ABHINAV
NATIONAL MONTHLY REFEREED JOURNAL OF RESEARCH IN SCIENCE & TECHNOLOGY
www.abhinavjournal.com

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC) ANALYTICAL COMPARISON AND SURVEY ON TRADITIONAL AND AGILE METHODOLOGY

Sujit Kumar Dora[1] and Pushkar Dubey[2]

[1]Programmer, Computer Science & Engineering, Padmashree Krutartha Acharya College of Engineering (PKACE), Bargarh, Odisha, India
Email : sujit_dora@rediffmail.com
[2]Lecturer, Management, Padmashree Krutartha Acharya College of Engineering (PKACE), Bargarh, Odisha, India
Email : pushkardubey@rediffmail.com

ABSTRACT

This research concerned with the software management processes that examine the area of software development through the development models, which are known as software development life cycle. Software Development Life Cycle (SDLC) methodologies are mechanisms to assure that software meet established requirements. These methodologies impose various degrees of discipline to the software development process with the goal of making the process more efficient and predictable. This paper review & explain the heavyweight methodologies (Traditional SDLC) & Lightweight methodology (agile SDLC) and also draws a predictable comparison between both methodologies.

Keywords: Software development life cycle (SDLC), traditional method, agile method.

INTRODUCTION

The process of building computer software and information systems has been always dictated by different development methodologies. A software development methodology refers to the framework that is used to plan, manage, and control the process of developing an information system. Formally, a software development methodology is known as SDLC short for Software Development Life Cycle. This methodology or process is divided into some phases such as Requirement Analysis, Design, Coding, Testing, Installation and Maintenance.

REQUIREMENT ANALYSIS

Requirement analysis is the first phase of the Software Development Life Cycle. The main aim of this phase is to know the actual requirement of client's and to document them properly. The emphasis in requirement analysis is on identifying what is needed from the system. It is one of the most essential phases in Software Development Life Cycle. The

output of requirement analysis is Software Requirement Specification (SRS); it is a complete and comprehensive description of the behavior of the software to be developed.

Design

It is the most creative phase in Software Development Life Cycle. The goal of this phase is to transform the requirement specification into a structure or plan. It is the process of planning and problem solving for a software solution. It implicates software developers and designers to define the plan for a solution. The output of this phase is Software Design Document (SDD).

Coding

In this phase Software Design Document (SDD) is converted into code by using some programming language. It is the logical phase of the Software Development Life Cycle. The output of this phase is program code.

Testing

Just after coding phase, testing is carried out to know the outcome of application. Testing is made to know the actual result and the expected result. This is most important and powerful phase. Effective testing will provide high quality software products, lower maintenance costs, and more accurate and reliable results.

Maintenance

This is the final stage of SDLC, where the software that is being developed distributed to end users who are responsible for maintaining and using it for proper operations.

SDLC Models

There is various software development approaches defined and designed which are used during development process of software, these approaches are also referred as "Software Development Process Models". Each process model follows a particular life cycle in order to ensure success in process of software development. Some of the SDLC Models are:

- Waterfall Model
- Spiral Model
- Iterative Model
- Incremental Model
- Prototyping Model
- V -Shaped Model
- RAD Model

Waterfall Model

The Waterfall model is the classical model of software engineering [1] also it is referred as a linear-sequential life cycle model. It defines some basic tasks, which are carried out in sequence: requirements definition, Architecture design, detailed design, implementation,

component verification, integration verification and requirements validation. In a waterfall model, each phase must be completed in its entirety before the next phase can begin. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. The waterfall model serves as a baseline for many other lifecycle models. [1][2]

Spiral Model

The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirement is gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. Requirements are gathered during the planning phase. In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. Software is produced in the engineering phase, along with testing at the end of the phase. The evaluation phase allows the customer to evaluate the output of the project to data before the project continues to the next spiral. [3]

Iterative Model

Iterative model is the model provides a new method of developing systems which could provide faster results, require less up-front information, and offer greater flexibility. With Iterative Development, the project is divided into small parts. This allows the development team to demonstrate results earlier on in the process and obtain valuable feedback from system users. Often, each iteration is actually a mini-Waterfall process with the feedback from one phase providing vital information for the design of the next phase. In iterative model we are building and improving the product step by step, we can track the defects at early stages. This avoids the downward flow of the defects. In iterative model we can get the reliable user feedback.

Incremental Model

The incremental model divides the product into builds, where sections of the project are created and tested separately. This approach will likely find errors in user requirements quickly, since user feedback is solicited for each stage and because code is tested sooner after it's written. In incremental models, as in sequential models, the overall requirements of the final system or product are known at the start of the development. In incremental models however a limited set of requirements is allocated to each increment and with each successive (internal) release more requirements are addressed until the final (external) release satisfies all requirements. [2]

Prototyping Model

A prototype is a working model that is functionally equivalent to a component of the product.[4] A prototype is a usable system or system component that is built quickly and at a lesser cost, and with the intension of being modifying or replacing it by full-scale and fully operational system.[5] Prototyping allows the users to see and interact with a prototype allowing them to provide better and more complete feedback and specifications.

V-Shaped Model

V-Shaped life cycle execute the number of processes in a sequential path like the water fall model. In this model each phase must be completed before the next phase begins. When each of the phases preceding implementation, before any coding is done the testing procedures are developed early in the life cycle. The requirement of this life cycle model is just like the waterfall model. A system test plan is created, before development is started. The focuses of the test plan is to meeting the functionality specified in the requirements. The high-level design phase focuses on system architecture and design and the low-level design phase is where the actual software components are designed, and unit tests are created in this phase. [1]

RAD Model

RAD model is Rapid Application Development model. It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

Traditional SDLC (Heavy Weight Methodology)

Traditional 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. [6]Due to these heavy aspects, this methodology became to be known as heavyweight. These methodologies are based on a sequential series of steps, such as requirements definition, solution building, testing and deployment. Heavyweight methodologies require defining and documenting a stable set of requirements at the beginning of a project. Software methodologies like Waterfall method, V-Model and RUP are called traditional software development methodologies and these are classified into the heavyweight methodologies.

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 drawings focusing on the need of the system and how to resolve those needs efficiently.

A main process in heavyweight methodologies is the big design upfront (BDUF) process, in which a belief that it is possible to gather all of a customer's requirements, upfront, prior to writing any code. Again this approach is a success in engineering disciplines which makes it attractive to the software industry. To gather all the requirements, get a sign off from the customer and then order the procedures to limit and control all changes does give the project a limit of predictability.

Characteristic of Traditional Methodology

The characteristic of traditional software development method based on four steps. The first step is to set up the requirements for the project and determine the length of time it will take to implement the various phases of development while trying to predict any problems that may arise in the project. The next step moves into the design and architectural planning phase where a technical infrastructure is produced in the form of diagrams or models. Once

the team is satisfied with the architectural and design plan, the project moves into the development phase where code is produced until the specific goals are reached. The testing phase often overlaps with the development phase to ensure issues are addressed early on. Once the project nears completion and the developers are close to meeting the project requirements, the customer will become part of the testing and feedback cycle and the project was delivered after the customer satisfy with it. [7]

The essential nature of the traditional software development life cycle is as follows:

- The goals are to thoroughly understand users' needs, craft a solid design, develop software flawlessly, and implement a functional system that satisfies user needs.
- There is a heavy emphasis on thorough planning to deal with risks.
- Such an approach assumes that problems are well defined and that an optimum solution can be arrived at by extensive, up-front planning.
- It also assumes that the processes are predictable and can be optimized and made repeatable.
- It is also based on the assumption that processes can be adequately measured and that sources of variations can be identified and controlled during the development life cycle.
- In summary, the traditional software development life cycle is highly process-centric.

Agile SDLC (Light Weight Methodology)

Agile Software Development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self organizing, cross functional teams.[8] It promotes the adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle. Software creation can be a complicated process if not done correctly. The key to a successful software project is communication, flexibility and good analysis. That is why agile approach is adopted for software development. It means the flexibility to adapt when things change.

Agile development methodology is a conceptual framework for undertaking any software engineering projects. There are a number of agile software development methods but the most popular agile methods are Extreme Programming (XP), crystal methods, Scrum, Feature Driven Development.

Extreme Programming (Xp)

XP is a more radical agile methodology, focusing on the software development process and addressing the analysis, development and test phases with novel approaches aimed at making a substantial difference to the quality of the end product.

Crystal Methods

Crystal Methods were developed to address the variability of the environment and the specific characteristics of the project. The Crystal family of lightweight SDLC methodologies is the creation of Alistair Cockburn [10]. Crystal is comprised of more than one methodology because of Cockburn's belief that differing project types require differing methodologies. Project types are classified along two lines: the number of people on the development team and the amount of risk . Crystal methodologies are divided into colors coded bands. "Clear" Crystal is the smallest and lightest. "Yellow", "Orange", "Red", "Maroon", "Blue", and "Violet" follow for use with larger groups using more complex methodologies. [9]

Scrum

Scrum is a lightweight management framework with broad applicability for managing and controlling iterative and incremental projects of all types. Scrum is becoming increasingly popular in the software community due to its simplicity and proven productivity. This method concentrates particularly on how to manage task within a team based development environment.

Feature Driven Development

Feature Driven Development (FDD) is a model-driven short-iteration software development process .The FDD process starts by establishing an overall model shape. [9] This is followed by a series of two-week "design by feature, build by feature" iterations. FDD consists of five sequential phases such as develop an overall model, build a features list, plan by feature, and design by feature, and build by feature. The first three phases are done at the beginning of the project. The last two phases are the iterative part of the process which supports the agile development with quick adaptations to late changes in requirements and business needs. [6]

Characteristic of Agile Methodology

The following are the main characteristic of agile methodology which differentiates from Heavy weighted methodology.

- People oriented.
- Team Competence
- Adaptive.
- Conformance to Actual.
- Balancing Flexibility and Planning.
- Empirical Process.
- Decentralized Approach.
- Simplicity.
- Collaboration.
- Small Self-organizing teams.[8]

The keys for successful use of choosing and using agile methodologies are

- Agile methods are a subset of iterative and evolutionary methods. Iterations are short to provide for more timely feedback to the project team.
- Extreme Programming is based upon four values and 12 specific software development practices.
- The Crystal family of methodologies is customizable based upon the characteristics of the project and the team.
- Scrum mainly deals with project management principles. The methodology allows the team freedom to choose its specific development practices.
- FDD has the most thorough analysis and design practices.[10]

COMPARISON BETWEEN TRADITIONAL AND AGILE METHODOLOGY

Although agile method is based on iterative development as some of the traditional approaches, Agile and Traditional methodologies have key differences. Traditional approaches use planning as their control mechanism, while agile models use the feedback from the users as the main control mechanism. Agile can be called a people-centric approach than traditional methods. Agile model delivers a working version of the product very early compared to traditional methodologies so that the customer can realize some of the benefits early on. Testing cycle time of Agile is relatively short compared to traditional methods, because testing is done parallel to development. Most traditional models are very rigid and relatively less flexible than the Agile model. Because of all these advantages, Agile is preferred over the traditional methodologies at the moment.

Some of the key differences between the Traditional and Agile methodology are given below.

Table 1. Difference between Traditional and Agile Methodology

Traditional Methodology	Agile methodology
Requirements are clear at the inception of the project.	Requirements are not clear at the inception of the project.
Limited communication, more stress in documentation.	More communication allows developing a product over short period of time, lesser importance on the documentation.
Elaborate process should be followed.	Limited process involved.
Time consuming as they develop the complete product at a single cycle.	Iterative model allow to develop easily in a short span of time.
No scrums calls and stand up calls.	Scrums calls are stand up calls at a regular interval to track the progress and get feedback from the clients.

Time consuming to understand, develop, test and defect fixing.	Save a lot of time due to frequent communication with stack holders, iterative development and proper feedback.
Testing happens only after the completion of the development.	Testing team work in parallel with the development team which helps to find the defect as soon as possible.
Automation is not a usual practice.	Continuous automated testing which ensures better quality.

Source: Compiled from different journal articles

CONCLUSION

Software Development Life Cycle is a methodology that governs the entire development process. In this paper various software development life cycle models are studied such as waterfall, spiral, incremental, iterative, prototyping, v-shaped, AD models. The Waterfall model provides base for other development models. Modern SDLC are divided into two main categories, which are traditional SDLC and agile SDLC. In this chapter we presented an overview of the traditional and agile software development model and the characteristics of the projects that may be suited for the use of this model. Additionally, we provided overviews of four representative methodologies, XP, Crystal, Scrum and FDD. Also we have shown the comparison between both models where agile SDLC excels traditional SDLC.

REFERENCES

1. Khurana gourav and s gupta(2012) " Study & Comparison of Software Development Life Cycle Models" IJREAS, Vol. 2(2), 1514-1515.
2. Wallin Christina and Land R. "Software development Life Cycle models the basic type", 2.
3. Nabil Mohammed Ali Munassar Ali and Govardhan A (2010) "A Comparison between Five Models of Software Engineering" *International Journal of Computer Science*, Vol. 7(5), 98-100.
4. Tuteja Maneela and Dubey G.(2012) "A Research Study on Importance of Testing and Quality Assurance in Software development life Cycle (SDLC)Models" *International Journal of Soft Computing and Engineering*, Vol. 2(3), 251-252.
5. http://www.iscanotes.com(2011), 5-6 (accessed 11th July 2013)
6. Awad M.A. "A Comparison between and Traditional Software Development Methodology" 1-7
7. Yu Beng Leau, Loo W.K. and Wai Yip Tham.(2012) "SDLC Agile vs Traditional Approach", *International Conference on Information and Network Technology, vol. 37 , 162-165.*

8. Harish Rohil and Syan Manisha(2012) "Ananlysis of Agile and Traditional Approach for Software development", *International Journal of Latest Trends in Engineering and Technology* ,Vol. 1(4),1-3
9. T Bhuvaneswari and Prabaharan S.(2013) "A Survey on Software development life cycle model", *Journal of Computer Science and Information Technology*, Vol2 (5), 263-265.
10. Laurie Williams(2007) "A survey of Agile development Methodologies",215-218.