Software Process Models Outline

Manan D Shah
Assistaant Professor
Information Technology Department
Charusat University
Mananshah0003@gmail.com

Charusmita Dhiman
Assistant Professor
Computer Engineering Department
MBICT, New V.V.Nagar
Charusmita.dhiman@gmail.com

***Abstract*—Software engineering works on different models. Models plays most important role while developing any software. Models may be considered as a skeleton of the software. The development models are the various processes or methodologies that are being selected for the development of the project depending on the project's aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out. Further detail description of each model is carried out in this paper by explaining its importance and pros and cons of the models. There are models like waterfall model, Incremental models and Evolutionary models.**

Keywords: - Software Models, Waterfall Model, Incremental Models, Evolutionary models

A. Introduction

A system too large for one person to build is usually also too large to build without an overall plan that coordinates the people working on it, the tasks that need to be done, and the artifacts that are produced. Researchers and practitioners have identified a number of software development formation Models for this coordination. Here are some of the main ones.

These formation Models are alternatives, but not exclusive ones: most describe different aspects of a formation, and it is common for a development group to be following two or more simultaneously. For example,

- The sashimi formation is a way of organizing a waterfall with feedback.
- Boehm's spiral Model example uses prototyping as the Model for each cycle, and portions of a waterfall Model for the delivered system stage of the prototyping Model.
- An incremental formation often uses a sashimi formation for its Produce a build stage.

Following is the classification of different Software Formation Model.

1. Waterfall Model
2. Incremental Formation Model
   - 2.1 Incremental Model
   - 2.2 RAD Model
3. Evolutionary Formation Model
   - 3.1 Prototyping
   - 3.2 Spiral Model
   - 3.3 Concurrent Model

1. 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 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 [7].

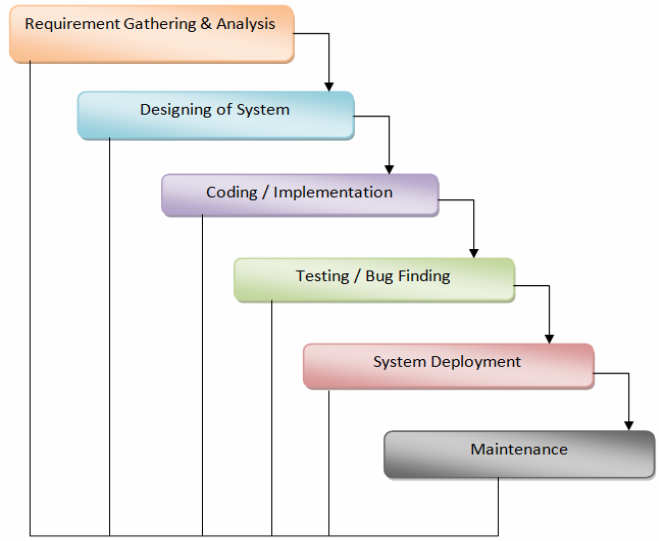

Figure 1. Waterfall-model

1.1 Fringe benefit of waterfall model:

- It is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- 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.

1.2 Shortfalls of waterfall model:

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

1.3 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.

2. Incremental Formation Model

2.1 Incremental Model:-

The model is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the waterfall model with the iterative philosophy of prototyping [10].

The product is decomposed into a number of components, each of which are designed and built separately (termed as builds). Each component is delivered to the client when it is complete. This allows partial utilization of product and avoids a long development time. It also creates a large initial capital outlay with the subsequent long wait avoided. This model of development also helps ease the traumatic effect of introducing completely new system all at once.

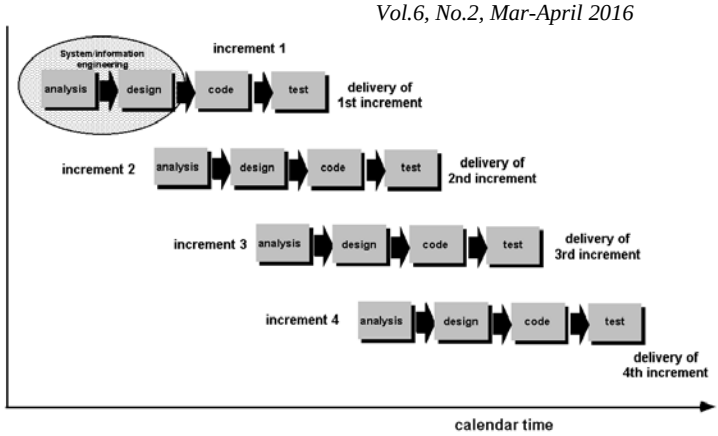

Figure 2 Incremental Model[Adapted From Google Images]

2.1.1 Fringe benefit of Incremental Model

- Generates working software quickly and early during the software life cycle.
- More flexible – less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Easier to manage risk because risky pieces are identified and handled during its iteration.
- Each iteration is an easily managed milestone.

2.1.2 Shortfalls of Incremental Model

- Each phase of an iteration is rigid and do not overlap each other.
- Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

2.1.3 When to use Incremental Model

- Such models are used where requirements are clear and can implement by phase wise. From the figure it's clear that the requirements ® is divided into R1, R2 to Rn and delivered accordingly.
- Mostly such model is used in web applications and product based companies.

2.2 RAD Model

RAD model is Rapid Application Development model. It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

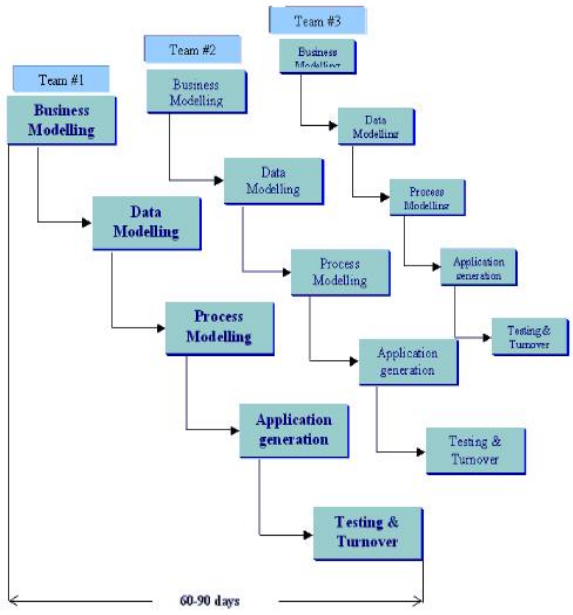

Figure 3. RAD Model[Adapted From Google Images]

Business modeling: The information flow is identified between various business functions.

Data modeling: Information gathered from business modeling is used to define data objects that are needed for the business.

Process modeling: Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.

Application generation: Automated tools are used to convert process models into code and the actual system.

Testing and turnover: Test new components and all the interfaces.

2.2.1 Fringe benefit of the RAD model:

- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.

2.2.2 Shortfalls of RAD model:

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

2.2.3 When to use RAD model:

- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

3. Evolutionary Software Formation Model

This approach is based on the idea of rapidly developing an initial software implementation from very abstract specifications and modifying this according to your appraisal. Each program version inherits the best features from earlier versions. Each version is refined based upon feedback from yourself to produce a system which satisfies your needs. At this point the system may be delivered or it may be re-implemented using a more structured approach to enhance robustness and maintainability. Specification, development and validation activities are concurrent with strong feedback between each. Evolutionary Software Formation Model of following types of Models

3.1 Prototyping Model

The basic idea here 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. 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. Prototyping is an attractive idea for complicated and large systems for which there is no manual formation or existing system to help determining the requirements. The prototypes are usually not complete systems and many of the

details are not built in the prototype. The goal is to provide a system with overall functionality [3].

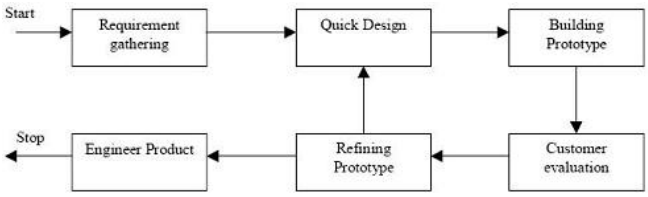

Figure 4. Prototype Model [Adapted From Google Images]

3.1.1 Fringe Benefit of Prototype Model:

- Users are actively involved in the development
- Since in this methodology a working Model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified Requirements validation, Quick implementation of incomplete application.

3.1.2 Shortfalls of Prototype Model:

- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Incomplete application may cause application not to be used as the full system was designed
- Incomplete or inadequate problem analysis.

3.1.3 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.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

3.2 The Spiral Model

The Spiral Model is an evolutionary Software Formation Model that couples the iterative nature of prototyping with the controlled and systematic aspects of the Linear Sequential Model. Using the Spiral Model the software is developed in a series of incremental releases. Unlike the Iteration Model where in the first product is a core product, in the Spiral Model the early iterations could result in a paper Model or a prototype. However, during later iterations more complex functionalities could be added [2].

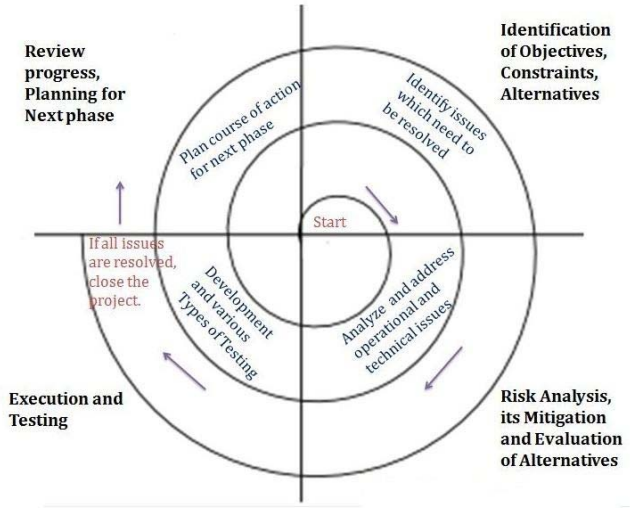

Figure 5. Spiral Model [Adapted From Google Images]

A Spiral Model, combines the iterative nature of prototyping with the controlled and systematic aspects of the Waterfall Model, therein providing the potential for rapid development of incremental versions of the software. A Spiral Model is divided into a number of framework activities, also called task regions. These task regions could vary from 3-6 in number and they are:

- Customer Communication - tasks required to establish effective communication between the developer and customer.
- Planning - tasks required to define resources, timelines and other project related information /items.
- Risk Analysis - tasks required to assess the technical and management risks.

- Engineering - tasks required to build one or more representation of the application.

- Construction & Release - tasks required to construct, test and support (eg. Documentation and training)
- Customer evaluation - tasks required to obtain periodic customer feedback so that there are no last minute surprises.

3.2.1 Fringe benefit of the Spiral Model

- Realistic approach to the development because the software evolves as the formation progresses. In addition, the developer and the client better understand and react to risks at each evolutionary level.
- The Model uses prototyping as a risk reduction mechanism and allows for the development of prototypes at any stage of the evolutionary development.
- It maintains a systematic stepwise approach, like the classic waterfall Model, and also incorporates into it an iterative framework that more reflect the real world.

3.2.2 Shortfalls of the Spiral Model

- One should possess considerable risk-assessment expertise
- It has not been employed as much proven Models (e.g. the Waterfall Model) and hence may prove difficult to 'sell' to the client.

3.3 Concurrent Development Model

The concurrent development Model - called concurrent engineering. It provides an accurate state of the current state of a project. Focus on concurrent engineering activities in a software engineering formation such as prototyping, analysis Modeling, requirements specification and design. Represented schematically as a series of major technical activities, tasks and their associated states. Defined as a series of events that trigger transitions from state to state for each of the software engineering activities. Often used as the paradigm for the development of client/server applications.

A client/server system is composed of a set of functional components. When applied to client/server, the concurrent formation Model defines activities in two dimensions :(i) System dimension -System level issues are addressed using three activities: design, assembly, and use.(ii) The component dimension is addressed with two activities: design and realization.

Two ways to achieve the concurrency:

System and component activities occur simultaneously and can be Modeling using the state-oriented approach

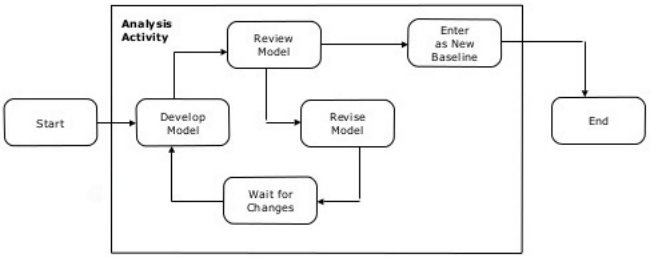

Figure 6. Concurrent Model[Adapted From Google Images]

A typical client/server application is implemented with many components, each can be designed and realized concurrently.

The concurrent formation Model is applicable to all types of software development and provides an accurate picture of the current state of a project. Rather than confining software engineering activities to a sequence of events, it defines a network of activities. Each activity on the network exists simultaneously with other activities. Events generated within a given activity or at some other place in the activity network trigger transitions among the states of an activity[6].

References

1. Shah, MR Manan D., MR Amit A. Kariyani, and MR Dipak L. Agrawal. "Allocation Of Virtual Machines In Cloud Computing Using Load Balancing Algorithm." *IJCSITS), ISSN* (2013): 2249-9555.

2.JJ Kuhl, "Project Lifecycle Models: How They Differ and When to Use Them" 2002 , www.businessesolutions.com.

3.Karlm, "Software Lifecycle Models', KTH, 2006.

4.Shah, Manan D., and Harshad B. Prajapati. "Reallocation and Allocation of Virtual machines in cloud computing." *arXiv preprint arXiv:1304.3978* (2013).

5.Shah Manan D, Dhiman Charusmita. "Cloud Computing Architecture & Services" IJCSMC, Vol. 4, Issue. 11, November 2015, pg.117 – 124 ISSN 2320–088X

6.Steve Easterbrook, "Software Lifecycles", University of Toronto Department of Computer Science, 2001.

7.National Instruments Corporation, "Lifecycle Models", 2006 , http://zone.ni.com.

8.JJ Kuhl, "Project Lifecycle Models: How They Differ and When to Use Them" 2002 , www.businessesolutions.com.

9.Karlm, "Software Lifecycle Models', KTH, 2006.

10.Rlewallen, "Software Development Life Cycle Models", 2005 ,http://codebeter.com.

11.Charusmita Dhiman, "Performance Evaluation Of AODV Routing Protocol In Vehicular Adhoc Networks", RACCCT 12, 29-30 March, 2012, Surat, India organized by IEEE SCET, **ISBN: 978-81-88894-34-5, pp 93-99.**

12. Charusmita Dhiman, "Novel approach to reduce routing overhead in dense VANETS", IJRCEE in volume 3 issue 5 Sep-Oct 2014 ISSN:2319-376X.

13.Kalpana Mudaliar & Charusmitha Dhiman. "Performance Comparison of Location Area Scheme and Ant Colony Optimization for Location Management in Cellular Network", IJFTET, Volume: II , Issue: IV Publication Year: 2015 , Page(s): 19-22