 1987),and the spiral method (Boehm, 1988)[4][9]. The code-and-fix method is one of the initial available process models. In this method, the software development actions are usually undertaken in two phases: writing the code, and then fixing the errors in the code. The major boundaries of the code-and-fix method were the following. Such a method became limited because there was no clear recognition of planning, requirements, and design phases before actually undertaking the coding work for the incorporation of feedback from users. The stage wise method was introduced by Benington in 1956. Contrasting the code-and-fix method, software development in the stage wise method is introduced in different stages: operational plan, operational specifications, coding specifications, coding parameter testing, assembly testing, shakedown, system evaluation but limitation of this method is that it does not have provisions for improving the latter stages of the software based on the experience gathered from the previous stages. McCracken and Jackson, in 1982, proposed the "Evolutionary Software Development Method", in which the operational experience of users was incorporated in the software development lifecycle in its different stages. In this method, the users specify the functional characteristics of the initial operational software, and then feed their experience for determining subsequent product improvements. However, one of the limitations of this approach is that it is often difficult to change the users' existing software to incorporate the additional changes. In other words, using such a process, the maintenance and evolution of operational software can be a challenge. The above methods still lacked the flexibility of easily modifying code through repeated re-optimizations as such code would often become poorly structured, difficult to test, maintain, and evolve. The transform method was proposed in 1983 by Balzer et al. to overcome this difficulty. In the transform method, all later modifications to code are made only to the specification, and not to the design, implementation, and testing stages. However, this method requires that one can convert formal specification of software into code

that specifies the requirements. This method is effective in reducing the amount of software development time, and cost.

The waterfall model was planned with a two-fold improvement over the stage wise method (Royce, 1987; Boehm, 1988). The waterfall method uses feedback loops between consecutive stages of the software development process and it incorporates "prototyping" as an independent phase that would be conducted in concurrence with the early software development stages of requirements analysis and design, before undertaking the actual development of the software. The water fall method for software development remained a popular method for many years, as it was able to reduce many of the restrictions of the previous models, and was used by different organizations for developing sweeping software, although waterfall method have some limitations. Whereas it worked well for the development of compilers, and operating systems, it had strong emphasis on document driven software development that was shown to be of limited help in the development of spreadsheets and small business applications. Additionally, excessive documentation of the specification of poorly understood applications, and their user interfaces was considered unnecessary. Then Boehm proposed the spiral method (Boehm, 1988) that evolved from the application of the above mentioned software development methods, especially the waterfall method. The proposal of the spiral method was an important contribution to software development because it was the first lifecycle method that clearly took a risk-management approach to software development, in compare to the document driven, and the code driven approaches of the preceding methods (Boehm, 1988). The spiral method has an iterative approach to software development in contrast to sequentially approaching the phases of software development as was done previously. In spiral method, the phases is presented in the shape of a spiral, in which the beginning of software development is shown at the epicenter of the spiral and proceeds iteratively through every cycle of the spiral as software development progresses. The above mentioned software development methods (including that of Boehm's) were greatly process based, with heavy importance on documentation. The boundaries of these methods were realized by many software practitioners and were criticized by them. Some of the apparent limitations of these methods include their difficulty to learn and use them quickly, some of them are labor intensive, they are time consuming and thus slow down the development process, many of them are poorly defined and ambiguous to implement in practical situations

(Tolvanen, 1998). Some of the other perceived limitations include the ability of the projects to adapt to the changing circumstances in the organizations and their projects, changing customer requirements. These limitations make them more restrictive and bureaucratic in their behavior (Kalermo and Rissanen, 2002). The agile philosophy came into being to overcome some of the above challenges so as to become more usable in modern day dynamic organizations. The primary move from the traditional software development practices to the agile ones involved laying lesser emphasis on process and documentation and paying more emphasis on quickly developing products and satisfying customers by incorporating their changing needs. Unlike the previous practices, the agile practices do not focus on plan-driven development. A group of 17 software practitioners, who have been following some light-weight software-development methods, met in Snowbird, Utah, USA in 2001 to rationalize their common philosophy, and termed their shared philosophy of software development as "agile". They developed the Manifesto for Agile Software Development, which states the values and principles of the ASD philosophy. The Agile Manifesto is explained in the next section.

3.0 Agile Manifesto and principles of ASD

The manifestofor ASD (Fowler and Highsmith, 2001; Fowler, 2002) [4] is described below:

That is, while there is value in the items on the right, we value the items on the left more (Agile Manifesto). Instead of evaluating whether the stated principle are right or wrong, which can, in fact be situation-dependent, in this section we explain the principles behind ASD. The 12 principles stated in the Agile Manifesto are formally written below:

(1) In Agile software highest priority is given to the customer satisfaction through early and continuous delivery of software.

(2) Software requirements changes are always welcome, in any phase of the Agile processes for the customer's competitive advantage.

(3) Working software is delivered frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

(4) Business people or customer and software developers work together daily during the project.

(5) Projects are built around experienced and motivated individuals. They are given the environment and support they need, and believed to get the job done.

(6) It is believed that the most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

(7) It is believed that working software is the basic goal of progress.

(8) It is believed that agile processes promote sustainable development.

(9) It is believed that continuous attention to technical excellence and good design enhances agility.

(10) It is believed that simplicity – the art of maximizing the amount of work not done is essential.

(11) It is believed that the well suited architectures, requirements, and designs come out from self-organizing teams.

(12) It is believed that success is achieved when at regular intervals the team decides on how to become more effective, then adjusts its behavior accordingly.

4.0 Important Agile Study (2009-2015)

Agile software development is a current trend for the software developers and researchers few research approach we are discussing in brief.

4.1 Study: Lucio Ribeiro (2009) conducted the work on "A Case Study for the Implementation of an Agile Risk Management Process in Multiple Projects Environments"[2].In Software Engineering field the software development process area under discussion to several uncertainties that need to be managed so that they do not interfere with the defined project goals. Few Technical reports describe that the quantity of successful projects is not available than desired. There may be few possible reasons one reason may be that many companies do not apply Risk Management or apply it not properly. In some situation, it is possible that there is a need to guide the Project Risk Management based on organizational environment, In this regard, the article's goal is to report the case study's experience for the accomplishment of an Agile Risk Management Process in multiple projects environments, called GARA (Gestão Ágil de Riscos de Ambiente). Some projects were selected to participate in this study.

in the process which aims to perform an active management of existing impediments mainly those affect more than one project because of its importance or being share resources among these projects. Benefits like the communication among Project Leaders and their experiences in solving impediments are also discussed.

Result: The success of the projects are directly related to the Project Risk Management (PRM), once any activity might present risks as it has been cited in this article, there are still many projects which cannot accomplish their aims. One of the reasons lay on the organizations: they do not apply the PRM, and when they do it, it is not satisfactory. Based on this context, the aim of this article was trying to apply the Agile Management of Environmental Risks process (or GARA) in an organizational environment, seeking for the evaluation of its efficiency and acceptance by the team. And also based on the results, it can be concluded that its application was successful, not damaging the chosen environment. The Multiple Projects Environment Management worries about the distribution and control of the projects resources, once they were chosen for the environment. During the experience, with GARA process, there have been made some necessary adaptations at the Matrix of Impediments for it to get adapted to the needs of the company. The importance of GARA process application in the organization was to provide a meaningful increase of the perception of the involved people at this process, in relation to the identification of the environmental and project impediments, before they could have an impact at the quality of the final product. There were few impediments totally solved, because the analyzed projects in the environment did not have an established deadline and they were always under modifications and/or evolution. Even being noticed that the major part of impediments were operational, and some could not be solved, they had ready replies strategies which were mitigated that could diminish its negative impact. With GARA application, the company which has not had well defined Risk Management Process, could be controlled and the Technology Director started to gather the Project Leaders every week to accompany the projects evolution and employees needs. As a consequence, this article served as a basis to collect improvements and evaluations of the process by the participants.

4.2 Study: Subhas C. Misra and Virender Singh(2013) conducted the research project on"Conceptualizing open agile software development life cycle (OASDLC) model"[6].Software development life cycle (SDLC) has always been the foundation methodology for any software engineer that describe the entire development process which an organization is bound to utilize to achieve successful software. The purpose of this paper is to bring forth a conceptual model after analyzing the best practices in SDLC, and extracting the best out of agile methodologies and the open source software, thereby bringing forward an optimized structure.

Result: The existing SDLC model being employed by "Brihaspati" project was evaluated to adopt best practices for the project. This finally led us to the conceptualization of a better model called as "Open Agile Software Development Life Cycle Model (OASDLC)". OASDLC is conceptualized keeping in mind the existing SDLC models and overcoming the issues faced by Brihaspati project. The OASDLC concept was given to evaluate whether it is able to give us with required solutions or not. Once deemed fit in terms of cost and effort we finally passed our model through survey research and statistical tests for validation. The statistical results were culled out through the survey research and analyzed to finally confirm the validity of our conceptual model. OASDLC holds the properties of both ASDLC and OSSDLC models together. The OASDLC model has been constructed through this work for Brihaspati project and needs to be implemented products as well. The proposed model allows a perfect process flow while undergoing the development phases. The actual validation would prove only after implementation. The quantitative test for all the methods have been based on certain assumptions with a view of type of developers involved. However, this can be tested for other factors as well.

4.3 Study: Subhas Misra(2011) , conducted the study on "Agile software development practices: evolution, principles, and criticisms"[4]. Purpose of study was Agile software development is an promising approach in software engineering, initially proposed and promoted by a group of 17 software professionals who practice a set of "lightweight" methods, and share a common set of values of software development. They consolidated their thoughts, and defined these methods as "agile". The approaches are based on experiences and best practices from the past. The purpose of this article is to sketch the history and evolution of agile software development practices, principles, and the criticisms as studied by the software development professional.

Result: ASD has recently gained extensive fame. The Agile Manifesto states valuing "individuals, and interactions over processes, and tools, working software over comprehensive documentation, customer association over contract negotiation, and responding to change over following a plan" (Fowler, 2002). Although there have been anecdotal and experiential reports available from practitioners of software development projects using agile practices, empirical studies on agile practices are inadequate. Specifically, in this article author outlined the concepts and the principles of ASD, the history and evolution, and the criticisms from different corners of the software development community. Thus, this paper will be extremely useful for the

practitioners who would like to adopt ASD practices in their organization. There are several areas in this domain that deserve the attention for the researchers.

Researchers are required to investigate the trust, privacy and security issues related to ASD. Researchers would also like to model the success factors, changes required and challenges involved in the adoption of ASD in the context of outsourcing/off shoring.

4.4 Study: Eltjo R. Poort(2014) accomplished his study on "Driving Agile Architecting with Cost and Risk"[7]. Five parts of advice can help architects become more effective in an agile without having to apply new methods or frameworks. They explain changes in approach or performance rather than complete practices or principles, so they're easy to digest and apply. The ideas are based on a solution architecting approach called Risk and Cost-Driven Architecture describes changes in attitude or behavior rather than complete practices ,so they're easy to absorb and apply. The ideas are based on a solution architecting approach called Risk and Cost-Driven Architecture.

Result: Decisions are your main deliverable. The decisions talk to concerns that you keep in an accumulation, prioritized by economic impact: risk and cost. These decisions outcome in a minimal architecture, with enough anticipation. There's a lot to say on agile architecting, regarding to topics like project organization and development process. These five parts of suggestion are limited to what you should be able to apply in any organization or process.

4.5 Study: Fernando Selleri Silva (2014) accomplished study on "Using CMMI together with agile software development: A systematic review"[10][11]. The search for adherence to maturity levels by using lightweight processes that require low levels of effort is regarded as a challenge for software development organizations. This study seeks to evaluate, manufacture, and present results on the use of the Capability Maturity Model Integration (CMMI) in grouping with agile software development, and thereafter to give an overview of the topics researched, which includes a conversation of their benefits and limitations, the strength of the findings, and the implications for research and practice.

The search strategy acknowledged 3193 results, of which 81 incorporated studies on the use of CMMI jointly with agile methodologies. The benefits found were grouped into two main categories: first related to the organization in general and second related to the development process, and were organized into subcategories, according to the area to which they refer. The

limitations were also grouped into these categories. Using the criteria defined, the strength of the facts found was measured low. The implications of the results for research and practice are discussed.

Result: Agile methodologies have been used by companies to reduce their efforts to reach levels 2 and 3 of CMMI, there even being reports of applying agile practices to achieve level 5. Among the benefits, improvements in organizational feature, larger team and customer approval, further incorporation, cost reduction, process incorporation, increasing productivity and reducing defects.

The viability of using CMMI together with the agile development is manifested on both sides. From the CMMI, the latest release of the CMMI-DEV Technical Report incorporates a set of suggestions for applying the model in environments with agile methodologies and these suggestions were also considered in its evaluation method (SCAMPI). Depending on the lead evaluator, confirmation of statements, for example, can replace the need for a direct artifact. This is a breakthrough in the appraisal method and favors the adoption of agile methodologies without the need to create (or generate) direct and objective evidence, although it is still subject to a relative dependence on the appraiser. On the agile side, the large number of papers published in events on agile development.

4.6 Study: Rafaela Mantovani Fontana (2014) conducted the study on " Progressive Outcomes :A frame work for maturing in agile software development"[13]. Maturity models are used to guide improvements in the software engineering field and a number of maturity models for agile methods have been proposed in the last years. These models differ in their underlying structure prescribing different possible paths to maturity in agile software development, neglecting the fact that agile teams struggle to follow prescribed processes and practices.

In this paper the objective was to empirically explore how agile teams develop to maturity, as a means to conceive a theory for agile software development evolvement that considers agile team's nature. The complex adaptive systems theory was used as a lens for analysis and four case studies were conducted to collect qualitative and quantitative data .As a result, we propose the Progressive Outcomes framework to describe the agile software development maturing process. It is a frame work in which people have the central role, ambidexterity is a key ability to

maturity, and improvement is guided by outcomes agile teams pursue, instead of prescribed practices.

Result: This study presented a framework for maturing in agile software development. We built this frame work based on the analysis of qualitative and quantitative data in four Brazilian agile teams. Our theoretical foundation, based on complex adaptive systems theory.

5.0 Conclusion and future Research:

In Agile development process it was experienced by the professionals that possibility of several kinds of risks enhanced during the development process as well as after the delivery of the project. To mitigate the different kind of risks it is required an efficient Project Risk Management System (PRMS) as per the project environment. Very few models have been developed so for to mitigate the agile software development risks, but those models are not sufficient to overcome the project risks efficiently, some works was done to check the maturity of the agile software and quality of the development process, but few more agile software challenges are still ahead. These challenges need attention of agile software researchers.

1- How to define risk indicators for the process and make experiences to accumulate them?

2- How to predict the residual risks in agile software while its occurrence is very high in agile process?

3- Which are the assessment approaches that enable estimation of the maturity of an agile team in each of the categories of the Progressive Outcomes frame work?

4- Due to the lack of appropriate evidence, we cannot say the degree to which combining CMMI and agile methodologies is feasible so as to further improvements in the software industry. This question requires further research.

5- The OASDLC model was proposed specially for the Brihaspati project and thus, the viability of the model needs to be ascertained for any other software project as well. The OASDLC model conceptualized out here can be further polished in terms different agile project processes and functionality.

6-Software Architecture is challenging in agile software development environment.

7-How the Dynamic schedule can be managed in agile environment?

8- Agile software methodology does not work well for safety critical and reliable systems.

9-How Agile teams systematize themselves in practice?

6.0 References

[1] Lucio Ribeiro, Cristine Gusmão, Wilmar Feijó, Vicente Bezerra,, "A Case Study for the Implementation of an Agile Risk Management Process in Multiple Projects Environments", pp 1396-1404,PICMET 2009 Proceedings, August 2-6, 2009.

[2] Peter Middleton , David Joyce, "Lean Software Management: BBC Worldwide Case Study", IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT, VOL. 59, pp 20-32 NO. Publisher IEEE, 1 FEBRUARY 2012.

[3] Bruce Powel Douglass, Chief Evangelist, IBM Rational,Leslie Ekas, Development Manager,"Adopting agile methods for safety-critical systems development", IBM Software Group, IBM Software Thought Leadership White Paper ,pp 1-11, October 2012.

[4] Subhas Misra, "Agile software development practices: evolution, principles, and criticisms", International Journal of Quality & Reliability Management, Vol. 29 No. 9, pp. 972-980, Emerald Group Publishing Limited,2012 .

[5] Rashina Hoda, James Noble, Stuart Marshall," Self-Organizing Roles on Agile Software Development Teams", IEEE Transactions on Software Engineering, Vol.39 , Issue: 3, pp 422-444,Sponsored by :IEEE Computer Society Publisher IEEE.,March 2013.

[6] Subhas C. Misra and Virender Singha, " Conceptualizing open agile software development life cycle (OASDLC) model", International Journal of Quality & Reliability Management, Vol. 32 Issue 3, pp. 214 -235, Emerald Group Publishing Limited 0265-671X, December 2013.

[7]Eltjo R. Poort ,"Driving Agile Architecting with Cost and Risk", The pragmatic Architect, Vol. 31, Issue 5, pp 20-23, PUBLISHED BY THE IEEE COMPUTER SOCIETY- IEEE, Oct-2014.

[8] Suprika Vasudeva Shrivastava and Urvashi Rathod," Risks in distributed agile development: A review", Procedia - Social and Behavioral Sciences , vol.133, pp. 417 – 424,Published by Elsevier Ltd, May 2014.

[9]Harleen K. Flora et al, "A Systematic Study on Agile Software Development Methodologies and Practices", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5, Issue-3, pp- 3626-3637,2014.

[10]Alistair Cockburn (Co-Authored Agile Manifesto 2001), "Disciplined Learning The Successor to Risk Management",HIGH MATURITY ORGANIZATIONAL CHARACTERISTICS, Online at <http://alistair.cockburn.us>. CrossTalk -July/August 2014, pp-15-18, August 2014.

[11] Fernando Selleri Silva,"Using CMMI together with agile software development: A systematic review", Information and Software Technology, Vol-58, pp 20–43, Published by Elsevier, February 2015.

[12]Patrick Rempel , Patrick Mäder,"Estimating the Implementation Risk of Requirements in Agile Software Development Projects with Traceability Metrics", Software Systems Group, Vol 9013, pp 81-97 ,Springer International publishing Switzerland, March 2015.

[13] Rafaela Mantovani Fontana, Vitae, Victor Meyer,"Progressive Outcomes: A framework format maturing in agile software development", Vol. 102, pp 88–108, The Journal of Systems and Software, April 2015.