Software Development Life Cycle: Traditional and Agile- A Comparative Study

Arpita Sharma[1] Mahender Kr. Beniwal [2]

Abstract- In the field of software development, software development lifecycle is the most important component. There is a number of software development methodologies used in software industry today. The paper discussed below focuses on the modern SDLC which are traditional methods and the agile methods. It also explains the compensation and shortcomings of the traditional as well as agile methods. Along with this, it suggests some improvements which could help in improving current agile development.

I. INTRODUCTION

For more than 50 years, software has been a part of modern civilization. In present scenario, several software development methodologies have been used. Software development life cycle (SDLC) is a process of building or maintaining software systems which includes various phases from preface to post development software testing and estimation. In this, the development team uses some models and methodologies in order to develop the software systems. There are mainly two kinds of methodologies, namely- heavyweight and lightweight. The heavyweight methodologies are also considered as the traditional way to develop software which claims its support to widespread planning, detailed documentation and expansive design. The light weight methodologies is also known as agile modeling have achieved considerable attention from the software engineering community from the last few years. Web based applications and traditional softwares differ from each other in some different aspects. The reason behind designing a software application is to perform a particular set of tasks. This set of tasks involves complex computation and processing and it gives well-defined results. In order to ensure that the end product comprises of high degree of reliability and robustness along with user acceptance, the complete development process is governed which is not an easy task.

II. TRADITIONAL SOFTWARE DEVELOPMENT

Waterfall method, V-Model and RUP are the software methodologies which are called traditional software development methodologies and are classified into the heavyweight methodologies. These Heavyweight methodologies are based on a sequential series of steps like requirements definition, solution building, testing and deployment. Traditional software development methodologies require defining and documenting a stable set of requirements at the beginning of a project.

There are four phases which are the features of traditional software development method.

– In the first step, the requirements for the project are analyzed and the time required to complete this project is determined which is the time to implement the various phases of development. Along with this, the problems are calculated that may arise in the project.

– The next step is the design and architectural planning phase in which a technical infrastructure is produced in the form of diagrams or models. It provides an effective road map for the developers to implement and brings to the surface the potential issues that the project may face as it proceeds.

– After the architectural and design plan phase, the project moves into the development phase. In this phase, the code is produced until the specific goals are reached. Often, development is broken down into smaller tasks which are then distributed among various teams based on their skills.

– The architectural and design phase if followed by the testing phase. Once the project moves to completion and the developers tend to meet the project requirements, the customer will become part of the testing and feedback cycle and the project is delivered after the customer satisfy with it.

The traditional software development methods depend on the processes which are predetermined, along with the on-going documentation which is written as the work progresses further. It helps in guiding the process for further development.

III. AGILE SOFTWARE DEVELOPMENT APPROACH

Agile is a very recent software development methodology or in general a group of methodologies which is based on agile manifesto. The reason behind developing this method was to solve some shortcomings in traditional software development methodologies. The Agile software development is based on the concept of incremental and iterative development. In this, the phases within a development life cycle are revisited over and over again. It takes customer feedback to improve the quality of software and to meet the end requirements. An agile method gives high priority to the customer participation early in the development cycle. It believes in including the customer in testing early in the development process and often as possible. Testing is performed at each point. The Agile method believes in starting the testing of the project at the beginning of the project and continuing it throughout the project. The popular variations of Agile are:

- Scrum
- Extreme Programming
- Lean Software Development
- Crystal methodologies
- Feature driven development
- Dynamic Software development method

In agile software development approach, the development life cycle is divided into smaller parts, which are called "iterations" or "increments". Each of these increments touches on each of the conventional phases of development. According to Agile program, the major agile factors include the following:

1. Early customer involvement
2. Iterative development
3. Self-organizing teams
4. Adaptation to change

The difference between the characteristics of Agile software development method and traditional software development approach can be listed below as:

Characteristics	Traditional Developments	Agile
Principal Objective	To build quality software products at minimum cost	To bring quality products to market as quickly as possible
Major engineering technologies used	Generators, Object oriented methods, modern programming languages(Dot Net/C++), CASE tools etc.	Component based methods 4^{th} and 5^{th} generation languages (HTML, Java, DotNet etc),Visualization (Motion, animation) etc.
Process Employed	Proper SDLC	Ad hoc
Products Developed	Code based systems, mostly new, some reuse, many external interfaces, often-complex applications	Object based systems, many reusable components (Shopping cart etc) few external interfaces, relatively simple
Stakeholders	Generally groups confined within the boundaries of departments, divisions or organizations	Wide range of groups, known and unknown, residing locally or overseas
Development Approach employed	Classical, requirement-based, phased and/or incremental delivery, use cases, documentation driven	Agile methods, extreme programming, building block-based, demo-driven, prototyping
Approach proceeding fashion	One-shot fashion	Moves step by step
Estimating Technologies Used	Source line of code, function point model	Design to fit based on available resources
Legal, Social and Ethical Issues	Less legal, social and ethical issues because application will work in limited boundaries	Contents can be easily copied and distributed without permission or acknowledgement of copyright an intellectual property rights. Application should take into account all groups of users
Quality Drivers	General product which can meet current requirement and manual process	Reliability, Usability, security, product which will work with future needs.
Time taken to deliver result	Comparatively slowly	Quickly
People Involved	Experienced software professionals	Graphics designer, web developer, less experienced software engineer
Cost for development	Costly approach	Inexpensive method

Table. 1: Characteristics Comparison

It is difficult for estimators to adapt and put their existing processes, metrics, and models to work operationally. And this is due to the differences between the agile and traditional projects. There are some challenges involved in web estimation. The table discussed below also identifies the approaches that are currently used to develop estimates for traditional software projects. These traditional approaches do not address the challenges that are faced with Agile projects. Estimating size and duration are the two major challenges. The differences between the challenges in traditional and agile methods are listed below as:

Characteristics	Traditional Approach	Agile challenges
Estimating Process	Most use analogy supplemented by lessons learned from past experience	Ad hoc using inputs from the developers
Model	Measurements from past	Measurements from past

Calibration	projects are used to calibrate models to improve accuracy	projects are used to identify estimating knowledge base
Size Estimation	Because systems are built to requirements, SLOC or function points are used. Separate models are used for COTS and reused software	Applications are built using a variety of web based object (html, applets, components, etc.)
What if analysis	Estimating models are used to perform quantitative what if and risk analysis	What is and risk analysis is mostly qualitative because models don't exists
Effort Estimation	Effort is estimated via regression formulas modified by cost drivers	Effort is estimated via analogy using job costing practices and experience as the guide. Little history is available

Table. 2: Comparison on the basis of challenges

CONCLUSION

On the basis of above discussion it is clear that the Agile method is a very useful methodology since it promotes communication over documentation which reduces knowledge of the system and therefore it can be adopted in modern software development process in order to replace the heavyweight development life cycle. It also provides complete code coverage and is able to identify the defects that testing process might have missed, as well as the root cause of the defect. To provide more customer satisfaction, to reduce bug rates, to shorten the development life cycle, and to accommodate changing business requirement agile software development methods were developed. In order to promote agile project development, developers should have good social and interpersonal skills instill in them. For this purpose, company should provide training to the developers. To take the feedback every effort should be taken to ensure whether all the requirements are being fulfilled or not. Only those tasks that add value to business processes supported by the system should be performed and lastly, processes and artifacts that do not add enduring value to the working software system should be discarded.

REFERENCES

[1] Szalvay, Victor. An Introduction to Agile Software Development. Danube Technologies Inc. 2004.
[2] IBM: Rational Unified Process: Best practices for software development teams (2003), http://www.ibm.com/developerworks/rational/library/253.html
[3] Dyba, Tore. Empirical studies of agile software development: A systematic review. 24 January 2008.
[4] Nikiforova, O., Nikulsins, V., Sukovskis, U.: Integration of MDA Framework into the Model of Traditional Software Development. In: Frontiers in Artificial Intelligence and Applications, Databases and Information Systems V, vol. 187, pp. 229–239. IOS Press, Amsterdam (2009).
[5] Systems Development Lifecycle: Objectives and Requirements. Bender RPT Inc. 2003.
[6] The Agile Manifesto is online at http://www.agilemanifesto.org/