International Journal of Advance Research in Science and Engineering
Volume No.06, Special Issue No.(01), Nov 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

Process Models in Software Engineering

Ramanpreet Kaur[1], Ramandeep Kaur [2]

[1,2]*Department of Computer science, Baba Farid Group of Institutions*

ABSTRACT

Modeling is a concept fundamental for software engineering. In this paper, the word is defined and discussed from various perspectives. The most important types of models are presented, and examples are given.Models are very useful, but sometimes also dangerous, in particular to those who use them unconsciously. Such problems are shown. Finally, the role of models in software engineering research is discussed.

Keywords— ***Software engineering, SDLC, Risk analysis, Software Management Processes, Software Development, Development Models***

I. INTRODUCTION

Models of Software engineering: Software systems come and go through a series of passages that account for their inception, initial development, productive operation, upkeep, and retirement from one generation to another. This article categorizes and examines a number of methods for describing or modeling how software systems are developed[2]. It begins with background and definitions of traditional software life cycle models that dominate most textbook discussions and current software development practices Software development life cycles are phased process with clearly identifiable goals and tasks. Software Process Models a software process model is an abstract representation of a process [3]. It presents a description of a process from some particular perspective as:

1. Specification.
2. Design.
3. Validation.
4. Evolution.

Component-based software engineering: The system is assembled from existing components. There are many variants of these models e.g. formal development where a waterfall-like process is used, but the specification is formal that is refined through several stages to an implementable design [1].

There are several types of models are given by below:-

II. WATERFALL MODEL

The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin. This type of **software development 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 **software testing** starts only after the development is complete.

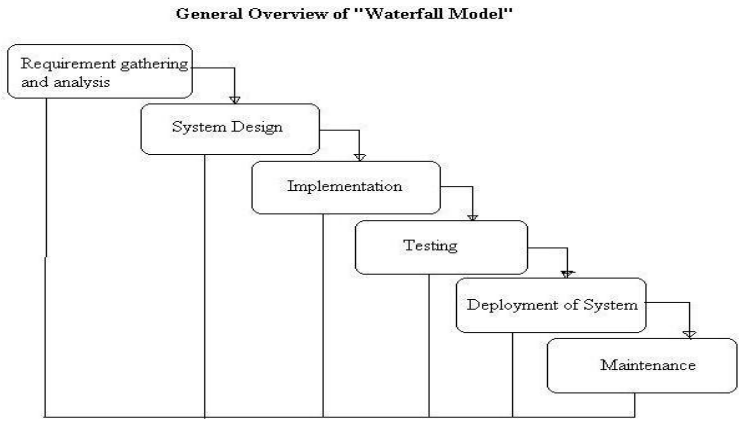


Fig. 1 Waterfall Model [4].

When to use the waterfall model:

- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements
- Ample resources with required expertise are available freely
- The project is short.

The waterfall method does not prohibit returning to an earlier phase, for example, returning from the design phase to the requirements phase. However, this involves costly rework. Each completed phase requires formal review and extensive documentation development. Thus, oversights made in the requirements phase are expensive to correct later. Because the actual development comes late in the process, one does not see results for a long time. This delay can be disconcerting to management and customers. Many people also think that the amount of documentation is excessive and inflexible. Although the waterfall model has its weaknesses, it is instructive because it emphasizes important stages of project development. Even if one does not apply this model, he must consider each of these stages and its relationship to his own project [4].

Advantages:•

1. Easy to understand and implement.
2. Widely used and known (in theory!).
3. Reinforces good habits: define-before- design, design-before-code.
4. Identifies deliverables and milestones. 5. Document driven
6. Works well on mature products and weak teams.

Disadvantages:•

1. Idealized doesn't match reality well.
2. Doesn't reflect iterative nature of exploratory development.

International Journal of Advance Research in Science and Engineering
Volume No.06, Special Issue No.(01), Nov 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

3. Unrealistic to expect accurate requirements so early in project.
4. Software is delivered late in project, delays discovery of serious errors.
5. Difficult to integrate risk management.
6. Difficult and expensive to make changes to documents,"swimming upstream".
7. Significant administrative overhead, costly for small teams and projects [8].

III. SPIRAL MODEL

The spiral model is similar to the **incremental model**, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations called Spirals in this model.

Planning Phase: Requirements are gathered during the planning phase. Requirements like 'BRS' that is 'Bussiness Requirement Specifications' and 'SRS' that is 'System Requirement specifications'.

Risk Analysis: In the **risk analysis phase**, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

Engineering Phase: In this phase software is **developed**, along with **testing** at the end of the phase. Hence in this phase the development and testing is done.

E**valuation phase:** This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

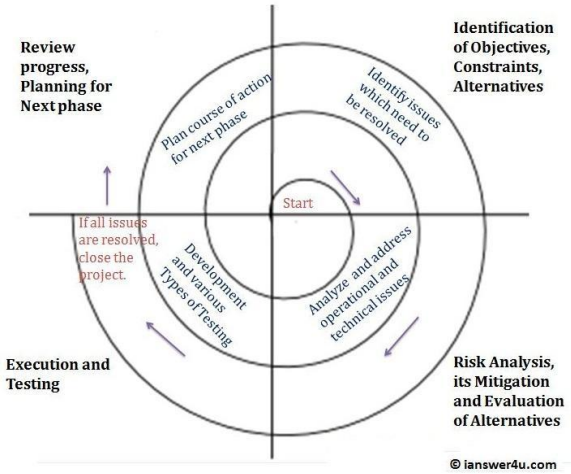


Fig. 2 Spiral Model[3].

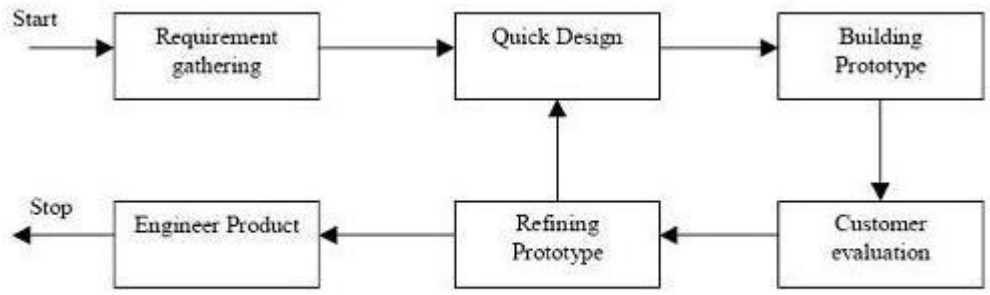
When to use Spiral model:

- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex

International Journal of Advance Research in Science and Engineering
Volume No.06, Special Issue No.(01), Nov 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

Prototype Model

The basic idea in **Prototype model** is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. Prototype model is a software development model. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system. The prototype are usually not complete systems and many of the details are not built in the proto type. The goal is to provide a system with overall functionality.



Prototyping Model

When to use Prototype model

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- They are excellent for designing good human computer interface systems.

IV. CONCLUSION AND FUTURE

Work After completing this research , it is concluded 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. Waterfall model and spiral model are used commonly in developing systems.
4. Each model has advantages and disadvantages for the development of systems ,

so each model tries to eliminate the disadvantages of the previous model Finally, some topics can be suggested for future works:

1. Suggesting a model to simulate advantages that are found in different models to software process management.
2. Making a comparison between the suggested model and the previous software processes management models.
3. Applying the suggested model to many projects to ensure of its suitability and documentation to explain its mechanical work.

REFERENCES

[1]. Ian Sommerville, "Software Engineering", Addison Wesley, 7th edition, 2004.

[2] CTG. MFA – 003, "A Survey of System Development Process Models", Models for Action Project: Developing Practical Approaches to Electronic Records Management and Preservation, Center for Technology in Government University at Albany / Suny, 1998 .

[3] Steve Easterbrook, "Software Lifecycles", University of Toronto Department of Computer Science, 2001.

[4] National Instruments Corporation, "Lifecycle Models", 2006 ,

[5] JJ Kuhl, "Project Lifecycle Models: How They Differ and When to Use Them" 2002 ,.

[6] Karlm, "Software Lifecycle Models', KTH, 2006 .

[7] Rlewallen, "Software Development Life Cycle Models", 2005 ,http://codebeter.com.

[8] Barry Boehm, "Spiral Development: Experience, Principles, and Refinements", edited by Wilfred J. Hansen, .