Volume 5, Issue 11, November 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Review of Software Development Life Cycle Models

Shubhmeet Kaur
B. Tech CE Final Year, Dept of Computer Engineering
Punjabi University, Punjab, India

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.

I. 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.

II. 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:

A. *Requirement Gathering & Analysis*
B. *Designing*
C. *Coding*
D. *Testing*
E. *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.

Fig. 1 Phases of SDLC

III. SDLC MODELS

These models describe the various phases and the order of their execution to be followed to develop the software product. Each phase produces deliverable 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:

A. Traditional models

1) Waterfall Model
2) Incremental Model
3) Spiral Model
4) V-shaped Model
5) RAD Model

B. Contemporary Models

1) Agile Model
2) Extreme Programming Model

IV. TRADITIONAL MODELS

Traditional software development models are those that are characterized by linear nature, that is, the various phases of SDLC are carried out sequentially. Building the software product initiates with the visualization of the final software, and continues working through to build the final visualized software product. A few traditional models are discussed below.

A. WATERFALL MODEL

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

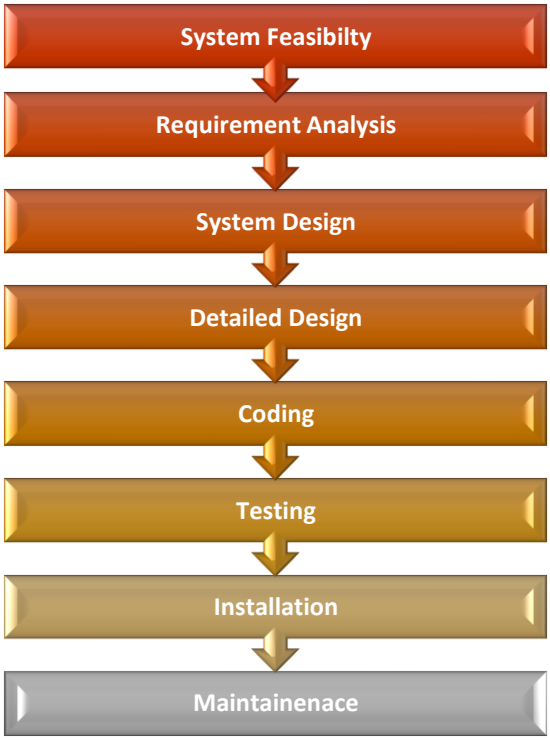

Fig. 2 Waterfall Model

1) Each phase must be completed before the next phase can begin.
2) 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, that is requirements are frozen.
2) Going back to a previous phase is not possible.
3) Life cycle of this model is too long.

4) No provision for user feedback
5) May use outdated technology and hardware.
6) Document driven model.
7) No intermediate product is formed, all or nothing approach exists.

Area of Usage of Waterfall Model

Requirements are well known and are fixed.

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

B. 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. The Incremental Model implements prototyping repeatedly in a sequential manner. This model prioritizes system requirements and then implements them subsequently to the previously developed prototype. This process is repeated until the desired functionality is achieved by software product.

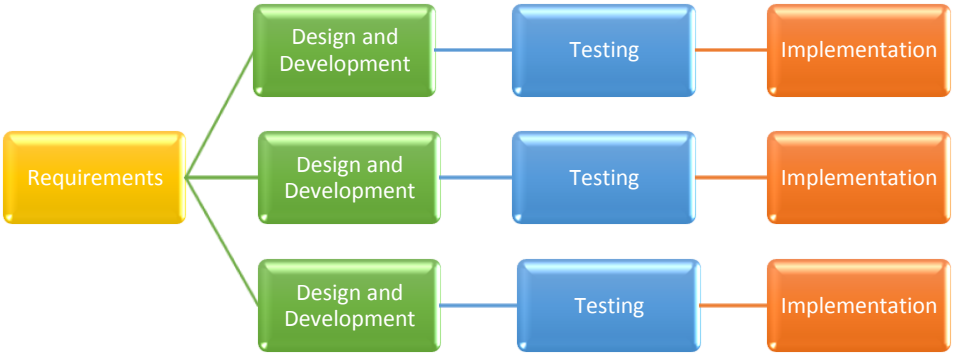

Fig. 3 Incremental Model

Advantages of Incremental Model

1) Very adaptive and changes are less costly.
2) Testing and debugging each increment is easy.
3) Chances of failure are less.
4) User feedback is taken after every iteration.
5) Generates working software quickly

Disadvantages in Incremental Model

1) Each phase is rigid and no overlapping is allowed.
2) Require high quality of planning and design.
3) Involves project management difficulties.
4) Requires clear understanding of entire system so that it can be broken down and developed incrementally.

Area of Usage of Incremental Model

1) Major requirements of the system are clearly defined and intricate details are not known.
2) Product has to reach market early.
3) New technology is being tested.

C. SPIRAL MODEL

Spiral model was defined by Barry Boehm in 1988. It is similar to Incremental Model with more focus on risk analysis. 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.

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) Chances of failure are very low.
5) Development can be terminated after any spiral and there will be working system available.

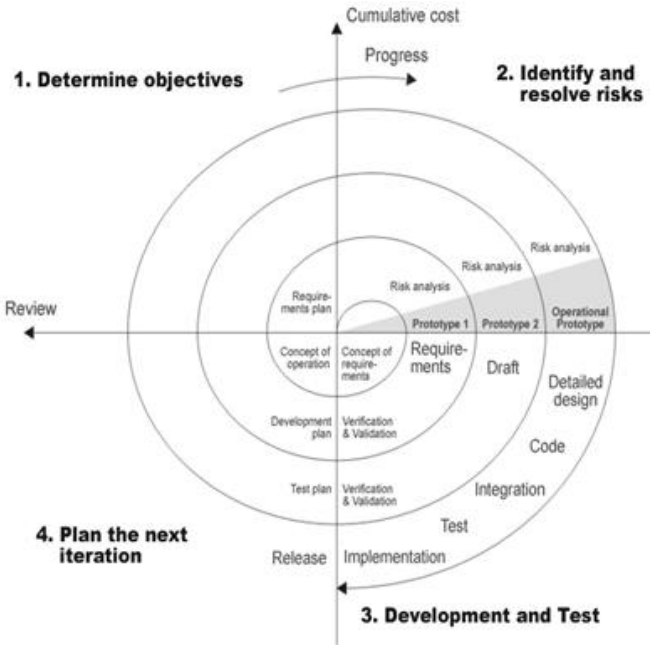

Fig. 4 Spiral Model [9]

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.
4) Success of the project depends on risk analysis.

Area of Usage of Spiral Model

Requirements are fuzzy and complex.

1) For medium to high risk projects.
2) New product line.
3) Changes are expected during development of the product.

D. V-SHAPED MODEL

V Model refers to verification and validation model. Like the waterfall model, the V-Shaped life cycle model is a sequential model that is each phase must be completed to begin the next phase. In this model, testing is done simultaneously with the development phase which means the earlier tasks are verified later.

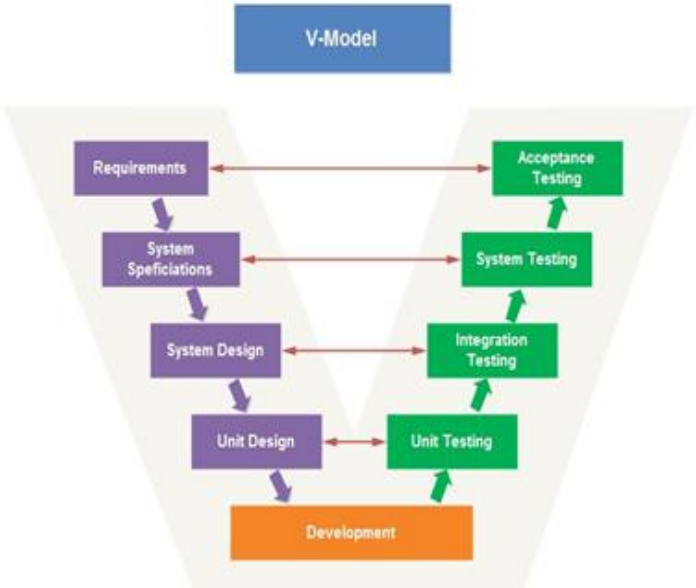

Fig. 5 V-Shaped Model [10]

Advantages of V-Shaped Model

1) Easy to plan and understand.
2) Defects found can be tracked and solved at early stages.

3) Testing activities happen before coding, thus saving a lot of time.
4) Higher chance of success over Waterfall Model.
5) Prevents downward flow of errors and defects.

Disadvantage of V-Shaped Model

1) Very rigid and least flexible model.
2) No early prototypes or end products available.
3) If changes happen in midway, test and requirement documents need to be updated.
4) No clear path indicated for problems found during testing phases.
5) High risk involved.

Area of Usage of V-Shaped Model

1) For small to medium size projects where requirements are clear and fixed.
2) Ample technical resources and expertise are available.
3) High confidence of customer exists since no prototype is produced.

V. CONTEMPORARY MODELS

The contemporary software development models are based upon the principle of iteration of SDLC processes to give more importance to adaptability during development of the software. A few contemporary models are discussed below.

A. RAPID APPLICATION DEVELOPMENT MODEL

Rapid Application Development model is a type of incremental model. In this model,
focus is on developing quality product in less time. In RAD Model, components are developed in parallel like several mini projects and are quickly delivered. This gives the customers something to see and use and also provide feedback regarding the delivered product and any change in their requirements can be handled. Thus, the development team can deliver a fully functional system within a very short period.

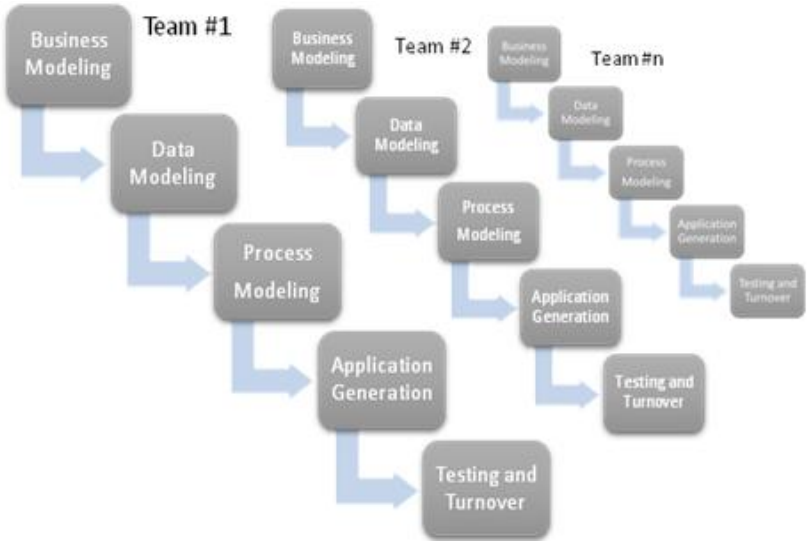

Fig. 6 RAD Model

Advantages of RAD Model

1) Reuses code and processes.
2) Provides great flexibility.
3) Encourages user involvement.
4) Chances of failure and defects are quite low.
5) Development time is reduced significantly.
6) Estimation of cost is easy.

Disadvantages of RAD Model

Not useful for large and complex projects.

1) Quality parameters are often ignored.

Area of Usage of RAD Model

1) Small and medium sized projects where time is an important aspect and quality can be ignored.

B. AGILE SOFTWARE DEVELOPMENT MODEL

The agile software development model was introduced by the agile team in 2001 through the agile manifesto. It focuses on early and continuous delivery of software, through iterative and incremental development, so as to achieve customer satisfaction. The main attributes of the agile model are:

1) Incremental: Small software releases, accompanied by rapid development cycles
2) Co-operative: Closer customer-developer interaction
3) Adaptive: flexible enough to adapt to last moment changes

Agile software development is largely dependent upon the agile team, which is a cross-functional group of individuals responsible for defining, building and testing the solution software. These teams are highly jelled and their working is based upon trust and interaction. They have the ability and the authority to develop the agile software.

Advantages of Agile Model

1) Quick Releases
2) Meets changing requirements
3) More emphasis on customer feedback
4) Measures real-time progress and is more dynamic
5) Incurs less overhead
6) Bad designs and wrong requirements identified and removed immediately.

Disadvantages of Agile Model

1) Cannot be used for large, complex projects
2) Works well only when teams are small

Area of usage of Agile Model

1) Small projects that are developed by small, self-organizing teams
2) Requirement changes and technology changes are frequent

C. EXTREME PROGRAMMING MODEL

The Extreme Programming (XP) is an agile development methodology used for developing software in an unstable environment. It allows flexibility to changes during the development process, thereby lowering the cost of change in requirements during the later development phases. The main features of extreme programming are:

1) Small releases and continuous feedback
2) Refactoring is done after each stage instead of waiting till the end of the process to correct flaws
3) Collective Ownership of the code by each programmer in the XP team facilitates code change whenever needed without ownership issues
4) Pair programming is done
5) Customer is available at all times to address issues and assign priorities
6) Integration and testing is done multiple times to ensure whole team is on the same stage of development

Advantages of Extreme Programming

1) Changes are less costly
2) Improved coordination & knowledge sharing among programmers during pair programming
3) Close customer participation

Disadvantages of Extreme Programming

1) Too much involvement by customer can be problematic for the developing team
2) Standardized coding guidelines have to be followed to ensure code is understandable by the whole team

Area of usage of Extreme Programming

1) Requirements changes are frequent
2) Releases must be quick
3) Management allows development and changes anytime, without documentation

VI. CONCLUSION

There are many software development life cycle models, each claiming to be the best. This paper provides an overview of few of these 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. 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.

ACKNOWLEDGMENT

I wish to express my sincere gratitude to Mrs Brahmaleen Kaur Sidhu, Assistant Professor, Department of Computer Engineering, University College of Engineering, Punjabi University Patiala. I would like to thank her for her immense support and guidance during the entire work. Her contribution made this review work a success.

REFERENCES

[1] Mishra A., Dubey D.," *A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios*", IJARSMS, Volume 1,Page 64-69,October 2013.

[2] Maheshwari S., Jain Dinesh Ch.,"*A Comparative Analysis of Different types of Models in Software Development Life Cycle*", IJARSMS, Volume 2, Issue 5, May 2012.

[3] Kute S., Thorat S. ," *A Review on Various Software Development Life Cycle(SDLC) Models"*, IJRCCT, Volume 3, Issue 7, July – 2014

[4] Abrahamsson P., Warsta J., Siponen M. and Ronkainen J., "New Directions on Agile Methods: A Comparative Analysis", *Proceedings of the International Conference on Software Engineering*, May 3-5, 2003, Portland, Oregon, USA.

[5] Singh G., Tamanna, "*An Agile Methodology Based Model for Software development*", IJARCSSE, Volume 4, Issue 6, June 2014

[6] Wallin C., Land R., "Software Development Lifecycle Models The Basic Types"

[7] Ragunath P., Velmourougan S., Davachelvan P., Kayalvizhi S., Ravimohan R., "*Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC)*", IJCSNS, VOL.10 No.1, January 2010

[8] Dora S., Dubey P., "*Software Development Life Cycle (SDLC) Analytical comparison and survey on Traditional and agile methodology*", NMRJRST, VOLUME NO.2, ISSUE NO.8

[9] http://testingart.com/software-development-life-cycle-models/

[10] http://leansoftwareengineering.com/2008/05/05/boehms-spiral-revisited/

