International Journal of Recent Research and Review, Vol. VI, June 2013

ISSN 2277 – 8322

Comparative Study of Agile Methods and Their Comparison with Heavyweight Methods in Indian Organizations

Uma Kumari[1], Abhay Upadhyaya[2]

[1]*Ph.D Scholar, Department of Computer Science, Jagannath University, Jaipur, India*

[2]*Professor, University of Rajasthan, Jaipur, India*

Email: umasecd@gmail.com, abhayu@rediffmail.com

***Abstract* : "Agile" means nimble or quick moving and "Agility" means the ability to think and draw conclusions quickly. In terms of software development agility means dynamic, content specific, growth oriented and able to adopt the changes quickly and easily. The roots of agile go back more than a decade but it is on the boom from the last two decades. It was February 2001, when a group of people kept a meeting in Utah, in order to find an alternative method to the existing heavy software development methodologies and the result was agile methodology. Although there are 13 agile methods and exact number of Agile method is still a debate question. I have included only 5 methods, Scrum, XP, DSDM, FDD, Lean Software**

***Keywords :* Agile, Scrum, Extreme Programming, Dynamic Systems Development Method, Feature-Driven Development Method, Lean Software**

I. INTRODUCTION

Traditional software development methods are not always feasible in rapidly changing business environment. This paper demonstrates the difference between heavyweight and agile software development. Agile Manifesto is collection of values and principles, which are found in most agile methods. Agile alliance formulated their ideas into values and further to twelve principles [1] that support those values which are as follows:

i. Highest priority is to satisfy the customer through early and continuous delivery of valuable software.
ii. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
iii. Deliver working software frequently from a couple of weeks to a couple of months, with a preference to shorter timescale.
iv. Business people and developers must work together daily throughout the project.
v. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
vi. The most efficient and effective method to conveying information to and fro within a development team is face to face conversation.
vii. Working software is the primary measure of progress.
viii. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain constant pace indefinitely.
ix. Continuous attention to technical excellence and good design enhances agility.
x. Simplicity- the art of maximizing the amount of work not done- is essential.
xi. The best architectures, requirements and designs emerge from self organizing teams.

xii. At regular intervals the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Values of Agile Manifesto are given below:

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

II. CHARACTERISTICS OF AGILE

Agile consists of following characteristics:

i. Modularity
ii. Incremental development
iii. Team composition
iv. People oriented
v. Cooperative
vi. Collaborative and communicating working style
vii. Maintaining the Integrity of the Specifications
viii. Lightness

III. SCRUM

Scrum is huddled mass of players engaged with each other to get a job done. Scrum for software development came out of rapid prototyping community because they wanted a methodology that would support an environment in which the requirements were not only incomplete at the start, but also could change rapidly during development. Scrum introduces the ideas of flexibility, adaptability and productivity. Scrum concentrates on how the team members should function together in order to produce the system flexibility in constantly changing environment. The main idea of Scrum is that system development involves several environment and technical values (e.g. requirements, time frame, resources and technology) that are likely to change during the process. This made the development process unpredictable and complex, requiring flexibility of the system development process for it to be able to respond to the changes. Scrum phases are introduced according to Schwaber [2].

A. Pre-game phase

This phase includes two sub-phases: planning and architecture/high level design.

Planning includes the definition of the system being developed. A product backlog is created containing all the requirements that are currently known. The requirements can originate from the customer, sales and marketing division, customer support or software developers. The requirements are prioritized and effort needed for their implementation is estimated. The product Backlog list is constantly updated with new and more detailed items, as well as with more accurate estimations and priority orders.

In the architecture phase, the high level design of the system including the architecture is planned based on the current items in the product Backlog. In case of enhancement to existing system, the changes needed for implementing the Backlog items are identified along with the problem they may cause. A design meeting is held to go over the proposals for the implementations and decisions are made on the basis of this review.

B. The development phase

This phase is called game phase is the agile part of scrum approach. This phase is treated as “black box” where unpredictable is expected. The different environment and technical variables (such as time frame, quality, requirements, resources, implementation technologies and tools and even development methods) identified in Scrum, which may change during the process are observed and controlled through various Scrum practices during the Sprints, of development phase. In the development phase, the system is developed in Sprints. Sprints are interactive cycles where the functionality is developed or enhanced to produce new increments. Each Sprint includes the traditional phases of system development. One Sprint is planned to last from one week to one month. There may

be 3 to 8 Sprints in one system. Also there can be more than one team building the increment.

C. *The post game phase*

This phase contains the closure of the release. This phase is entered when an agreement has been made that the environmental variables such as the requirements are completed. In this case, no more items and issues can be found nor can any new ones be invented. The system is now ready for release and preparation for this is done during post-game phase, including the tasks such as integration, system testing and documentation.

IV. EXTREME PROGRAMMING (XP)

Extreme Programming has evolved from the problems caused by the development cycles in traditional development models [3]. It first started as an opportunity to get job done with practices that had been found effective in software development processes. After a number of successful trials, XP methodology was developed on the key principles and practices used. The term 'extreme' comes from taking these commonsense principles and practices to extreme levels. The life cycle of XP consists of 5 phases:

A. Exploration phase

In this phase the customers write out the story cards that they wish to be included in first release. Each story card describes a feature to be included into the program. At the same time the project team familiarize themselves with the tools, technology and practices they will be using in the project. The technology to be used will be tested and architecture possibilities for the system are explored by building a prototype of the system. The exploration phase takes time between weeks to a few months, depending largely on how familiar the technology is to programmers.

B. Planning phase

This phase sets the priority order for the stories and agreement of the contents of the first small release. Programmers estimate the effort required by each story and schedule is then agreed upon. The time span of schedule of the first release doesn't take more than two months. Planning phase itself takes about few days.

C. Iteration to release

This phase includes several iterations of the system before the first release. The schedule set in the planning stage is broken down to a number of iterations that will each take one to four weeks to implement. The first iteration creates a system with the architecture of the whole system. This is achieved by selecting the stories that will enforce building the structure for the whole system. The customer decides the stories to be selected for each iteration. The functional tests created by the customer are run at the end of each iteration. At the end of last iteration the system is ready for production.

D. The productionizing phase

This phase requires extra testing and checking of the performance of the system before the system can be released to the customer. At this phase, new changes may still be found and the decision has to be made if they are included in the current release. During this phase, the iterations may need to be quickened from 3 weeks to 1 week. The postponed ideas and suggestions are documented for later implementations during maintenance phase.

E. Maintenance phase

After the first release is produced for customer use, the XP project must both keep the system in the production running while also producing new iterations. In order to do this, this phase has been introduced. Maintenance phase may require incorporating new people into the team and changing the team structure.

F. The death phase

This phase is near when the customer has no longer any stories to be implemented. This requires that the system satisfies customer needs in other respects. This is the time in XP process when necessary documentation of the system is finally written as no more changes to the architecture, design or code are made. Death may also

occur if the system is not delivering the desired outcomes, or if it becomes too expensive for further development.

V. Dynamic Systems Development Method (DSDM)

DSDM is widely used in U.K. DSDM is a non-profit and non proprietary framework for RAD development, maintained by DSDM consortium [4]. DSDM provides a framework of controls for RAD, supplemented with guidance on how to efficiently use these controls. The fundamental idea behind DSDM is that instead of fixing the amount of functionality in a product, and then adjusting time and resources to reach that functionality, it is preferred to fix time and resources and then adjust the amount of functionality accordingly.

DSDM consists of 5 phases: Feasibility study, business study, functional model iteration, design and build iteration and implementation. The first two phases are sequential and done only once. In the last three phases, the actual developments are done which are iterative and incremental. DSDM approaches iterations as timeboxes. A timebox lasts for a predefined period of time, and the iteration has to end within the timebox.

A. Feasibility study phase

It is where the suitability of DSDM for a given project is assessed. Judging by the type of project and most of all, organizational and people issues, the decision is made, whether to use DSDM or not. Two work products are prepared- a feasibility report and an outline plan for development. Optionally, a fast prototype can also be made if the business or technologies are not known well enough to be able to decide whether to proceed to the next phase or not. The feasibility study phase is not expected to take more than a few weeks.

B. Business study

It is the phase where essential characteristics of the business and technology are analyzed. The recommended approach is to organize workshops, where a sufficient number of customer's experts are gathered to be able to consider all relevant facets of the system to be able to agree on the development priorities.

C. Functional model iteration phase

It is the first iterative and incremental phase. In each iteration the contents and approach for itertations are planned [5]. The iteration goes through and results are analyzed for further iterations. Both analysis and coding are done, prototypes are build and experiences gained from them are used in improving the analysis models. The prototypes are not to be entirely discarded. A functional model is produced as an output containing the prototype code and analysis models. Testing is also continuing and essential part of this phase. There are four outputs in the phase. Prioritized function is the prioritized list of the function that is delivered at the need of the iteration. Functional prototyping review documents collect the user comments about the current increment, working as input for subsequent iterations. Non functional requirements are listed, mainly to be dealt within the next phase. Risk analysis of further development is important document in the function model iteration phase because from next phase onwards, encountered problem will be more difficult to address.

D. Design and built iteration

This is where the system is mainly built. The output is a tested system that fulfils at least the minimum agreed set of requirements. Further development is based on user's comment. The final implementation phase is where the system is transferred from development environment into actual production environment. Trainings are given to users and the system is handled over to them.

VI. Feature-Driven Development (FDD)

It does not cover the entire software development process, but focus on the design and building phase. FDD consists of 5 sequential processes.

A. Develop an overall model phase

In the beginning of this phase domain experts are already aware of the scope, context and requirements of the system to be built. Documented requirements such as use cases or functional specifications are likely to exist at this phase. After each walkthrough, a development team works in small group in order to produce object models for the domain area at hand. The development team then discusses and decides upon the appropriate object model for each domain areas.

B. Build a feature list phase

Walkthrough object models and existing requirement documentation give a good basis for building a comprehensive features list for the system being developed.

C. Plan by feature phase

It includes creation of a high-level plan, in which the feature sets are sequenced according to their priority and dependencies and assigned to chief programmers. Further, the classes identified in the process are assigned to individual developers, i.e. class owners.

D. Design by feature and build by feature phase

In this a small group of features are selected from the feature sets, and feature teams needed for developing the selected features are formed by the class owners. The design by feature and build by feature processes are iterative producers, during which the selected features are produced. One iteration should take few days to a maximum of two weeks.

VII. LEAN SOFTWARE

Lean Software Development helps software organizations to optimize their processes and production methods in order to deliver their products to the market much faster and with better quality. Lean puts main focus on people and communication. People who produce the software are respected and they communicate efficiently. It is more likely that they will deliver good product and the final customer will be satisfied [6].

Lean Software Development subsequently gave birth to agile software Development methods and its main branches are Scrum and Crystal Clear. Lean Software Development is not a management or development methodology, but it offers principles that are applicable in any environment to improve software development.

Lean software development follows seven principles which are

i. Eliminate Waste
ii. Amplify learning
iii. Decide as late as possible
iv. Defer Commitment
v. Deliver as fast as possible
vi. Empower the team
vii. Build integrity

VIII. FEATURES AND DRAWBACKS OF DIFFERENT AGILE METHODS

Features and drawback of Scrum, XP, DSDM, FDD and Lean Software are as per given in the Table I.

IX. DIFFERENCE BETWEEN AGILE AND HEAVYWEIGHT METHODS

Every method has its weakness and strengths, so depending upon need methodology is decided. Major factors that effect this selection can be categorized as per table II.

X. REFERENCES

[1] Agile manifesto website online available at www.agilemanifesto.org

[2] Schwaber Ken., & Beedle Mike,"Agile software development with Scrum",Upper Saddle River, New Jersey, Prentice Hall, pp. 142.

[3] Elssamadisy, A.(2001), XP on a large scale project-a developer's view, In M. Marchesi,G. Succi, D.Wells & Williams(Eds)," Exreme Programming prespectives", Addison –Wesley, pp.123.

[4] Dynamic Systems Development Method (2nd ed.), Ashford Tesseract , DSDM_Consortium (1995), pp. 239.

[5] V. R. Basili and A. J. Turner, "Iterative Enhancement: A Practical Technique for Software Development," *IEEE Transactions on Software Engineering*, vol. 1, no. 4, pp. 266 - 270,1975.

[6] M. Poppendieck and T. Poppendieck, *Lean Software Development Boston: Addison Welsey, 2003, pp. 203*

[7] Glass, R. L. , "Agile Versus Traditional: Make Love, Not War!,", Cutter IT Journal 14(12): pp. 12–18, 2001.

[8] Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, Highsmith, J.A., 2000 New York: Dorset House, ISBN 0-932633-40-4, pp. 392.

TABLE I
Agile methods features and drawbacks

S. No	Method Name	Features	Drawbacks
1.	Scrum	Independent, self-organizing and small development teams,30 ay release cycle	Integration and acceptance tests are not detailed
2.	XP	Customer driven development, small teams, daily builds	While individual practices are suitable for many situations, overall view and management practices are less attractive
3.	DSDM	Application of controls to RAD, use of timeboxing and empowered DSDM teams. First truly Agile software development method, use of prototyping	While the method is available only consortium members have access to white papers dealing with actual use of the method.
4.	FDD	5 step process, object oriented component based development, method simplicity, design and implement the system by features, object modeling	It focuses only on design and implementation, it needs other supporting approaches as well
5.	Lean software	Focuses on people and communication, follow 7 principals, elimination of waste reduces project time and cost.	Decisions have to made promptly

TABLE II
Comparison of Agile methods with heavyweight methods

S. No	Criteria	Agile Methods	Heavyweight Methods
1.	Approach	Adaptive [7]	Predictive
2.	Aim	To deliver software quickly [8]	Software is delivered at a defined speed
3.	Benefits	(a) Can work when requirements constantly changes (b) At the end of each stage it can adopt new ideas (c) Agile team has launchable product at the end of each tested stage (d) Allow specification changes as per user's requirement	(a) With frequent requirement changes it cannot work (b)Cannot adopt new ideas in-between stages (c) These do not deliver product in-between production. (d) Does not allow specification changes as per user's requirement
4.	Team size	Small	Large
5.	Documentation	Low	High
6.	Testing	Bugs are eliminated in the development cycle and product is double tested after the bug elimination	Bugs are eliminated at the end only
7.	Success measurement	Business Value	Confirmation to plan
8.	Management Style	Decentralize	Autocratic
9.	Time period	Measured in weeks	Measured in months
10.	Emphasis	On cowboy coding (i.e. absence of defined method), People oriented	There exists a defined method, process oriented
11.	Cycles	Numerous	Limited
12.	Return on investment	Early	Late
13.	Risks	Unknown risks	Well understood risks
14.	Architecture	Designed for current requirements	Designed for current and foreseeable requirements