Thakur Educational Trust's (Regd.)

THAKUR RAMNARAYAN COLLEGE OF ARTS & COMMERCE

ISO 21001:2018 Certified

PROGRAMME: B.Sc (I.T)

CLASS: S.Y.B.Sc (I.T)

SUBJECT NAME: SOFTWARE ENGINEERING

SEMESTER: IV

FACULTY NAME: Ms. SMRITI DUBEY

UNIT I

Chapter 4 – Software Development Process Models

Concepts :

Software Process

Software Development Process Models

Software Process:

A software process is a set of activities that leads to the production of software product. These activities may involve the development of software from scratch in a standard language like Java or There is no ideal process, and many organizations have developed their own approach to software development. For some systems, a very structured development process is required. For business systems, with rapidly changing requirements, a flexible, agile process is likely to be more effective.

The major fundamental activities which are common to all software processes:

1) **Software Specification**: The functionality of the software and constraints on its operation must be defined.

2) **Software design and implementation**: The software to meet the specification must be produced

3) **Software validation**: The software must be validated to ensure that it does what the customer wants.

4) **Software evolution**: The software must evolve to meet the changing customer needs.

Software Development Process Models:

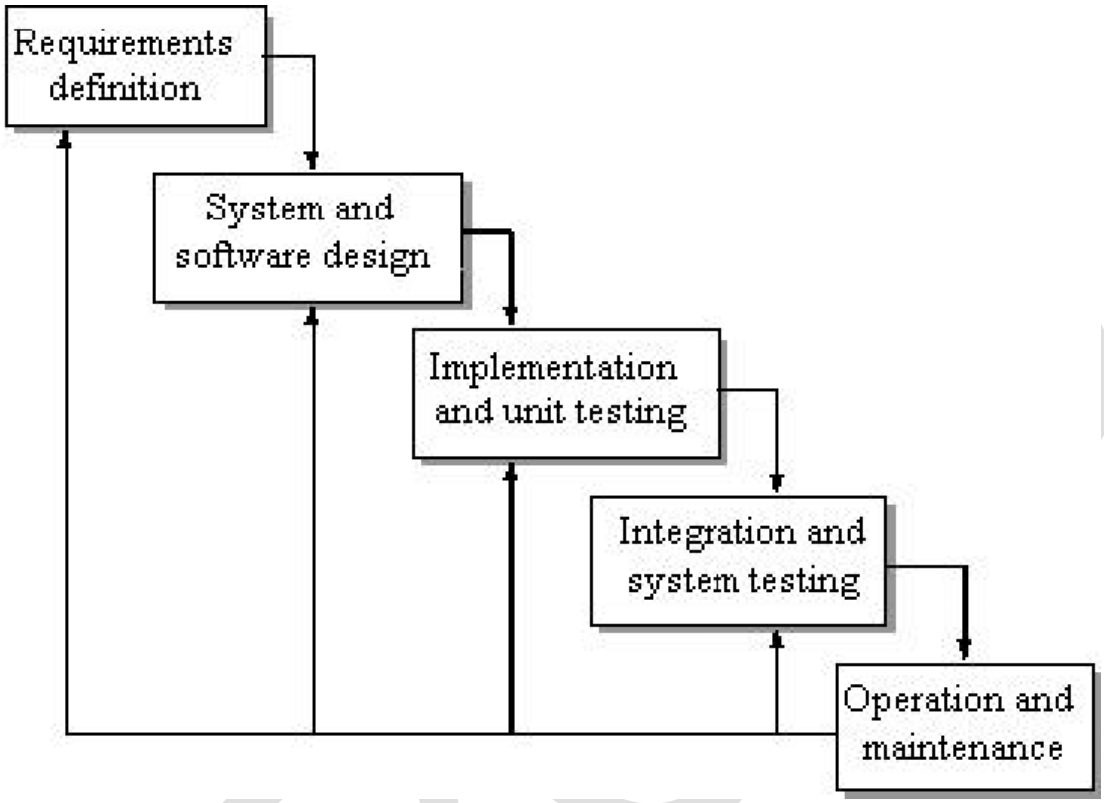
A software process model is an abstract representation of a software process. Each process model represents a process from a particular perspective and thus provides only partial information about that process.

The models specify the stages and order of a process. So, think of this as a representation of the **order of activities** of the process and the **sequence** in which they are performed. **The goal of a software process model is to provide guidance for controlling and coordinating the tasks to achieve the end product and objectives as effectively as possible.**

There are many kinds of process models for meeting different requirements. We refer to these as SDLC models (Software Development Life Cycle models). **The most popular and important SDLC models are as follows:**

1) Waterfall model
2) Incremental model
3) RAD model
4) Iterative model
5) Prototype model
6) Spiral model
7) RUP
8) Time boxing model

1) Waterfall model



The Waterfall Model was the 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 before the next phase can begin and there is no overlapping in the phases.**

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. **In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.**

The sequential phases in Waterfall model are −

1) **Requirement definition** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

2) **System and software design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

3) **Implementation and unit testing** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

4) **Integration and system testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures. Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

5) **Operation and Maintenance –** This is the longest lifecycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the lifecycle, improving the implementation of system units and enhancing the system services as new requirement are discovered.

Waterfall Model – Advantages

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

1) Documentation is produces at each phase

2) It fits with other engineering process models

3) The product of waterfall model always defines all constraints of the organization

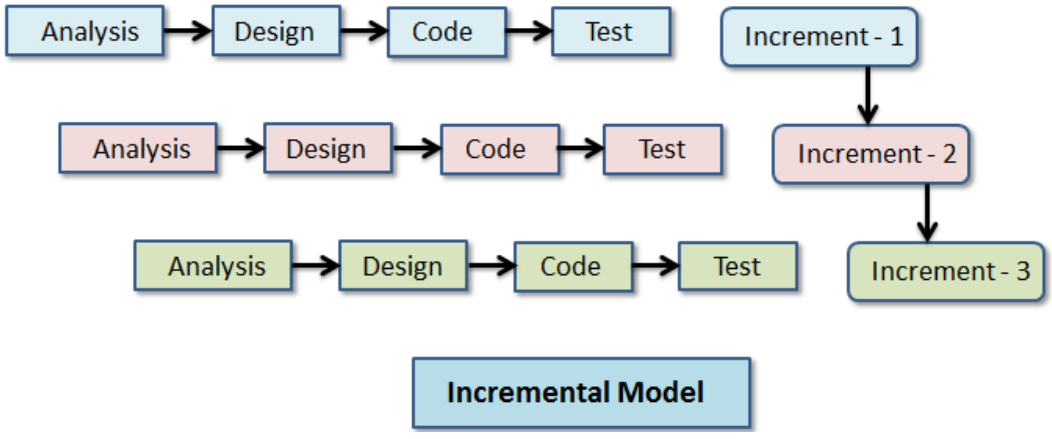
Waterfall Model - Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows –

1) No working software is produced until late during the life cycle.
2) High amounts of risk and uncertainty.
3) Not a good model for complex and object-oriented projects.
4) Poor model for long and ongoing projects.
5) Cannot move to the previous phases once the phase is frizzed. The premature freezing may lead to badly structured system which means that the system won't do what the user wants.

2) Incremental Model



1) Incremental development model is an in between approach that combines the advantages of waterfall model and evolutionary development. In an incremental development process customers identify in outline the services to be provided by the system. They identify which of the services are most important and which are least important to them.

2) A number of delivery increments are then defined with each increment providing a sub set of the system functionality. The allocation of services to increments depends on the service priority with the highest priority services delivered first.

3) Once the system increments have been identified the requirements for the services to be delivered in the first increment are defined in detail, and that increment is developed. During development further requirements analysis for later increments can take place but requirements change for the current increment are not accepted.

4) Once an increment is completed and delivered customers can put it into service. This means that they early delivery of the part of the system functionality. They can experiment with the system that helps them clarify their requirements for later increments and later versions of the current increment.

5) As new increments are completed, they are integrated with existing increments so that the system functionality improves with each delivered increment. The common services may be implemented early in the process or may be implemented incrementally as functionality as required by an increment.

Incremental model – Advantages

1) Customers do not have to wait until the entire system is delivered before they can gain value from it. The first increment satisfies their most critical requirements so they can use the software immediately

2) Customers can use the early increments as prototypes and gain experience that informs their requirements for later system increments.

3) There is lower risk of overall project failure. Although problems may be encountered in some increments it is likely that some will be successfully delivered to the customer

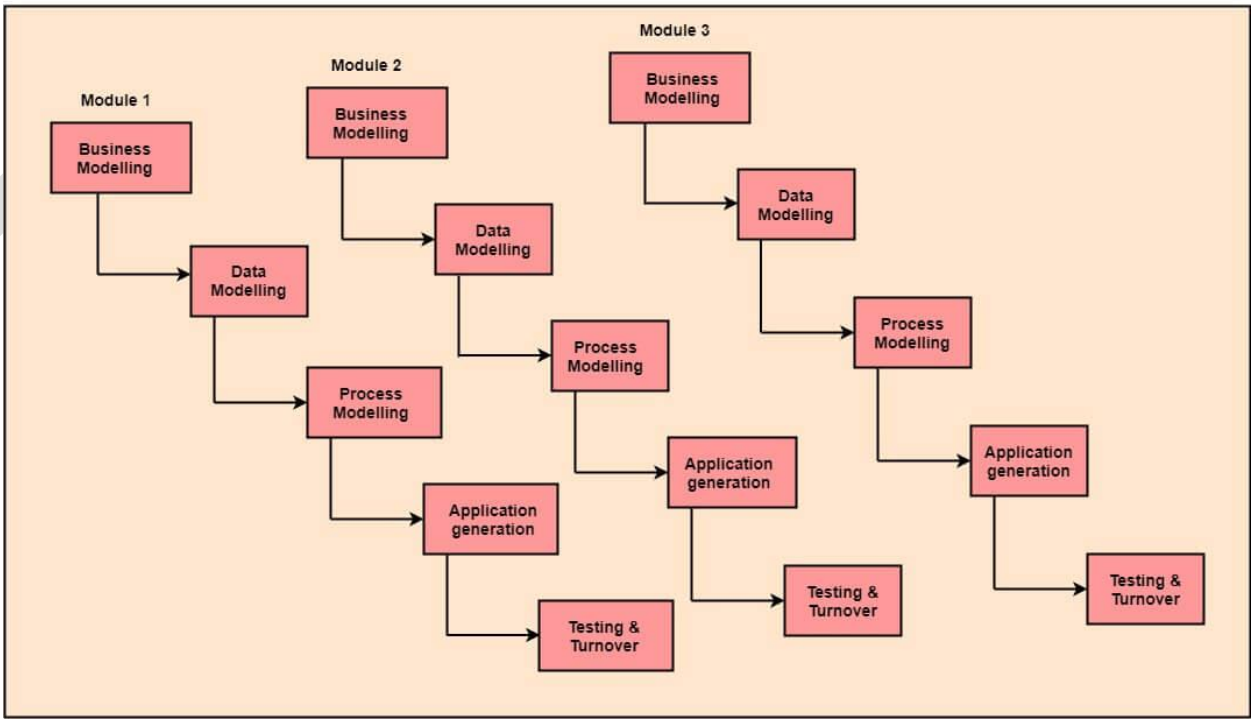
4) As the highest priority services are delivered first and later increments are integrated with them it is inevitable that the most important system service receive the most testing. This means that the customer is less likely to encounter software failure in most important parts of the system.

Incremental model – Disadvantages

1) Need for good planning
2) Total Cost is high.
3) Well defined module interfaces are needed.

3) RAD MODEL (Rapid Application Development)

Fig: RAD Model



RAD model 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 developed into working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

The various phases of RAD are as follows:

1.**Business Modelling**: The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.

2. **Data Modelling**: The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.

3**. Process Modelling**: The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

4. **Application Generation**: Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.

5. **Testing & Turnover**: Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.

RAD model Advantages

1) This model is flexible for change.

2) In this model, changes are adoptable.

3) Each phase in RAD brings highest priority functionality to the customer.

4) It reduced development time.

5) It increases the reusability of features.

RAD model Disadvantages

1) It required highly skilled designers.

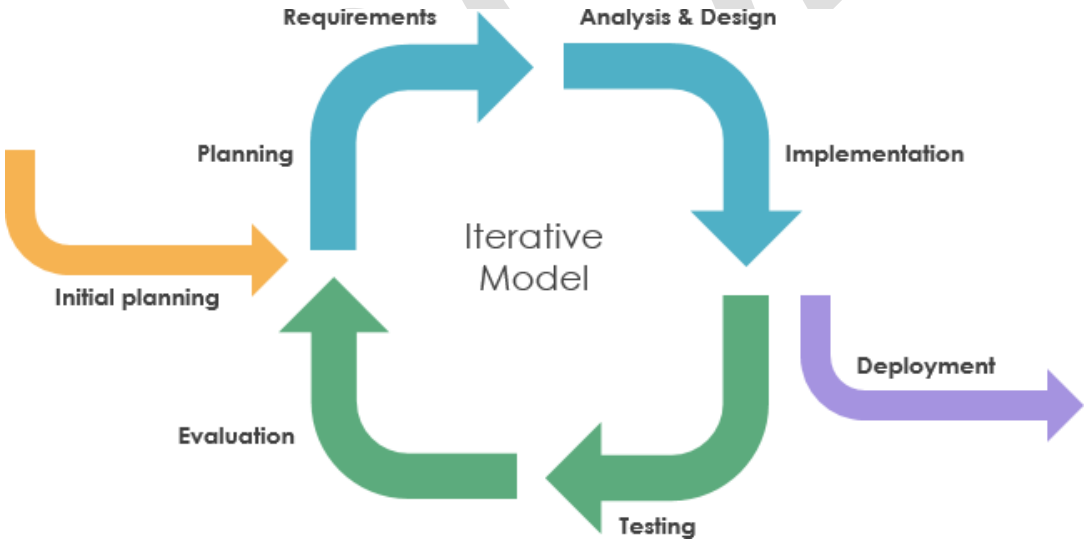
2) All application is not compatible with RAD.

3) For smaller projects, we cannot use the RAD model.

4) On the high technical risk, it's not suitable.

5) Required user involvement.

4) Iterative Development Model



An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software which can then be reviewed in order to identify further requirements. This process is then repeated producing a new version of the software for each cycle of the model.

The various phases of Iterative model are as follows:

1) **Requirement gathering & analysis:** In this phase, requirements are gathered from customers and check by an analyst whether requirements will fulfil or not. Analyst checks that need will achieve within budget or not. After all of this, the software team skips to the next phase.

2) **Design:** In the design phase, a software solution to meet the requirements is designed. This may be a new design or an extension to an earlier design.

3) **Implementation and Test phase:** The software is coded, integrated and tested

4) **Review phase:** The software is evaluated; the current requirements are reviewed and changes and additions to requirements are proposed.

5) For each cycle of the model a decision has to be made as to whether the software produced by the cycle will be discarded or kept as the starting point for the next cycle. Eventually a point will be reached where the requirements are complete and the software can be delivered or it becomes impossible to enhance the software as required and fresh start has to be made.

6) The iterative lifecycle model can be likened to producing software by successive approximation.

7) The key to successful use of an iterative software development life cycle is rigorous validation of requirements and verification of each version of the software against those requirements within each cycle of the model

8) The first three phases of the iterative model is in fact an abbreviated form of sequential V model or Waterfall model of development. Each cycle of the model produces software that requires testing at the unit level, for software integration, for system integration and for acceptance testing

9) As the software evolves through successive cycles test have to be repeated and extended to verify each version of the software.

Iterative model – Advantages

1. Testing and debugging during smaller iteration is easy.
2. A Parallel development can plan.
3. It is easily acceptable to ever-changing needs of the project.
4. Risks are identified and resolved during iteration.
5. Limited time spent on documentation and extra time on designing

Iterative model – Disadvantages

1. It is not suitable for smaller projects.
2. More Resources may be required.
3. Design can be changed again and again because of imperfect requirements.
4. Requirement changes can cause over budget.
5. Project completion date not confirmed because of changing requirements.

5) Prototype model

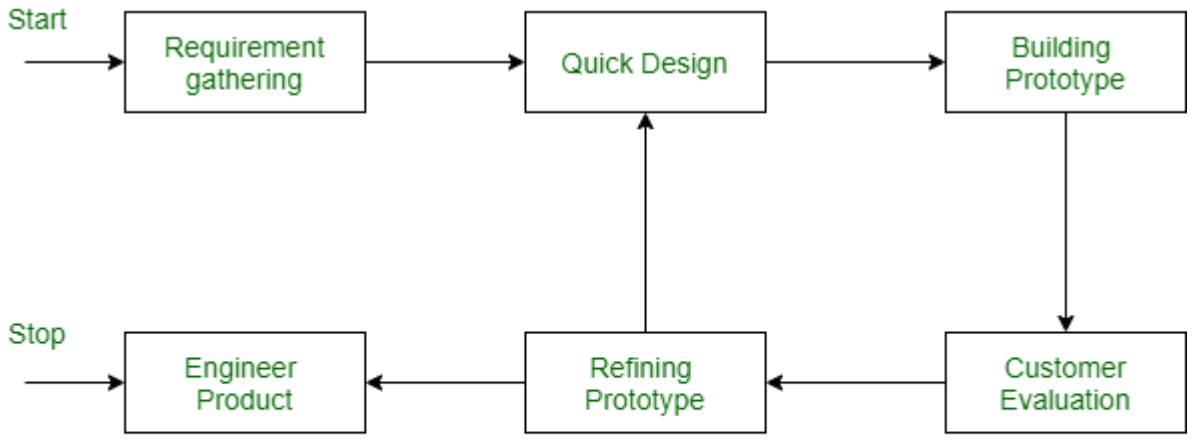


Figure - Prototype Model

Prototyping Model is a software development model in which prototype is built, tested, and reworked until an acceptable prototype is achieved. It also creates base to produce the final system or software. It works best in scenarios where the project's requirements are not known in detail. It is an iterative, trial and error method which takes place between developer and client.

By using the 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 desired system.

Prototyping Model has following six SDLC phases as follow:

Step 1: Requirements gathering and analysis

A prototyping model starts with requirement analysis. In this phase, the requirements of the system are defined in detail. During the process, the users of the system are interviewed to know what is their expectation from the system.

Step 2: Quick design

The second phase is a preliminary design or a quick design. In this stage, a simple design of the system is created. However, it is not a complete design. It gives a brief idea of the system to the user. The quick design helps in developing the prototype.

Step 3: Build a Prototype

In this phase, an actual prototype is designed based on the information gathered from quick design. It is a small working model of the required system

Step 4: Initial user evaluation

In this stage, the proposed system is presented to the client for an initial evaluation. It helps to find out the strength and weakness of the working model. Comment and suggestion are collected from the customer and provided to the developer.

Step 5: Refining prototype

If the user is not happy with the current prototype, you need to refine the prototype according to the user's feedback and suggestions.

This phase will not over until all the requirements specified by the user are met. Once the user is satisfied with the developed prototype, a final system is developed based on the approved final prototype.

Step 6: Implement Product and Maintain

Once the final system is developed based on the final prototype, it is thoroughly tested and deployed to production. The system undergoes routine maintenance for minimizing downtime and prevent large-scale failures.

Prototype model – Advantages

1) Users are actively involved in development. Therefore, errors can be detected in the initial stage of the software development process.

2) Customer satisfaction exists because the customer can feel the product at a very early stage.

3) Quicker user feedback helps you to achieve better software development solutions.

4) There will be hardly any chance of software rejection.

Prototype model – Disadvantages

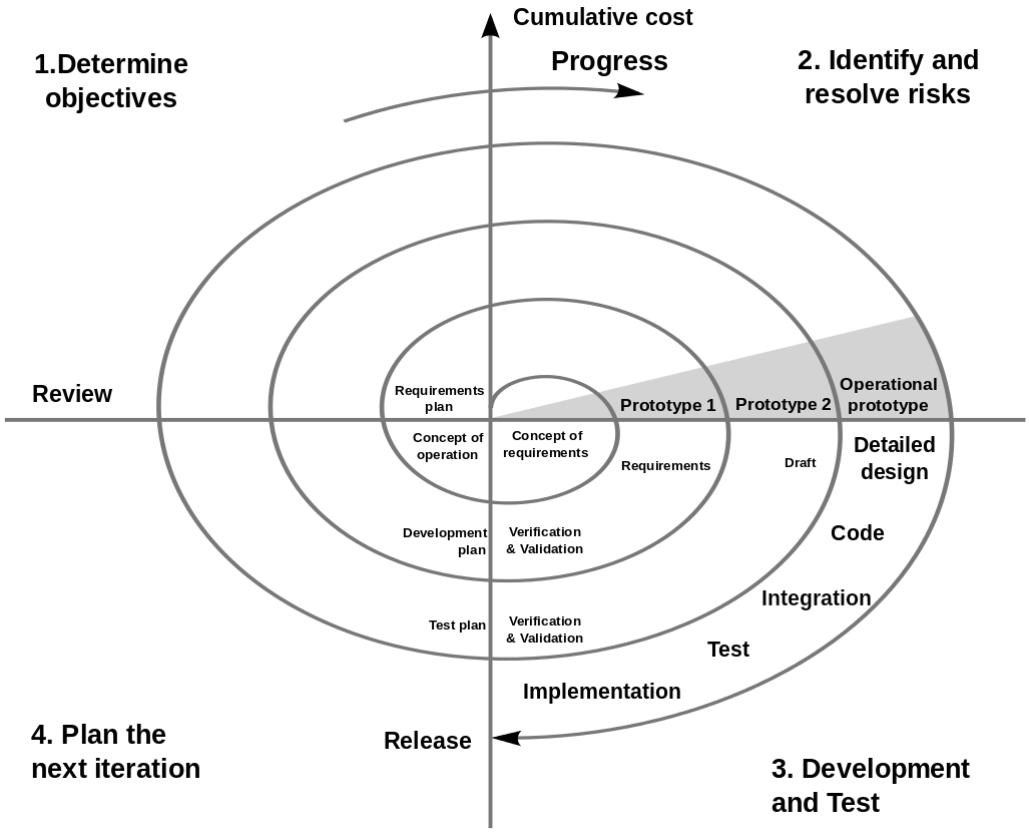
1) The cost of developing a prototype is a total waste as the prototype is ultimately thrown away.

2) Prototyping may encourage excessive change requests**.**

3) It is very difficult for software developers to accommodate all the changes demanded by the clients.

4) Leads to implementing and then repairing way of building systems

6) Spiral Model



The spiral model of the software process was originally proposed by Boehm in 1998.Rather than representing the software process as a sequence of activities with some backtracking from one activity to another the process is represented as a spiral. **Each loop in the spiral represents a software process. Thus, the innermost loop might be concerned with system feasibility, the next loop with requirements definition, the next loop with system design and so on.**

Spiral Model is a risk-driven software development process model. The exact number of loops of the spiral is unknown and can vary from project to project. The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

Objectives determination and identify alternative solutions: Requirements are gathered from the customers and the objectives are identified, elaborated, and analysed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

Identify and resolve Risks: During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

Develop next version of the Product: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

Review and plan for the next Phase: In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started**.**

Risk Handling in Spiral Model

A risk is any adverse situation that might affect the successful completion of a software project. The most important feature of the spiral model is handling these unknown risks after the project

has started. Such risk resolutions are easier done by developing a prototype. The spiral model supports coping up with risks by providing the scope to build a prototype at every phase of the software development.

Spiral model Advantages

1) Additional functionality or changes can be done at a later stage

2) Cost estimation becomes easy as the prototype building is done in small fragments

3) Continuous or repeated development helps in risk management

4) Development is fast and features are added in a systematic way in Spiral development

5) There is always a space for customer feedback

Spiral model Disadvantages

1) Risk of not meeting the schedule or budget

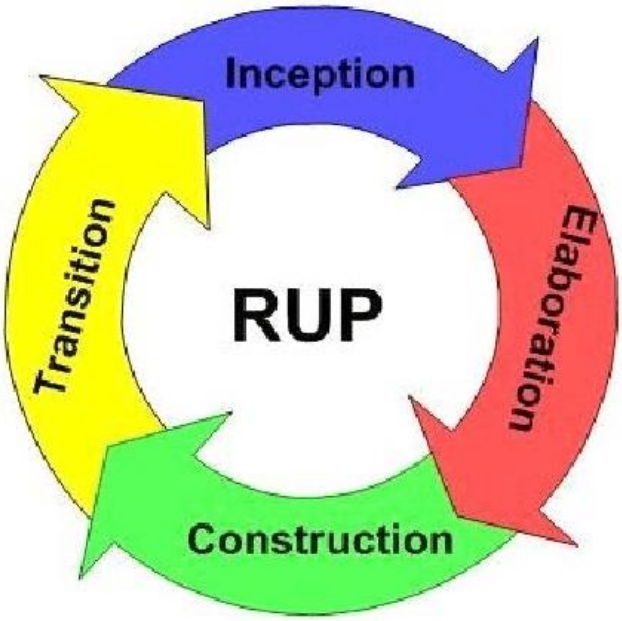
2) Spiral development works best for large projects only also demands risk assessment expertise

3) For its smooth operation spiral model protocol needs to be followed strictly

4) Documentation is more as it has intermediate phases

5) Spiral software development is not advisable for smaller project, it might cost them a lot

7) RUP (Rational Unified Process)



Rational Unified Process (RUP) is a software development process for object-oriented models. It is also known as the Unified Process Model.

Each phase is finalised with a milestone. A milestone is a point in time where decisions of critical importance must to be made. In order to be able to make those decisions, the objectives must have been accomplished.

The phases in the RUP are:

1. Inception Phase

It is the initial phase of the developing process. During this phase, the project's basic ideas and structure will be determined to prepare a business suite, i.e. the team will decide the purpose of the project, success criteria, estimated cost, risk assessment, scheduled time, and resources required to

complete it etc. It is just like an evaluation of the project. The project may be cancelled or consider depends on if it fails to pass the below criteria.

2. Elaboration Phase

This is the second phase of the development process. During this phase, to analyze the project's requirements and necessary architecture, i.e. to review the problems, develop the project plan and architect, and eliminate the high-risk elements from the project. It is the most critical phase among the four phases. The actual development and coding will take place in the following phase.

3. Construction Phase

This is the third phase of the development process. During this phase, the project is developed and completed. Here all the features are developed and integrated into the product, i.e., the software is designed, written, and tested successfully. So the development product will be a deployable product. It measures the completeness of the product.

4. Transition Phase

This is the last phase of the development process. During this phase, the software is released and delivered to the public or customers. Based on the feedback from the end-users, the product will be made update or change. It is the process of deployment.

RUP model Advantages

1) It allows us to deal with changing requirements within the project's development life cycle as per the client or customer needs, i.e., it welcomes change.

2) It supports incremental build the software product.

3) It provides proper documentation of the software product.

4) It helps to use the resources efficiently.

5) It helps to identify issues early in the process life cycle.

RUP model Disadvantages

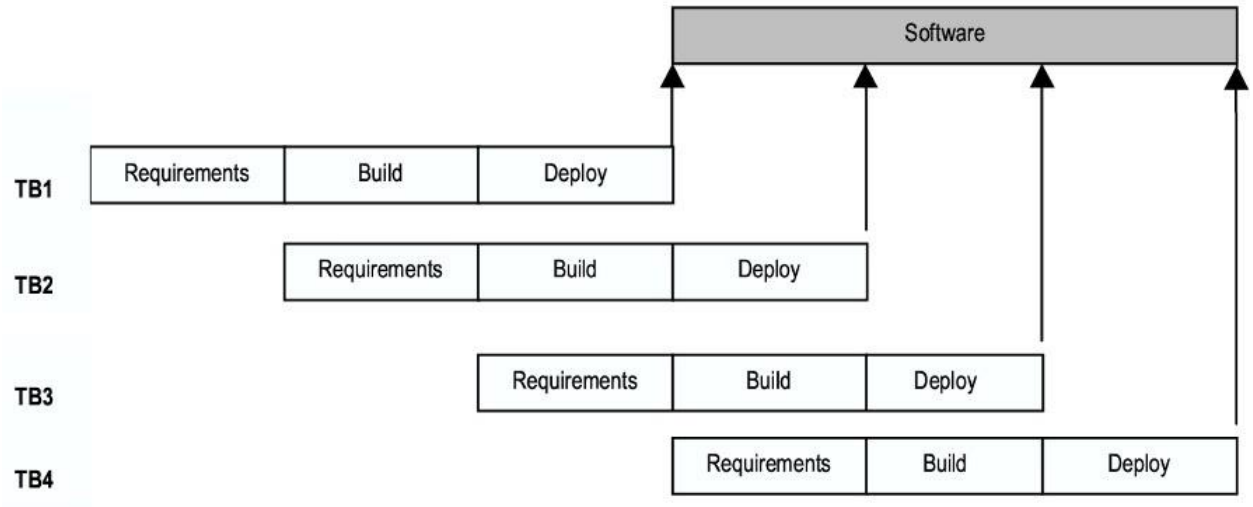
1) It is a complex model to implement as it has multiple stages of the workflow.

2) It is challenging for organizations to implement which has, small team size or projects.

3) It should be highly result-oriented from individuals or teams.

4) It emphasizes the integration of modules throughout the software development process, so this creates trouble during the testing phase.

8) Timeboxing model



In the timeboxing model, the basic unit of development is a time box, which is of fixed duration. Since the duration is fixed, a key factor in selecting the requirements or features to be built in a time box is what can be fit into the time box. This is in contrast to regular iterative approaches where the functionality is selected and then the time to deliver is determined. Timeboxing changes the perspective of development and makes the schedule a non-negotiable and a high-priority commitment.

Each time box is divided into a sequence of stages, like in the waterfall model. Each stage performs some clearly defined tasks for the iteration and produces a clearly defined output. The model also requires that the duration of each stage, that is, the time it takes to complete the task of that stage, is approximately the same. Furthermore, the model requires that there be a dedicated team for each stage. That is, the team for a stage performs only tasks of that stage—tasks for other stages are performed by their respective teams.

Consider a time box consisting of three stages: requirement specification, build, and deployment. The requirement stage is executed by its team of analysts and ends with a prioritized list of requirements to be built in this iteration along with a high-level design. The build team develops the code for implementing the requirements, and performs the testing. The tested code is then handed over to the deployment team, which performs pre-deployment tests, and then installs the system for production use. These three stages are such that they can be done in approximately equal time in an iteration.

With a time, box of three stages, the project proceeds as follows. When the requirements team has finished requirements for timebox-1, the requirements are given to the build team for building the software. The requirements team then goes on and starts preparing the requirements for timebox-2. When the build for timebox-1 is completed, the code is handed over to the deployment team, and the build team moves on to build code for requirements for timebox- 2, and the requirements team moves on to doing requirements for timebox-3.

Timeboxing is well suited for projects that require a large number of features to be developed in a short time around a stable architecture using stable technologies. These features should be such that there is some flexibility in grouping them for building a meaningful system in an iteration that provides value to the users.

Difference between Waterfall model and Incremental model

Basis	Waterfall Model	Incremental Model
Working Version	The working version of the software is delivered at end of the model's life cycle.	The working version of the software is delivered in each iteration.
Work Flow	This model proposes a sequential workflow.	This model proposes a linear and parallel workflow.
User Involvement	User involvement occurs in the first phase of the model i.e., communication.	The user involvement occurs in first phase of each iteration.
Feasibility	This process model does not accommodate changes in the software.	This process model accommodates changes in software easily.
Team Size	Team size is large.	Does not require a large team.
Documentation	Focuses too much on documentation.	Provides documentation but not enough.
Maintenance	This model provides the least maintenance.	This model promotes maintenance.
Testing	After the coding phase completes.	At each iteration.
Retracking	Not possible.	Possible.

Difference between RAD model and Spiral model

Sr.no	RAD MODEL	SPIRAL MODEL
1	RAD model is a software development model where by the components or functions are developed in parallel as if they were mini projects	Spiral model is a software development model and is made with features of incremental, waterfall or evolutionary prototyping models.
2	Rapid development is its main objective.	High assurance is its main objective.
3	RAD model requirements and early-stage planning is not necessary.	Spiral model requirements and early-stage planning is required.
4	It is necessary to have detailed documentation but in a limited manner.	Detailed documentation is required.
5	Requirements are specified as time boxed release manner.	Requirements are specified in the beginning.
6	In RAD model small team size is required.	In spiral model large team is required.
7	Customer involvement is only at the beginning.	Customer involvement is high as compared to RAD model.
8	Testing is done in RAD model after completion of coding.	Testing is done in spiral model at the end of the engineering phase.
9	In RAD model overlapping of phases is possible.	In spiral model overlapping of phases is not possible.
10	Cost of RAD model is Low.	Cost of spiral model is very expensive.
11	RAD model is used between large and small project.	Spiral model is used for large project.
12	Flexibility to change in RAD model is Easy.	Flexibility to change in spiral model is not that difficult.