© 2014 IJIRT | Volume 1 Issue 4 | ISSN : 2349-6002

Envisioning Software Process Models in SDLC

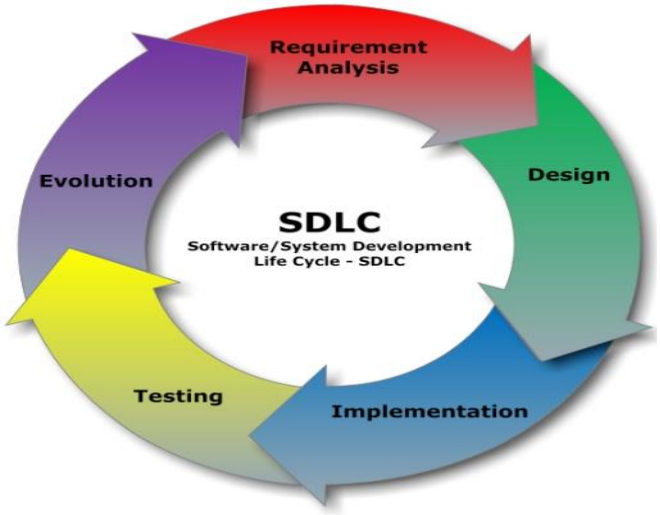
Aabha Sharma, Nikhita Upreti, Divya Bali

***Abstract*- This paper categorizes and examines a number of methods for describing or modelling how software systems are developed. It is concerned with the software management processes that examine the area of software development through the development models, which are known as software development life cycle. It represents five of the development models namely, waterfall, Iteration, V-shaped, spiral and Extreme programming. The life – cycle model (or process model) is a development strategy that encompasses the process, methods, and tools layers that are used to build software.**

***Index Terms*- SDLC, Models, Requirement, Design, Testing**

I. INTRODUCTION

A software development process, also known as a software development life cycle (SDLC), is a structure imposed on the development of a software product. Software development life cycle also known as software development process or simply a software process is a division of software development work into distinct phases or activities with the intent of better planning and management.



A Process Model describes the sequence of phases for the entire lifetime of a product. Therefore it is sometimes also called Product Life Cycle. A software process model is a standardised
format for
• planning
• organising, and
• running
a development project.
The goal of a software project model is to provide guidance for systematically coordinating and controlling the tasks that must be performed in order to achieve the end product and the project objectives.

II. SOFTWARE PROCESS MODELS

The life – cycle model (or process model) is a development strategy that encompasses the
process, methods, and tools layers that are used to build software.It is basically an abstract representation of a process. It presents a description of a process from some particular perspective as:
1. Specification.
2. Design.
3. Validation.
4. Evolution.
Some of the basic software process models are:
1. Waterfall model.
2. Iteration model.
3. V-shaped model.
4. Spiral model.
5. Extreme model.
This research paper comprises of these models because their features correspond to most software development programs.

A. The WaterFall Model

The waterfall model is a sequential software development model in which development is seen as flowing steadily downwards (like a waterfall) through several phases. The waterfall model is believed to have been the first process model which was introduced and widely followed in software engineering. In a waterfall model, each phase must be completed fully before the next phase can begin. This type of model is basically used for the for the project which is small and there are no uncertain requirements. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model the testing starts only after the development is complete. In waterfall model phases do not overlap. Phases in waterfall model are depicted below:

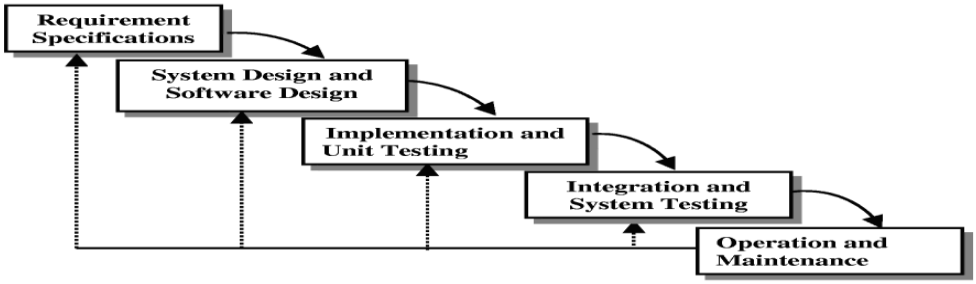
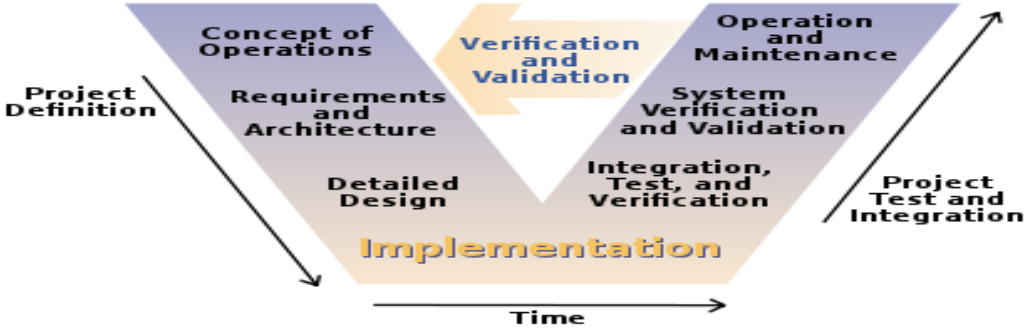


Fig. 1 Waterfall model

Advantages:	Disadvantages:
• Each stage has well defined deliverable or milestone. • It is simple to use and understand. • In this model phases are processed and completed one at a time. Phases do not overlap. • Waterfall model works well for smaller projects where requirements are very well understood.	• You cannot go back a step; if the design phase has gone wrong, things can get very complicated in the implementation phase. • High amounts of risk and uncertainty. • Not a good model for complex and object-oriented projects. • Difficult to integrate risk management. • 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.

B. V SHAPE MODEL

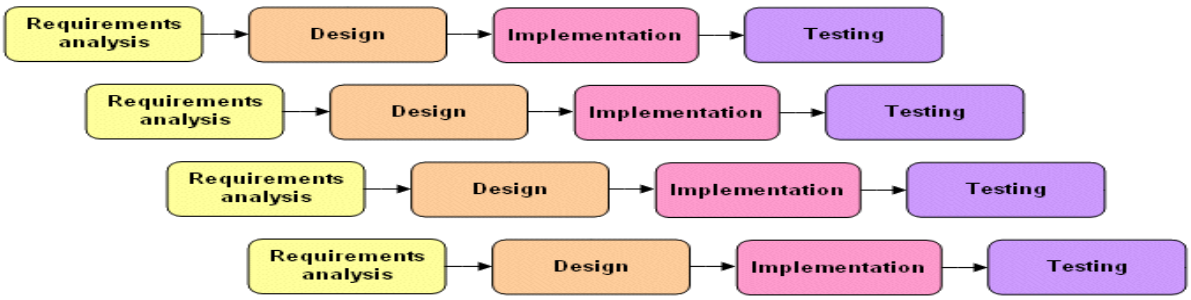
The V-model represents a software development process (also applicable to hardware development) which may be considered an extension of the waterfall model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. Each subsequent phase is begun at the completion of the deliverables of the current phase. It is representative of a comprehensive approach to defining the phases of the software development process. It emphasizes the relationship between the analytical and design phases that precede coding with the testing phases that follow coding.



Advantages:	Disadvantages:
• The model emphasizes planning for verification and validation of the product in the early stages of product development. • It defines the products that the development process should generate; each deliverable must be testable. • It enables project management to track progress accurately; the progress of the project follows a timeline, and the completion of each phase is a milestone. • It is easy to use.	• It does not easily handle concurrent events. • It does not handle iterations of phases. • The model does not contain risk analysis activities. • The model is not equipped to handle dynamic changes in requirements throughout the life cycle.

C. THE INCREMENTAL MODEL

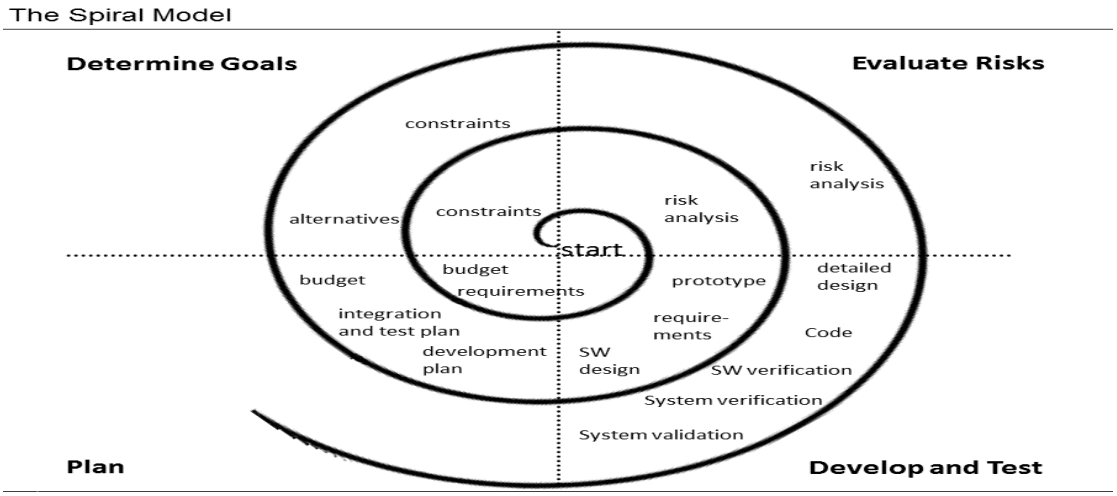
With Iterative Development, the project is divided into small parts. This allows the development team to demonstrate results earlier on in the process and obtain valuable feedback from system users.
Often, each iteration is actually a mini-Waterfall process with the feedback from one phase providing vital information for the design of the next phase. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during development of earlier parts or versions of the system. Multiple development cycles take place here, making the life cycle a"multi-waterfall" cycle. Cycles are divided up into smaller, more easily managed modules. Each module passes through the requirements, design, implementation and testing phases.



Advantages:	Disadvantages:
• Generates working software quickly and early during the software life cycle. • This model is more flexible – less costly to change scope and requirements. • It is easier to test and debug during a smaller iteration. • In this model customer can respond to each built. • Lowers initial delivery cost.	• Needs good planning and design. • Needs a clear and complete definition of the whole system before it can be broken down and built incrementally. • Total cost is higher than waterfall.

D. THE SPIRAL MODEL

The spiral model comprises activities organized in a spiral, and has many cycles. This model combines the features of the prototyping model and waterfall model and is advantageous for large, complex, and expensive projects. It determines requirements problems in developing the prototypes. In addition, it guides and measures the need of risk management in each cycle of the spiral model. The objective of the spiral model is to emphasize management to evaluate and resolve risks in the software project. Different areas of risks in the software project are project overruns, changed requirements, loss of key project personnel, delay of necessary hardware, competition with other software developers and technological breakthroughs, which make the project obsolete.It is similar to the incremental model, with more emphases placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.



Advantages:	**Disadvantages:**
• High amount of risk analysis. • Good for large and mission-critical projects. • Software is produced early in the software life cycle.	• Can be a costly model to use. • Risk analysis requires highly specific expertise. • Project's success is highly dependent on the risk analysis phase. • Doesn't work well for smaller projects

E. EXTREME PROGRAMMING

Extreme Programming is one of several popular Agile Processes. Extreme Programming is successful because it stresses customer satisfaction. Instead of delivering everything you could possibly want on some date far in the future this process delivers the software you need as you need it. Extreme Programming empowers your developers to confidently respond to changing customer requirements, even late in the life cycle. Extreme Programming emphasizes teamwork. Managers, customers, and developers are all equal partners in a collaborative team. Extreme Programming implements a simple, yet effective environment enabling teams to become highly productive. The team self-organizes around the problem to solve it as efficiently as possible. Below is a table on how XP is extreme.

good practices are	pushed to the *extreme*.
Code reviews	review code all the time (pair programming)
Testing	everybody will test all the time (unit testing) even the customers (functional testing)
Design	make it part of everybody's daily business (refactoring)
Simplicity	always leave the system with the simplest design that supports current functionality (simplest thing that could possibly work)
Architecture	everybody will work defining and refining the architecture all the time (metaphor)
Integration testing	integrate and test several times a day (continuous integration)
Short iterations	make iterations really short-seconds, minutes, hours not weeks, months, years (the planning game)

Why it's called eXtreme

Advantages:	**Disadvantages:**
• Robustness: Extreme Programming leverages the power of simplicity. • Resilience: Extreme Programming in-builds accommodation of such changed requirements through getting user stories at the start of iterations, and through feedback during the course of iterations. • Cost Savings: Extreme Programming trims unproductive activities to reduce costs • Lesser Risks • Employee Satisfaction	• It assumes the constant involvement of the customer. • Its success depends on data collection at many stages of the development process. • Extreme Programming code is a centred approach rather than a design-centred approach, and the lack of proper documentation creates problems in large products when project members leave and new members come in later.

III. CONCLUSION

After analysis of all models through the various factors, it has been found that:

1. There are many existing models for developing systems for different sizes of projects and requirements.
2. These models were established between 1970 and 1999.
3. The original water fall model is used by various big companies for their internal projects
4. Each model has its own strengths and weaknesses for the development of systems , so each model tries to eliminate the disadvantages of the previous model.

REFERENCES

[1] http://en.wikipedia.org/wiki/Software_development_process

[2] http://www.the-software-experts.com/e_dta-sw-process.php

[3] http://www.ijarcsse.com/docs/papers/May2012/Volum2_issue5/V2I500405.pdf

[4] http://www.nada.kth.se/~karlm/mvk/mvk08_lec2.pdf

[5] http://samples.jbpub.com/9780763785345/85345_CH04_Tsui.pdf

[6] http://www.ijcsi.org/papers/7-5-94-101.pdf

[7] http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/

[8] http://www.spminfoblog.com/post/73/the-v-shaped-software-development-life-cycle-model/

[9] http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/

[10] http://www.umsl.edu/~sauterv/analysis/f06Papers/Hutagalung/