ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

COMPARATIVE ANALYSIS OF SOFTWARE DEVELOPMENT MODELS

Sandeep Kaur*
*Department of computer Science & Engineering, Guru Nanak Dev University Regional Campus, Sathiala, India

DOI: 10.5281/zenodo.1036644

ABSTRACT

No geek is unfamiliar with the concept of software development life cycle (SDLC). This research deals with the various SDLC models covering waterfall, spiral, and iterative, agile, V-shaped, prototype model. In the modern era, all the software systems are fallible as they can't stand with certainty. So, it is tried to compare all aspects of the various models, their pros and cons so that it could be easy to choose a particular model at the time of need

KEYWORDS: Software development life cycle, development phases, waterfall model, spiral model, V-shaped model, Prototype style, Iterative model, associate between models.

I. INTRODUCTION

[7] Software engineering is an application of systematic, disciplined, quantifiable approach to development, operation and maintenance of software. It integrates significant mathematics, computer science as well as practices whose origins are in engineering.[6] Software development process is also known as Software development life cycle(SDLC) which is a structure that represents the various phases during the development of the software product.

[1] Development model can be understood as a prescriptive as well as a descriptive model. Descriptive model as it describes the history of how a particular software system was developed. It can be used as basis for understanding & improving development process. Prescriptive model as it gives us prescription how a new software system should be developed and used as a guideline to organize software development activities. Prescriptive model dominates over descriptive one as prescriptive model is easier and common to use and also one must have to collect the data throughout the life cycle of system in descriptive model.

Software projects are still vulnerable to unfamiliar problems that are large in size, complicated and are large in size. So, there is a great need to make some kind of standard that defines all the tasks that are required to develop & maintain software. We are having various models that uses different approaches.

II. PHASES

Lifecycle is a characteristic of how software should be developed. As it is known that software has to go through various phases, so let them study in brief:

2.1 Planning: It is the first phase in which all the components to build the system are established and all the requirements to be met are decided. It is deciding a plan for a solution.
2.2 Feasibility study: [9] it is assessment of the practicality of a proposed project. The two criteria to judge feasibility are cost required and value to be attained.
2.3 Designing: it determines the framework of a system to meet the specific requirements. It produces a specification for how each component is implemented.
2.4 Implementation: it is the coding phase in which program code is made in relevance to design. The coding should be done using suitable language.[8] It implements the detailed design specification.
2.5 Testing: It is used to check out and correct the errors, if any found in the system. It is done to check whether software meets the specified requirements.
2.6 Documentation: It is written text or illustration that accompanies software.[9] It either explains how it operates or how to use it.

2.7 Maintenance: It is modification of software product after delivery to correct faults, to improve the performance.

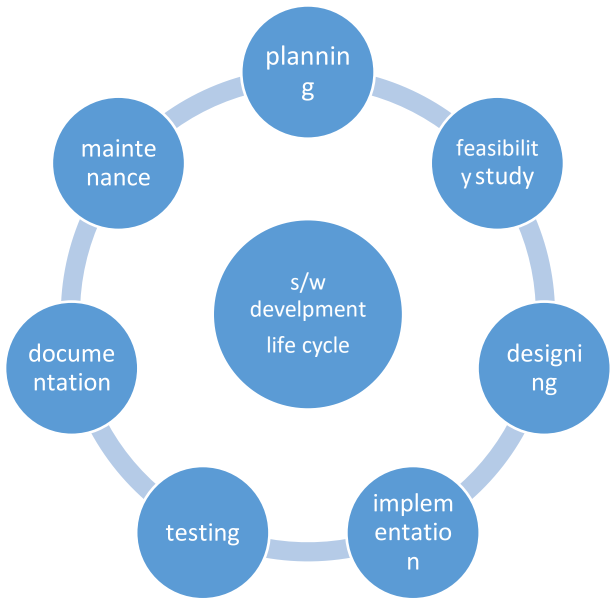


Figure 1: Phases of development life cycle

III. WATERFALL MODEL

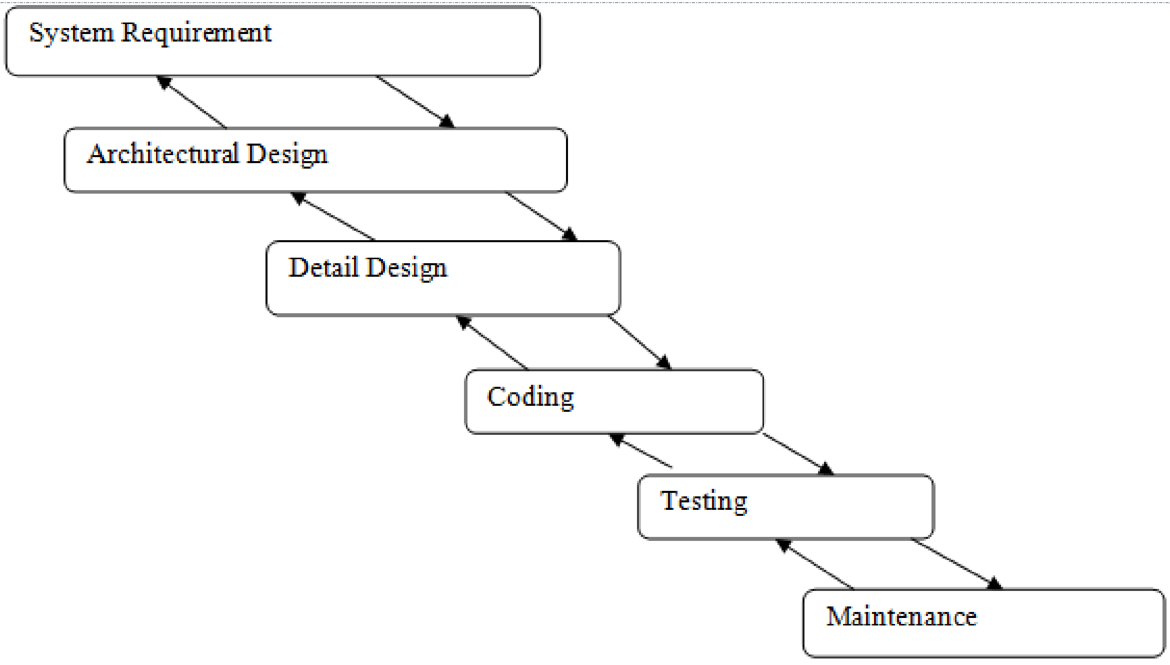
[5] It is classical model of software engineering proposed by Royce . As, it emphasis planning in early stages, it ensures design flaws before they develop.

IV. ORIGINAL WATERFALL MODEL

It serves as a baseline for many other models.[5] It comprises of various non-overlapping stages, from establishing system & software requirements and continues with system design, detailed design, coding ,testing & maintenance.

Table 1: Strengths & weaknesses of original waterfall model

Strengths	Weaknesses
Minimizes planning overhead as it can be done at initial stages.	Coming back to the back point is difficult.
Works well for technically weak & inexperienced staff	The final phase produces non-documentation that can be derivable.
Acts as a template into which methods for analysis, design , code and test can be placed.	Small changes or errors may cause serious problems.
Simple to use and understand.	Software can't be handled to clients until final phase is completed.
Define-before-design & design-before-code.	Idealized, not very much real.
Document driven.	Difficult to integrate risk management.
Identifies deliverables & milestones.	Difficult & expensive to swim upward.



V. MODIFIED WATERFALL MODEL

[8] It uses the same phases as that of original but is not based on discontinuous basis. Using this, the phases can be overlapped if needed.

Table 2. Strengths & weaknesses of modified waterfall model

Strengths	Weaknesses
Implementation of easy areas don't need to wait for hard ones.	Unforeseen inter-dependencies can create problems.
More flexible.	Concurrency can't be achieved.

VI. ITERATIVE MODEL

It is proposed to overcome to downfalls of waterfall model. [5]It don't attempt to start with a full specification of requirements. Instead, development begins by specifying & implementing just part of the software, which can be reviewed in order to identify further requirements. This process is repeated, until all specifications are met. Each sub-process is a mini-waterfall process that provide feedback to other phases.

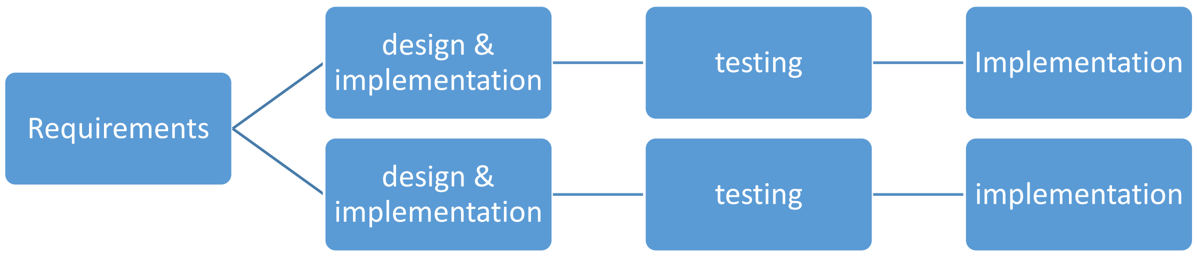


Figure 2:Iterative Model [5]

Table 3:Strengths & Weaknesses of iterative model

Strengths	Weaknesses
Better than waterfall model	No clear milestones
Provides feedback	Difficult to manage
Used when requirements are not well clear	No stage is really finished.

VII. PROTOTYPE MODEL

[6] It is the development approach of activities during software development, the creation of prototypes, i.e., incomplete versions of software program being developed. It don't freeze the requirements before design or coding. Prototypes are usually not complete systems. By using prototype, the client can get the actual feel of the system.

When to use: When the desired system needs to have a lot of interaction with end users. Online systems, web interfaces have a very high amount of interaction with end users. They are excellent to design good human computer interface system.

Table 4: Strengths & Weaknesses of prototype model

Strengths	Weaknesses
User involvement	Unsuitable for large apps.
User get a better understanding as working model is provided	May produce system inadequate for overall organization needs
Error detection is easy Missing functionality can be identifies easily	Possibility of causing systems to be left unfinished.
Quick user feedback	Difficult to manage the project.
Cost effective	Increase the complexity

VIII. SPIRAL MODEL

It was the first model to explain why the iteration matters.[6] It combines the elements of both design & prototyping-in-stages, in order to combine advantages of top down and bottom up approaches. It moves in clockwise direction beginning from center position and provide deliverable at each traversal.
When to use:For big and mission- critical projects where the cost of risk mgmt is not an issue.

Table 5: Strengths & Weaknesses of Spiral Model.

Strengths	Weaknesses
Software is produced in early stages.	Risk identification, assessment & mgmt is difficult.
More customer involvement.	Cost & time estimations are also not easy.
Good for large & mission-critical projects.	Not suitable for small small projects.
Better productivity through reuse capabilities.	Risk analysis requires highly specific expertise.

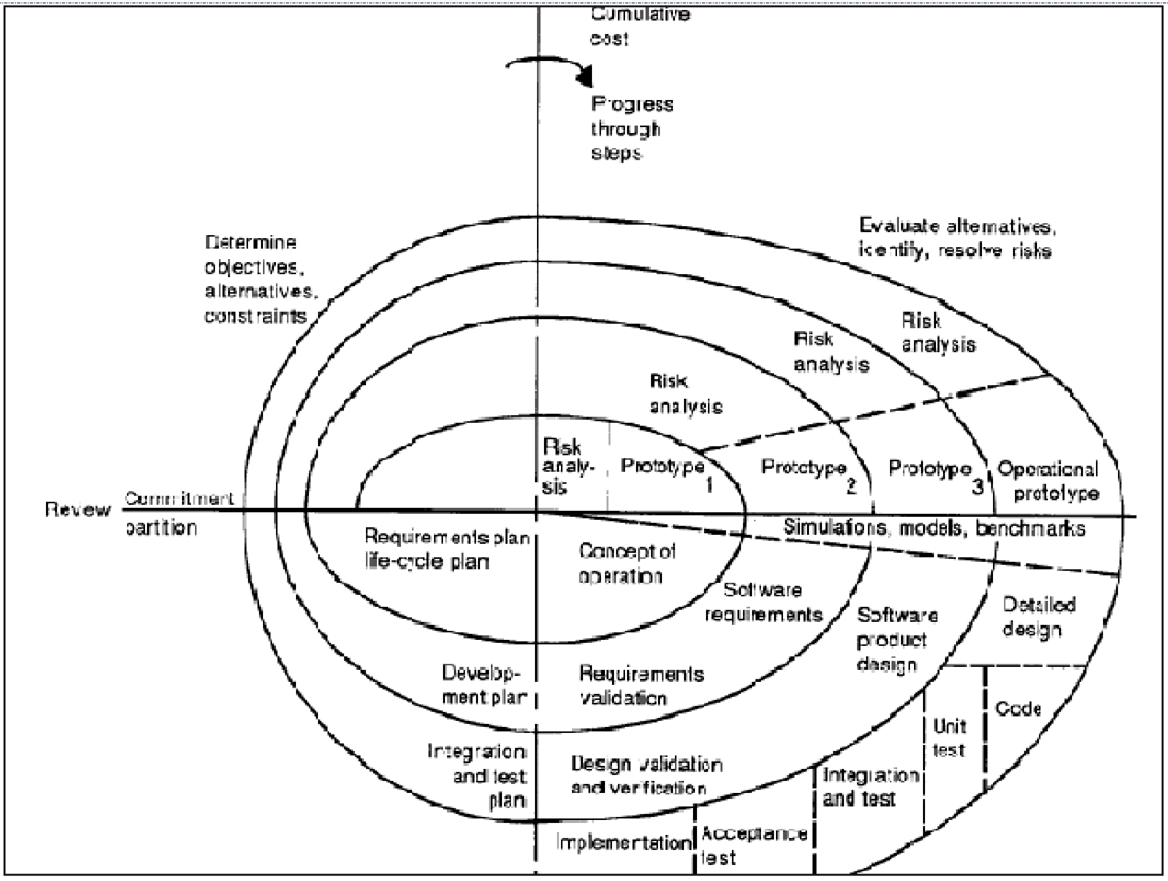


Figure 3: spiral model

IX. V-SHAPED MODEL

It is quite similar to the waterfall model, as it is a sequential path of execution of processes , initializing with the requirement phase. Each phase must be completed before the next phase begins. More emphasis is given to testing phase. [6] The high level design phase focuses on system architecture & design. An integration test plan is created in order to test the pieces of software systems ability to work together. The low level design phase is where the actual software components are designed and unit tests are created. Once the coding is finished, the path of execution continues up the right side of V where the test plans developed earlier are now put to use.

Table 6: Strengths & Weaknesses of V-Shaped Model

Strengths	Weaknesses
Simple and easy to use	Very rigid
Each phase has specific deliverables.	Adjusting scope is expensive
Works well when requirements are clear	No early prototypes are developed

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

Table 7: Comparison Table

Features	**Waterfall Model**	**Iterative Model**	**Prototype Model**	**Spiral Model**	**V-Shaped Model**
Requirement Specification	Beginning	Beginning	Changes Frequently	Beginning	Beginning
Understanding Requirements	Well Understood	Not well understand	Not well understand	Well understand	Well understood
Flexibility	Rigid	Less flexible	Highly Flexible	Flexible	Rigid
Implementation time	Long	Less	Less	Project Dependent	Long
Phase overlap	No	No	Yes	Yes	No
Cost	Low	Low	High	Expensive	Expensive
Complexity	Simple	Simple	Complex	Complex	Simple
Reusability of components	No	Yes	Yes	Yes	No
User Involvement	Only at Beginning	Intermediate	High	High	Intermediate
Risk Analysis	Only at beginning	No Risk Analysis	No Risk Analysis	Yes	Yes
Gurantee of success	Less	High	Good	High	High
Plus points	Minimizes planning overhead, simple to use & understand.	Better than waterfall, provides feedback	Quick user feedback, cost effective, error detection is easy.	Better productivity, More user involvement.	easy to understand, each phase provides deliverables.
Downsides	Freeze requirement, difficult to manage risk.	No stage is actually finished, difficult to manage.	Increased complexity, Difficult to manage.	Not suitable for small projects,risk mgmt is difficult.	Very rigid, don't provide early prototypes.
Suitable	Technically weak & inexperienced staff	When user requirements are not clear.	For small applications.	For big & mission critical systems.	Small projects.

X. CONCLUSION

This paper presents various software development models. No doubt, there are numerous models, but only 5 have been discussed that are mostly used. It is tried to include all the strengths, weaknesses and their suitable use. Every model tries to uplift the downsides of the previous model . Research on this topic will never end. Now, it would be much easy to choose the best model as per the requirements

XI. REFERENCES

[1] Walt Scacchi, Institute for Software Research, University of California, Irvine: Process Models in Software Engineering, research paper ,Feb 2001.
[2] Joruts LBwgret1:" Applying Design Methodology to Software Development", research paper.
[3] Rupinder Kaur, Jyotsna Sengupta :A New Approach to Software Development Fusion Process Model ,J. Software Engineering & Applications, 2010, 3, 998-1004 doi:10.4236/jsea.2010.310117 Published Online October 2010 (http://www.SciRP.org/journal/jsea).
[4] Jasmine K.S, Dr. R. Vasantha:" A New Process Model For Reuse Based Software Development Approach"Proceedings of the World Congress on Engineering 2008 Vol IWCE 2008, July 2 - 4, 2008, London, U.K.

[5] Vanshika Rastogi:”Software Development Life Cycle Models-Comparison, Consequences”, research paper(IJCSIT) , 2015.
[6] Ms.shikha maheshwari, prof. Dinesh Ch.Jain:”A Comparative Analysis of different types of models in software development life cycle”.research paper (www.ijarcsse.com) , May 2012.
[7] IanSommerville, SoftwareEngineering, Addison Wesley, 9th ed. ,2010.
[8] Nabil Mohammed Ai Munassar , A. Govardhan:”A Comparison between five models of software engineering”, research paper(www.ijcsi.org), sept 2010.
[9] wikipidea.

CITE AN ARTICLE

Kaur, S. (2017). COMPARATIVE ANALYSIS OF SOFTWARE DEVELOPMENT MODELS. *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY, 6*(10), 611-617.