IJITKM Volume 7 • Number 1 • December 2013 pp. 117-120 (ISSN 0973-4414)

A Complete Model for Software Development Life Cycle

Bhupendra Bhadana, Meenal Borkar
Department Computer Science and Engineering, B.S Anangpuria Institute of Technology& Management
Satyug Darshan Institute of Technology & Management Faridabad, India
bhupendra.bhadana@faculty.anangpuria.com, meenal. borkar@gmail.com

Abstract-
This paper focuses and examines a new model for software development life cycle (SDLC model) used in the software development. But, if we discuss this point further, it become quickly apparent that many developers are still practicing and analyzing, *' loop until you get it right'* method of development, while these method takes a lot of time and cost in development. As we know, in the software development the major issues are concerned with the testing and maintenance phase. According to an analysis 67% of software development cost is due to maintenance and 15% due to testing. So, these areas are of crucial importance in software industry. This model focuses on these areas to reduce the software cost. This model reduces the maintenance cost and testing cost using the documentation of each module which will help in system and integration testing and the maintenance.

I.INTRODUCTION

At present time IT sector is improving very fast and its need is rapidly increasing in all other sectors also. At the same time there is a great demand of good software development skills and platform. But, if we discuss this point further, it become quickly apparent that many developers are still practicing and analyzing " loop until you get it right" method of development, but this method takes a lot of time and cost in development. As observed from past experience, the software based on these models takes many months to complete a cycle. These models give a good product in end but problem is that these take a lot of time due to which when the product completes its technology become old and in terms of maintenance, it takes more time and cost. This problem is realm for software industry.

However, there existed a distinct difference in the level of enthusiasm between those of the traditional SDLC and the new complete model proponents. Given this difference, for those in attendance the question became: Which is best and why? Which should we practice, and why? And what about Design for Complete model? Is this a new replacement for the SDLC, or simply something else? Further discussion revealed that, of all of the important topics such as documentation, adoption, user involvement, financial management, management control, outsourcing, quality control, risk management, time management and scope, topics stood out as central to Complete: cost, time and maintenance. These topics are indeed mirrored in SDLC history. However, they have been represented with various levels of emphasis and often with limited success.Using these topics, cost and time and maintenance, the following looks first at the discordant array of SDLCs and then reviews complete model. Finally comparisons will be drawn and summary given.

II.THE SYSTEM DEVELOPMENT LIFE CYCLE (SDLC): A REVIEW

A. Waterfall Model

The waterfall model is the oldest and most widely used model in the field of software development. The waterfall model is generally attributed to Royce (1970). The model encourages the product development team to specify what the software is supposed to do (gather & define requirements) before implementing the system. Product development is split into multiple sequential steps (design, implement and test) with intermediate deliverables leading to a final product.

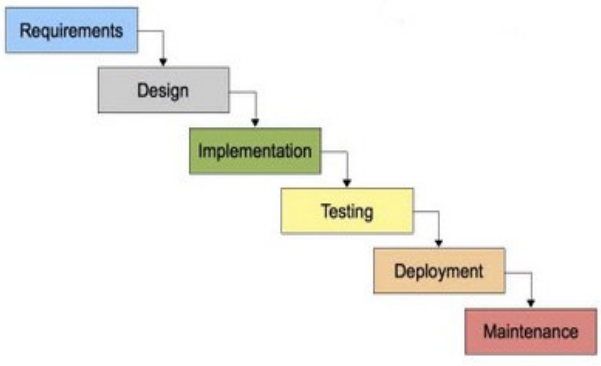

Fig. 1 Waterfall Model

Advantage of Waterfall Model

- Needless to mention, it is a linear model and of course, linear models are the simplest to be implemented.
- The amount of resources required to implement this model is very minimal.
- One great advantage of the waterfall model is that documentation is produced at every stage of the waterfall model development. This makes the understanding of the product designing procedure simpler.
- After every major stage of software coding, testing is done to check the correct running of the code.

Disadvantages of Waterfall Model

The question that must be bothering you now is that with so many advantages at hand, what could be the possible disadvantages of the waterfall model. Well, there are some disadvantages of this widely accepted model too. Let us look at a few of them.

- Ironically, the biggest disadvantage of the waterfall model is one of its greatest advantages. You cannot go back, if the design phase has gone wrong, things can get very complicated in the implementation phase.
- Many a times, it happens that the client is not very clear of what he exactly wants from the software. Any changes that he mentions in between may cause a lot of confusion.
- Small changes or errors that arise in the completed software may cause a lot of problem.
- The greatest disadvantage of the waterfall model is that until the final stage of the development cycle is complete, a working model of the software does not lie in the hands of the client. Thus, he is hardly in a position to mention if what has been designed is exactly what he had asked for.

B. Incremental Model*:*

Incremental model is an evolution of waterfall model. The product is designed, implemented, integrated and tested as a series of incremental builds. It is a popular model for software evolution used in many commercial software companies and system vendor. Incremental software development model may be applicable to projects where Software Requirements are well defined, but realization may be delayed and basic software functionality is required early.

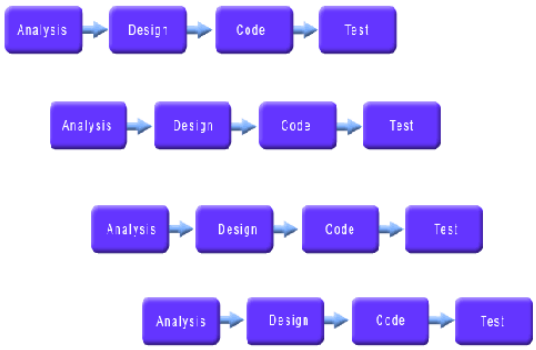

Fig. 2 Incremental Model

Advantage of Incremental Model:

- Generates working software quickly and early during the software life cycle.
- More flexible - less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Easier to manage risk because risky pieces are identified and handled during its iteration.

Disadvantage of Incremental Model:

- Each phase of an iteration is rigid and do not overlap each other.
- Problems may arise pertaining to system architecture because not all requirements are gathered up on front for the entire software life cycle.

C. Spiral Model*:*

The Spiral Model is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the Linear Sequential Model. Using the Spiral Model the software is developed in a series of incremental releases. Unlike the Iteration Model where in the first product is a core product, in the Spiral Model the early iterations could result in a paper model or a prototype. However, during later iterations more complex functionalities could be added. Spiral Model combines the iterative nature of prototyping with the controlled and systematic aspects of the Waterfall Model, therein providing the potential for rapid development of incremental versions of the software. A Spiral Model is divided into a number of framework activities, also called task regions.

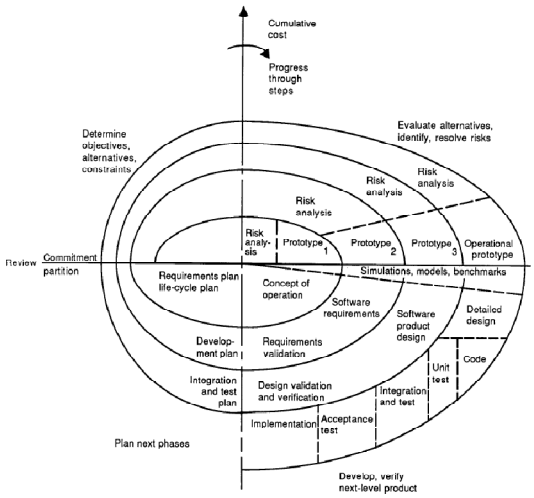

Fig. 3 Spiral Model

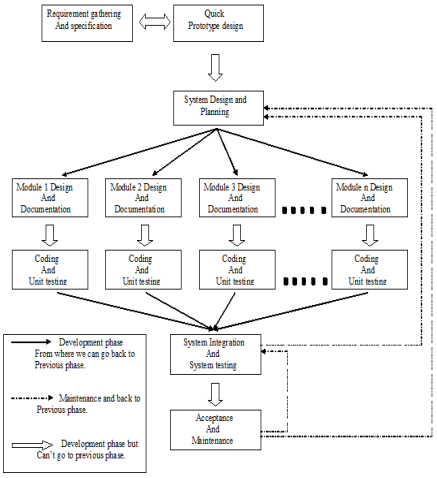

Fig. 4 Complete Model

Advantages of the Spiral Model

- Realistic approach to the development because the software evolves as the process progresses. In addition, the developer and the client better understand and react to risks at each evolutionary level.
- The model uses prototyping as a risk reduction mechanism and allows for the development of prototypes at any stage of the evolutionary development.
- It maintains a systematic stepwise approach, like the classic waterfall model, and also incorporates into it an iterative framework that more reflect the real world.

Disadvantages of the Spiral Model

- One should possess considerable risk-assessment expertise
- It has not been employed as much proven models (e.g. the Waterfall Model) and hence may prove difficult to 'sell' to the client.
- These models give a good product in end but problem is that these takes a lot of time due to that when product complete technology become old one and for maintenance it takes more time and cost.

III.**COMPLETE MODEL: A Discussion**

Now, after these traditional SDLCs here comes new SDLC Complete. In this SDLC, we put the effort to mix all possible advantages from all the SDLCs on the basis of areas such as cost, time and maintenance. This model tries to remove disadvantages of the all SDLCs. A detailed design of model is given here:

A. Design:

As we can see in the figure of this SDLC, it includes the following phases in development:

1). Requirement gathering and specification

Like all other SDLCs it also accepts the detailed requirement of user to develop a software according to his needs. It also take the requirement from user on the behalf of quick design. This process loops untill user have not specified all his needs.

2). Quick Prototype design

Like protype model in this we have include an protype phase but unlike in prototype model it does not spend as much time as the prototype model takes in software prototype. It only contain the quick design of software to take all the requirement of product from user.

3). System design

Here our detailed design of the product comes. In this phase developers create an detailed design of the software product and planning of the developing of that software product are included. In this phase, to develop the product with time and cost, system is boken down into independent modules and these modules are further processed.

4). Module Documentation and designing

It contains a separate SRS for each module. In which each module is discussed in terams of system and a detailed document of each module is generated. This document part help further in coding, testings and maitenance. By these documents, we can go easily and can interact to each module in testing and maintenance. This documentation clears the module requirement in system and give a good way to develop system.

5). Coding and unit testing

After the finishing of the documentation part of each module, coding and testing part of each module comes into the development.

6). System Integration and system testing

After succesfull development the every module, we integrate these modules into a system and testing of whole the system is done in this phase.

7). Maintenance

Like all other SDLCs, it also include the maintenance phase which is very important to maintain the product as the user requirement and technology changes with the time.

These are the some phases covered by software development in this model. But apart from these we have some other issues also covered by this model. One of them is the ability to go back on each phase whenever needed. For example, from the system testing phase we can go back to design phase if the product not filled the user requirement. Like this, from the maitenance phase we can go on any stage of software development for maintenance of the software product.This facility makes it easy to implement the things in implementation phase.

Software systems come and go through a series of passages that account for their inception, initial development, productive operation, updating, and retirement from one generation to other. As it is clear before, this SDLC is generated totally keeping in mind cost, time and maintenance because these are the main factors which vital for the software development. If we keep these factors in mind and develop a product then there is no need or a little need of risk management. So there is not much need of risk management in this SDLC. If there is some risk then we can work on them in the planning and design phase.

B. Advantages of Complete Model*:*

Because this SDLC is developed according to all the other models. So the best of this SDLC is that it gives the possible advantages of other models and try to keep disadvantages out processes in develepment. If we combine and find out some advantages of that model then, we can mention followings:

- A linear approach and easy to use like the Waterfall model.
- Documentation part on each phase make testing and maintenance part so easy. There is no complexity if want to make some changes in product.
- Easier to test and debug a smaller module and easier to manage risk because risky pieces are identified and handled during its module documentation part.
- In addition, the developer and the client better understand and react to risks and it gives a brief idea to both about the product and requirements. The model uses prototyping as a risk reduction mechanism and allows for the development of prototypes at any stage.
- Maintenance takes not so much time and cost.

CONCLUSION

This paper focused and examined a new model for software development life cycle (SDLC model) used in the software development. While there are a lot of the model and a lot of time and cost so high for development.

According to an analysis 67% of software development cost is due to maintenance and 15% due to testing. So, these areas are of crucial importance in software industry. This Complete model focused on these areas to reduce the software cost and development time.

IV. IMPLEMENTATION AND COMPARATIVE ANALYSIS WITH VARIOUS MODELS:

On the basis of a project developed by author . Project was a about a website of **Job Portal**. In which all the user of the website has the different privileges. According to their privileges they have different power different pages and different work.

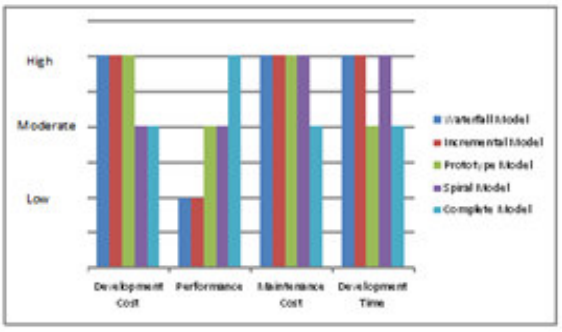

Fig. 5 Comparative Analysis of Various Models

V. APPLICATIONS

- Applicable for large industrial projects.
- Applicable for ERP etc.

REFERENCES:

1. Royce, Winston, “Managing the development large software system”
2. John Morrison’s Blog: http://www.techmanageronline.com/2009/11/waterfall-model.html
3. Incremental model details: http://www.softdevteam.com/Incremental-lifecycle.asp
4. Nasib Singh Gill, “Software Engineering Vol. 1
5. Software Engineering – A Practitioner’s Approach, Roger S. Pressman, 1996, MGH.
6. Fundamentals of software Engineering, Rajib Mall, PHI
7. Software Engineering by Ian sommerville, Pearson Edu, 5th edition, 1999, AW
8. Dr. Roy A.Boggs, “SDLC and Six Sigma”