A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios

Article · October 2013

CITATIONS
28

READS
4,427

1 author:

Apoorva Mishra
Indian Institute of Information Technology, Pune
28 PUBLICATIONS **108** CITATIONS

SEE PROFILE

ISSN: 2321-7782 (Online)
Volume 1, Issue 5, October 2013
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios

Apoorva Mishra[1]
Assistant Professor
Computer Science & Engineering, C.S.I.T.
Durg - India

Deepty Dubey[2]
Assistant Professor
Computer Science & Engineering, C.S.I.T.
Durg - India

Abstract: ***There are various SDLC models widely used for developing software. SDLC models give a theoretical guide line regarding development of the software. SDLC models are very important for developing the software in a systematic manner such that it will be delivered within the time deadline and should also have proper quality. Employing proper SDLC allows the project managers to regulate whole development strategy of the software. Each SDLC has its advantages and disadvantages according to which we decide which model should be implemented under which conditions. For this we need to compare SDLC models. In this paper we will compare different famous life cycle models like-waterfall model, prototype, rapid application development, V-shaped model, spiral model & incremental model.***

Keywords: Software Development Life Cycle, Activities involved in SDLC models, Comparative analysis of models

I. INTRODUCTION

Software development life cycle (SDLC) is a method by which the software can be developed in a systematic manner and which will increase the probability of completing the software project within the time deadline and maintaining the quality of the software product as per the standard. The System Development Life Cycle framework provides a sequence of activities for system designers and developers to follow for developing software. It is often considered as a subset of system development life cycle. Any software development process is divided into several logical stages that allow a software development company to organize its work efficiently in order to build a software product of the required functionality within a specific time frame and budget. All software projects go through the phases of requirements gathering, business analysis, system design, implementation, and quality assurance testing [1]. Employing any SDLC model is often a matter of personal choice entirely dependent on the developer. Each SDLC has its strengths and weaknesses, and each SDLC may provide better functionalities in one situation than in another. Then the challenge is to decide which model should be selected to provide a particular set of functionalities under certain circumstances. One life cycle model theoretically may suite particular conditions and at the same time other model may also look fitting into the requirements but one should consider trade-off while deciding which model to choose [2]. Rodriguez-Martinez et al. [3] focused on lifecycle frameworks models and detailed software development life cycles process and reported the results of a comparative study of Software development life cycles that permits a plausible explanation of their evolution in terms of common, distinctive, and unique elements as well as of the specification rigor and agility attributes. Jovanovich D. et al. [4] presented basic principles and comparison of software development models. Davis A.M. et al. [5] provided a framework that can serve as a basis for analyzing the similarities and differences among alternate life-cycle models as a tool for software engineering researchers to help describe the probable impacts of a life-cycle model and as a means to help software practitioners decide on an appropriate life-cycle model to utilize on a particular project or in a particular application area. A Software Development Life Cycle Model is a set of activities together with an ordering relationship between activities performed in a manner that satisfies the ordering relationship that will produce desired product. A software

development life cycle model is broken down into distinct activities and specifies how these activities are organized in the entire software development effort. In response to traditional approaches to software development, new lightweight methodologies have appeared [6]. A high percentage of software development efforts have no process and might best be described as a chaotic "code and fix" activity. Light SDLC techniques are compromise between no process and too much process.

II. Phases Involved in SDLC Model

The phases that are generally present in each and every software development life cycle model are;

1. Understanding the problem (through requirements gathering).

2. Deciding a plan for a solution (Designing)

3. Coding the planned solution

4. Testing the actual program

5. Deployment & maintenance of the product.

For large systems, each activity can be extremely complex and methodologies and procedures are needed to perform it efficiently and correctly. Furthermore, each of the basic activities itself may be so large that it cannot be handled in single step and must be broken into smaller steps. For example, design of a large software system is always broken into multiple, distinct design phases, starting from a very high level design specifying only the components in the system to a detailed design where the logic of the components is specified.

The common phases of an SDLC can be represented by the following diagram

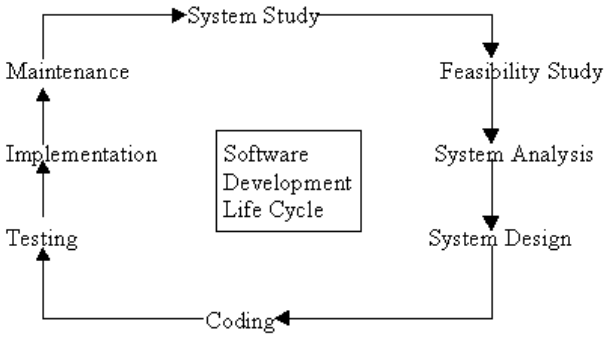

FIGURE 1.Phases of an SDLC model.

In addition to the activities performed during software development, some activities are performed after the main development is complete. There is often an installation phase, which is concerned with actually installing the system on the client's computer systems and then testing it. Maintenance is an activity that commences after the software is developed. Software needs to be maintained not because some of its Components "wear out" and need to be replaced, but because there are often some residual errors remaining in the system which must be removed later as they are discovered. Therefore, maintenance is unavoidable for software systems.

III. Common Software Development Life Cycle Models

WATERFALL MODEL

Waterfall model was proposed by Royce in 1970 which is a linear sequential software development life cycle (SDLC) model. The various phases followed are **r**equirements analysis, design, coding, testing and implementation in such a manner that the phase once over is not repeated again and the development does not move to next phase until and unless the previous phase is completely completed**.** Hence it is not very much useful when the project requirements are dynamic in nature.

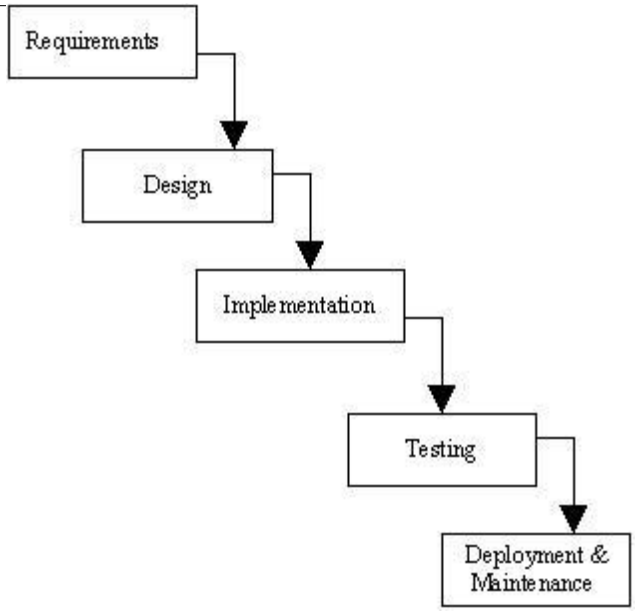

FIGURE 2.WATER FALL SDLC MODEL

SPIRAL MODEL

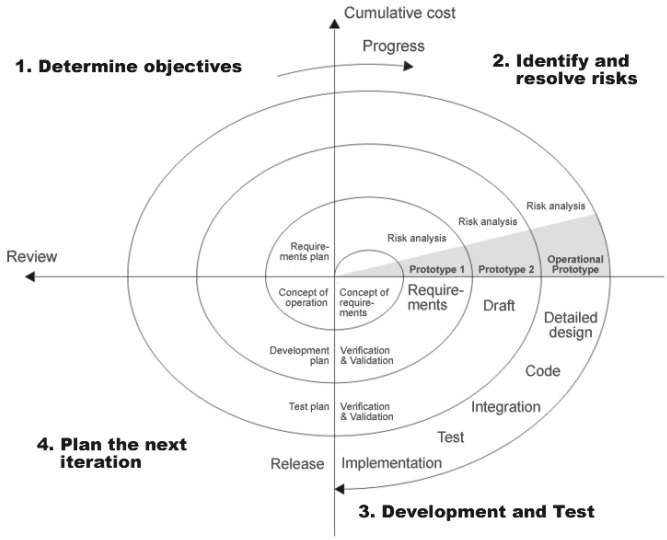

FIGURE 3. SPIRAL MODEL

In response to the weaknesses and failures of the Waterfall SDLC Model, many new models were developed that add some form of iteration to the software development process. In the Spiral SDLC Model as in figure 2 , the development team starts with a small set of requirements and goes through each development phase (except Installation and Maintenance) for those set of requirements [8]. Based on lesson learned from the initial iteration, the development team adds functionality for additional requirements in ever-increasing "spirals" until the application is ready for the Installation and Maintenance phase.

RAD MODEL

If requirements are well understood and project scope is constrained, the Rapid application development (RAD), figure 4 process enables a development team to create a "fully functional system" within very short time periods (e.g., 60 to 90 days).

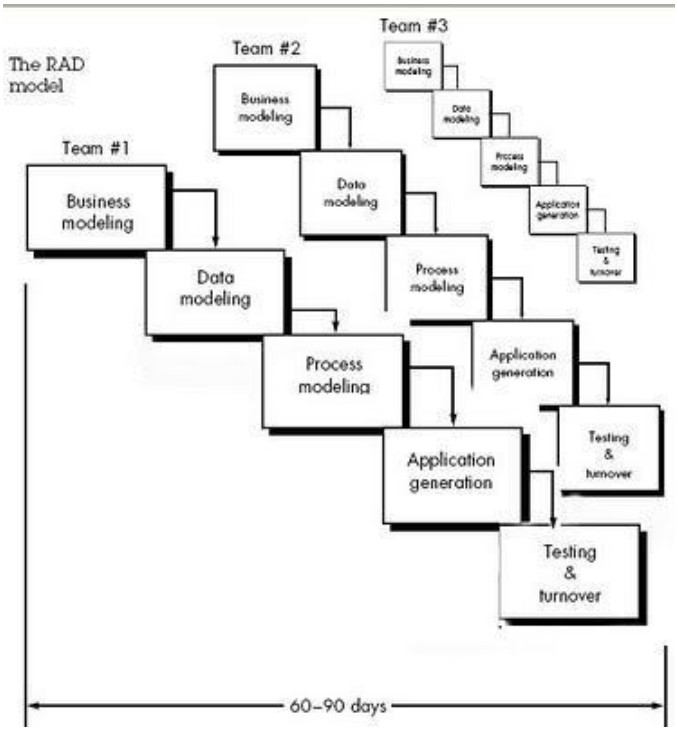

FIGURE 4: RAD MODEL

INCREMENTAL MODEL

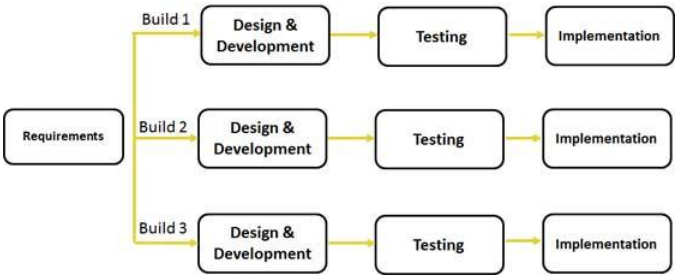

FIGURE 5. INCREMENTAL MODEL

V-SHAPED SDLC MODEL

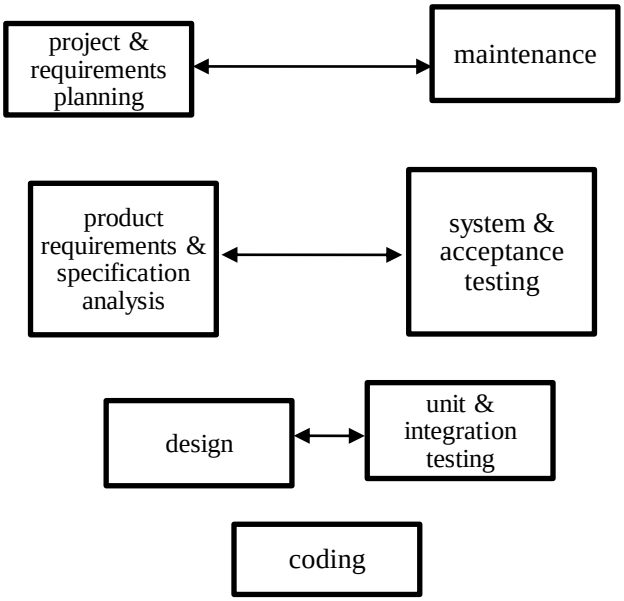

FIGURE 6 V-SHAPED MODEL

A variant of the Waterfall that emphasizes the verification and validation of the product, testing of the product is planned in parallel with a corresponding phase of development.

IV. Comparison of Different SDLC Models

As there are various models of software development life cycle, each has its own advantages and disadvantages depending upon which we have to decide, which model we should choose. For instance if the requirements are known before hand and well understood and we want full control over the project at all time, then we can use waterfall model. V-shaped Model has higher chance of success over the waterfall model due to the development of test plans during the life cycle. It works well for small projects where requirements are easily understood. Incremental model is at the heart of a cyclic software development process. It starts with an initial planning and ends with deployment with the cyclic interactions in between. Easier to test and debug during a smaller iteration. Easier to manage risk because risky pieces are identified and handled during its iteration. Spiral model is good for large and mission critical projects where high amount of risk analysis is required like launching of satellite. RAD Model is flexible and adaptable to changes as it incorporates short development cycles i.e. users see the RAD product quickly. It also involves user participation thereby increasing chances of early user community acceptance and realizes an overall reduction in project risk. The comparison of the different models is represented in the following table on the basis of certain features.

FEATURES	WATERFALL	V-SHAPED	INCREMENTAL	SPIRAL	RAD
Requirement specifications	Beginning	Beginning	Beginning	Beginning	Time boxed release
Cost	Low	Expensive	Low	Expensive	Low
Simplicity	Simple	Intermediate	Intermediate	Intermediate	Very Simple
Risk involvement	high	Low	Easily manageable	Low	Very low
Expertise	High	Medium	High	High	Medium
Flexibility to change	Difficult	Difficult	Easy	Easy	Easy
User involvement	Only at beginning	At the beginning	Intermediate	High	Only at the beginning
Flexibility	Rigid	Little flexible	Less flexible	flexible	High
Maintenance	Least	Least	Promotes maintainability	Typical	Easily maintained
Duration	Long	According to project size	Very long	Long	Short

Table 1(Comparison of Different SDLC Models)

V. Conclusion

There are many SDLC models such as, Waterfall, RAD, spiral, incremental, V-shaped etc. used in various organizations depending upon the conditions prevailing there. All these different software development models have their own advantages and disadvantages. In the Software Industry, the hybrid of all these methodologies is used i.e with some modification. In this paper we have compared the different software development life cycle models on the basis of certain features like- Requirement specifications, Risk involvement, User involvement, Cost etc. on the basis of these features for a particular software project one can decide which of these software development life cycle models should be chosen for that particular project. Selecting the correct life cycle model is extremely important in a software industry as the software has to be delivered within the time deadline & should also have the desired quality. This study will make the process of selecting the SDLC model easy& hence will prove to be very effective for software industry.

References

1. Klopper, R., Gruner, S., & Kourie, D. (2007), "Assessment of a framework to compare software development methodologies" Proceedings of the 2007 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries, 56-65. doi: 10.1145/1292491.1292498.

2. Roger Pressman, titled "Software Engineering - a practitioner's approach"

3. Laura C. Rodriguez Martinez, Manuel Mora ,Francisco,J. Alvarez, "A Descriptive/Comparative Study of the Evolution of Process Models of Software Development Life Cycles", Proceedings of the 2009 Mexican International Conference on Computer Science IEEE Computer Society Washington,DC, USA, 2009.

4. Jovanovich, D., Dogsa, T.,"Comparison of software development models," Proceedings of the 7th International Conference on, 11-13 June 2003, ConTEL 2003, pp. 587-592.

5. A. M. Davis, H. Bersoff, E. R. Comer, "A Strategy for Comparing Alternative Software Development Life Cycle Models", Journal IEEE Transactions on Software Engineering ,Vol. 14, Issue 10, 1988

6. Fowler, M. (2000), "Put Your Process on a Diet", Software Development".

7. Sanjana Taya, Shaveta Gupta, "Comparative Analysis of Software Development Life Cycle Models".

8. Vishwas Massey, K.J Satao, " Comparing Various SDLC Models And The New Proposed Model On The Basis Of Available Methodology".

AUTHOR(S) PROFILE

Apoorva Mishra, received the B.E (hons) degree in computer science and engineering from R.C.E.T, Bhilai in 2011, then worked at Tata Consultancy Services, Mumbai till May 2012.Now he is working as an assistant professor at C.S.I.T, Durg.

Deepty Dubey, received the B.E degree in computer science and engineering from S.S.C.E.T, Bhilai in 2005, then completed her M.Tech from R.C.E.T, Bhilai, in 2010.Now she is working as an assistant professor at C.S.I.T, Durg.

