ISSN:0975-9646

Software Development Life Cycle Models- Comparison, Consequences

Vanshika Rastogi

Asst. Professor, Dept. of ISE, MVJCE
Bangalore

Abstract- Software Development Life Cycle is a well defined and systematic approach, practiced for the development of a reliable high quality software system. There are tons of SDLC models available. This paper deals with five of those SDLC models, namely; Waterfall model, Iterative model, V-shaped model, Spiral model, agile model. Each development model has certain advantages and disadvantages. The paper begins with the discussion to the introduction of SDLC, followed by the comprehensive comparison among the various SDLC models.

Key words: ***software development life cycle, development models, comparison between models.***

1. INTRODUCTION

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. [1] Software Engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software because it integrates significant mathematics, computer science and practices whose origins are in Engineering. Various processes and methodologies have been developed over the last few decades to improve software quality, with varying degrees of success. However it is widely agreed that no single approach that will prevent project over runs and failures in all cases. Software projects that are large, complicated, poorly-specified, and involve unfamiliar aspects, are still particularly vulnerable to large, unanticipated problems. A software development process is a structure imposed on the development of a software product. There are several models for such processes, each describing approach test to a variety of tasks or activities that take place during the process. It aims to be the standard that defines all the tasks required for developing and maintaining software. [2]

These classic software life cycle models usually include some version or subset of the following activities:

- Planning and Visualization
- Requirement Analysis
- Software Modeling and Design
- Coding
- Documentation
- Testing
- Deployment and Maintenance

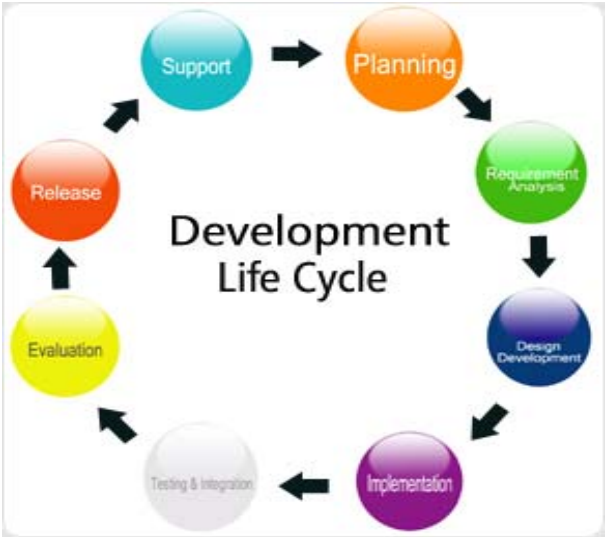

fig.1: SDLC

2. VARIOUS SDLC MODELS

The various SDLC models are discussed as:

a) Waterfall 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. In addition, it is intensive document and planning makes it work well for projects in which quality control is a major concern. The waterfall life cycle consists of several non overlapping stages; 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 life cycle models.[3]

Basic principle:

- Project is divided into sequential phases, with some overlap and splash back acceptable between phases.
- Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
- Tight control is maintained over the life of the project via extensive written documentation, formal reviews, and approval/signoff by the user and information technology management to occurring at the end of most phases before beginning the next phase.[2]

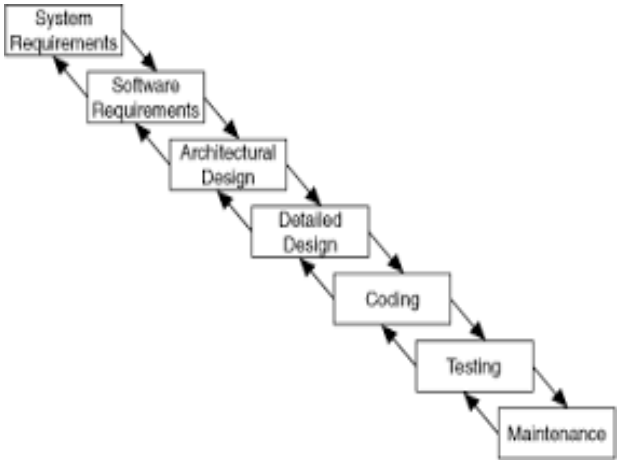

Fig 2: waterfall model

b) Iterative model: An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.[4]

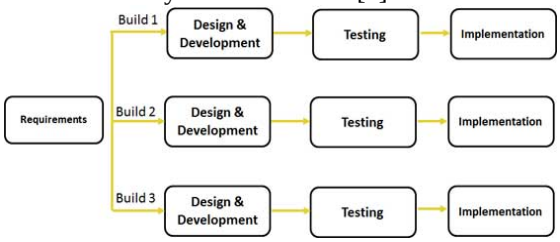

Fig3: iterative model [5]

Basic Principle:

- The problems with the Waterfall Model created a demand for a new method of developing systems which could provide faster results, require less up front information and offer greater flexibility.
- Iterative model, the project is divided into small parts. This allows the development team to demonstrate results earlier on in the process and obtain valuable feedback from system users.
- Each iteration is actually a mini-Waterfall process with the feedback from one phase providing vital Information for the design of the next phase.[2]

c) c) V-shaped Model: Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing is emphasized in this model more so than the waterfall model though. The testing procedures are developed early in the life cycle before any coding is done, during each of the phases preceding implementation. Requirements begin the life cycle model just like the waterfall model. Before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in the requirements gathering. The high- level design phase focuses on system architecture and design. An integration test plan is created in this phase as well in order to test the pieces of the software systems ability to work together. The low- level design phase is where the actual software components are designed, and unit tests are created in this phase as well. The implementation phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use. [6]

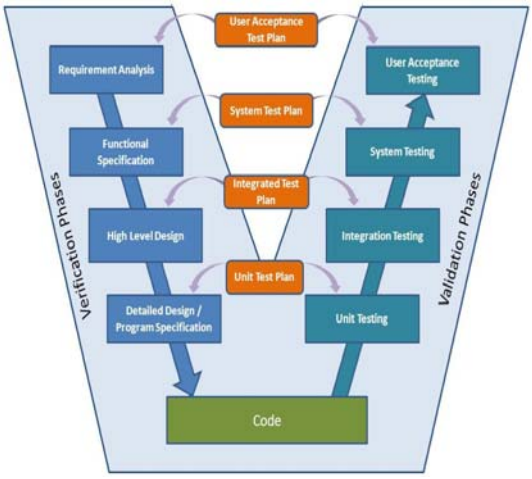

Fig4: iterative model

d) Spiral model: This model was not the first model [7] to discuss iterative development, but it was the first model to explain why the iteration matters. As originally envisioned, the iterations were typically 6months to 2years long. Each phase starts with a design goal and ends with the client (who may be internal) reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project. The process begins at the center position. From there it moves clock wise in traversals. Each traversal of the spiral usually results in a deliverable [8]. It is not clearly defined what this deliverable is. This changes from traversal to traversal. For example, the first traversals may result in a requirement specification. The second will result in a prototype, and the next one will result in another prototype or sample of a product, until the last traversal leads to a product which is suitable to be sold. Consequently the related activities and their documentation will also mature towards the outer traversals. E.g. a formal design and testing session would be placed into the last traversal.[9]

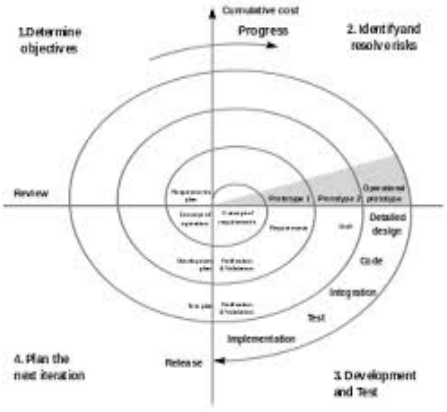

Fig5: spiral model

e) Agile model: Agile software development is a style of software development that emphasizes customer satisfaction through continuous delivery of functional software". The Process of Agile Software Development involves the following:

1. Starts with a kick-off meeting
2. The known requirements are understood and prioritized. The development is plan is drawn accordingly.
3. Relative complexity of each requirement is estimated
4. Sufficient design using simple diagrams is done
5. Test Driven Development (TDD) approach may be used. TDD emphases on "writing test first and then writing code to pass the test". It can help in avoiding over-coding.
6. Development is done, sometimes in pairs, with lot of team interaction. Ownership of code is shared when pair programming is done.
7. The code is tested more frequently. Sometime a dedicated "Continuous Integration" Server/Software may be used to ease the integration testing of the code.
8. Depending on the feedback received, the code is refractor. Refactoring does not impact the external behavior of the application but the internal structure may be changed to provide better design, maintainability. Some ways of refactoring may be add interface, use super class, move the class etc. [10]

3. COMPARISON

The comparison between different models is shown by their advantages and disadvantages in two different tables respectively:

Table1: Comparison of advantages [6]

S.N	Waterfall model	Iterative model	Spiral model	V shaped Model
1.	Simple and easy to use.	More flexible than the basic waterfall model.	High amount of risk analysis	Simple and easy to use.
2.	Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.	If there is personnel continuity between the phases, documentation can be substantially reduced.	Good for large and mission-critical projects	Each phase has specific deliverables.
3.	Phases are processed and completed one at a time.	Implementation of easy areas does not need to wait for the hard ones.	Software is produced early in the software life cycle.	Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle.
4.	Works well for smaller projects where requirements are very well understood.	Works well for smaller and moderate size projects	Works well for projects where risk analysis contains higher priority.	Works well for small projects where requirements are easily understood

Table2: Comparison of Disadvantages [6]

S.N	Waterfall model	Iterative model	Spiral model	V shaped Model
1.	Adjusting scope during the life cycle can kill a project.	Milestones are more ambiguous than the waterfall.	Can be a costly model to use.	Very rigid, like the waterfall model.
2.	No working software is produced until late during the life cycle.	Activities performed in parallel are subject to miscommunication and mistaken assumptions.	Risk analysis requires highly specific expertise.	Little flexibility and adjusting scope is difficult and expensive.
3.	High amounts of risk and uncertainty.	Unforeseen interdependencies can create problems.	Project's success is highly dependent on the risk analysis phase.	Software is developed during the implementation phase, so no early prototypes of the software are produced.
4.	Poor model for complex and object-oriented projects. Poor model where requirements are at a moderate to high risk of changing.	Changes are possible as it is iterative model	Doesn't work well for smaller projects.	Model doesn't provide a clear path for problems found during testing phases.

4. COMPARISON BETWEEN WATERFALL MODEL AND AGILE MODEL

Table3: Comparison of Waterfall and Agile model [11]

	Waterfall model	Agile Model
History	Waterfall model in software engineering got formally introduced as an idea, through a paper published by Winston Royce in 1970.	Agile model of software development, evolved in the 1990s. After further improvements, In 2001, a group of pioneers in agile software development came together and declared the 'Agile Manifesto', which is a set of canonical rules of sorts, for agile software development methods.
Conceptual Difference	It is a sequential process of software development. Just like in a waterfall, the water progressively falls from one altitude to the lower, in a similar way, the production cycle progresses sequentially, from one stage to the other.	Agile breed of models, focus on 'agility' and 'adaptability' in development. Instead of one time-consuming and rigid development schedule, agile models involve multiple iterative development schedules that seek to improve the output with each iteration.
Efficiency	The '*One Phase*' and '*Rigid*' development cycle of a waterfall model, makes it difficult to make last minute changes in requirements or design.	Agile methods, due to their iterative and adaptable nature, can incorporate changes and release a product, in lesser time.
Suitability	Waterfall model is a natural choice when the customer has provided a clear list of requirements, which are not likely to be modified.	Agile models are applicable in every area of software development. It's best suited for web based applications where its iterative nature helps in incorporating and correcting the various bugs that arise over time.

5. CONCLUSION

There are more than tons of sdlc models today. Here only a study of five of those models is given. This paper focused on basic five models; their advantages, disadvantages, so that one can select the best suited model as per his requirements.

6. FUTURE WORK:

This paper focused on the existing models. There are various shortcomings in the existing models; in future we can have models that can overcome the drawbacks of the existing models.

REFERENCES:

[1] IanSommerville,SoftwareEngineering,AddisonWesley,9th ed., 2010.
[2] Comparative Analysis of Different types of Models in Software Development Life Cycle-Ms.Shikha Maheshwari, Prof. Dinesh Ch. Jain
[3] A Comparison between Five Models of Software Engineering Nabil Mohammed Ali Munassar and A.Govardhan [4] http://istqbexamcertification.com/what-is iterative- model-advantages-disadvantages-and-when-to- use-it/
[5] http://www.tutorialspoint.com/sdlc/sdlc_iterative_mod el.html
[6] Raymond Lewallen,"SoftwareDevelopment Life Cycle", 2005
[7] weblog.erenkrantz.com/~jerenk/phase-ii/Boe88.pdf
[8] www.ccs.neu.edu/home/matthias/670s05/Lectures/2.html
[9] Software Engineering Models Consequences And Alternatives Nitin Mishra, Shantanu Chowdhary, Arunendra Singh, Anil Sharma
[10] Study & Comparison Of SoftwareDevelopment Lifecycl Models-Gourav Khurana ,Sachin Gupta
[11] Gray Pilgrim, "Waterfall Model Vs Agile", Website http://www.b uzzle.com/articles/waterfall- model-vs-agile.html, Jan, 2012