International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 6 | ISSN : 2456-3307

Analysis the Strength of Agile Methodologies in Software Development

K M. Jyoti[1], Mr. Tinku Singh[2], Parul Saharavat[3]
[1]M.Tech Scholer J.P.I.E.T, Meerut, Uttar Pradesh, India
[2]Department of computer science J.P.I.E.T, Meerut, Uttar Pradesh, India
[3]Department of computer science D.N. Polytechnique, Meerut, Uttar Pradesh, India

ABSTRACT

There are several software development methodologies in use today. Some companies have their own customized methodology for developing their software but the majority speaks about two kinds of methodologies: heavyweight and lightweight. The strengths and weakness between the two opposing methodologies are discussed and the challenges associated with implementing agile processes in the software industry are provided. According to our findings agile methodologies can provide good benefits for small scaled and medium scaled projects but for large scaled projects traditional methods seem dominant. The study also focuses the different success factors of agile methods, the success rate of agile projects and comparison between traditional and agile software development.

Keywords : Software Process, Software Development Methodology, Agile, Heavyweight Methodologies.

I. INTRODUCTION

Software has been part of modern society for more than 50 years. Software development started off as a messy activity often mentioned as “code and fix”. The software was written without much of a plan, and the design of the system was determined from many short term decisions. This worked well for small systems but as systems grew it became more difficult to add new features and bugs were harder to fix. This style of development was used for many years until an alternative was introduced: Methodology. Methodologies impose a disciplined process upon software development with the aim of making software development more predictable and more efficient. Traditional or heavyweight methodologies are plan driven in which work begins with the elicitation and documentation of a complete set of requirements, followed by architectural and high level design development and inspection. Due to these heavy aspects, this methodology became to be known as heavyweight. These methodologies and practices are based on iterative enhancements, a technique that was introduced in 1975 and that has become known as agile methodologies.

II. ORGANIZATION OF TITLE

The goal, therefore, is to begin filling in the gap of methodologies by conducting a detailed review of both heavyweight and agile methodologies. For heavyweight method, several methods are reviewed such as Waterfall, Unified Process and Spiral. Further an overall view of the characteristics of heavyweight methods is discussed. Next, the same procedure for agile methodologies is followed. Some agile approaches such as Extreme Programming, Scrum, Dynamic System Development Method, Feature Driven Development and Adaptive Software Development underlining the characteristics of agile methods are introduced. Furthermore, a comparison of the different agile methods in order to highlight

the similarities and differences between them are carried out. The next section criticizes the limitations of each heavyweight and agile methods. Following this, the challenges associated with implementation of agile processes in the software industry according to software practitioners and anecdotal evidence.

III. PROBLEM STATEMENT

Now a days in any field like business, education, sports etc. the success depend on the software being used, due to the fast development in technology the organizations needs its software updated and the software should meet all the business needs. The rapid growing completions between the organizations have created a challenge for the software development companies (Czarnacka-Chrobot, 2010).
The agile software process is the combination of the best practices and their previous success and failure experiences with many software development projects regarding what works and what not. Each of these two practitioners (best practice and previous success and failure experience) had their own different philosophies about how they approached software development. However, all of them advocated close collaboration between software development and business teams, as opposed to silo development by software teams; face-to-face communication, as opposed to over-emphasis on written documentation in projects; frequent delivery of portions of working software, as opposed to final delivery of the complete product at the end; accepting changing requirements by customers, as opposed to defining a fixed set of requirements "cast-in-stone"; and adaptive organizational capability of teams according to changing business requirements (Misra et al., 2009).

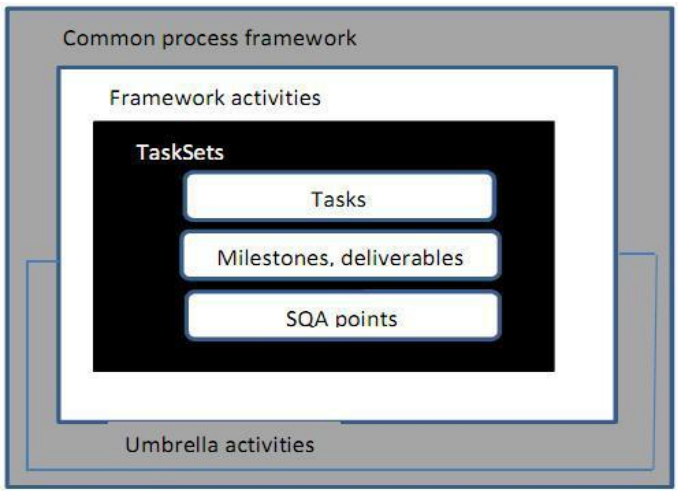

Figure 1. Software process (Pressman, 2001)

The word agile means light weight; the main theme of agile method is the simplicity and speed. The main points of agile methods according to Fowler and Highsmith (2001) are Incremental (small software releases, with rapid cycles) Cooperative (customer and developers working constantly together with close communication) Straightforward (the method itself is easy to learn and to modify, well documented) Adaptive (able to make last moment changes).

IV. HEAVYWEIGHT METHODOLOGIES

Software development Process models are used for all most all type of software development projects and according to Sommerville (2011) a software model is the simplified form of a software process that represents a particular perspective and provides partial information about the process.Software process is a framework of activities that are involved in all most all the software projects regardless of the size and complexity of the tasks (Pressman, 2001).
Those phases are listed below.

- ✓ **Software specifications**: Here the functionality of the software is defined and the constraints on the operations of those functionalities are defined.

- ✓ **Software design and implementation**: The software which can complete the required specification can be produced.
- ✓ **Software validation**: The software should satisfy the customer, means the software should perform all the tasks for which it is produced.
- ✓ **Software evolution**: The software must be ready to meet the changing customer need.

V. HEAVYWEIGHT METHODOLOGIES CHARACTERISTICS

The heavyweight development methodology is based on a sequential series of steps, such as requirements definition, solution build, testing and deployment, whereas lightweight methodologies propose executing the project steps in parallel. For example, the manager of a project that is based on the heavyweight methodology won't agree to build the IT solution until the full requirements have been determined, and so it continues for each project phase. Still, any project team larger than 10-20 people and working in multiple locations may be a good candidate for a heavyweight methodology. Heavyweight methodologies can be the better choice when you have multiple teams working at different locations and you need tighter control to formalize key parts of the project.

VI. COMPARISON OF VARIOUS HEAVYWEIGHT METHODOLOGIES ON THE BASIS OF DIFFERENT PARAMETERS

Table 1. Comparison of different Heavyweight Methodologies

MODEL/ MERITS	WATERFALL MODEL	ITERATIVE WATERFALL MODEL	PROTOTYPE	V-SHAPED MODEL	SPIRAL MODEL
Success rate	Low	High	Good	High	High
Cost	Low	Low	High	Very high	High
Flexibility	Rigged	Less flexibility	Highly flexible	Little flexible	Flexible
Risk analysis	High	No risk analysis	Low	Low	Low
Cost control	Yes	No	No	Yes	Yes
Reusability	Limited	Yes	Weak	Low	Yes
Risk analysis	Only at beginning	Low	No risk analysis	Low	Yes
Implementation time	Long	Less	Less	Less	Depends on project
Interface	Minimum	Crucial	Curial	Minimum	Crucial
Security	Vital	Limited	Weak	Limited	High
Expertise required	High	High	Medium	Medium	High
Simplicity	Simple	Simple	Simple	Intermediate	Intermediate
User involvement	Only at beginning	At the beginning	High	At the beginning	High
Resource control	Yes	Yes	No	Yes	Yes

VII. AGILE MODELING

Agile – devoting "the quality of being agile; readiness for motion; nimbleness, activity, dexterity in motion" as mentioned in the Oxford Dictionary [13] – software development methods are attempting to offer once again an answer to the eager business community asking for lighter weight along with faster and nimbler software development processes.
Following are the Agile Manifesto principles:

- ✓ **Individuals and interactions** - In agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- ✓ **Working software** - Demo working software is considered the best means of communication with the customer to understand their requirement, instead of just depending on documentation.
- ✓ **Customer collaboration** - As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.

Responding to change

Agile development is focused on quick responses to change and continuous development. All these methodologies acknowledged that high quality software and more importantly customer satisfaction could only be achieved by bringing "lightness" to their processes. Some of the most used agile methodologies are listed below.

VIII. EXTREME PROGRAMMING (XP)

Extreme programming (XP) has evolved from the problems caused by the long development cycles of traditional development models [14]. The term 'extreme' comes from taking these commonsense principles and practices to extreme levels. A summary of XP terms and practices is shown below [14]:

- ✓ **Planning:** The programmer estimates the effort needed for implementation of customer stories and the customer decides the scope and timing of releases based on estimates.
- ✓ **Small/short releases:** An application is developed in a series of small, frequently updated versions. New versions are released anywhere from daily to monthly.
- ✓ **Metaphor:** The system is defined by a set of metaphors between the customer and the programmers which describes how the system works.
- ✓ **Simple Design:** The emphasis is on designing the simplest possible solution that is implemented and unnecessary complexity and extra code are removed immediately.
- ✓ **Refactoring:** It involves restructuring the system by removing duplication, improving communication, simplifying and adding flexibility but without changing the functionality of the program
- ✓ **Pair programming:** All production code is written by two programmers on one computer.
- ✓ **Collective ownership:** No single person owns or is responsible for individual code segments rather anyone can change any part of the code at any time.
- ✓ **Continuous Integration:** A new piece of code is integrated with the current system as soon as it is ready. When integrating, the system is built again and all tests must pass for the changes to be accepted.
- ✓ **40-hour week:** No one can work two overtime weeks in a row. A maximum of 40-hour working week otherwise it is treated as a problem.
- ✓ **On-site customer:** Customer must be available at all times with the development team.

- ✓ **Coding Standards:** Coding rules exist and are followed by the programmers so as to bring consistence and improve communication among the development team.

Adaptive Software Development (ASD)

Adaptive Software Development (ASD), developed by James A. Highsmith, offers an agile and adaptive approach to high-speed and high-change software projects [25]. It is not possible to plan successfully in a fast moving and unpredictable business environment. In ASD, the static plan-design life cycle is replaced by a dynamic speculate-collaborate-learn life cycle. ASD focal point is on three non-linear and overlapping phases [23]:

- ✓ **Speculate:** To define the project mission, make clear the realization about what is unclear.
- ✓ **Collaborate:** Highlights the importance of teamwork for developing high change systems
- ✓ **Learn:** This phase stresses the need to admit and react to mistakes, and that requirements may well change during development.

Lean Software Development (LSD)

Lean Software Development (LSD) is the application of lean principles to the craft of software development. So what is Lean? According to the National Institute of Standards and Technology Manufacturing Extensions Partnership's Lean Network, Lean is: "A systematic approach to identifying and eliminating waste through continuous improvement, flowing the product at the pull of the customer in pursuit of perfection." [26]

"Lean Software Development reduces defects and cycle times while delivering a steady stream of incremental business value." [27]

IX. CONCLUSION AND FUTURE WORK

It's a common fact that software plays a very important role in business and any type of business are dependent on software. If we go 30 years back then at that time there was not that much competitions in business and the organizations were using their old technology and were fulfilling their business requirements. During the current situation due to advancement in the internet field and e-business the business of majority of the companies are expanded globally and the those companies are using very advanced technology due to high competition between the business companies and because of that competition the organizations business requirements are changing at a very fast rate and it is not possible for the software companies to use the traditional software model for their software development. Those companies need such software development methods that are flexible to requirements and changed requirements and can deliver the software product in short possible time and within budget.

Agile methods have been used by software companies and the success rate of agile methods is much more than the traditional methods. I argue that agile methods are very simple and easy and can complete the customer requirements easily within time and budget because agile methods involve customer during the software develop phases. There is a close communication between the customer and developers, and also among the developers (team members).

X. REFERENCES

[1]. Cong Liu,David Umphress, Heavyweight or Lightweight: A Process Selection Guide for Developing Grid Software, ACM-SE '08 March 28-29, 2008, Auburn, AL, USA

[2]. Sheetal Sharma, Darothi Sarkar, Divya Gupta, Agile Processes and Methodologies: A Conceptual Study, / International Journal on Computer Science and Engineering (IJCSE), ISSN: 0975-3397, Vol. 4 No. 05 May 2012

[3]. Kaushal Pathak, Anju Saha, Review of Agile Software Development Methodologies, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 2, February 2013, ISSN: 2277 128X

[4]. B. Grady, C. Robert, J. Newkirk, Object Oriented Analysis and Design with Applications, 2nd edition, Addison Wesley Longman, 1998

[5]. P. Kruchten, "What is Rational Unified Process?", The Rational Edge, http://www.therationaledge.com/content/jan_01/f_rup_pk.html Accessed 2/2/2005

[6]. C. Larman, Agile & Iterative Development: A Manager's Guide. Addison-Wesley, 2004.

[7]. B. Boehm, "A Spiral Model of Software Development and Enhancement," IEEE Computer, May 1998.

[8]. W. Cunningham, "Agile Manifesto." http://www.agilemanifesto.org/, Accessed on 10/7/2004

[9]. S. W. Ambler, "Duking it out", Software Development, July 2002

[10]. First eWorkshop on Agile Methods, Centre for Experimental Software Engineering Maryland, April 8 2002

[11]. J. Highsmith and A. Cockburn, "Agile Software Development: The Business of Innovation", IEEE Computer, http://www.jimhighsmith.com/articles/IEEEArticle1Final.pdf Accessed on 10/10/2004