Open access Journal International Journal of Emerging Trends in Science and Technology

A Comparative Study between Waterfall and Incremental Software Development Life Cycle Model

Authors
Deepti Singh[1], Ankit Thakur[2], Abhishek Chaudhary[3]
[1]M.Tech (CSE) Scholar, Bhagwant University, Sikar Road, Ajmer, Rajasthan, India
Email: *deepti.singh426@gmail.com*
[2]M.Tech (CSE) Scholar, Bhagwant University, Sikar Road, Ajmer, Rajasthan, India
Email: *ankit.thakur1011@gmail.com*
[3]Assistant Professor (CSE), Bhagwant University, Sikar Road, Ajmer, Rajasthan, India
Email: *abhishek02mar@rediffmail.com*

Abstract
Software development life cycle or SDLC for short is a methodology for designing, building, and maintaining information and industrial systems. There are various SDLC models widely used for developing software. SDLC models give a theoretical guide line about development of the software. 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. 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. Each SDLC has its advantages and disadvantages according to which we decide which model should be implemented under which conditions. In the present scenario all software systems are imperfect because they cannot be built with mathematical or physical certainty. According SDLC each and every model has the advantage and drawbacks. The concept of system lifecycle models came into existence that emphasized on the need to follow some structured approach towards building new or improved system. For this we need to compare SDLC models. In this paper we will compare two different famous life cycle models like-waterfall model & incremental model.

Keywords: *Increment model, waterfall model, Software engineering, Software process model and Software*

1. INTRODUCTION

In today's life computer is considered a time- saving device and its progress helps in executing complex, long, repeated processes in a very short time with a high speed. In addition to using computer for work, people use it for fun and entertainment. Noticeably, the number of companies that produce software programs for the purpose of facilitating works of offices, administrations, banks, etc, has increased recently. During the previous few decades, software has been developed from a tool used for analyzing information or solving a problem to a product in itself. However, the early programming stages have created a number of problems turning software an obstacle to software development particularly those relying on computers. Software consists of documents and programs that contain a collection that has been established to be a part of software engineering procedures. Moreover, the aim of software engineering is to create a suitable work that constructs programs of high quality. A software development process, also known as a software development life cycle (SDLC), is a structure imposed on the development of a software product. 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. 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.

The development models are the various processes or methodologies that are being selected for the development of the project depending on the project's aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out.The selection of model has very high impact on the testing that is carried out. It will define the what, where and when of our planned testing, influence regression testing and largely determines which test techniques to use.There are various Software development models or methodologies. They are as follows:

1. Waterfall model
2. V model
3. Incremental model
4. RAD model
5. Agile model
6. Iterative model
7. Spiral model

This paper focused on the comparative study between two different models-waterfall and incremental model. In the Section -2, SDLC phases are discussed, Section -3 summarizes waterfall model, Section-4 summarizes incremental model and Section – 5, includes the conclusion.

2. SDLC PHASES

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. Software life cycle models describe phases of the software cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced according to the design which is called development phase. After coding and development the testing verifies the deliverable of the implementation phase against requirements.

There are following six phases in every Software development life cycle model:

2.1 Requirement gathering and analysis:

Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements like; Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development. Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

2.2 Design: In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

2.3 Implementation /Coding: On receiving system design documents, the work is divided in modules/units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

2.4 Testing: After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase unit testing, integration testing, system testing, acceptance testing are done.

2.5 Deployment: After successful testing the product is delivered / deployed to the customer for their use.

2.6 Maintenance: Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

3. WATERFALL MODEL

Waterfall model was proposed by Royce in 1970 which is a linear sequential software development life cycle (SDLC) model. The waterfall model is the classical model of software engineering. This model is one of the oldest models and is widely used in government projects and in many major companies. As this model emphasizes planning in early stages, it ensures design flaws before they develop. Hence it is not very much useful when the project requirements are dynamic in nature The model begins with establishing system requirements and software requirements and continues with architectural design, detailed design, coding, testing, and maintenance. The waterfall model serves as a baseline for many other lifecycle models. In some organizations, a change control board maintains the quality of the product by reviewing each change made in the maintenance stage. Consider applying the full waterfall development cycle model when correcting problems or implementing these enhancement requests. In each stage, documents that explain the objectives and describe the requirements for that phase are created. At the end of each stage, a review to determine whether the project can proceed to the next stage is held. Many people believe that this model cannot be applied to all situations. In real-world development, however, one can discover issues during the design or coding stages that point out errors or gaps in the requirements. The waterfall method does not prohibit returning to an earlier phase, for example, returning from the design phase to the requirements phase. However, this involves costly rework. Each completed phase requires formal review and extensive documentation development. Thus, oversights made in the requirements phase are expensive to correct later. Because the actual development comes late in the process, one does not see results for a long time. This delay can be disconcerting to management and customers. Many people also think that the amount of documentation is excessive and inflexible. Although the waterfall model has its weaknesses, it is instructive because it emphasizes important stages of project development. Even if one does not apply this model, he must consider each of these stages and its relationship to his own project.

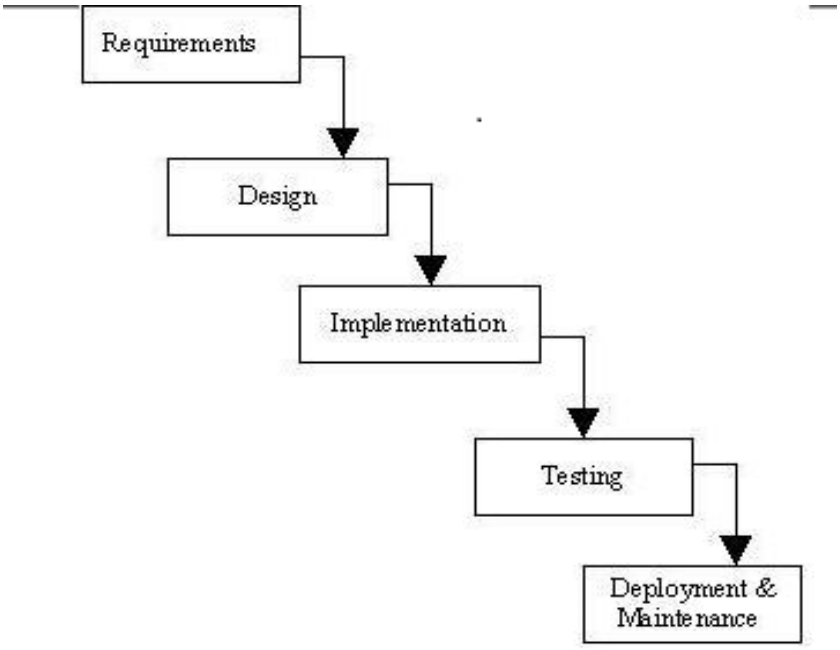

fig1. Waterfall Model

3.1 Advantages:

- Easy to understand and implement. 2.
- Widely used and known (in theory!).
- Reinforces good habits: define-before- design, design-before-code.
- Identifies deliverables and milestones.
- Document driven, URD, SRD, … etc. Published documentation standards, e.g. PSS-05.
- Works well on mature products and weak teams.

3.2 Disadvantages:

- Idealized, doesn't match reality well.
- Doesn't reflect iterative nature of exploratory development.
- Unrealistic to expect accurate requirements so early in project.
- Software is delivered late in project, delays discovery of serious errors.
- Difficult to integrate risk management.
- Difficult and expensive to make changes to documents.
- Significant administrative overhead, costly for small teams and projects

4. INCREMENTAL MODEL

The Incremental approach is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. The incremental model is an intuitive approach to the waterfall model. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. Cycles are divided up into smaller, more easily managed iterations. Each iteration passes through the requirements, design, implementation and testing phases. 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. A

project which is using the incremental model may start with general objectives, which is also defined as provisions and those requirements are implemented by following the next coming portion of the objectives, till all objectives are performed it push difficult problems to the future to prove early success and this model can be useful for the projects where the basic software functionality is required at the beginning.

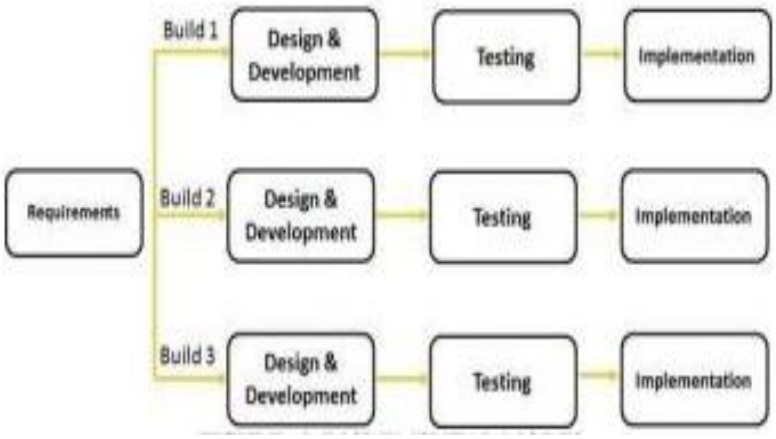

Fig 2. Incremental Model

4.1 Advantages

- 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.
- Each iteration is an easily managed milestone.

4.2 Disadvantages

- 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 front for the entire software life cycle.

5. COMPARISON

The waterfall model consists of phases that are completed sequentially before proceeding to the next phase. For comparison to other models, the salient attributes of the waterfall model are that it is

- A formal method.
- A type of top-down development.
- Composed of independent phases to be done sequentially.
- Used in varied ways.
    - Steps are combined.
    - There are different starting and ending points.

Because of the weaknesses shown above, the application of the waterfall model should be limited to situations where the requirements and the implementation of those requirements are very well understood.

It is developed to overcome the weaknesses of the waterfall model. The incremental model performs the waterfall in overlapping sections attempting to compensate for the length of waterfall model projects by producing usable functionality earlier. This may involve a complete upfront set of requirements that are implemented in a series of small projects. As an alternative, a project using the incremental model may start with general objectives. Then some portion of these objectives is defined as requirements and is implemented, followed by the next portion of the objectives until all objectives are implemented. But, use of general objectives rather

than complete requirements can be uncomfortable for management. Because some modules will be completed long before others, well-defined interfaces are required. Also, formal reviews and audits are more difficult to implement on increments than on a complete system. Finally, there can be a tendency to push difficult problems to the future to demonstrate early success to management.

Table 1- Comparison of Waterfall model and Incremental Model

Features	Waterfall Model	Incremental Model
Requirement specifications	Beginning	Beginning
Cost	Low	Low
Simplicity	Simple	Intermediate
Risk involvement	high	Easily manageable
Expertise	High	High
Flexibility to change	Difficult	Easy
User involvement	Only at beginning	Intermediate
Flexibility	Rigid	Less flexible
Maintenance	Least	Promotes maintainability
Duration	Long	Very long

6. CONCLUSIONS

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 focused on the two different models of software development life cycle-waterfall model & incremental model and compare them with different aspects. For example, if a company has experience in building accounting systems, I/O controllers, or compilers, then building another such product based on the existing designs is best managed with the waterfall model and if it is too risky to develop the whole system at once, then the incremental development should be considered 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. 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
2. 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.
3. 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
4. Khurana gourav and s gupta(2012) " Study & Comparison of Software Development Life Cycle Models" IJREAS, Vol. 2(2), 1514-1515.
5. 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.

6. 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.
7. Eduardo Malaga Chicano, "A comparative study if iterative protyping vs waterfall model processed by small and medium sized projects by system engineering", 1996.
8. Craig Larman and Victor R. Basili,"Iterative and Incremental Development: A Brief History". IEEE Computer (IEEE Computer Society) 36 (6): 47–56. Doi:10.1109/MC.2003.1204375. ISSN 0018-9162. Retrieved 2009-01-10.], June 2003.
9. Ernest Mnkandla, "About Software Engineering Frameworks and Methodologies", IEEE AFRICON 2009.
10. Sanjana Taya, Shaveta Gupta, "Comparative Analysis of Software Development Life Cycle Models"
11. Mohamed Sami Abd EI-Satar" Software Development Life Cycle Models and Methodologies", 2012
12. Kevin Forsberg and Harold Mooz, "The Relationship of System Engineering to the Project Cycle," in Proceedings of the First Annual Symposium of National Council on System Engineering, October 1991: 57–65.
13. Royce, Winston, "Managing the Development of Large Software Systems", Proceedings of IEEE WESCON 26, 1970
14. Jeffrey A. Livermore Walsh College, "Factors that Impact Implementing an Agile Software Development Methodology".
15. Lillian Doric and Elias Niemelae "A Survey on Software Architecture Analysis Methods", IEEE Transaction on Software engineering, VOL. 28, NO. 7, JULY 2002.
16. M, Hue, 1. Varner, L. Zhu, and M, A. Babar, "Software quality and Agile Methods," Pros,28th Annual International Computer Software and Applications Conference (COMPSAC'04), pp., 520-525,2004
17. Steve Easterbrook, "Software Lifecycles", University of Toronto Department of Computer Science, 2001.
18. JJ Kuhl, "Project Lifecycle Models: How They Differ and When to Use Them" 2002.
19. 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
20. Fowler, M., "Put Your Process on a Diet", Software Development",2000.