Research article Available online www.ijsrr.org ISSN: 2279–0543

International Journal of Scientific Research and Reviews

A Review of Conventional SDLC Process Models

K. Sunil Manohar Reddy[1*] **and V. Vinay Kumar**[2]

Associate Professor, Dept. of CSE, Matrusri Engineering College, Hyderabad.
Email: sunil186@gmail.com

ABSTRACT

This paper reviews software development life cycle models that are used in the area of software development. It elucidates about various advantages and disadvantages of each model, according to which, it can be decided which model should be used under which conditions. These models are of the following two types: traditional models and contemporary models. The Waterfall model, Incremental Model, Spiral Model and V-Shaped Model are traditional models which follow a set of prescribed steps. The Contemporary models widely used in industries are Rapid Application Development Model, Agile Development Model and Extreme Programming Model.

KEYWORDS: Software development life cycle (SDLC), Software models, Traditional Models, Contemporary Models and Agile teams.

***Corresponding author:**

K. Sunil Manohar Reddy

Associate Professor,

Dept. of CSE, Matrusri Engineering College,

Hyderabad.

Email: sunil186@gmail.com

INTRODUCTION

Software engineering is a coherent, methodical and structured approach used for development, performance and maintenance of software products. This implies that when different group of people apply same methodologies of software, similar software will be produced. It is the application of engineering to software to develop efficient software products that are able to meet the constraints of cost, quality and time. System Development Life Cycle in Software engineering refers to the process of constructing or fabricating systems, models and techniques used for their development. Software Development Life Cycle provides sequence of operations for software developers to develop the software in a manner such that it is completed within deadlines and quality of the product of software is maintained as per the standards laid down by the developers. It is often considered as a subset of System Development Life Cycle.

PHASES OF SDLC

The various activities carried out for software development can be divided into manageable sections called as phases. These phases may be carried out in different way as per the need. Generally, there are five common phases in SDLC:

1. Requirement Gathering & Analysis
2. Designing
3. Coding
4. Testing
5. Maintenance and Support

The requirement gathering and analysis phase helps to understand the problem. This is followed by deciding a plan to solve the problem in the designing phase. The planed solution is then implemented during coding. The Solution program is tested against various test cases. Deployment and maintenance follows this stage. There are various software development approaches designed to develop the software products. These are known as Software Life Cycle Models.

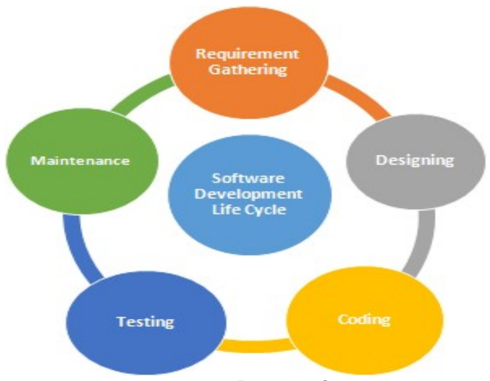

Figure1. Phases of SDLC

SDLC MODELS

These models describe the various phases and the order of their execution to be followed to develop the software product[1]. Each phase produces deliverables that are fed as input to the next phase in the life cycle. The product generated by the last phase serves as the final product called as software product. Different models proposed so far are:

Conventional software models are characterized by linear nature, that is, the various phases of SDLC are carried out sequentially[2]. Building the software product initiates with the visualization of the final software, and continues working through to build the final visualized software product. Some of the traditional models are:

1) Waterfall Model
2) Incremental Model
3) Spiral Model

1. WATERFALL MODEL: Waterfall model was proposed by Winston Royce in 1970. It is known as the classical and basic model of Software Engineering. It is a linear sequential SDLC model as various phases are carried out in a sequence and development is seen as flowing downwards through following phases:

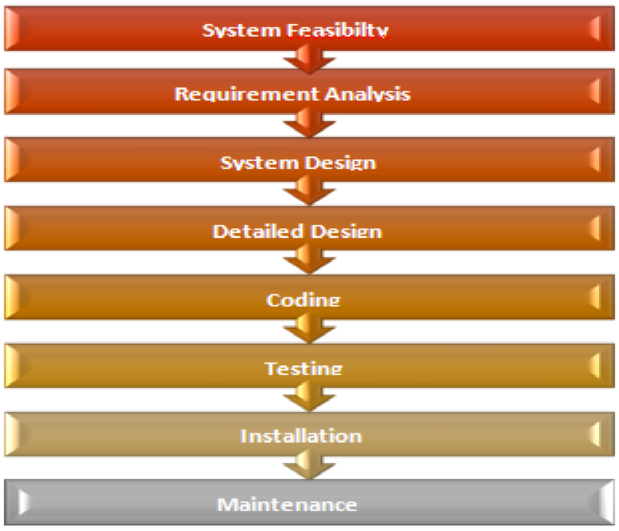

Figure2. Waterfall Model

Each phase must be completed before the next phase begins. Extensive documentation along with validation and verification is involved.

Advantages of Waterfall Model:

1. It is simple to use and understand.
2. Verification and Validation prevent error propagation.
3. Each stage has well defined milestone.

Disadvantages of Waterfall Model:

1. Requirements once decided cannot be changed, i.e., requirements are frozen.
2. Going back to a previous phase is not possible.
3. No intermediate product is formed, all or nothing approach exists.

Areas of Usage of Waterfall Model:

1. Project is of short duration.
2. Automating any existing manual system.

2. INCREMENTAL MODEL: The Incremental Model combines the elements of linear model with iterative prototyping. In prototyping, incomplete versions of the software program are created which are either discarded or developed further into final products based upon user response. Incremental Model implements prototyping repeatedly in a sequential manner. This model prioritizes system requirements and then implements them subsequently to previously developed prototype[3]. This process is repeated until the desired functionality is achieved by software product.

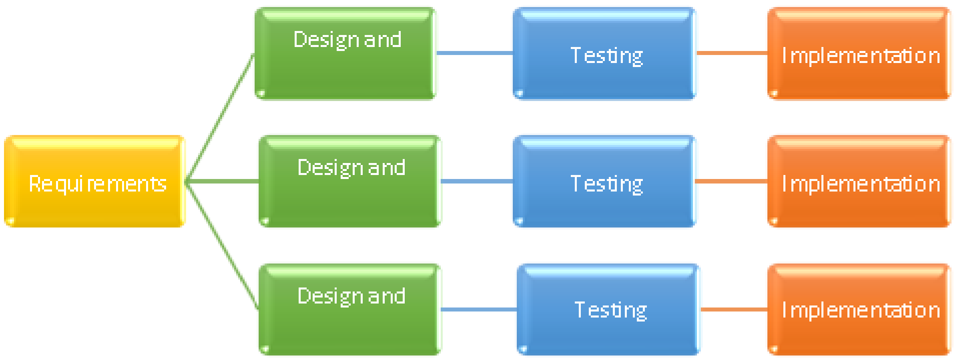

Figure3. Incremental Model

Advantages of Incremental Model:

1. Testing and debugging each increment is easy.
2. Chances of failure are less.
3. Generates working software quickly

Disadvantages in Incremental Model:

1. Require high quality of planning and design.
2. Involves project management difficulties.
3. Requires clear understanding of entire system to break it down and develop incrementally.

Area of Usage of Incremental Model:

1. When product has to reach market early.
2. When new technology is being tested.

3. SPIRAL MODEL: Spiral model was defined by Barry Boehm in 1988. It is similar to Incremental Model with more focus on risk analysis[4]. The Spiral Model involves four phases:

planning, risk analysis, engineering and evaluation. A software project passes through these phases repeatedly in iterations corresponding to various spirals in the model. The baseline spiral starts in the planning phase. The requirements are gathered and risk assessment is done. Subsequent spirals are built on the baseline spiral. During risk analysis phase, risks are identified and alternative solutions are suggested. Prototype is produced at the end of risk analysis phase. At the end of engineering phase, tested software is produced. During evaluation phase, the customer evaluates output of project before it continues to next spiral. In this model angular component measures progress while radii represent cost.

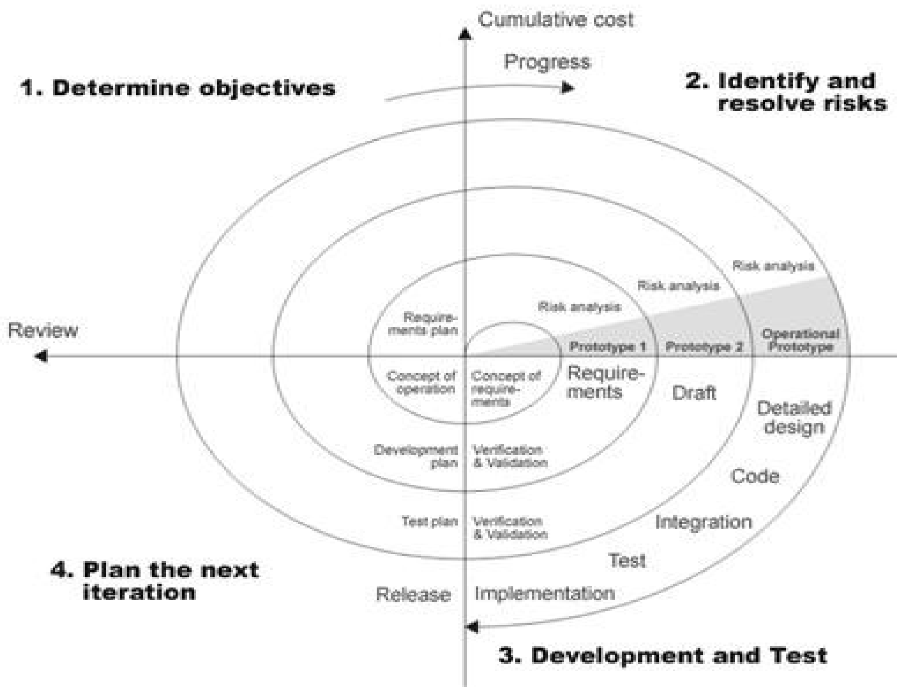

Figure4. Spiral Model

Advantages of Spiral Model:

1. High amount of risk analysis.
2. Early production of software in the life cycle.
3. Good for large projects.
4. Development can be terminated after any spiral and the working system will be available.

Disadvantages of Spiral Model:

1. Cost and time estimations are difficult.
2. Not suitable for small projects as cost of risk analysis is greater than cost of entire project.
3. Risk analysis requires high expertise.

Area of Usage of Spiral Model: Requirements are fuzzy and complex.

1. For medium to high risk projects.
2. New product line.

CONCLUSION

There are many software development life cycle models, each claiming to be the best[5]. This paper provides an overview of few of the conventional software models which laid a base to the contemporary software process models. No model is superior to any other and these models are used depending upon the requirements and prevailing conditions. Each model has its own advantages and disadvantages; therefore a hybrid of these methodologies is developed and chosen for different projects according to the requirements.[6] Recent time has shown a shift from traditional models to contemporary models of software development. Selecting the correct life cycle model can help to deliver the required software product within deadline, meeting quality and cost constraints. Further, more changes and evolutions are continuously being performed to increase the quality of software being developed by improving the methodologies used.

REFERENCES

1. Mishra A., Dubey D., "*A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios*", IJARSMS, October 2013; 1: 64-69.
2. Maheshwari S., Jain Dinesh Ch., "*A Comparative Analysis of Different types of Models in Software Development Life Cycle*", IJARSMS, May 2012; 2(5).
3. K. Sunil Manohar Reddy, "*Requirements Engineering: An Overview*", IJRECE**,** June 2019; 7(2): 2930-2939.
4. Kute S., Thorat S., "*A Review on Various Software Development Life Cycle(SDLC) Models"*, IJRCCT, July – 2014; 3(7).
5. Wallin C., Land R., "Software Development Lifecycle Models The Basic Types"
6. K. Sunil Manohar Reddy, "*A Review on Agile Unified Process Model*", IJRECE**,** June 2019; 7(2): 2768-2770.