ISSN: 2321-7782 (Online)
Volume 2, Issue 3, March 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Paper / Case Study
Available online at: www.ijarcsms.com

Analysis of Strengths and Weakness of SDLC Models

Shikha Verma
Delhi – India

Abstract: ***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.***

Keywords: Build and Fix model, Iterative model, Prototype model, Spiral model

I. INTRODUCTION

A software development process also known as software development life cycle is a structure imposed on the development of software products. There are several models for software to a variety of task or activities that take during the process. SDLC stands for Software Development Life Cycle. SDLC is the process consisting of a series of planned activities to develop or alter the software products. It is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

II. SDLC MODEL I - BUILD AND FIX MODEL

It is a simple two phase model:

- The phase is to write code.
- The next phase is to fix it.

Fixing in this context may be error correction or addition of further functionality.

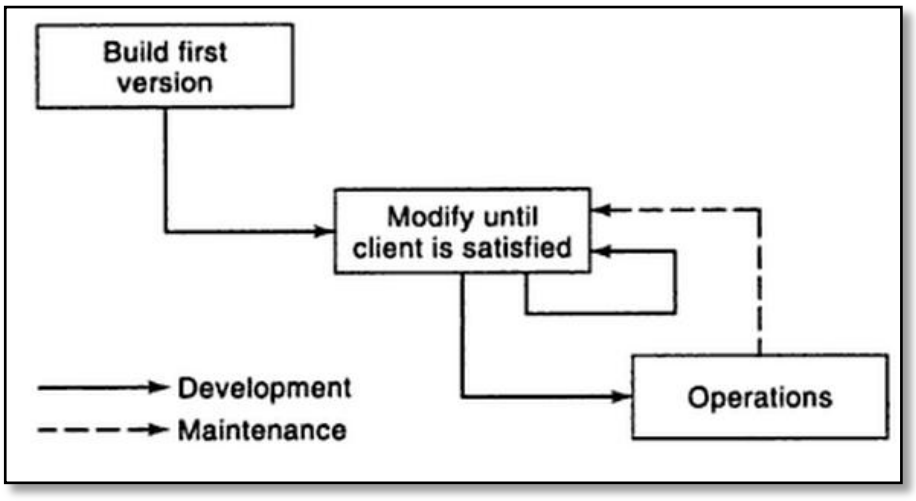

Figure 1: Build and Fix Model

III. SDLC Model II - Waterfall Model

This is of the simplest known as the linear sequential life cycle model. In a waterfall model, each phase must be completed before moving onto the next. This model has five phases:

Requirement analysis and specification phase- The goal of this phase is to understand the exact requirements of the customer and to document them properly.

Design phase-The goal of this phase is to transform the requirement specification into a structure that is suitable for implementation in some programming language.

Implementation and unit testing-Design is implemented, small modules are tested in isolation from the rest of the software products.

Integration and system testing phase-This is very important phase. It is also a very expensive and consumes one third to one half of the cost of a typical development project.

Operation and maintenance-Software maintenance is very broad activity includes :

- Error correction
- Enhancement of capabilities
- Deletion of capabilities and optimization

The purpose of this phase is to preserve the value of software overtime

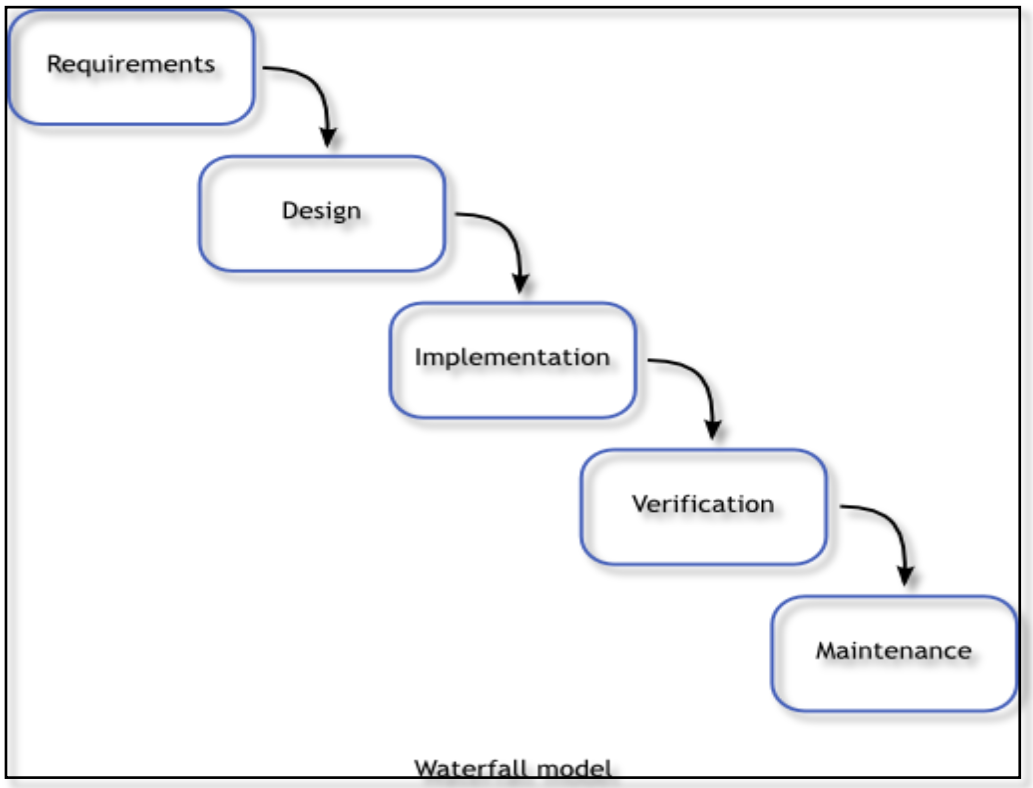

Figure 2: Waterfall Model

IV. SDLC Model III - Prototyping Model

This model is used to overcome the limitation of waterfall model. In this model instead of freezing the requirements before coding or testing or design, a prototype is built to clearly understand the requirements. This prototype is built based on the e current requirements. Through examining this prototype, the clients get a better understanding of the features of the final product. The prototype may be a usable program, but is not suitable as the final software product.

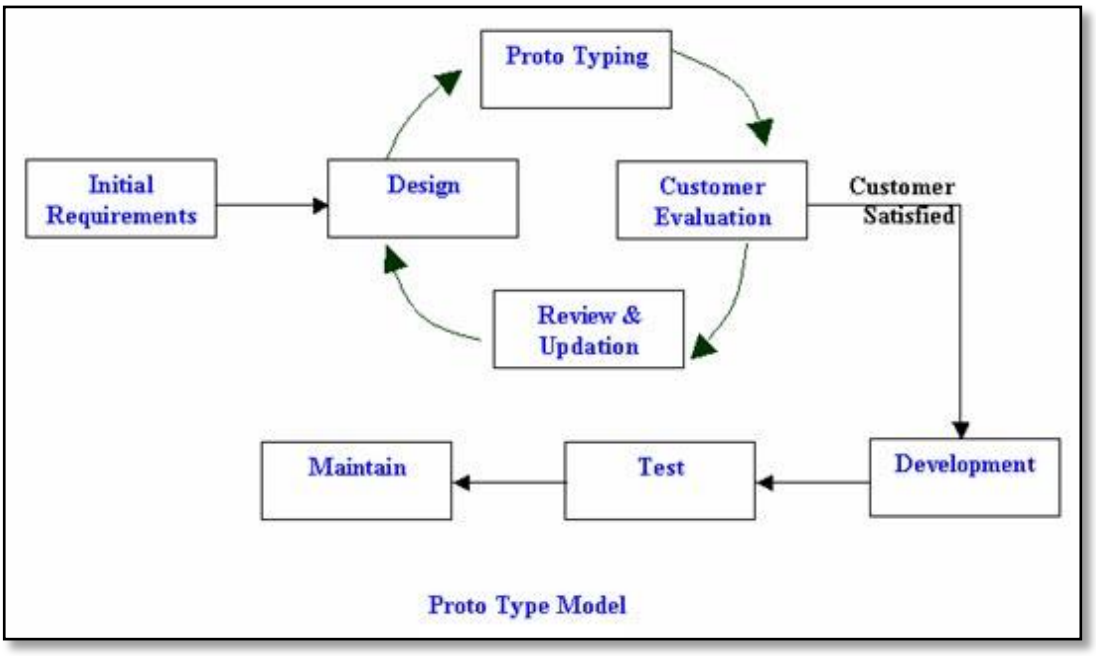

Figure 3: Prototype Model

V. SDLC Model IV – Iterative Enhancement Model

The iterative model can be thought of as a "multi –waterfall" cycle. Cycles are divided into smaller and easily managed iterations. Each iteration passes through a series of phases, so after each cycle you will get working software. This model delivers an operational quality of product at each release but one that satisfies only a subset of the customer's requirements. The complete product is divided into reeases and the developer delivers the product release by

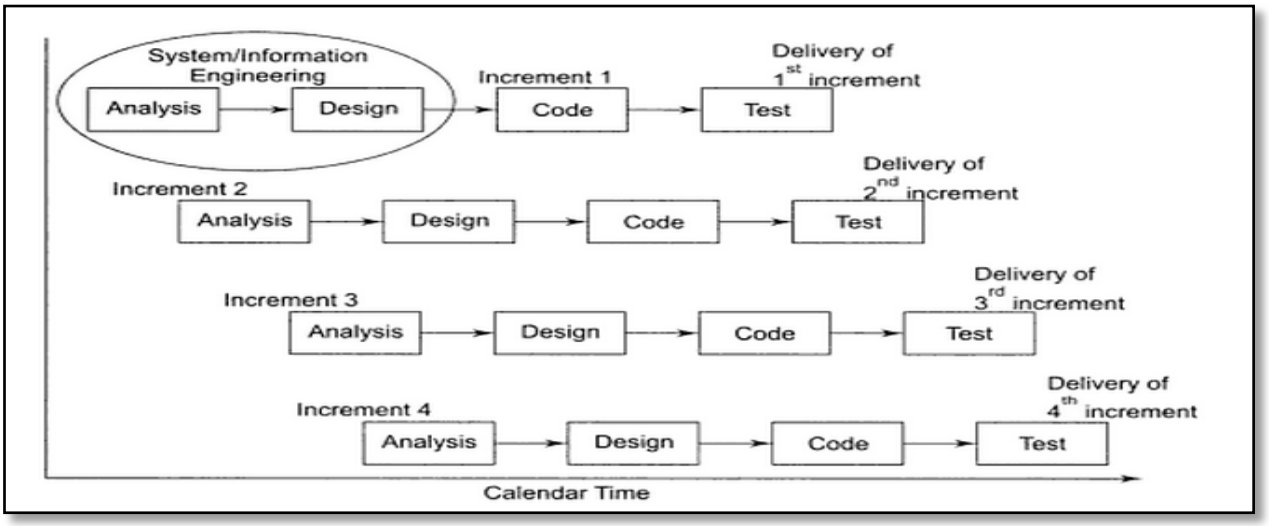

Figure 4: Iterative Model

VI. SDLC Model V – Evolutionary Development Model

This model resembles iterative enhancement model. It does not require a useable product at the end of each cycle. In evolutionary development, requirements are implemented by category rather than by priority.This are useful for projects using new technology that is not well understood. This is also used for complex projects where all functionality must be delivered at one time, but the requirements are unstable or not well understood at the beginning.

VII. SDLC Model VI – Spiral Model

The spiral model is similar to iterative model but places more emphasis on risk analysis. The system requirements are defined in as much as possible by involving various users so as to identify the various aspects of the system. The preliminary design is created. This is the most crucial step in the spiral model as it helps in developing cost effective strategies for working on a project. Using the preliminary design, a prototype for the new system is developed this is a scaled down system, which represents an approximate characteristics of final product /output. Each loop of the spiral represents one phase. One phase is split roughly into four sectors of major activities:

- Planning
- Risk analysis
- Development
- Assessment

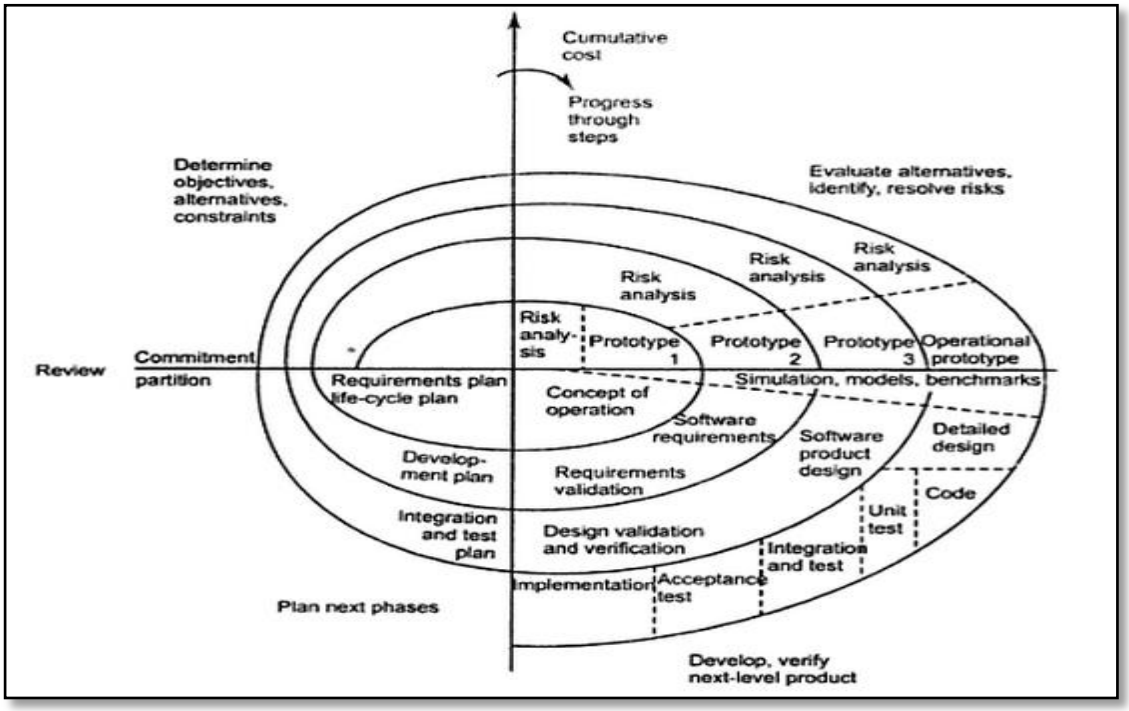

Figure 5: Spiral Model

VIII. Comparison of Strength and Weakness of SDLC

SDLC MODEL	STRENGHTS	WEAKNESSES	TYPES OF PROJECTS
BUILD AND FIX MODEL	1. No time spent on overhead tasks such as planning, documentation, quality assurance, standards, enforcement and other non-coding activities. 2. Requires little experience	1. Fundamental flaws in approach do not show up quickly 2. Rarely produces useful results 3. Dangerous as there is no way to access progress, quality or risk 4. Reworking and maintenance are costly	1. Small projects and programming exercises like proof of concept , demos and prototypes
WATERFALL MODEL	1. Simple , easy to execute , intuitive and logical 2. Controllable	1. Inflexible and does not support iterations 2. Delivers results late	1. For well-understood problems 2. Short duration

	results 3. Consistent with many technology practices 4. Minimizes planning overhead 5. Minimizes wasted effort	in the development cycle 3. Addressing mistakes is difficult task 4. User feedbacks are not taken during development	projects 3. Automation of existing manual system
PROTOTYPING	1. Benefits from user input 2. As a working model of the system is provided, user gets a better understanding of the system that is being developed 3. Errors and risks can be detected at a much earlier stage, as the system is developed using a prototypes	1. Increases complexity of the overall system 2. Involves higher risks 3. Involves implementing and then repairing the way a system is built so errors are an inherent part of the development process 4. Disallows later changes	1. System with novice users 2. When there are uncertainties in requirements
ITERATIVE MODEL	1. Produces working software early during the lifecycle 2. More flexible as scope and requirements changes can be implemented at low cost 3. Allows user feedbacks 4. Testing and debugging is easier , as the iterations are small 5. Low risks factor as the risks can be identified and resolved during each iteration	1. This model has phases that are very rigid and do not overlap 2. Not all the requirements are gathered before starting the development; this could lead to problems related to system architecture at later iterations. 3. Each iteration can have planning overhead	1. For businesses where time is of essence 2. Where risk of a long project cannot be taken 3. Where requirements are not known
EVOLUTIONARY MODEL	1. This model can be used even the requirements cannot or will not be specified 2. The user can experiment with the system to improve the requirements	1. Constitutes high risk and strong management is required 2. This method is used as an excuse for hacking to avoid documenting the requirements or design , even if they are well	1. System requirements early are not known in advance 2. Creating fundamentally new software 3. Developers are not confident in software architecture and

		understand	algorithms
SPIRAL MODEL	1. Almost reflects reality of software development 2. Reports of usage quite positive 3. Minimizes risks at early stage thereby prevents collapse 4. Lowest total cost 5. Each iteration of the spiral can be tailored to suit the needs of the whole project	1. Complex and intricate 2. Requires extensive management infrastructure 3. Without attentive and knowledgeable management , it fails	1. Only for major multimillion dollar efforts 2. Ill suited for small efforts

IX. CONCLUSION

After analysis of all models, it has been found that the original water fall model is used by various big companies for their internal projects .Prototype model used to develop online systems for transaction processing. Evolutionary model is useful when system requirements are not known in advance or when creating fundamentally new software. Spiral model is used for development of large, complicated and expensive projects like scientific Projects .Since spiral model approach enables the project term to address the highest risk at the lowest total cost.

References

1. P.K. Agarwal and Manoj Kumar, Consumer Behaviour – An Indian Perspective, Pragati Prakashan, Meerut, 2011, p.60.
2. Software Engineering -Sommerville
3. Software Engineering - Jibitesh Mishra
4. Essentials of Software Engineering -Frank F. Tsui

AUTHOR(S) PROFILE

Shikha Verma Completed the MCA degree in Computer Applications from Guru Gobind Sing Indraprastha University and received B.A degree in Computer Applications from Delhi University in 2013 and 2010, respectively.