Paper Name: Software Engineering

Topic: Process Models

Paper Code: BCA CC409

By Ms. Amrita Prakash

Assistant professor,

Department of Computer Science

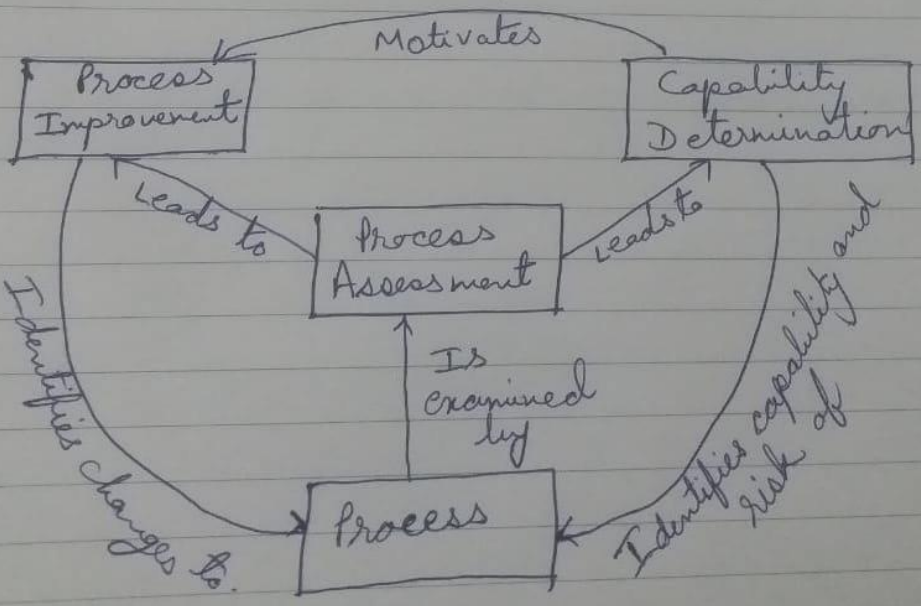
Patna Womens College

Process Assessment

* The process needs to be assessed so as to ensure that it meets a set of basic process criteria, which is essential for implementing the principles of software engineering in an efficient manner.

* The process is assessed to evaluate methods, tools and practices, which are used to develop and test the software.

* The aim of process assessment is to identify the areas for improvement and suggest a plan for making that improvement.



In the above figure, it is shown that the process is examined by Process Assessment, which leads to Capability Determination and Process Improvement. The capability of a process determines whether a process with some variations is capable of meeting user requirements or not.

The main focus areas of process assessment are

* Obtaining guidance in improving software development and test process.
* Obtaining an independent and unbiased review of the process.
* Obtaining a baseline for improving quality and productivity of processes.

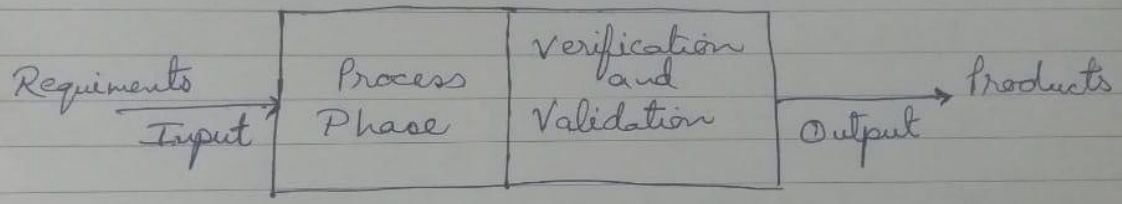
Note: Baseline is defined as a set of software components and documents that has been formerly reviewed and accepted, that serves as the basis for further development.

Software Life Cycle Models

* A process model can be defined as a strategy, comprising processes, methods, tools or steps for developing a software.
* These models provide a basis for controlling various activities required to develop and maintain the software.
* A process model for Software Engineering depends on the nature and application of a software project.
* IEEE defines a process model as 'a framework containing the processes, activities and tasks involved in the development, operation and maintenance of a software product, spanning the life of the system from the definition of its requirements to the termination of its use'.
* A process model reflects the goals of software development, such as developing a high quality product and meeting the schedule on time. It also provides a flexible framework for enhancing the processes.

Advantages of Process Model:

1. It Enables Effective Communication
2. It Facilitates Process Reuse.
3. It is Effective
4. It Facilitates Process Management.



Phases in the Development Process.

→ Verification is the process of evaluating a system or its components for determining the product developed at each phase of software development.

→ Validation is the process of evaluating the product at the end of each phase to ensure compliance with the requirements.

Various kinds of Process Models are :-

1. Waterfall Model / Linear Sequential Model
2. Prototyping Model
3. Spiral Model
4. Incremental Model
5. RAD Model
6. V Model
7. Build and Fix Model and
8. Formal Method Model.

SOFTWARE LIFECYCLE MODELS

1. Waterfall Model

- The waterfall model is the classic model or oldest model and is known as mother of all the model. It is widely used in government projects and many vital projects in company.
- The waterfall model is also called as **'Linear sequential model'** or **'Classic life cycle model'.**
- In this model, each phase is executed completely before the beginning of the next phase. Hence the phases do not overlap in waterfall model.
- This model is used for small projects.
- In this model, feedback is taken after each phase to ensure that the project is on the right path.
- Testing part starts only after the development is completed.

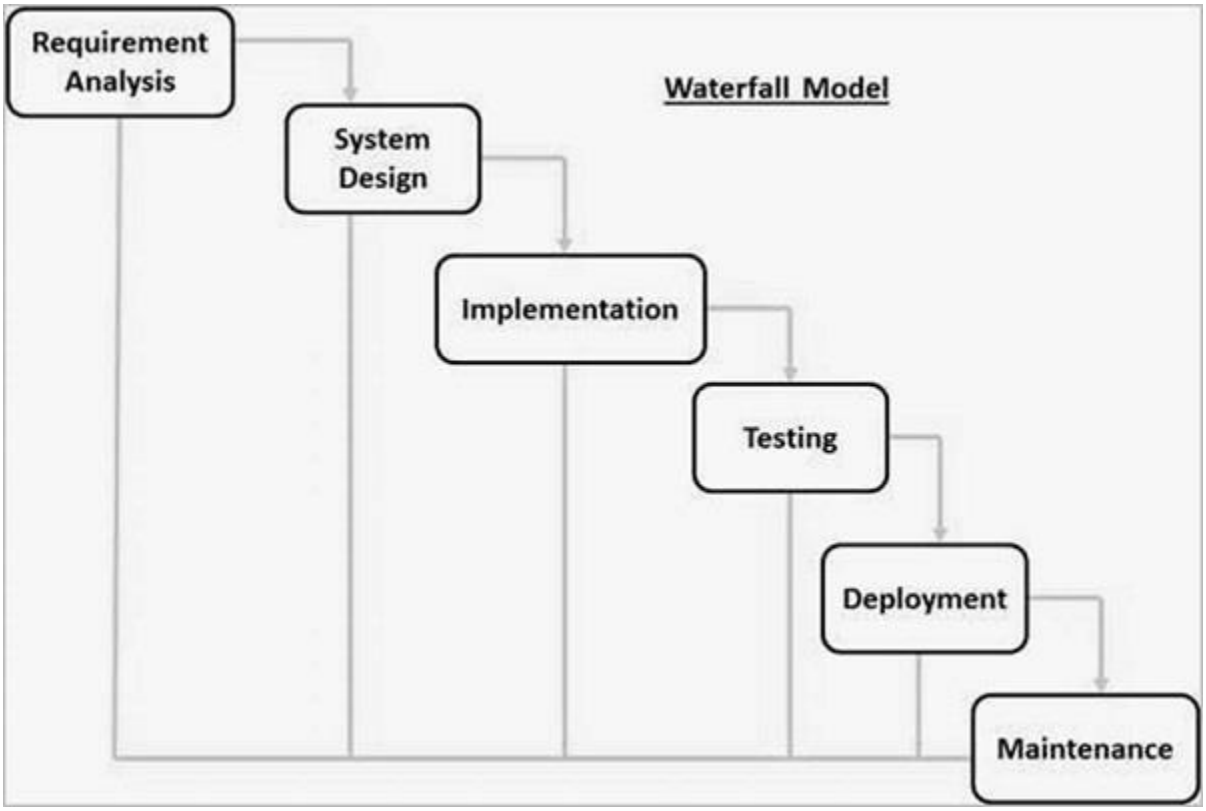
The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall Model - Design

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 following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are −

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System 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.
- **Implementation** − 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.
- **Integration and 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.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

Waterfall Model - Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model - Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

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 −

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

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 −

- 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. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

2. Prototyping Model

- Prototype is defined as first or preliminary form using which other forms are copied or derived.
- Prototype model is a set of general objectives for software.
- It does not identify the requirements like detailed input, output.
- It is software working model of limited functionality.

The prototyping model is applied when there is an absence of detailed information regarding input and output requirements in the software. This model is developed on the assumption that it is often difficult to know all the requirements at the beginning of the project.

This model increases flexibility of the development process by allowing the user to interact and experiment with a working representation of the product known as **Prototype.**

The different phases of Prototyping model are:

1) Communication
2) Quick design
3) Modeling and quick design
4) Construction of prototype
5) Deployment, delivery, feedback

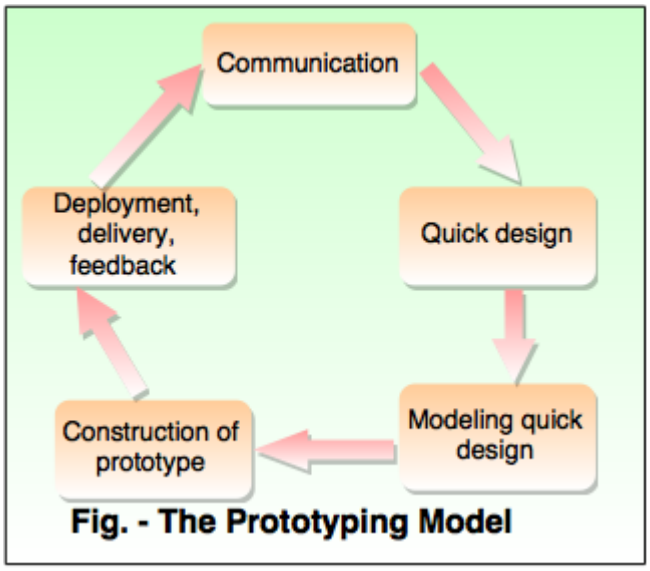


Fig. - The Prototyping Model

1. Communication
In this phase, developer and customer meet and discuss the overall objectives of the software.

2. Quick design

- Quick design is implemented when requirements are known.
- It includes only the important aspects i.e input and output format of the software.
- It focuses on those aspects which are visible to the user rather than the detailed plan.
- It helps to construct a prototype.

3. Modeling quick design

- This phase gives the clear idea about the development of software as the software is now constructed.
- It allows the developer to better understand the exact requirements.

4. Construction of prototype
The prototype is evaluated by the customer itself.

5. Deployment, delivery, feedback

- If the user is not satisfied with current prototype then it is refined according to the requirements of the user.
- The process of refining the prototype is repeated till all the requirements of users are met.
- When the users are satisfied with the developed prototype then the system is developed on the basis of final prototype.

Advantages of Prototyping Model

- In the development process of this model users are actively involved.

- The development process is the best platform to understand the system by the user.
- Earlier error detection takes place in this model.
- It gives quick user feedback for better solutions.
- It identifies the missing functionality easily. It also identifies the confusing or difficult functions.

Disadvantages of Prototyping Model

- The client involvement is more and it is not always considered by the developer.
- It is a slow process because it takes more time for development.
- Many changes can disturb the rhythm of the development team.
- It is a throw away prototype when the users are confused with it.

3. Spiral Model

The Spiral Model comprises activities orgranised in a spiral and has many cycles. This model combines the features of the prototyping model and waterfall model and is advantageous for large, complex and expensive projects.

The Spiral Model determines requirement problems in developing the prototype.

This model also guides and measures the need of risk management in each cycle of the spiral model.

The objective of the spiral model is to emphasize management to evaluate and resolve risks in the software project.

Spiral Model - Design

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

Identification

This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

Design

The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

Construct or Build

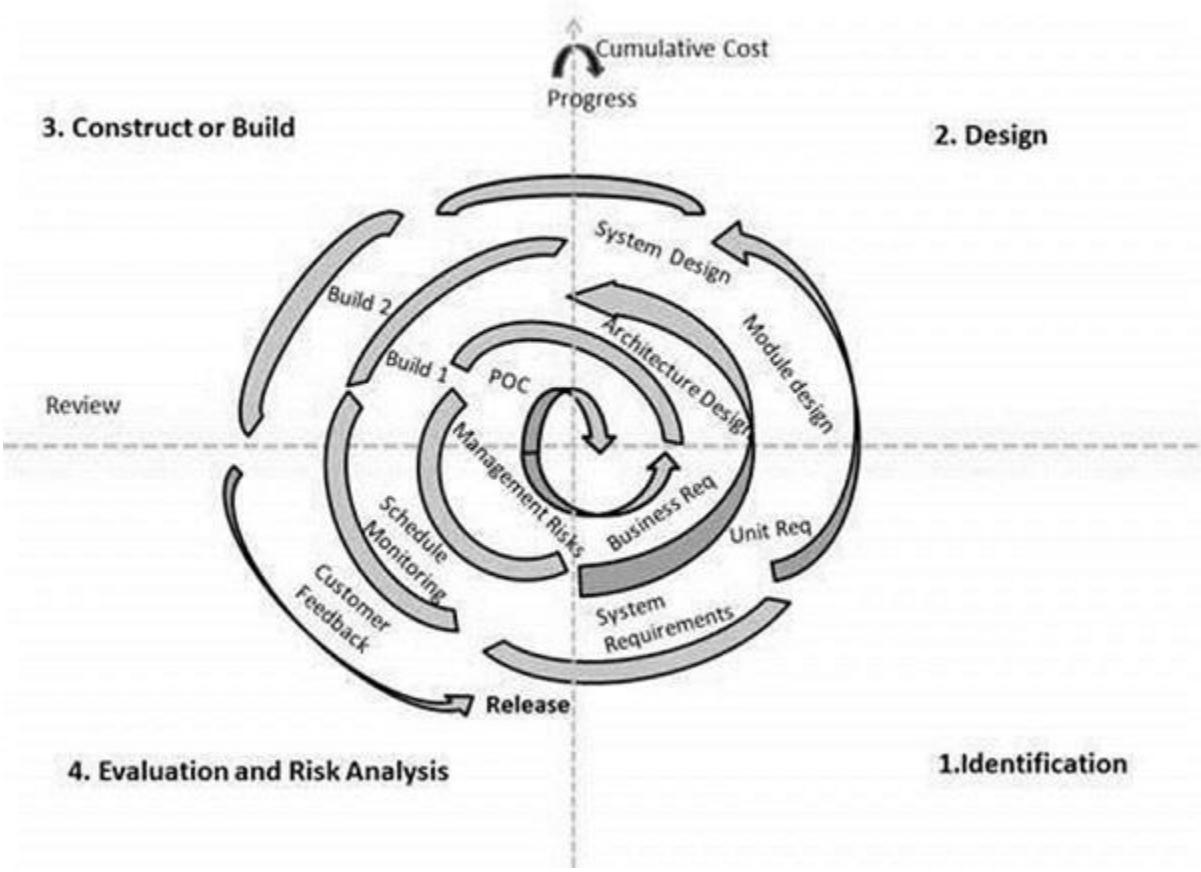
The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

Evaluation and Risk Analysis

Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

The following illustration is a representation of the Spiral Model, listing the activities in each phase.



Based on the customer evaluation, the software development process enters the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

Spiral Model Application

The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e. learning with maturity which involves minimum risk for the customer as well as the development firms.

The following pointers explain the typical uses of a Spiral Model −

- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

Spiral Model - Pros and Cons

The advantage of spiral lifecycle model is that it allows elements of the product to be added in, when they become available or known. This assures that there is no conflict with previous requirements and design.

This method is consistent with approaches that have multiple software builds and releases which allows making an orderly transition to a maintenance activity. Another positive aspect of this method is that the spiral model forces an early user involvement in the system development effort.

On the other side, it takes a very strict management to complete such products and there is a risk of running the spiral in an indefinite loop. So, the discipline of change and the extent of taking change requests is very important to develop and deploy the product successfully.

The advantages of the Spiral SDLC Model are as follows −

- Changing requirements can be accommodated.
- Allows extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

The disadvantages of the Spiral SDLC Model are as follows −

- Management is more complex.
- End of the project may not be known early.

- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go on indefinitely.
- Large number of intermediate stages requires excessive documentation.

4. Incremental Model

- The incremental model combines the elements of waterfall model and they are applied in an iterative fashion.
- The first increment in this model is generally a core product.
- Each increment builds the product and submits it to the customer for suggesting any modifications.
- The next increment implements the customer's suggestions and add additional requirements in the previous increment.
- This process is repeated until the product is completed.

For example, the word-processing software is developed using the incremental model.

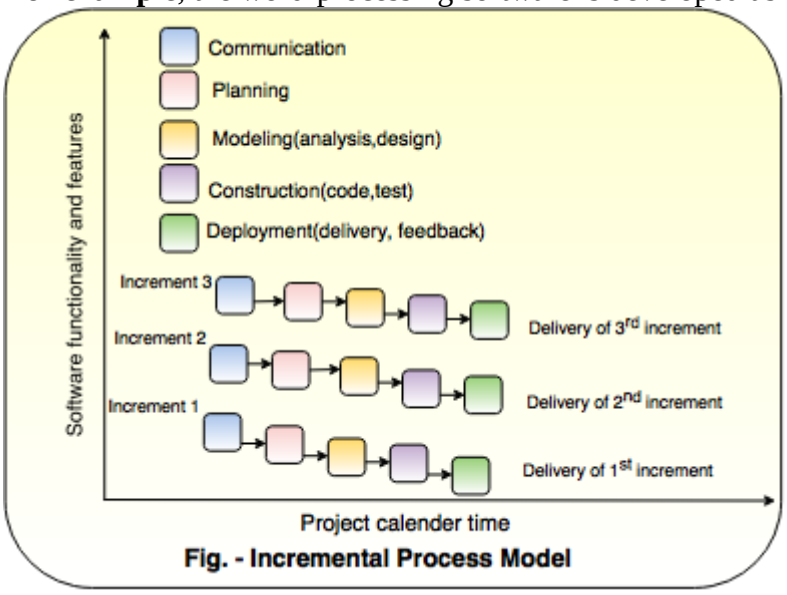


Fig. - Incremental Process Model

Following are the phases of Incremental model:

i) Communication
The software development starts with the communication between customer and developer.

ii) Planning
It consists of complete estimation, scheduling for project development.

iii) Modeling

- Modeling consists of complete requirement analysis and the design of the project like algorithm, flowchart etc.
- The algorithm is a step-by-step solution of the problem and the flow chart shows a complete flow diagram of a program.

iv) Construction

Construction consists of code generation and the testing part.

- Coding part implements the design details using an appropriate programming language.
- Testing is to check whether the flow of coding is correct or not.
- Testing also checks that the program provides desired output.

v) Deployment

- Deployment step consists of delivering the product to the customer and taking feedback from them.
- If the customer wants some corrections or demands for the additional capabilities, then the change is required for improvement in the quality of the software.

Advantages of Incremental model

- This model is flexible because the cost of development is low and initial product delivery is faster.
- It is easier to test and debug in the smaller iteration.
- The working software is generated quickly in the software life cycle.
- The customers can respond to its functionalities after every increment.

Disadvantages of the incremental model

- The cost of the final product may cross the cost initially estimated.
- This model requires a very clear and complete planning.
- The planning of design is required before the whole system is broken into smaller increments.

- The demands of customer for the additional functionalities after every increment causes problem in the system architecture.

5. RAD Model

The **RAD (Rapid Application Development)** model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product.

Rapid Application Development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

What is RAD?

Rapid application development is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.

In the RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process.

RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype.

The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable.

RAD Model Design

RAD model distributes the analysis, design, build and test phases into a series of short, iterative development cycles.

Following are the various phases of the RAD Model −

Business Modeling

The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels. A complete business analysis is performed to find the vital information for business, how it can be obtained, how and when is the information processed and what are the factors driving successful flow of information.

Data Modeling

The information gathered in the Business Modeling phase is reviewed and analyzed to form sets of data objects vital for the business. The attributes of all data sets is identified and defined. The relation between these data objects are established and defined in detail in relevance to the business model.

Process Modeling

The data object sets defined in the Data Modeling phase are converted to establish the business information flow needed to achieve specific business objectives as per the business model. The process model for any changes or enhancements to the data object sets is defined in this phase. Process descriptions for adding, deleting, retrieving or modifying a data object are given.

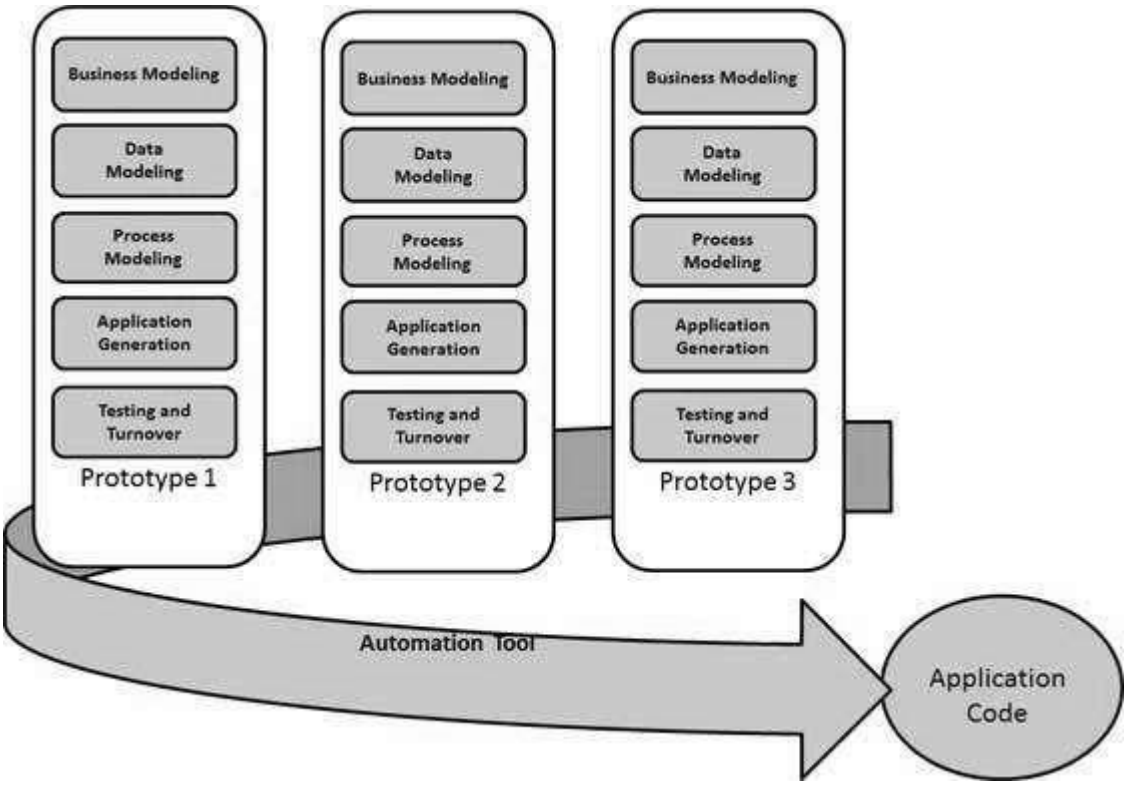
Application Generation

The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.

Testing and Turnover

The overall testing time is reduced in the RAD model as the prototypes are independently tested during every iteration. However, the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues.

The following illustration describes the RAD Model in detail.



RAD Model Vs Traditional SDLC

The traditional SDLC follows a rigid process models with high emphasis on requirement analysis and gathering before the coding starts. It puts pressure on the customer to sign off the requirements before the project starts and the customer doesn't get the feel of the product as there is no working build available for a long time.

The customer may need some changes after he gets to see the software. However, the change process is quite rigid and it may not be feasible to incorporate major changes in the product in the traditional SDLC.

The RAD model focuses on iterative and incremental delivery of working models to the customer. This results in rapid delivery to the customer and customer involvement during the complete development cycle of product reducing the risk of non-conformance with the actual user requirements.

RAD Model - Application

RAD model can be applied successfully to the projects in which clear modularization is possible. If the project cannot be broken into modules, RAD may fail.

The following pointers describe the typical scenarios where RAD can be used −

- RAD should be used only when a system can be modularized to be delivered in an incremental manner.
- It should be used if there is a high availability of designers for modeling.
- It should be used only if the budget permits use of automated code generating tools.
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
- Should be used where the requirements change during the project and working prototypes are to be presented to customer in small iterations of 2-3 months.

RAD Model - Pros and Cons

RAD model enables rapid delivery as it reduces the overall development time due to the reusability of the components and parallel development. RAD works well only if high skilled engineers are available and the customer is also committed to achieve the targeted prototype in the given time frame. If there is commitment lacking on either side the model may fail.

The advantages of the RAD Model are as follows −

- Changing requirements can be accommodated.
- Progress can be measured.
- Iteration time can be short with use of powerful RAD tools.
- Productivity with fewer people in a short time.
- Reduced development time.
- Increases reusability of components.

- Quick initial reviews occur.
- Encourages customer feedback.
- Integration from very beginning solves a lot of integration issues.

The disadvantages of the RAD Model are as follows −

- Dependency on technically strong team members 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.
- Management complexity is more.
- Suitable for systems that are component based and scalable.
- Requires user involvement throughout the life cycle.
- Suitable for project requiring shorter development times.

6. V Model

- V model is known as Verification and Validation model.
- This model is an extension of the waterfall model.
- In the life cycle of V-shaped model, processes are executed sequentially.
- Every phase completes its execution before the execution of next phase begins.

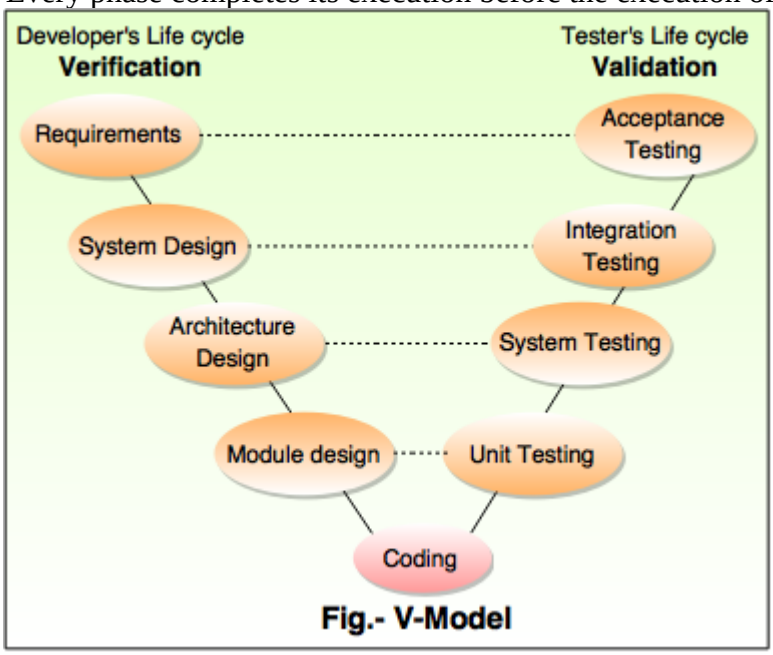


Fig.- V-Model

Following are the phases of V-model:

i) Requirements

- The requirements of product are understood from the customers point of view to know their exact requirement and expectation.
- The acceptance test design planning is completed at requirement stage because, business requirements are used as an input for acceptance testing.

ii) System Design

- In system design, high level design of the software is constructed.
- In this phase, we study how the requirements are implemented their technical use.

iii) Architecture design

- In architecture design, software architecture is created on the basis of high level design.
- The module relationship and dependencies of module, architectural diagrams, database tables, technology details are completed in this phase.

iv) Module design

- In module phase, we separately design every module or the software components.
- Finalize all the methods, classes, interfaces, data types etc.
- Unit tests are designed in module design phase based on the internal module designs.
- Unit tests are the vital part of any development process. They help to remove the maximum faults and errors at an early stage.

v) Coding Phase

- The actual code design of module designed in the design phase is grabbed in the coding phase.
- On the basis of system and architecture requirements, we decide the best suitable programming language.
- The coding is executed on the basis of coding guidelines and standards.

Advantages of V-model

- V-model is easy and simple to use.
- Many testing activities i.e planning, test design are executed in the starting, it saves more time.
- Calculation of errors is done at the starting of the project hence, less chances of error occurred at final phase of testing.

- This model is suitable for small projects where the requirements are easily understood.

Disadvantages of V-model

- V-model is not suitable for large and composite projects.
- If the requirements are not constant then this model is not acceptable.

Difference between the Verification and Validation

Verification	Validation
Verification is the process to find whether the software meets the specified requirements for particular phase.	The validation process checks whether the software meets requirements and expectations of the customer.
It evaluates an intermediate product.	It evaluates the final product.
The objective of verification is to check whether software is constructed according to requirement and design specification.	The objective of validation is to check whether the specifications are correct and satisfy the business need.
It describes whether the outputs are as per the inputs or not.	It explains whether outputs are accepted by the user or not.
Verification is completed before the validation.	It is completed after the verification.
Plans, requirement, specification, code are evaluated in the verifications.	Actual product or software is tested under validation.