I.J. Education and Management Engineering, **2018, 6, 59-68**
Published Online November 2018 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2018.06.06

Available online at http://www.mecs-press.net/ijeme

An Efficient Software Development Life cycle Model for Developing Software Project

Mr. Madhup Kumar [a], Dr Ekbal Rashid [b]

[a] *M.Tech Research Scholar, Jharkhand Rai University, Ranchi, Jharkhand*
[b] *Associate Professor, Deptt. Of Computer Science & Engineering Aurora Technological and Research Institute Uppal, Parvathapur, Hyderabad*
www.atri.edu.in

Received: 01 December 2017; Accepted: 06 August 2018; Published: 08 November 2018

Abstract

There are different life cycle models available for developing various types of software. Every Software Development Life Cycle (SDLC) model has some advantages and some limitations. In that case software developers decide which SDLC model is suitable for their product. Further, we need development of software in a systematic and disciplined manner. This is advantage of using a life cycle model. A life cycle model forms a common understanding of the activities among the software engineers and helps to develop software in a proper manner, so that time can be reduced. The objective of this paper is to compare all traditional or existing SDLC model with our Proposed SDLC model for development of software in effective and efficient manner.

Index Terms: Software Engineering, SDLC, Comparative Life cycle model.

1. Introduction

Computers have been used for commercial purpose for last 5 to 6 decades [9]. Software engineering is a useful method to solve large and more complex program in cost effective and efficient way with their past experience [1]. We can say software engineering is an engineering approach to develop software. Software engineering method based on error prevention, cost effective to prevent error from occurring than to correct them as and when they detected, well defined stages such as Software Requirement Specification (SRS), Designing, Coding, Testing Maintenance, and various design techniques used [2]. Previous method that was

* Corresponding author
E-mail address: madhup.kumar@bitmesra.ac.in, ekbalrashid2004@yahoo.com

exploratory method that is Based on error correction, error are detected only during the final project testing, in this method there are various limitation like hard to maintain product, break down when used to develop large product. A software life cycle is the sequence of identifiable stages or process that a software product undergoes during its life time [3]. A SDLC model is a descriptive and pictorial representation of software life cycle [4]. A life cycle model map the different activities performed on a software product from its inspection to retirement [5].Business organization follow steps-Business process, manufacturing industries-manufacturing process same as for software development use software process model [6] .The first life cycle of any software product is generally feasibility study, requirement analysis, Design, Coding, testing and maintenance [7]. A software/system life cycle model is a description of the series of activities carried out in a Software Engineering project, and the relative order of these activities [8].

2. Related Work

Any organization or company if they want to achieve the highest quality of software product then they should follow the complete software development life cycle model[11] and should be free from defects or errors, because customers or users are increasingly intolerant of the hit-and-miss approach that characterized software products [10]. The explicitly stated functional requirements can be derived from the requirements stated by the customer which are generally documented in some form. However, implicit requirements are requirements which are not stated explicitly but are intended [11] [12]. As we know large numbers of models have already been work out in the area of software development life cycle models. This research paper enlisted and presented an overview of various software development cycle models which has been given below.

2.1. Classical Waterfall Model

I. When the requirements are very well understand and fixed.
II. When Product definition is constant.
III. When Technology is understood.
IV. When there are no confusing requirements.
V. When sufficient resources with required expertise are available without restraint
VI. When project is short.

2.2. Iterative Waterfall Model

I. When the requirements are very well understand and fixed.
II. When Product definition is constant.
III. When project is big
IV. When Technology is understood

2.3. Prototyping Model

I. Needs a lot of interface with the client/customer.
II. System that have very high amount of interaction with end users, can go for Prototype model.
III. Interaction with end user to result in a useable system for designing good human computer interface systems.

2.4. Evolutionary Model/ Incremental Model

I. When the requirements of the system are clearly defined and understood.
II. Major requirements must be defined; however, some details can evolve with time.

III. There is a need to get a product to the market early.
IV. A new technology is being used
V. Resources with needed skill set are not available
VI. There are some high risk features and goals.

2.5. Spiral Model

I. When costs and risk estimation is important
II. For medium to high-risk projects.
III. For Long-term project assurance risky because of possible changes to economic priorities
IV. When Users are uncertain of their needs
V. When Requirements are complex.
VI. New product line.
VII. Considerable changes are expected.

2.6. V- Model

I. This model should be used for small to medium sized projects.
II. When requirements are clearly defined and set.
III. Chosen when sufficient technical resources are available with needed technical expertise.
IV. Confidence of customer is required for choosing this model approach.
V. No prototypes are produced,
VI. There is a very high risk involved in meeting customer expectations.

2.7. RAD Model

I. RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
II. It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
III. RAD should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (few months).

3. Drawbacks/limitations of Different Life Cycle Models

In this section we have given emphasis on many limitations or drawbacks found in existing SDLC models. It can be seen in table 1.

Table 1.

Classical Waterfall Model	• Iteration not possible. • Late delivery of software. • High risk and uncertainty. • Not for complex and object-oriented projects. • Not for long and ongoing projects. • Requirement changing not possible
Iterative Waterfall Model	• Iterations possible but may cause confusion as the project proceeds. • In this model we freeze software and hardware. But not advisable especially in long-term projects. • Requirement gathering tough Task. • Change in any previous stage can cause big problem (Dependency). • Iteration can be a costly.

Prototyping Model	• Leads to implementing and then repairing way of building systems. • Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans. • Incomplete application.
Evolutionary Model/ Incremental model	• Incomplete planning and design phase. • Difficult to broken down and built incrementally. • Total cost is higher than waterfall
Spiral Model	• Costly model to use. • Risk analysis requires highly specific expertise. • Project's success is highly dependent on the risk analysis phase. • Doesn't work well for smaller projects.
V- model	• Very rigid and least flexible. • Software is developed during the implementation phase, so no early prototypes of the software are produced. • If any changes happen in midway, then the test documents along with requirement documents has to be updated.
RAD model	• Depends on strong team and individual performances for identifying business requirements. • Only system that can be modularized can be built using RAD • Requires highly skilled developers/designers. • High dependency on modeling skills • Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

4. The New proposed SDLC Model

We have proposed a new SDLC model with various features which makes effective and efficient. Detailed information regarding new SDLC phases which is given below such as Customer Requirement (CR) to Maintenance i.e., 1 to 9. See figure 1.

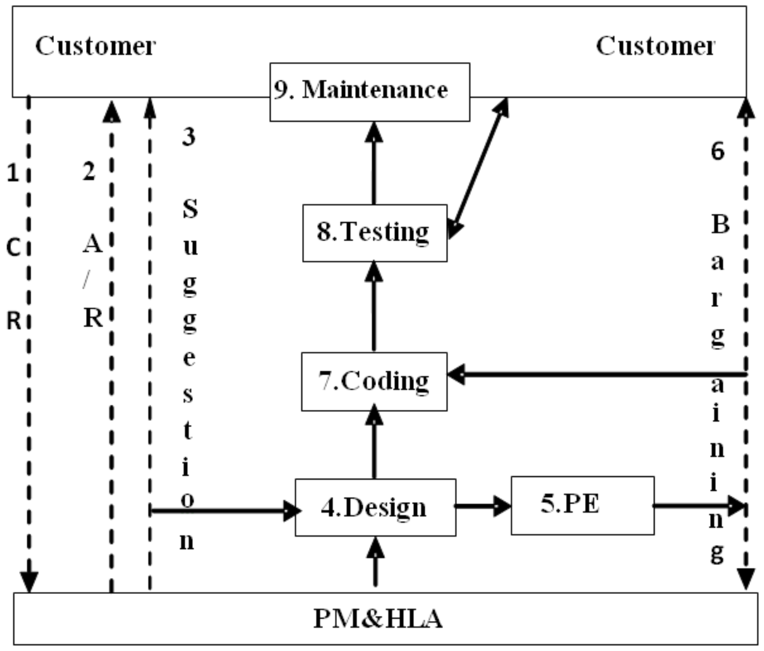

Fig.1. New Proposed SDLC Model

1. **Customer Requirement**: As it is evident from various studies in the field of software development, customer's requirement is very crucial and basic essential element for the development phase. There may be a drastic software failure or delay due to wrong requirement analysis. The whole process of any software is dependent on requirement gather form user. There are different method form which we can collect requirement from customer or user. The objective of software engineering is that developed software should meet or fulfill the user requirement. The output of this phase is to prepare a document called customer requirement document this is same as software requirement specifications document. The project manger (PM) prepares this document. In this document there are all functional and non functional requirements are prepared. And after preparing this document PM will also send to high level authority (HLA) of the company.
2. **Approval/Rejection (A/R):** In this life cycle model we introduce new process that is involvement of higher authority or management team in SDLC. As we know how many project fail or delay due to bypassing or not involvement of higher authority or higher management team members of company. The project manager (PM) controls the project during development and has overall responsibility of develop the project in systematic and discipline manner. In this lifecycle model introduce new phase that is Approval /Rejection phase for project .In this phase the project manager after getting customer requirements he /she will send a prepare customer requirement Document (CRD) to higher authority of company or higher management to the company for approval or rejection. After getting approval or rejection from the H.L.A project manager can proceed. The H.L.A can give suggestion to PM also regarding project.
3. **Suggestion from PM:** The next new introduce phase of life cycle model is suggestion form PM .The project manager has full control over project during its life time. After receiving approval /rejection or suggestion from H.L.A .The project manager can ensure or reject project to customer and can also give some suggestions to customer about project. The PM is a person who is technically sound and know more about latest technology .Its depend on customer he/she can take suggestion or not
4. **Design:** software design is an important phase of any SDL model for software development. In new proposed life cycle model design phase is also an important phase for developing software because from this phase the PM will estimate the size of the software and that is also useful for other estimation like time, cost, schedule etc. In this phase we shall use unified modeling language tools for designing class and use case diagram and after all necessary diagram from which we can estimate the size, time, cost, schedule etc. of the project.
5. **Project Estimation:** The main objective of this new developed software life cycle model is to estimate accurate size of project from design phase .Due to underestimating size of the product may cause project failure or delay. There are various size estimation technique are available like LOC, Function point and feature point estimation technique but have also some drawback of each estimation technique. In all prior SDLC model have same problem Estimating size of the product at beginning of product but this is not in nature how can we estimate or judge size of the product form requirement gathering phase. After this we take as this estimated size we calculate Cost, Effort, scheduling etc. this is really become unnatural. These are one of the main causes of project failure or delay. In this new proposed life cycle after designing phase expert (experience Member of team or good programmer) can calculate the size of the product. This method is more genuine form older one and minimum chance to wrong estimate size of product.
6. **Bargaining within PM and customer:** The next and one of the important new introduce phase of this life cycle model is bargaining within PM and Customer. In this phase the PM and customer will finalize about cost and time of the project i.e. how much it will take time to deliver the project in what cost? In this phase of life cycle model the project manager will estimate total cost and time to deliver the project with the help of well defined estimation technique and the customer can have bargaining power for time and cost. In all previous SDLC models the customer were annoying form time and cost .there is not any involvement of customer in SDLC model. In this life cycle model overcome this limitation of all prior SDLC model.

7. Coding: The process of writing an algorithm using the correct syntax of any programming language called coding. A coding should be brief and concise, clear-cut and easily translated into executable code.

The steps for developing a program are therefore:

I. Understanding the requirement.
II. Produce the design document from CRS document.
III. Translate the design into program code using suitable programming language.

Features of good programming:

I. According to specification:
II. In time:
III. Flexible:
IV. Easily modifiable:
V. Less execution time and space required:
VI. Minimum testing cost:
VII. Easy to maintain:

These are some Coding guidelines:

I. Meaningful Variables and functions name:
II. Meaningful comment:
III. Avoid go-to like statement:
IV. Minimum coupling and high cohesion:
V. Reusability:

8. Testing: Testing is traditionally used mean testing of program code. When the product is tested with appropriate and realistic tests that reflect typical usage patterns by the intended users, the chances of the product satisfying the customer's requirement is much higher. While testing does not guarantee zero defects, effective testing certainly increases the chances of customer acceptance of the software. Testing is done by a set of people with in a software product (or service) organization whose goal and agreement is to uncover the defects in the product before it reaches the customer. The process of analyzing a software item to detect difference between existing and required condition (i.e., bugs) And to evaluate the feature of the software items.

8.1. Levels of Testing:

I. Unit Testing
II. Integration testing
III. System testing
IV. IV. Functional Testing
V. Performance testing
VI. Acceptance testing
VII. Alpha testing
VIII. Beta testing

8.2. Types of Testing

- I. Non-execution based Testing
- II. Execution based Testing

8.2.1. Black Box Testing

- I. Requirement based Testing
- II. Positive and negative Testing
- III. Boundary value analysis
- IV. Equivalence partitioning
- V. Graph Based Testing
- VI. Compatibility Testing

8.2.2. White box Testing

- I. Basic path Testing.
- II. Structural Testing:
- III. Statement coverage Testing
- IV. Branch coverage Testing
- V. Condition coverage Testing
- VI. Loop coverage Testing
- VII. Path coverage Testing
- VIII. Domain Value Testing
- IX. Data-Flow Testing
- X. logic-based Testing
- XI. Fault -based Testing

In general, non-execution based testing is less expensive than execution based testing because:

- I. Execution based testing can be extremely time consuming.
- II. Reviews lead to detection of fault earlier in life cycle.

9. Maintenance: The term software maintenance denotes any changes made to a software product after it has been delivered to the customer. The maintenance phase of software life cycle is the period in which a software product performs useful task. The development cycle for a software product span our one or two years, while maintenance phase spans for five to ten years.

9.1. Need for Software Maintenance

- I. Correct programs.
- II. Enhance software products.
- III. Adaptation of product to new environments.

9.2. Types of Maintenance

- I. Perfective Maintenance.
- II. Adaptive Maintenance.

III. Corrective Maintenance.

5. Algorithmic form of proposed SDLC Model

Our proposed SDLC model works like algorithm which is easy and understandable.

Step 1: Request from customer.
Step 2: Request accepted go to step 3, else stop.
Step 3: Suggestions from PM and HMA if required.
Step 4: Prepare design for software development.
Step 5: Estimate size, cost, and time form Step 4.
Step 6: Result from Step 5, Bargaining with customer.
Step 7: Coding.
Step 8: Testing.
Step 9: Maintenance.

6. Comparative Study

In this section we have done comparison between new proposed software development life cycle with existing life cycle and it can be seen in table 2.

Table 2. Comparative Study of New Purposed Software Life cycle model with other Models.

Classical Waterfall Model	**New proposed SDLC Model**
1.Iteration problem	1.No need of Iteration
2.High amounts of risk and uncertainty	2. Directly communication with PM
3. Not a good model for complex and object-oriented projects.	3. Good model for any type of project.
4. Not suitable for the requirements changing projects.	4. Requirement gathers in different cycle and use requirement gathering technique.
Iterative Waterfall Model	**New proposed SDLC Model**
1. Requirement gathering tough Task.	1.Requirement gathering technique used
2. Change in any previous stage can cause big problem (Dependency).	2. No chance to major changes due to user involvement.
3.Iteration can be a costly	3.No need of iteration
Evolutionary Model/ Incremental model	**New proposed SDLC Model**
1. Incomplete planning and design phase.	1. There is planning and design phase.
2. Difficult to broken down and built incrementally.	2. Not needed.
Prototype Model	**New proposed SDLC Model**
1.Incomplete application	1.Complete application
Spiral Model	**New proposed SDLC Model**
1. Project’s success is highly dependent on the risk analysis phase.	1. All phases are important
2.Doesn’t work well for smaller projects	2.Work for any type of projects

7. Conclusions & Future Scope

In this research work we compare all the traditional SDLC models with each other every model have some advantages and some limitation and we also proposed a new model for software development with different life cycle. In this research work we also compare a new model with other traditional SDLC model. Some of the limitations of different SDLC model can be overcome by this new proposed model but some rest limitations of different SDLC model are not been overcome like late delivery and cost. In our future work we shall improve this new model and try to add more features in this model.

References

[1] Ian Sommerville, "Software Engineering", Addison Wesley, 7th edition, 2004.

[2] Rajib mall, “Software Engineering princilple” PHI.

[3] Nabil Mohammed Ali Munassar and A. Govardhan: (2010) A Comparison Between Five Models Of Software Engineering, IJCSI, Vol. 7, Issue 5, ISSN (Online): 1694-0814, pp. 94-101.

[4] Rajendra Ganpatrao Sabale, Dr. A.R. Dani. (2012): Comparative Study of Prototype Model For Software Engineering with System Development Life Cycle, (IOSRJEN), ISSN: 2250-3021, PP. 21-24.

[5] Adel Alshamrani and Abdullah Bahattab. (2015): A Comparison between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model, IJCSI, 2015ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784, pp. 106-111.

[6] Vanshika Rastogi. (2015): Software Development Life Cycle Models-Comparison, Consequences, (IJCSIT), pp. 168-172.

[7] Aleš ŽIVKOVICˇ, Marjan HERICˇKO, Boštjan BRUMEN, Simon BELOGLAVEC, Ivan ROZMAN.(2005): The Impact of Details in the Class Diagram onsoftware Size Estimation, INFORMATICA, 2005, Vol. 16, No. 2, pp. 295–312.

[8] Vishwas Massey. (2012): Comparing Various SDLC Models And The New Proposed Model On The Basis Of Available Methodology, IJARCSSE, ISSN: 2277 128X, pp. 170-177.

[9] T Bhuvaneswari1, S Prabaharan. (2013): A Survey on Software Development Life Cycle Models, IJCSMC, ISSN 2320–088X, pp. 262-267.

[10] Mark Sherriff and Laurie Williams Defect Density Estimation through Verification and Validation North Carolina State University Raleigh, NC, USA 27695 {mssherri, lawilli3}@ncsu.edu).

[11] Waman S Jawadekar Software Engineering principles and practice, computer engineering series, third reprint 2006 by Tata McGraw-Hill publishing company limited.

[12] Stephen H. Kan Metrics and Models in Software Quality Engineering, second edition by Addison Wesley.

Authors’ Profiles

Mr. Madhup Kumar is pursuing M.Tech by Research in Computer Science from Jharkhand Rai University, Ranchi under the guidance of Dr. Ekbal Rashid. He has completed his MCA from BIT Mesra Ranchi. He is working as Asssociate Lecturer in department of CSE, BIT Mesra, Patna Campus,. His research area is Machine Learning(CBR),Software Engineering and Networking.

Ekbal Rashid, he is working as an Associate Professor in the Deptt. of Computer Science and Engineering with Aurora's Technological and Research Institute, Uppal, Hyderabad. He received his M.Tech in Computer Science in year 2009 from BIT, Mesra and his Ph.D in Computer Science and Engineering in May 2015 from Siksha 'O' Anusandhan University (Deemed University), Bhubaneswar, Orissa. Dr. Rashid has more than 30 International and National publications in journals and conference proceedings of repute including Springer, IEEE, Inderscience. His research area is software engineering, machine learning, data mining and artificial intelligence. Dr. Rashid has written two books on the topic of "Analogy-Based Software Cost Estimation" published by Advance Academic Publisher, India and "Enhancing Software Fault Prediction with Machine Learning: Emerging Research and Opportunities" that has been published by IGI Global, USA sep 2017. He has more than fifteen years of teaching experience in various reputed Institutes/Universities across the country.

How to cite this paper: Madhup Kumar, Ekbal Rashid,"An Efficient Software Development Life cycle Model for Developing Software Project", International Journal of Education and Management Engineering(IJEME), Vol.8, No.6, pp.59-68, 2018.DOI: 10.5815/ijeme.2018.06.06