ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

# A Transition from Traditional to Agile Software Development

**Anupama Kaushik**

Assistant Professor, Department of IT, Maharaja Surajmal Institute of Technology, New Delhi, India

**Abstract:** The traditional software development methodologies are also known as "Heavy Weight" methodologies. They are based on a strict sequential series of steps such as requirement phase, implementation, testing and deployment. They require defining and documenting a stable set of requirements at the beginning of a project itself. The three most significant traditional methodologies are Waterfall, Spiral Model and Unified Process. Agile software development (ASD) is a new buzz word within software engineering community. These are also known as "Light Weight Methodologies". Agile processes, or development methods, represent an apparently new approach for planning and managing software development projects. ASD differs from traditional approaches as it puts less emphasis on up-front plans and strict plan-based control and more on mechanisms for change management during the project. In this paper a study on characteristics of both traditional software development as well as agile software development (ASD) and the reasons for transition from traditional to agile development is done.

**Keywords:** Traditional software development; Heavy Weight Methodologies; Agile software development; Light Weight Methodologies.

## I. INTRODUCTION

The term traditional software development or Heavy Weight" methodologies mean the traditional ways of developing the software, where the software companies follow the fixed sequence of steps for software development present in different methodologies. Many developers found this process very frustrating and these methodologies accommodate very few changes. As a result, several consultants have independently developed methodologies and practices to embrace and respond to the inevitable change they were experiencing. These methodologies and practices are based on iterative enhancements, a technique that was introduced in 1975 and that has become known as agile methodologies [1].

Over the past few years software development organizations have learned about the benefits of Agile Methodologies, such as Scrum and XP. The scientific literature and business journals present numerous success stories highlighting the benefits of organizations which successfully adopted agile practices. As a result many organizations are now aspiring to adopt agile practices [2].

## II. CHARACTERISTICS OF HEAVY WEIGHT METHODOLOGIES

From past many decades heavyweight methodologies were around. They have a disciplined approach upon software development with the aim of making software development more predictable and more efficient.

The heavy weight methodologies have these similar characteristics [1]:

**Predictive approach** – Heavyweight methodologies have a tendency to first plan out a large part of the software process in great detail for a long span of time. This approach follows an engineering discipline where the development is predictive and repeatable. A lot of emphasis is put on the requirements of the system and how to resolve them efficiently. These requirements are then handed over to another group who are responsible for building the system. The project planning team predicts the task for construction team and reasonably predicts the schedule and budget for construction.

**Comprehensive Documentation** – Heavy weight methodologies view the requirements document as the key piece of documentation. In this approach, all the customers' requirements are gathered prior to writing any code. They then get a sign off from the customer to limit and control the changes. After that a more comprehensive documentation is made.

**Process Oriented** - The goal of heavyweight methodologies is to define a process that will work well for whoever happens to be using it [3]. The process would consist of certain tasks that must be performed by the managers, designers, coders, testers etc. For each of these tasks there is a well-defined procedure.

**Tool Oriented –** Project management tools, Code editors, compilers, etc. must be in use for completion and delivery of each task.

## III.CHARACTERISTICS OF LIGHT WEIGHT METHODOLOGIES

According to Highsmith and Cockburn [4], "what is new about agile methods is not the practices they use, but their recognition of people as the primary drivers of project success.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

The light weight methodologies have these similar characteristics.

**People Oriented**- Agile methodologies consider people – customers, developers, stakeholders, and end users – as the most important factor of software methodologies. As Jim Highsmith and Alistair Cockburn state, "The most important implication to managers working in the agile manner is that it places more emphasis on people factors in the project: amicability, talent, skill, and communication" [5].

**Adaptive** –Agilists welcome changes at all stages of the project. They view changes to the requirements as good things, because they mean that the team has learned more about what it will take to satisfy the market [3]. Today the challenge is not stopping change but rather determining how to better handle changes that occur throughout a project. "External Environment changes cause critical variations. Because we cannot eliminate these changes, driving down the cost of responding to them is the only viable strategy" [4].

**Conformance to Actual** – Agile methodologies value conformance to the actual results as opposed to conformance to the detailed plan. Highsmith states, "Agile projects are not controlled by conformance to plan but by conformance to the business value" [6]. Each iteration or development cycle adds business value to the ongoing product. For agilists, the decision on whether business value has been added or not is not given by the developers but instead by end users and customers.

**Balancing Flexibility and Planning** – Plans are important, but the problem is that software projects cannot be accurately predicted far into the future, because there are so many variables to take into account. A better planning strategy is to make detailed plans for the next few weeks, very rough plans for the next few months, and extremely crude plans beyond that [7]. In this view one of the main sources of complexity is the irreversibility of decisions. If you can easily change your decisions, this means it's less important to get them right – which makes your life much simpler. The consequence for agile design is that designers need to think about how they can avoid irreversibility in their decisions. Rather than trying to get the right decision now, look for a way to either put off the decision until later or make the decision in such a way that you will be able to reverse it later on without too much difficulty [8].

**Empirical Process –** Agile methods develop software as an empirical (or nonlinear) process. In engineering, processes are either defined or empirical. In other words, defined process is one that can be started and allowed to run to completion producing the same results every time. In software development it cannot be considered a defined process because too much change occurs during the time that the team is developing the product. Laurie Williams states, "It is highly unlikely that any set of predefined steps will lead to a desirable, predictable outcome because requirements change technology changes, people are added and taken off the team, and so on" [9].

**Decentralized Approach** – Integrating a decentralized management style can severely impact a software project because it could save a lot of time than an autocratic management process. Agile software development spreads out the decision making to the developers. This does not mean that the developers take on the role of management. Management is still needed to remove roadblocks standing in the way of progress. However management recognizes the expertise of the technical team to make technical decisions without their permission.

**Simplicity** – Agile teams always take the simplest path that is consistent with their goals. Fowler states, "They (agile teams) don't anticipate tomorrow's problems and try to defend against them today" [3]. The reason for simplicity is so that it will be easy to change the design if needed on a later date. Never produce more than what is necessary and never produce documents attempting to predict the future as documents will become outdated. "The larger the amount of documentation becomes, the more effort is needed to find the required information, and the more effort is needed to keep the information up to date" [10].

**Collaboration –** Agile methods involve customer feedback on a regular and frequent basis. The customer of the software works closely with the development team, providing frequent feedback on their efforts. As well, constant collaboration between agile team members is essential. Due to the decentralized approach of the agile methods, collaboration encourages discussion. As M. Fowler describes, "Agile teams cannot exist with occasional communication. They need continuous access to business expertise" [3].

**Small Self-organizing teams –** An agile team is a self-organizing team. Responsibilities are communicated to the team as a whole, and the team determines the best way to fulfill them. Agile teams discuss and communicate together on all aspects of the project. That is why agility works well in small teams. As Alistair Cockburn and Jim Highsmith highlight, "Agile development is more difficult with larger teams. The average project has only nine people, within the reach of most basic agile processes. Nevertheless, it is interesting to occasionally find successful agile projects with 120 or even 250 people" [5].

## IV. NEED OF TRANSITION FROM TRADITIONAL TO AGILE

The main difference between the traditional and agile methodologies is the acceptance of change. It is the ability to respond to change that often determines the success or failure of a software project [9]. Traditional methods

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

freeze product functionality and disallow change. However one of the key factor for the success of agile processes is its response to change at any stage of the project. It becomes very difficult to provide a set of stable requirements in this volatile and constantly changing environment. Martin Fowler and Jim Highsmith founders of the agile manifesto mention that, "Facilitating change is more effective than attempting to prevent it. Learn to trust in your ability to respond to unpredictable events; it's more important than trusting in your ability to plan for disaster," [11]. Furthermore, B. Berry and P. Philip [11] and C. Jones [12] both concluded that during their project development experience, requirements change at 25% or more.

A research study was conducted by a Standish Group of 365 respondents and regarding 8,380 projects representing companies across major industry segments. From their findings, 16.2% of the projects were completed on-time and on-budget with all features and functions specified. However 52.7% of the projects are completed but over-budget, over the time estimate and offering less features and functions while 31.1% of projects were canceled at some point during the development cycle [13] . The study further reveals that the three major reasons that a project will succeed are user involvement, executive management support, and a clear statement of requirements.

Another limitation of heavyweight methodologies is the handling of complexity. "Complex rules and regulation give rise to simple stupid behaviour," says the former CEO of Visa International [3]. The approach to plan everything and then to follow the plan works smoothly for stable and less complex environment but for larger and more complex environments, this technique would fall apart.

The agilists believe that people can respond quicker and transfer ideas more rapidly when talking face-to-face than they can in heavyweight methodologies when reading or writing documentation. When developers talk with customers and sponsors, they could work out difficulties, adjust priorities, and examine alternate paths forward in ways not possible when they are not working together. According to Cockburn the most significant single factor is "communication" [14].

Agile methodologies focus on the talents and skills of individuals and molds processes to specific people and teams, not like heavyweight methods where all tasks and roles are assigned to individuals and it is expected that the individuals will perform their tasks accordingly.

Another argument between agile and heavyweight methodologies is the measurement of project success. A predictive heavyweight project considers a project to be successful that is on-time and on-cost [15].However agilists measures project success by questioning if the customer got software that is more valuable to them than the cost put into it. According to Martin Fowler, "A good predictive project will go according to plan, a good agile project will build something different and better than the original plan foresaw" [3].

## V. CONCLUSION

Traditional software development methodologies are always the choice of software developers but the use of agile methodologies is like a feather on a cap. In this paper the need of agile methodologies and why transition from traditional to agile software development is required is studied.

## ACKNOWLEDGMENT

The author is thankful to all the authors whose references are included.

## REFERENCES

[1] M.A. Awad, "A Comparision between Agile and Traditional Software Development Methodologies".
[2] Naftanaila Ionel, " Agile Software Development Methodologies: An Overview of the Current State of Research"pp 381-385
[3] M. Fowler, "The New Methodology," http://www.martinfowler.com/articles/newMethodology.html
[4] J. Highsmith and A. Cockburn, "Agile Software Development: The Business of Innovation", IEEE Computer,
[5] J. Highsmith and A. Cockburn, "Agile Software Development: The People Factor", IEEE Computer, http://www.jimhighsmith.com/articles/IEEEArticle2Final.pdf
[6] J. Highsmith, Agile Software Development Ecosystem. Addison Wesley, 2002
[7] J. A. Highsmith, Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Addison Wesley 2000.
[8] M. Fowler, "Is Design Dead?", Software Development, http://www.martinfowler.com/articles/designDead.html
[9] Williams and A. Cockburn, "Agile Software Development: It's about Feedback and Change," IEEE Computer, June 2003, pp. 39-43
[10] P. Wendorff, An Essential Distinction of Agile Software Development Processes Based on Systems Thinking in Software Engineering Management. Addison Wesley; page 218.
[11] B. Barry and P. Philip, "Understanding and Controlling Software Costs", IEEE Transactions on Software Engineering, Vol. 14 No 10, October 1988
[12] C. Jones, Applied Software Measurements. McGraw Hill,1997
[13] The Standish Group International, "The CHAOS Report," http://www.standishgroup.com/sample_research/chaos_1994_1.php
[14] A. Cockburn, "Characterizing People as Non-Linear, First-Order Components in Software Development," http://alistair.cockburn.us/crystal/articles/cpanfocisd/characterizingpeopleasnonlinear.html
[15] Dynamic System Development Method Consortium, "DSDM Tour", http://www.dsdm.org.