IJCSC Volume 5 • Number 1 March-Sep 2014 pp. 164-169 ISSN-0973-7391

Conceptual Study of Agile Software Development

Sharnil Pandya, Ankur Kumar Yadav, Nikunj Dalsaniya, Vivek Mandir

Department of Computer Science & Engineering,Nirma University, Ahmedabad, India

sharnil.pandya@nirmauni.ac.in, ankurayadav@gmail.com, nikunj.dalsaniya@yahoo.in, vivek.mandir19@gmail.com

ABSTRACT

This paper consists of comparative study of agile processes. Agile processes have important applications in the areas of software project management, software schedule management, etc. In particular the aim of agile processes is to satisfy the customer, faster development times with lower defects rate. Agile development is invented for handling change. This paper compares the agile processes with other software development life cycle models. Agile processes are not always advantageous in every case, they have some disadvantages as well. So the advantages and disadvantages of agile processes are also discussed in this paper.

Keywords: Agile Software Development, Software Engineering Methodologies, Software Measurement, Software Development Life-cycle (SDLC), SCRUM, XP, COSE , Crystal.

1. INTRODUCTION

Software has been part of modern society for more than 50 year. Likewise, so have software development processes. The software process is the foundation for engineering software. Within the context of his book, Pressman defines “a software process as a framework for the tasks that are required to build high-quality software.” [1].

In software development life cycle, there are two main considerations, one is to emphasize on process and the other is the quality of the software and process itself. Agile software processes is an iterative and incremental based development, where requirements are changeable according to customer needs. It helps in adaptive planning, iterative development and time boxing. It is a theoretical framework that promotes foreseen interactions throughout the development cycle. There are several SDLC models like spiral, waterfall, RAD, Incremental, RUP which has their own advantages. Software Development Life Cycle (SDLC) is a framework that describes the activities performed at each stage of a software development cycle.

The software development activities such as planning, analysis, design, coding, testing and maintenance which need to be performed according to the demand of the customer. It depends on the various applications to choose the specific model. Agile process is itself a software development process [2]. Agile process is an iterative approach in which customer satisfaction is at highest priority as the customer has direct involvement in evaluating the software [3].

In modern competitive era changes are frequent to any software product or module which is under development, due to the market competitions priority of requirements changes frequently and only specific development is done which is urgently required and then later on changes and improvements comes into the picture for the rest developed modules. So requirement engineering is done in parallel to software development and requirement changes often happen to survive in the competitive market. Refactoring is also an important type of change requirement which sticks the development policies with the developed code and which comes into the picture once the development task in bulk is over. Refactoring process improves code. It is increasing the function while reducing code bulk.

Agile process is design for change, without refactoring and rebuilding. Its objective is to design programs that are receptive to change. It lets changes be applied in a simple way to reduce or avoid major refactoring, system builds and retesting.

This paper is organized as follows: Characteristics of Agile processes and agile process philosophy are presented in section II. Section III discusses the phases of agile process methodologies. Section IV consists of Agile Processes. Section V discusses the advantages of Agile Process. Section VI is based on Disadvantages of Agile Processes. Section VII is based on COSE, and Section VIII discusses the comparison of agile processes. Finally, conclusions are discussed in the section IX.

2. Agile Process Philosophy

Agile Software development philosophy has it roots in the reality of today’s markets. Main aim of agile development is attempt to deal with some issues introduced by rapidly changing and unpredictable markets. The Agile Manifesto [5] is not an Agile Software Development Methodology itself but the "spiritual" background for Agile Software Development, so it called "Meta-Methodology". Or as [6] said with respect to the Agile Manifesto: "Agile is an Umbrella - Methodologies are Implementations”. Reading the “Manifesto for Agile Software Development” [4] the basic ideas of the philosophy are introduced through four basic values:

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

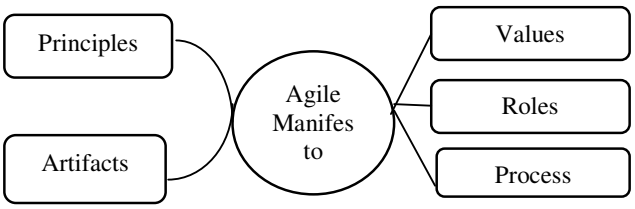

Figure 1. Basic Contents of the Agile Manifesto

2.1 Characteristics of Agile Projects

Agile process requires less planning and it divides the tasks into small increments. Software development life cycle includes the following phases:

1. Requirements gathering, 2. Analysis, 3.Design, 4.Coding,
5. Testing, 6. Maintenance.

The involvement of software team management with customers reduces the risks associated with the software. This agile process is an iterative process in which changes can be made according to the customer satisfaction. In agile process new features can be added easily by using multiple iterations.

1. Iterative
The main objective of agile software processes is satisfaction of customers, so it focuses on single requirement with multiple iterations.
2. Modularity
Agile process decomposes the complete system into manageable pieces called modules. Modularity plays a major role in software development processes.
3. Time Boxing
As agile process is iterative in nature, it requires the time limits on each module with respective cycle.
4. Parsimony
In agile processes parsimony is required to mitigate risks and achieve the goals by minimal number of modules.
5. Incremental
As the agile process is iterative in nature, it requires the system to be developed in increments, each increment independent of others, and at last all increments are integrated into complete system.
6. Adaptive
Due to the iterative nature of agile process new risks may occurs. The adaptive characteristic of agile process allows adapting the processes to attack the new risks and allows changes in the real time requirements.
7. Convergent
All the risks associated with each increment are convergent in agile process by using iterative and incremental approach.
8. Collaborative
As agile process is modular in nature, it needs a good communication among software development team.
Different modules need to be integrated at the end of the software development process.
9. People Oriented
In the agile processes customer satisfaction is the first priority over the technology and process. A good software development team increases the performance and productivity of the software.

3. PHASES OF AGILE PROCESS

Figure shows the life cycle of AGILE process.

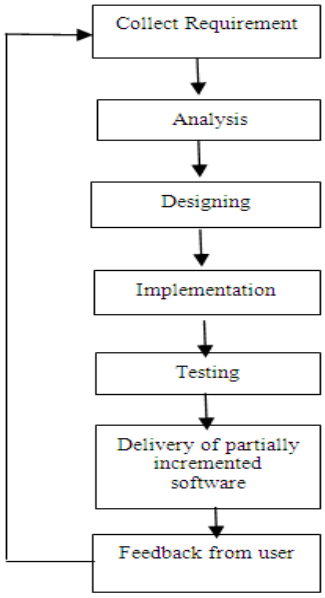

FIGURE 2. PHASES OF AGILE PROCESS. (ITERATIVE AND INCREMENTAL)

(ALL PHASES ARE OVERLAPPED)

4. Agile Processes

This section consists of the agile processes studied in this paper. There are many Agile Methods.

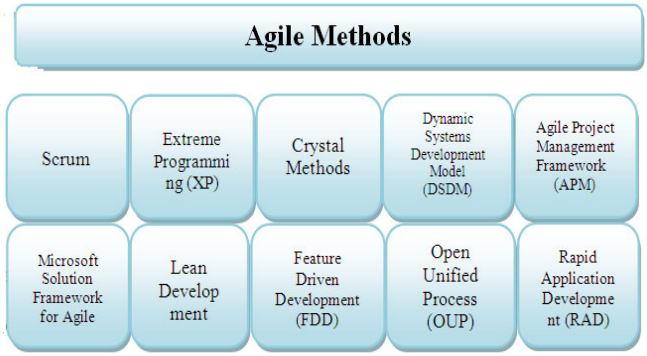

Fig 3. Various Methods

The processes selected in this study are the following:

- Scrum
- Extreme Programming (XP)
- Crystal Methodologies (CM)

4.1 Scrum

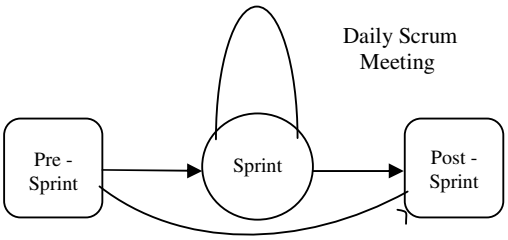

Figure 4 Three Stages in Scrum [7].

The Scrum methodology gets its name from the huddle formed by rugby players to clash with the players from opposition. It was developed by Ken Schwaber and Jeff Sutherland. The primary thought behind this methodology is that the world is totally unpredictable and hence, it is impossible to accurately plan for the future. Scrum is an agile, lightweight framework for managing and controlling software development in rapidly changing and distributed environments
[8]
"Scrum relies on self-commitment, self-organization, and emergence rather than authoritarian measures." [Schwaber, Ken 1996][9].In scrum method the entire development cycle is divided into a series of iteration where each iteration is called as a sprint. Maximum duration of a sprint is 30 days [9]. The Scrum framework is divided into three stages:
Pre-Sprint: The Pre-Sprint stage involves Sprint planning. This is a process of creating a list of features to be incorporated in the system. The owner determines which feature is to be taken up in the next Sprint. A Sprint Goal is also established which provides a purpose to the team to achieve.

Sprint: Sprint stage leads to the development of the software. The feature picked up from the list is implemented in a 30-day cycle. There is a daily Scrum meeting which improves the visibility of each

person's work. Changes to any feature during a Sprint are not allowed, except under extraordinary circumstances.

Post-Sprint: This stage involves customer demonstration, progress review and technical review. This stage ensures that the customer and the team have an early preview to the system.

At the end of this stage, the entire Scrum process is repeated.

Scrum requires at least one daily integration and regression test of the code. In addition, a sprint review session of four hours maximum is organized regularly to discuss and report to the manager and the customer what has been accomplished so far during the sprint. The sprint review session is also a way to receive feedback on regular basis from the various stakeholders involved in the project.

4.2 XP

XP is the most successful method of developing agile software because of its focus on customer satisfaction. XP requires maximum customer interaction to develop the software. It divides the entire software development life cycle into several number of short development cycles. Main advantage of this is that it welcomes and does changes or requirements from the customers at any phase of the SDLC.

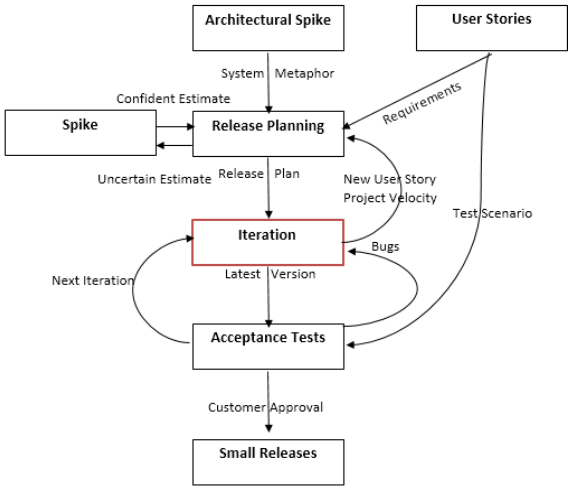

Figure 5 XP

Extreme Programming has defined practices and guidelines that implementers should follow. The process begins by gathering stories. These are short use cases, small enough to fit on an index card. Here each story is business-oriented, estimable and testable. From the stories, the customer selects the most valuable set. This set com prises iteration and is coded first. Coding is done in pairs, two people coding on one machine, and iteration is typically one to two weeks long. Once complete the set is tested and put into production. "The goal of each iteration is to put into production some new stories that are tested and ready to go." [11].

Testing plays a major role in XP. Each iteration is subjected to unit testing. Writing all unit tests prior to writing any code is mandatory. A particular iteration must pass its unit testing prior to going into production. Customers determine system wide tests. Considering their needs and referencing the stories, customers think about what it would take to satisfy them that the iteration is successful. These needs are translated into system wide tests. Testing regularly and often at the unit level and system level provide s feedback and confidence that the project is moving ahead and the system is functioning according to the customer's requirements.

4.3 Crystal Methods

The Crystal methodologies are a set of processes that can be applied to different projects depending on the size and the complexity of a project. The more critical the project, the more rigorous and formal processes are required. Crystal methods define four levels of critically [13]:

- **Life (L):** A system failure is critical and may cause loss of life.
- **Essential money (E):** A system failure may cause loss of money.
- **Discretionary money (D):** A system failure may cause loss of money but can be fixed by referring to the system's user manual.
- **Comfort (C):** A system failure may cause a loss of customer comfort.
-

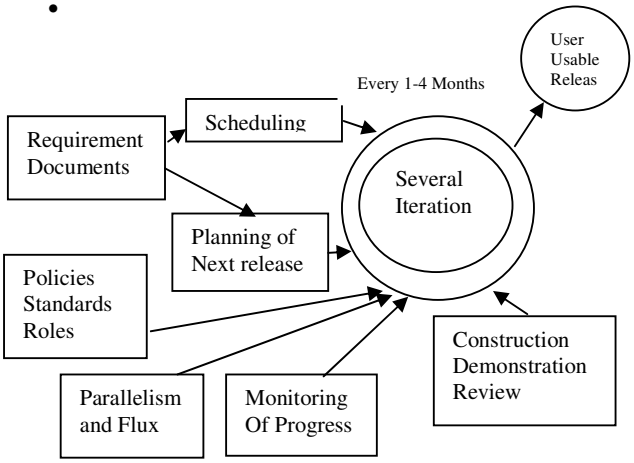

Fig 6.Iterations

Small and non-critical projects can be developed using less rigorous Crystal methods. Larger and more critical projects, however, demand more attention and therefore, a more rigorous Crystal process is used. A system failure for the fourth level may cause a loss of com fort whereas a system failure for the first level may cause a loss of life [12].

In this each member of the Crystal family is marked with a colour indicating the 'heaviness' of the methodology, (ex. the darker the colour the heavier the methodology). Currently two crystal methodologies have been defined: Crystal clear and crystal orange.

The functions or practices are explicitly defined. Staging, this is the scheduled time frame for one increment ranging from one to four months duration. Increments include several iterations during which construction, demonstration, and reviews are included activities. Iterations are monitored and team deliverables gage the progress and stability of iteration. User viewings, (i.e. reviews by users), are conducted one to three times per iteration depending on the criticality of the project. Methodology-tuning is a basic technique used to fix and improve the process applying knowledge gained in one iteration to the next iteration. In addition to the methodology-tuning workshops there are reflection workshops conducted at the start of an increment and midway into it [14].

Crystal methods are based on the principle that how to achieve a maximum extent by which a written communication or documents communication can be reduced to a verbal communication for faster development. All Crystal methods begin with a core set of roles, work products, techniques, and notations. There is no limit on team size in crystal methods. Iteration lengths are large generally 4 months and more. It is built to support distributed team.

The multiple processes offered by the Crystal methodologies are adaptive and agile. They are adaptive because they offer multiple solutions for projects having different criteria. They are agile because they deliver work products at the completion of each project increment and are able to apply lessons learned to the next iteration. Additionally, they demand customer involvement, and decisions are ma de based on outcome s rather than trying to force conformance to a plan developed early in a project's phase.

5. ADVANTAGES OF AGILE PROCESS

1) Adaptive to the changing environment: In agile software development method, software is developed over several iterations. Each iteration is characterized by analysis, design, and implementation and testing. After each iteration the mini project is delivered to the customer for their use and feedback. Any changes that upgrade the software are welcome from the customer at any stage of development and that changes are implemented.

2) Ensures customer satisfaction: This methodology re quires active customer involvement throughout the development [15]. The deliverables developed after each iteration is given to the user for use and improvement is done based on the customer feedback only. So at the end what we get as the final product is of high quality and it ensures the customer satisfaction as the entire software is developed based on the requirements taken from customer.

3) Least documentation: The documentation in agile methodology is short and to the point though it depends on the agile team. Generally they don't make documentation on internal design of the software. The main things which should be on the documentation are product features list, duration for each iteration and date. This brief documentation saves time of development and delivers the project in least possible time.

4) Reduces risks of development: As the incremented mini software is delivered to the customers after every short development cycle and feedbacks are taken from the customers, it warns developers about the upcoming problems which may occur at the later stages of development. It also helps to discover errors quickly and they are fixed immediately.

6. DISADVANTAGES OF AGILE PROCESS

1) Customer interaction is the key factor of developing successful software: Agile methodology is based on customer involvement because the entire project is developed according to the requirements given by the customers. So if the customer representative is not clear about the product features, the development process will go out of the track.

2) Lack of documentation: Though the least documentation saves development time as an advantage of agile method, on the other hand it is a big disadvantage for developer. Here the internal design is getting changed again and again depending on user requirements after every iteration, so it is not possible to maintain the detail documentation of design and implementation because of project deadline [16].

3) Time consuming and wastage of resources because of constant change of requirements: If the customers are not satisfied by the partial software developed by certain iteration and they change their requirements then that incremented part is of no use. So it is the total wastage of time, effort and resources required to develop that increment.

4) More helpful for management than developer: The agile methodology helps management to take decisions about the software development, set goals for developers and fix the deadline for them. But it is very difficult for the baseline developers to cope up with the ever changing environment and every time changing the design, code based on just in time requirements.

7. CHANGE-ORIENTED SOFTWARE ENGINERING

A change model for Change-Oriented Software Engineering (COSE) is proposed in this section. Based on an evolution scenario, a lack of support in current Interactive Development Environments (IDEs) is identified to apply COSE. A set of extensions to an existing model of first-class changes and describe the desired behaviour of change-oriented IDEs to support COSE is introduced.

Model development information as change operations is proposed that is retrieve directly from the programming environment the developer is using, while developer is effecting changes to the system. This accurate and incremental information opens new ways for both developers and researchers to explore and evolve complex systems.

New change request can come at any point of development i.e. at the start of any project or at the middle of development or during the bug fixing for a project or once the project is released and due to market standards and any other associated parameter change, changes in the product. So the model should be enough flexible to support the new change request at any moment of time [17].

Figure depicts the one iteration of proposed model. When a new change requirement will come into the system first of all it is categorized. It means it is filtered in terms of functional requirements, the new change request could be feature change request, it could be database change related request or it could be in terms of a bug or defect in the system or it could be change due to any other reason i.e. change in specifications etc. Once the newly change request is classified it will enter into the second phase where it is fitted to a suitable agile model and technique for this. Here model storming is also done so that in next time same type of change request can be handled faster. The priority of the change request is also decided in terms of execution. In the last phase this change request is sent to the SDLC cycle for the execution.

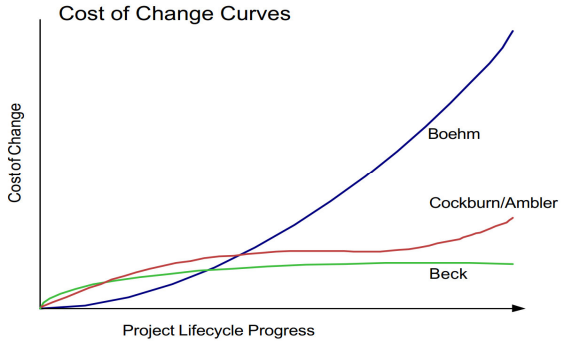

Fig 7. Life Cycle

PROPOSED MODEL

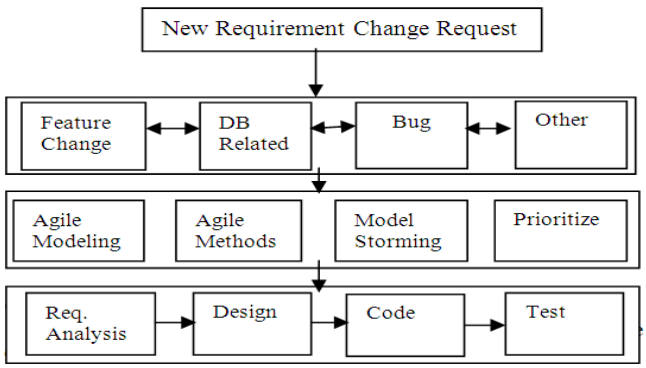

Fig 8 Requirement Requests

8. COMPARISON

	Customer Collaboration	Time to Market	Responding to Change	Documentation	Verification and Validation	Team Mangt
XP	User Stories Onsite Customers	2-3 Months	One Site Customer Short Release	User Stories Test Cases Acceptance Test	TDD Unit Testing Integration Testing Acceptin g Testing	1team/project (3-16 member)
Scrum	Create and prioritize the product backlog Review meetings Onsite Customer	30 days	Daily Scrum Meeting Sprint Review Meeting Short Release	Product Backlog List Sprint Backlog List	Sprint Review Unit Testing Integration Testing Regression Testing	1-4 teams (5-9 team member)
Crystal Clear	Direct User Involvement	1-4 Months	Direct User Involvement Multiple Tuning Technique Short Release	Object Models User Manuals Test Cases	Unit Testing Regression Testing	1-10 teams (3-9 team member)
Crystal Orange	Direct User Involvement	3-6 months	Direct User Involvement Multiple Tuning Technique Short Release	Object Models User Manuals Test Cases Feature Description	Regression Testing Formal Testing	1-20 team (10-40 team member)

Fig.9 Comparision between various techniques

9. CONCLUSION

In this paper we have discussed the software development life cycle models, the characteristics of agile process, methodologies of agile process, advantages and disadvantages. In the comparative study of agile software development with other software development models we conclude that agile project is much better than other software development process in terms of productivity, performance, faster time cycles, risk analysis. Agile processes are implemented in important applications such as web based, testing tools, etc.

In this paper, we presented our analysis of three agile software processes and compared them based on criteria relate to software development projects. Here objective of Agile Process is to help project managers and software engineers understand the key characteristics of these processes and therefore select the most suitable process with respect to the type of software projects they develop. In this paper benefit of agile development is adopted to simplify the change-oriented software engineering.

ACKNOWLEDGMENT

We thank our guide Prof. Sharnil Pandya and faculty Prof. Kruti Lavingia for their kind support and the proper guidance for this survey paper which enhance our knowledge about Agile Software Development.

We would like to thank Prof. Sharnil Pandya for his most support and encouragement. He kindly read my paper and offered invaluable detailed advices on grammar, organization, and the theme of the paper. Also we would like to thank Prof. Kruti Lavinagia to read my thesis and to provide valuable advices. This research paper is made possible through the help and support from everyone, including: parents, teachers, family, and friends.

REFERENCES

[1] Software Engineering a Practitioner's Approach; Pressm an, Roger S. McGraw Hill; 2001; p. 20.

[2] A. Ahmed, S. Ahmad, Dr. N Ehsan, E. Mirza, S.Z. Sarwar,"Agile Software Development: Impact on Productivity and Qulaity", in the Proceedings of IEEE ICMIT.(2010).

[3] B.Boehm and R.Turner, "Balancing Agility and Discipline: A Guide for the Perplexed, Addison, Wesley, 2003. M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in *Proc. ECOC'00*, 2000, paper 11.3.4, p. 109.

[4] The Manifesto for Agile Software Development; http://agilema nifesto.org/ ; last referenced Nov 1, 2002.

[5] R.Dumke, A.Schmietendorf, H. Zuse, Formal Descriptions of Software Measurement and Evaluation – A Short Overview and Evaluation, University of Magdeburg 2005,http://www ivs.cs.unimagdeburg.de/sw-eng/agruppe/forschung/paper/FormalM.pdf

[6] S. Ambler, Quality in an Agile World, September 2005 issue of Software Quality Professional (http://www.asq.org)

[7] Scrum Log, Jeff Sutherland's Website at http://jeffsutherland.com http://jeffsutherland.com/scrum/index.html.

[8] J.O. Coplien. (2011, August 2012). It's Ordered - Not Prioritized. Scrum Alliance [Online]. Available: http://www.scrumalliance.org/articles/367-its-ordered--not-prioritized, accessed 22 Au-gust 2012.

[9] http://en.wikipedia.org/wiki/Scrum_(development)

[10] Mass hysteria and the delusion of crowds; Kenny, Michael; itopia Technoparkstr 1 8005 Zurich Switzerland; 2001; p. 7.

[11] Embracing Change with Extreme Programming; Beck, Kent; IEEE Computer; October 1999; p. 72.

[12] Agile Software Development Methods; Abrahamsson, Pekka, Salo, Outi; VTT Publications ISBN 951-38-6009-4; Sept 2002; p. 39.

[13] A. Cockburn. Crystal Clear: A Human-Powered Methodology for Small Teams. Addison-Wesley Professional, 2004.

[14] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and Analysis" . Espoo, Finland: Technical Research Centre of Finland, VTT Publications 478.

[15] B.Boehm, "Anchoring the Software Process," IEEE Software, July 1996.

[16] B. Boehm and D.Port,"Balancing Discipline and Flexibility w ith the Spiral Model and MBASE". Crosstalk, Dec. 2001.

[17] Romain Robbes and Michele Lanza, "Change-based Approach to Software Evolution", Electronic Notes in Theoretical Computer Science (ENTCS) archive, Vol. 166 , Pages 93-109, 2007.

[18] Barry Boehm, "Software Engineering Economics", Prentice Hall PTR, 1981.

[19] Tobin J Lehman, Akhilesh Sharma , "Software Development as a service: Agile Experiences", in annual SRII Global Conference (2011).

Agile Software Development Ecosystems,Alistair Cockburn and Jim Highsmith,Addison-Wesley, 2004.