International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 06 | June -2017 www.irjet.net p-ISSN: 2395-0072

Comprehensive Analysis of Software Development Life Cycle Models

Harshad S. Modi[1], Nikhil Kumar Singh[1], Harsha Pradeepbhai Chauhan[2]

[1] *Lecturer, Department of Computer Engineering, Government Polytechnic Gandhinagar, Gujarat*
[2]*Lecturer, Department of Information Technology, Government Polytechnic for Girls, Ahmedabad, Gujarat*

---***---

Abstract - *There is increasing demand of software with cheaper cost, having more functionality, faster delivery, and of high quality than previously—it was. Recently the software development has become more and more diverse and complex. SDLC models are utmost important for developing the software in a systematic manner such that it will be delivered within deadline and should also have proper quality. There are tons of SDLC models available. Each development model has certain advantages and disadvantages. The paper starts with the introduction of SDLC, followed by the discussion and comprehensive comparison among the various SDLC models.*

Key Words: Software life cycle, development models, Comparative analysis of models

1. INTRODUCTION

Software development life cycle (SDLC) is used to develop quality software in specified time as per to the need of customer. Quality of product is maintained using SDLC. Every development model includes different kind of activities like requirements gathering and analysis, system analysis, system design, coding, testing, implementation. Selection of SDLC model depends developer or the team of developers. There are advantages and disadvantages in each SDLC model depending on different kind situations. The main challenge is to select the best suitable model.

1.1 Literature Review of Various SDLC Models

This section Consist various models and techniques of software development. This models and techniques are analyzed on the basic of their strengths and weaknesses.

1.1 Waterfall Model

In 1991, David Whitgift [1] points out that in the earliest days of software development; code was written and then debugged. There was no formal approach for design and analysis. Due to the complex software systems requirements, this code and debug approach became less than optimal. Since the approach to developing complex hardware systems was well understood, it provided a model for developing software.

In waterfall model only after completing one phase we can go for other phase. After completing certain phases a baseline is set that freezes the products of the development at that point. If there is a need to change these products, a formal change process is followed to make the change. The graphical representation of these phases results in the downward flow of a waterfall.

In 2009, Kai Petersen et. al. [2] Perform a case study to provide more detailed explanations of the issues and identified four new issues, namely -

1. Which version of requirement is implemented by whom is a matter of Confusion
2. Great effort is required for maintenance
3. Specialized competence focus and peoples having less confidence
4. It's very tedious to locate the fault because of communication barrier.

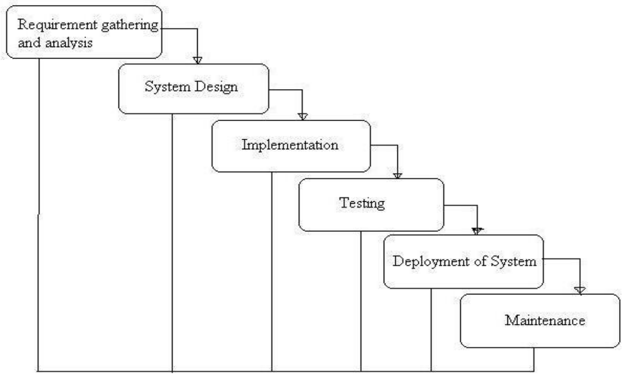

Fig-1: Waterfall Model

Comparing the Waterfall Model

As shown in Figure 1, the phases in waterfall model are totally sequential i.e. after completing one phase only you may proceed for the next phase. To compare this model with the other models, the key attributes of the waterfall model are that –

1. It is very much formal method.

2. Support top to down development.
3. Consist of independent phases that are to be performed sequentially.

Where to Use the Waterfall Model

The waterfall model is not advisable to be used for large-scale product. If the requirements and the implementation of those requirements are clearer than one can go for using the waterfall model. For example, if a company ABC has experience in building accounting software's, Input -Output controllers or compilers, then making such kind of another product that is based on the existing designs is easily and effectively managed with the waterfall model.

2.2 Incremental Model

Incremental model is mixture of linear and iterative prototyping model. It is joining of more than one Waterfall Models. This model gives the weightage to requirements and repeatedly implementing prototyping from the previously developed prototype until get the desired software product. [3]

Project requirements are contained in multiple modules and each module is designed and developed independently. At the end all the developed modules are joined with other modules. Each individual module is developed using the waterfall model.

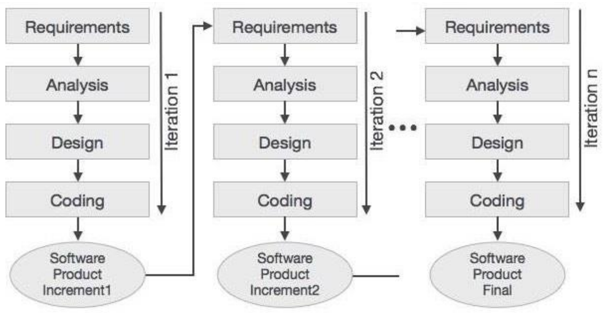

Fig-2: Incremental Model [4]

Where to Use the Incremental Model

1. If it is very much risky to build the overall system at once, in such case the incremental model of development is advisable.
2. Product is required to reach the market as soon as possible.
3. When the testing of new technology is required.

2.3 Spiral Model

One view of the incremental model is spiral model. The spiral view has one strong point than that of the incremental model: resources are kept constant and the size of the system grows. The spiral model size is nothing but the size of the system, where the gap between the spiral coils shows the resources. In Figure 3, the distance in between the coils in not altering, that indicates unchanging amount of the resources.

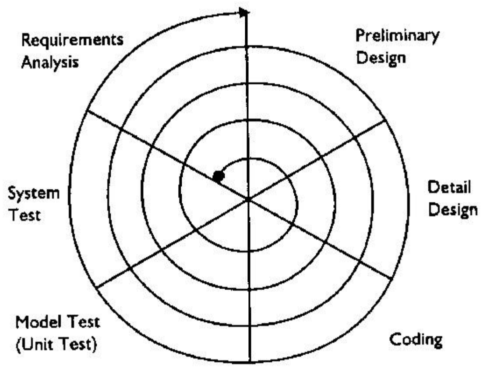

Fig-3: Spiral Model

Barry Boehm [5] proposed another spiral model the uses a prototyping model for controlling the cost. In this model you may see the continuous increase in resources as the distance in between the spiral coils is increasing as shown in Figure 4.

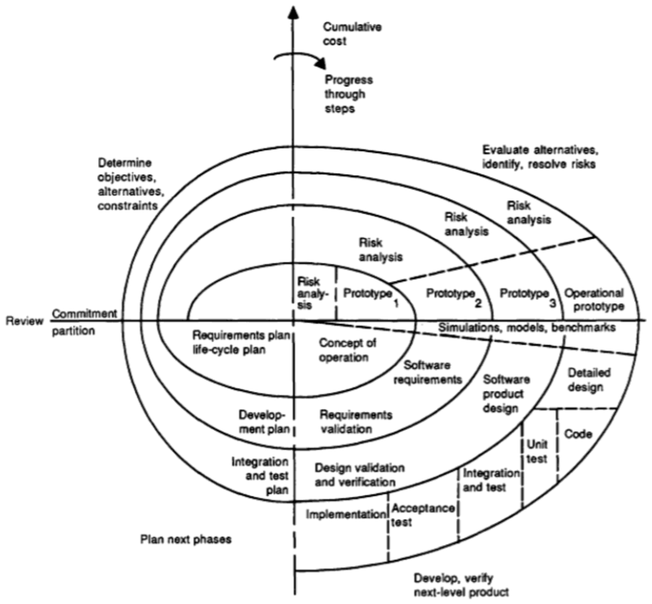

Fig-4: Boehm Spiral Model [5]

When to Use the Boehm Spiral Model

ADE (Aerospace, Defense and Engineering) specialists mostly use Boehm model because ADE project are considered to be more risky. It is not much used among the developers. Business projects are more conservative.

According to DeGrace [6] the spiral model may be actually used with many business applications, especially the projects where the success is not certain or the project requires lot of computation, like in decision support systems.

Table-1: Advantage and Disadvantage of Waterfall, Incremental, and Boehm Spiral Models

ADVANTAGE		
Waterfall	**Incremental**	**Boehm Spiral**
Suitable for smaller projects for which the requirements are very much clear	Suitable for Smaller as well as medium size products	Suitable for medium and large sizes of products.
The Phases are completed and processed one at a time.	Easy part is completed first as it do not need to wait for hard ones.	Software is produced earlier in the lifecycle.
It is simple and easy to use.	It is flexible with respect to Waterfall model	High risk management is required. Good for projects where Risk Analysis is prioritized.
DISADVANTAGE		
Waterfall	**Incremental**	**Boehm Spiral**
Redefining the scope during the life cycle can kill the project.	Changes are possible.	Changes in requirements are very much possible so not suitable for small size projects
No working software is produced until the life cycle completes.	Activities performed in parallel may lead to miscommunication and leads to mistook.	Highly costly model as it require risk analysis in each cycle.
High amount of risk and uncertainity.	Risk is involved in unforeseen interdependencies.	Risk analysis requires highly expertise for specific fields.

2.4 Prototyping

Prototyping is the task of creating a working replica of a system. This model can be used with the waterfall model in a way like Boehm Spiral or might completely replace it.

DeGrace [6] says that you may find some sort of list of requirements. Sometimes it may be informal. Requirements of your customers may arrive in form of memos

After that you may transform the requirements in a working model by altering your prototype to include them. By using the fourth generation language, you may transform the requirements into language and macro commands.

With the help of available libraries, you may write a driver and then insert and select calls to the functions of library that depicts the requirements. After that you may integrate all of them by writing the code to handle output, input, and error processing functions, operator messages, and connections between functions.

In the next step, you present the results in front of the customer or may decide in any way, whether it is doing what the customer want or not. Repeat the process if there is involvement of new set of requirements. [6]

Use of Prototyping with the Waterfall

The prototyping model may be used with the waterfall model. This model may be used to represent the technical feasibility when the technical risk is quite high. This may also be used for extraction and understanding of Client requirement. The major goal of the model is to the project economic by getting the problem before taking more number of resources.

Use of Prototyping with Objects

Object-oriented model focuses on real-world objects. Coad and Yourdon [7] say that prototyping must be used with the analysis and design portions of Object Oriented. Object-Oriented Design based prototyping tools carry out the development life cycle tasks a little bit faster in an easier way.

Strengths of Prototyping

- Get the product in early stage
- Provides precise and perfect requirement.
- Risk management is done.
- Final product documentation.

Weaknesses of Prototyping

- Not compatible with already existing system.
- Documentation is not up to the level [8].
- Many times product suffers from poor performance.

2.5 Cleanroom

The cleanroom technique keeps the software bugs away from the product. In this the concept is to detect the bugs in the earlier stage when it is more economic. Mike Dyer describes a COBOL project which uses the Cleanroom technique that has 52,000 lines of code with 179 errors, and is 15 to 20 times better than industrial norms [9].

Using cleanroom technique the programs are more reliable than normal lifecycle model. Also the time required to develope a verified program is less to design, code, and debug a program. The functional verification technique scales up to large programs, and that statistical quality control is better than the time-honored technique of finding and removing bugs [10].

When to Use Cleanroom

Cleanroom technique can be used with waterfall, incremental, or spiral models to produce software of random size and complexity. Cleanroom results in higher quality software rather than increasing the direct productivity.

An organization that wants to use cleanroom technique needs to have only in place systematic design methods, documented requirements in a natural language, formal inspection procedures, developer-performed unit testing, and configuration management of software after its release to the independent test organization, and ad hoc functional testing. The basic cleanroom process is shown in Figure 5.

Strengths of Cleanroom

- Produce reliable software.
- Cleanroom may be gradually introduced to an organization

Weaknesses of Cleanroom

- Requires a complete set of requirements.
- Disciplined style may stifle creativity.

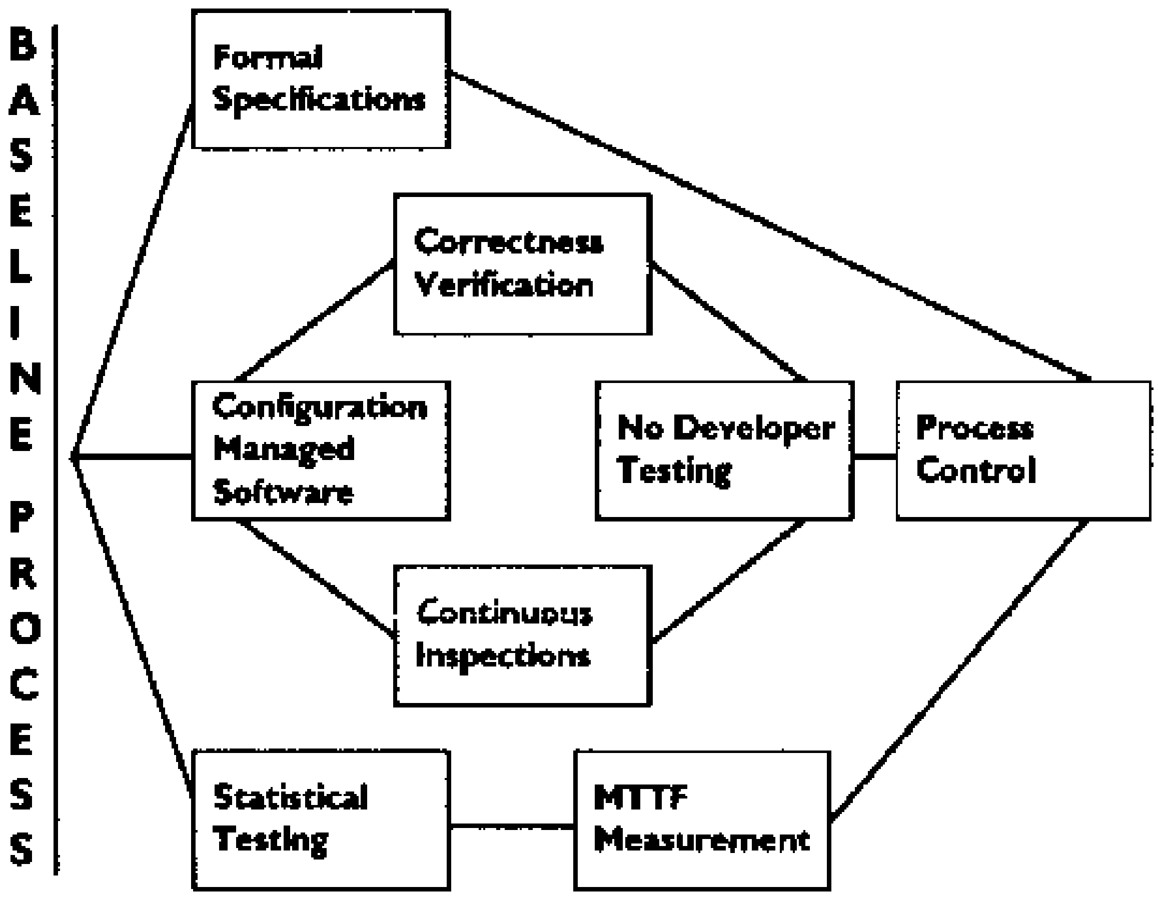

Fig-5: Roadmap for Introducing Cleanroom Component Techniques [9]

2.6 Object-Oriented

The object-oriented technique for software development is based on real-world objects. It is basically related with the condition that there exists basic human limitation to manage more than seven different objects or concepts at a time. Grady Booch [11] suggested using software engineering principals to help us to decompose systems so that we never simultaneously deal with more than seven entities. The popularity of object-oriented techniques is increasing with the enhanced complexity of software systems. Object-oriented includes Object Oriented Programming (OOP), object-oriented analysis (OOA), Object Oriented design (OOD).

Use of Object-Oriented Model:

Use it in following situations-

1. If system is data-oriented and functional complexity is somewhat less [12].
2. If a decent object-oriented implementation technology is there, and the organization gives you proper tools for its practitioners to efficiently and effectively use It.
3. Object-oriented Analysis is not much effective for systems with very limited responsibilities [12].

Strengths of Object-Oriented: [13]

1. Client may be involved in getting the result.
2. Reduced maintenance cost..
3. This model represents reality from users point of view.

Weaknesses of Object-Oriented

1. Due to non-structured background of analysis it may be difficult for one who is having structured analysis background.
2. It’s very tedious task to reconcile with DoD-STD-2167A [14].

Conclusion

This paper consist analysis of three different models and three different techniques. There are several other models and techniques that have not been taken into consideration. The models discussed are waterfall model, incremental model and spiral model. The techniques discussed are prototyping, Cleanroom, and object-oriented which are believed to be those most relevant to the Organizations like Department of Defense. Proper understanding of these models and techniques provides the idea to understanding others models and techniques.

REFERENCES

[1] Whitgift, David, "Methods and Tools for Software Configuration Management", J. Wiley, 1991.

[2] Petersen K., Wohlin C., Baca D. , "The Waterfall Model in Large-Scale Development", Lecture Notes in Business Information Processing, vol 32. Springer, Berlin, Heidelberg, 2009.

[3] Shubhmeet Kaur, " A Review of Software Development Life Cycle Models", In International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 5, No. 11, 2015.

[4] https://www.tutorialspoint.com/scrum/scrum_overview.htm

[5] Boehm, Barry, "A Spiral Model of Software Development and Enhancement," from Proceedings of an International Workshop on the Software Process and Software Environments, 1985.

[6] DeGrace, Peter, and Stahl, Leslie Hulet, "Wicked Problems, Righteous Solutions: A Catalogue of Modern Software Engineering Paradigms", pp. 116, 117, 127. Reprinted with permission of Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[7] Coad Peter and Edward Yourdan, Object-Oriented Design, 1991.

[8] Software Management Guide, Vol. I, Software Technology Support Center, October 1993, p. 23.

[9] Dyer, Mike, "he Cleanroom Approach to Quality Software Development", 1993.

[10] Blum Bruce I., "Software Engineering: A holistic View", 1992.

[11] Booch, Grady, Software Engineering with Ada, 1994, p. 25.

[12] Coad, Peter, and Edward Yourdon, Object-Oriented Analysis, 1991.

[13] Morris, Randy, and Boyd Tidwell, "Object-Oriented Ada Using State Controlled Implementation," CrossTalk, June 1994, p. 17.

[14] Atkinson, Shane, Documentation Technology Report, Software Technology Support Center, April 1994.

BIOGRAPHIES

Harshad S. Modi is working as Lecturer in Government Polytechnic Gandhinagar, Gujarat Since last 6 years. He also worked as Lecturer in UVPCE for 5 years. He is pursuing his M-Tech from GEC, Gandhinagar.

Nikhil Kumar Singh is working as Lecturer in Government Polytechnic Gandhinagar, Gujarat. He also worked as Assistant Professor in U. V. Patel college of Engineering for more than 2 years. He completed his M-tech from SVNIT Surat, Gujarat.

Harsha Pradeepbhai Chauhan is working as lecturer (I.T.) in Government Polytechnic for Girls Ahmedabad, Gujarat. She has more than 15 year's experience in teaching field. She completed her B. E. From MSU Vadodara, Gujarat.