© INTERNATIONAL JOURNAL FOR RESEARCH PUBLICATION & SEMINAR
ISSN: 2278-6848 | Volume: 07 Issue: 04 | July – September 2016
Paper is available at www.jrps.in | Email : info@jrps.in

THE SOFTWARE DEVELOPMENT OF ORGANIZATION ON AGILE METHODOLOGY

Saurabh[1], Mrs. Savita Bishnoi[2]

[1] M. tech. Research Scholar, Rohtak Institute of Engg. & Mgt. College

[2] Rohtak Institute of Engg. & Mgt. College

Abstract:

The basic requirement of Software development methodology is delivery, adaptation to requirements and feedback collection on required information. Agile software development is a group of software development methodologies that well promotes the development iterations, open collaboration, and process adaptability throughout the project. This paper includes performance analysis and comparison of agile methodology with traditional one.

However, when some projects are implemented in a business environment, there is often a lack of well established method of project management or skilled project implementers, so that methods can be used that were used in large organizations. A good project management method that can help with successful implementation could prove beneficial to many organizations of small and medium size. Agile project management has great potential to fill this role, and it was with this goal in mind that this research was conducted. This paper is a survey for all agile development methods and their comparison with simple software development methodologies. In this Paper there will be discussed about some agile methodology with respect to parameters of performance analysis and process estimation.

I. Introduction

In recent years, organizations have realized the very real value of Big Data. In order to preserve the competitive edge, we need to change the way we store and manage your data - but main problem is, how will we remain agile with the onset of information?

Agility means that the quality of being agile. These days, the web software industry and Mobile application development industry are searching for a better approach of software development. Conventional software development methods have the feature to completely close all the requirements process before analysis and design phases. Feasibility and compatibility of this process with all projects is very less. Agile methods have an advantage to allow developers, so that they can make late changes in the requirement specification document.

"Agile Software Development Manifesto" is presented as following:

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

a) The communication between the individual who are in development team is very important, because development centers are located at different places. The necessity of interaction between Individuals over tools and versions and processes is very important [1].

b) One objective of software development team is to continuous delivery of the working software for the customers. New releases must be produced for frequent intervals. The property of developers is that, they try to keep the code simple, straight forward and technically as advanced as possible and will try to decrease the documentation.

c) The pace and size of project will depend on the relationship between developers and the stakeholders. The key part of the relationship is the cooperation and negotiation between clients. Agile methods can be used in maintaining good relationship with clients.

d) The development team members are being well-informed and they should be authorized to consider the possible adjustments and enhancements emerging during the development process [2].

II. Related Work

During the 1980s, a number of conferences in information systems were devoted to defining, analyzing, and comparing methodologies, primarily system design methodologies. At that stage methodologies were seen as a way to bring control and repeatability to the development of automated business systems (Olle, Sol, & Tully, 1983).

By the 1980s and 1990s, methodologies for all aspects of development were available; however a parallel stream of critique and empirical research reported the failure or misuse of methodologies in practice. Parnas and Clements (1986) advised the software engineering community to 'fake it'. Rather than follow the ideal, that is the rational, systematic way to develop a software system proposed by SDLC-based methodologies, which involves writing requirements and design documents before beginning development, developers should construct the necessary documentation as system details emerge during development. In this way, the final versions of documents describing requirements, design decisions, and software modules would match the final version of the delivered system. This brief history of system development methodologies up until the late 1990s shows a change in thinking within the IS development research community, from assuming methodology use is universally beneficial, to a recognition that methodology use can be problematic. It also shows that the business environment, with its faster pace, and new technology environments, such as object-orientation and the internet, influence methodology creation. Further, methodologies are an 'ideal' that is seldom achieved in practice, and there is no one universal methodology appropriate for all types of development. In the late 1990s, agile methods emerged to contribute to this landscape.

Table – I Agile Methods by Publication Date

	Agile Method	***Acronym***	***Key Source***
1	Dynamic Systems Development method	DSDM	DSDM (Dynamic Systems Development Method, Version 2, 1995) Stapleton (1997)
2	Crystal methods	Crystal	Cockburn (1998) Cockburn (2002)
3	Extreme Programming	XP	Beck (1999) Beck (2000) 1st EditionBeck and Andres (2005) 2nd Edition
4	Adaptive Software Development	ASD	Highsmith (2000)
5	Scrum	Scrum	Beedle, Devos, Sharon, Schwaber, & Sutherland (1999) Schwaber & Beedle (2002)
6	Feature Driven Development	FDD	Palmer & Felsing (2002)
7	Lean Development	LD	Charette (2002) Poppendiek & Poppendiek (2003)
8	EVO	EVO	Gilb (2005)
9	AgileUP	AUP	Ambler (2008) First published online 2005

III. Agile methods

Individual agile methods were created in reaction to persistent problems in software development not adequately addressed by traditional system development methodologies or software engineering techniques, and the need to expedite software development in the business and technology environment of the late 1990s and early 2000s (Beck, 2000; Cockburn, 2002). Agile methods share a common basis in the practical experiences of software engineers, and ideas from new product development literature such as Takeuchi and Nonaka's (1986) work identifying how to best manage projects when developing new products under intense time-pressure [3][4].

Agile methods produce the first delivery of the project in one or two weeks, to achieve rapid feedback. To decrease in change agile methods will invent simple. Agile methods are based on iterative and incremental methods, so they improve design issues and quality [5][6].

Definition of Agile method:

Process is agile when :

a) Incremental: rapid iterations and small releases
b) Cooperative: Strong Customer-developer relation
c) Straight: The methods are easy to learn and to modify with documentation
d) Adaptive: Instant ability to entertain changes

IV. The need for agility:

The business conditions are constantly changing and technology is rapidly evolving so enterprises must preserve their freedom of action. They can't afford to get locked in by a specific approach to analyzing data, specific technology architecture, or a specific vendor product stack. When they will lock the specific approach for a specific data and technology, then there will not be any efficiency enhancements. Agility can have different meanings in different technological contexts [7]. In business perspective agility means having the ability to readily entertain new approaches to addressing competitive challenges as the market changes. For IT, agility means to enhance the efficiency, the freedom to change different layers of the technology stack to avoid getting locked in to a particular method and process.

V. Analyzing Agile Methodology

Agile methods are based on adaptive software development methods, while traditional SDLC models (waterfall model, for example) are based on a predictive approach. In traditional SDLC models, teams work with a detailed plan and have a full list of characteristics and tasks that must be completed in the next few months or the entire life cycle of the product. Predictive methods completely depend on the requirement analysis and careful planning at the beginning of the cycle. Any change that is to be included will go through a strict change control management and prioritization. The agile model uses an adaptive approach where there is no detailed planning and only clear future tasks are those related to the characteristics that must be developed. The team adapts to dynamic changes in the product requirements. The product is frequently tested, minimizing the risk of major faults in the future. Interaction with the clients is the strong point of agile methodology and open communication and minimal documentation are typical characteristics of the agile development environment. Teams collaborate closely and often are located in the same geographical space [9].

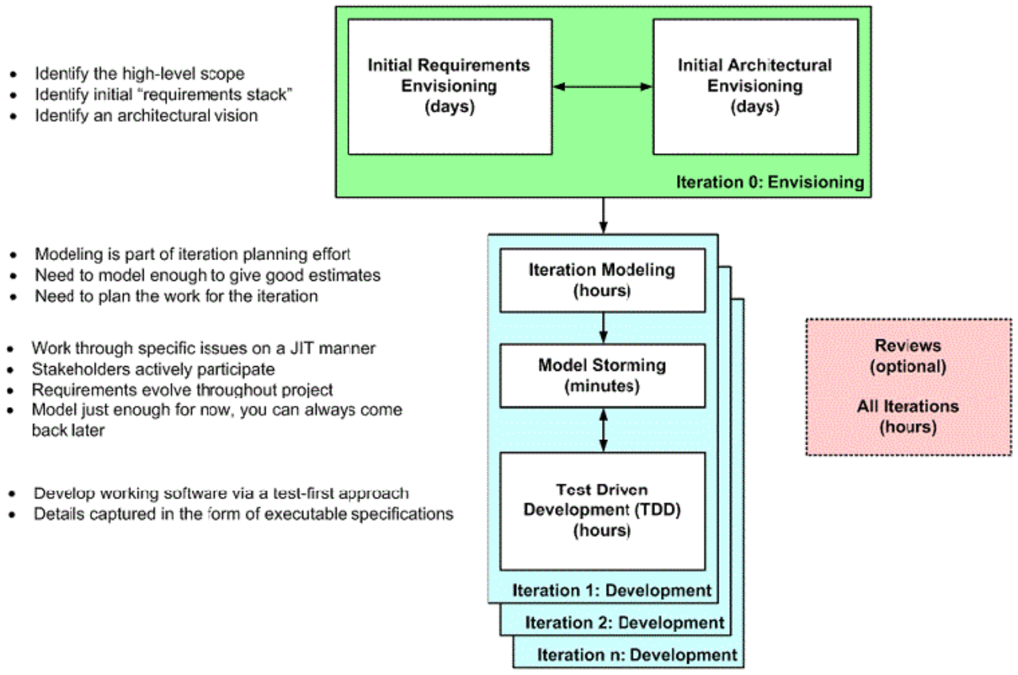
Identify the high-level scope
Identify initial "requirements stack"
Identify an architectural vision
Initial Requirements Envisioning (days)
Initial Architectural Envisioning (days)
Iteration 0: Envisioning
Modeling is part of iteration planning effort
Need to model enough to give good estimates
Need to plan the work for the iteration
Work through specific issues on a JIT manner
Stakeholders actively participate
Requirements evolve throughout project
Model just enough for now, you can always come back later
Develop working software via a test-first approach
Details captured in the form of executable specifications
Iteration Modeling (hours)
Model Storming (minutes)
Test Driven Development (TDD) (hours)
Iteration 1: Development
Iteration 2: Development
Iteration n: Development
Reviews (optional)
All Iterations (hours)

Figure 1: The Agile Model Driven Development (AMDD) lifecycle for software projects

Figure 1 depicts the lifecycle of Agile Model Driven Development (AMDD). During "iteration 0", the first iteration of an agile project, you need to get your project organized and going in the right direction. Part of that effort is the initial envisioning of the requirements and the architecture so that you are able to answer critical questions about the scope, cost, schedule, and technical strategy of your project. Details about the business domain are identified on a just-in-time (JIT) basis during iterations via initial iteration modeling at the beginning of each iteration; or by modeling storming throughout the iteration. Analysis is so important to agilists that we do it every day.

Although agile methodologies triumph over traditional ones in several aspects, there are any difficulties in making them work. One of them is the significant reduction of documentation and the claim that the source code itself should be the documentation. [8] Thus, developers used to agile methods tend to insert more comments in source code in order to clarify and explain. They ask lots of questions to the experienced developers and this may delay completion of the iteration, which can lead to increased development costs. On the other hand, traditional methods emphasize documentation in orientation and clarification of the project for the development team, so there is no concern about not knowing the project details or not having a knowledgeable developer. The fact that agile development allows changes in requirements in an incremental way lead to two dependency problems in design: rigidity and mobility. Rigidity means a change in the system leads to a cascade of changes in other modules, while mobility means the inability of the system to include reusable components because they involve too much effort or risk.

- **Agile Software Analysis Process**

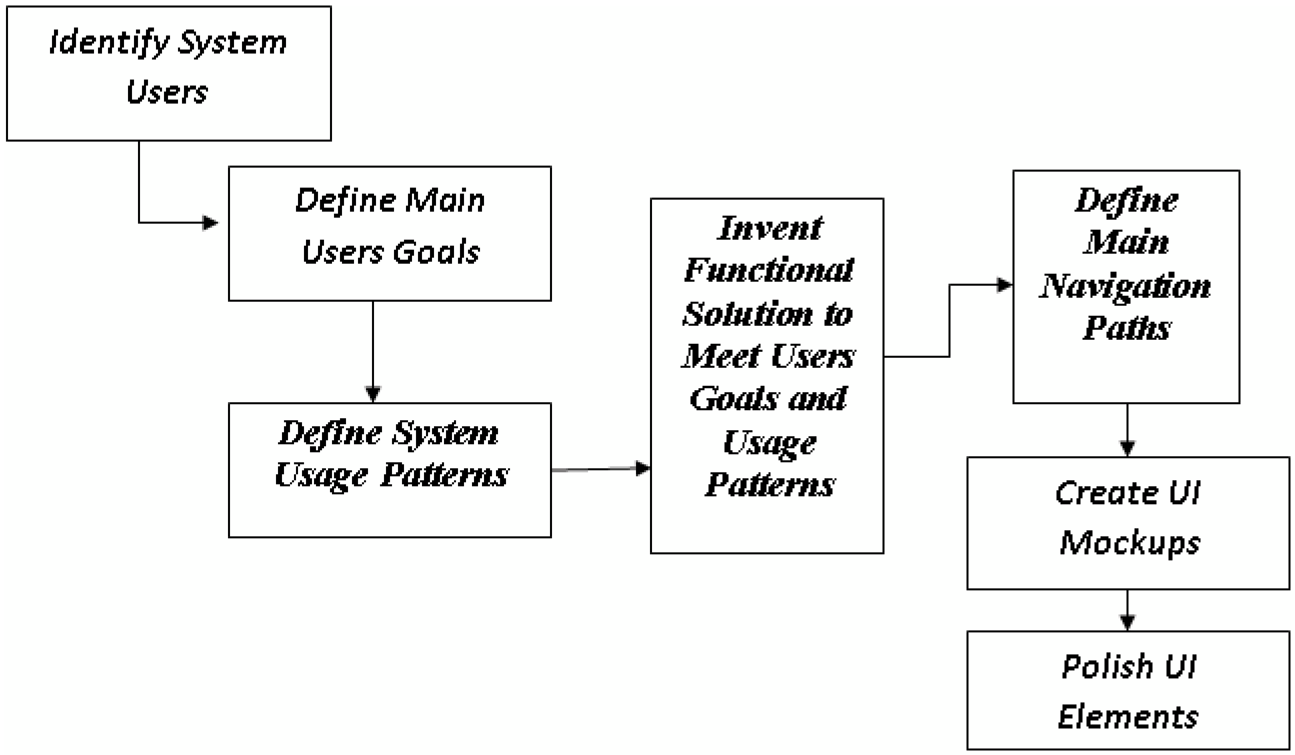
Identify System Users
Define Main Users Goals
Define System Usage Patterns
Invent Functional Solution to Meet Users Goals and Usage Patterns
Define Main Navigation Paths
Create UI Mockups
Polish UI Elements

Figure 2: Steps for Agile Software Analysis Process

- **Composing An Agile Estimating Process**

Teams can spend huge amounts of time breaking down features to create their estimates, but the actual time needed is usually a vastly different number. An average software project begins when a team or person outlines a project and receives approval to go forward. In the first some stages of a project, someone guesses how long it will take to deliver. This person may be a salesperson, project manager, or development manager. They may make a guess based on their experience, or they may have some quick chats with seasoned employees and solicit their opinions [11].Agile estimation techniques address the shortcomings of the methods used.

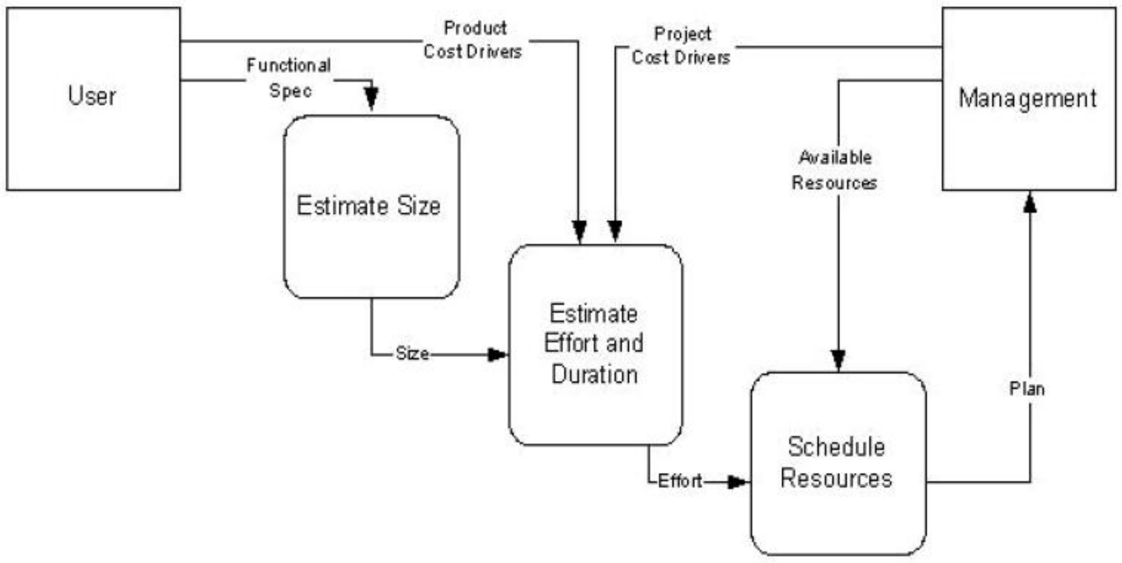

Figure 2: Steps for Agile Software Analysis Process

VI. Conclusions

This survey paper shows the how the agile software development is better than other software development approaches. The efficiency can be achieved and the performance of agile methodology. As in the traditional software development the specific methods and techniques are locked down. Traditional methodologies concentrate more on Processes, tools, contracts and plans. In contrast to traditional methods, agile methods keep emphasis on interaction, working software, embracing change at any moment of the project, customer relationships. So this can be concluded that the agile is more people centric than traditional one. And these can't be defined by small set of principles and methods or techniques.

References

1. Agile Software Development methods-Review and analysis by Pekka Abrahamsson, Outi Salo, Jussi Ronkainen and Juhani Warsta.

2. Abrahamsson, P., Warsta, J., Siponen, M.T. and Ronkainen, J. New Directions on Agile Methods: A Comparative Analysis, Proceedings of the International Conference on Software Engineering, 2003 (Oregon, USA).

3. L. Williams and A. Cockburn, —Agile Software Development: It‘s about Feedback and Change, IEEE Computer, June 2003, pp. 39-43

4. Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, *48*(5), 72-78.

5. Manifesto for Agile software development; http://agilealliance.com

6. Lindvall, M., Basili, V. R., Boehm, B., Costa, P., Dangle, K., Shull, F., Tesoriero, R.,Williams, L., & Zelkowitz, M. (2002). Empirical findings in agile methods. In Proceedings of Extreme Programming and Agile Methods – XP/Agile Universe Conference 2002, Chicago, IL, 97-207.

7. Agile software development: Evidence from the field. Alan MacCormackhttp://www.agile develop-pmentconference.com/2003/files/AlanAgileSoftwar eJun03.ppt

8. Boehm, B.W. Software Engineering Economics, 1981 (Prentice Hall, Upper Saddle River, New Jersey)

9. K. Peterson, A Comparison of Issues and Advantages in Agile and Incremental Development between State of the Art and an Industrial Case. Journal of System and Software. 2009

10. Boehm, B., Port, D., & Brown, A. W. (2002). Balancing plan-driven and agile methods in software engineering project courses. *Computer Science Education*,*12*(3), 187-195.

11. Meso, P., & Jain, R. (2006). Agile software development: adaptive systems principles and best practices. *Information Systems Management*, *23*(3), 19-30.

12. L.R. Vijayasarathy, Agile Software Development: A survey of early adopters. Journal of Information

Technology Management Volume XIX, Number 2. 2008

13. Cho, J. (2008). Issues and Challenges of Agile Software Development with Scrum. *Issues in Information Systems*, *9*(2), 188-195.

14. M. Poppendieck and T. Poppendieck, Lean Software Development. Boston: Addison Wesley, 2003.

15. T. Potok and M. Vouk, "The Effects of the Business Model on the Object-Oriented Software Development Productivity," IBM Systems Journal, vol. 36, no. 1, pp. 140-161, 1997.

16. K. Schwaber and M. Beedle, Agile Software Development with SCRUM. Upper Saddle River, NJ: Prentice-Hall, 2002.

17. J. Stapleton, DSDM: The Method in Practice, Second ed: Addison Wesley Longman, 2003.

18. M. Vouk and A. T. Rivers, "Construction of Reliable Software in Resource-Constrained Environments," in Case Studies in Reliability and Maintenance, W. R. Blischke and D. N. P. Murthy, Eds. Hoboken, NJ: Wiley-Interscience, John Wiley and Sons, 2003, pp. 205- 231.

19. L. Williams, "The XP Programmer: The Few Minutes Programmer," IEEE Software, vol. 20, no. 3, pp. 16-20, May/June 2003.

20. L. Williams and A. Cockburn, "Special Issue on Agile Methods," IEEE Computer, vol. 36, no. 3, June 2003.

21. L. Williams and R. Kessler, Pair Programming Illuminated. Reading, Massachusetts: Addison Wesley, 2003.