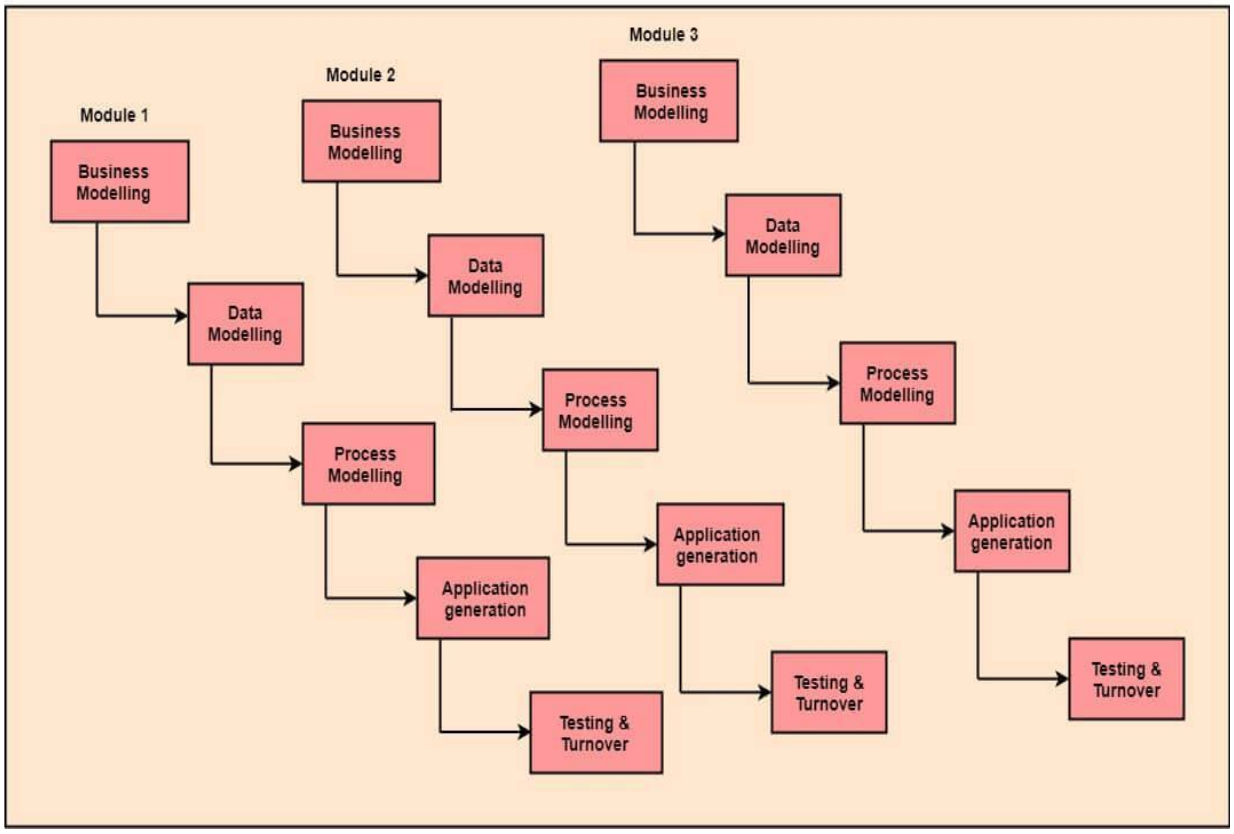
Chapter 1

RAD (Rapid Application Development) Model

RAD is a linear sequential software development process model, It is a type of **incremental model**. that emphasizes a Short development cycle i.e. from 60 to 90 days It uses component based construction approach we develop the modules in parallel. If the requirements are well understood and described, RAD (Rapid Application Development) is a concept that products can be developed faster and of higher quality through:

- Gathering requirements using workshops or focus groups or brain storming
- Prototyping and early, reiterative user testing of designs
- The re-use of software components
- A rigidly paced schedule that refers design improvements to the next product version
- Less formality in reviews and other team communication

Fig: RAD Model



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.

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.

When to use RAD Model?

- When the system should need to create the project that modularizes in a short span time (2-3 months).
- When the requirements are well-known.
- When the technical risk is limited.
- When there's a necessity to make a system, which modularized in 2-3 months of period.
- It should be used only if the budget allows the use of automatic code generating tools.

Advantage of RAD Model

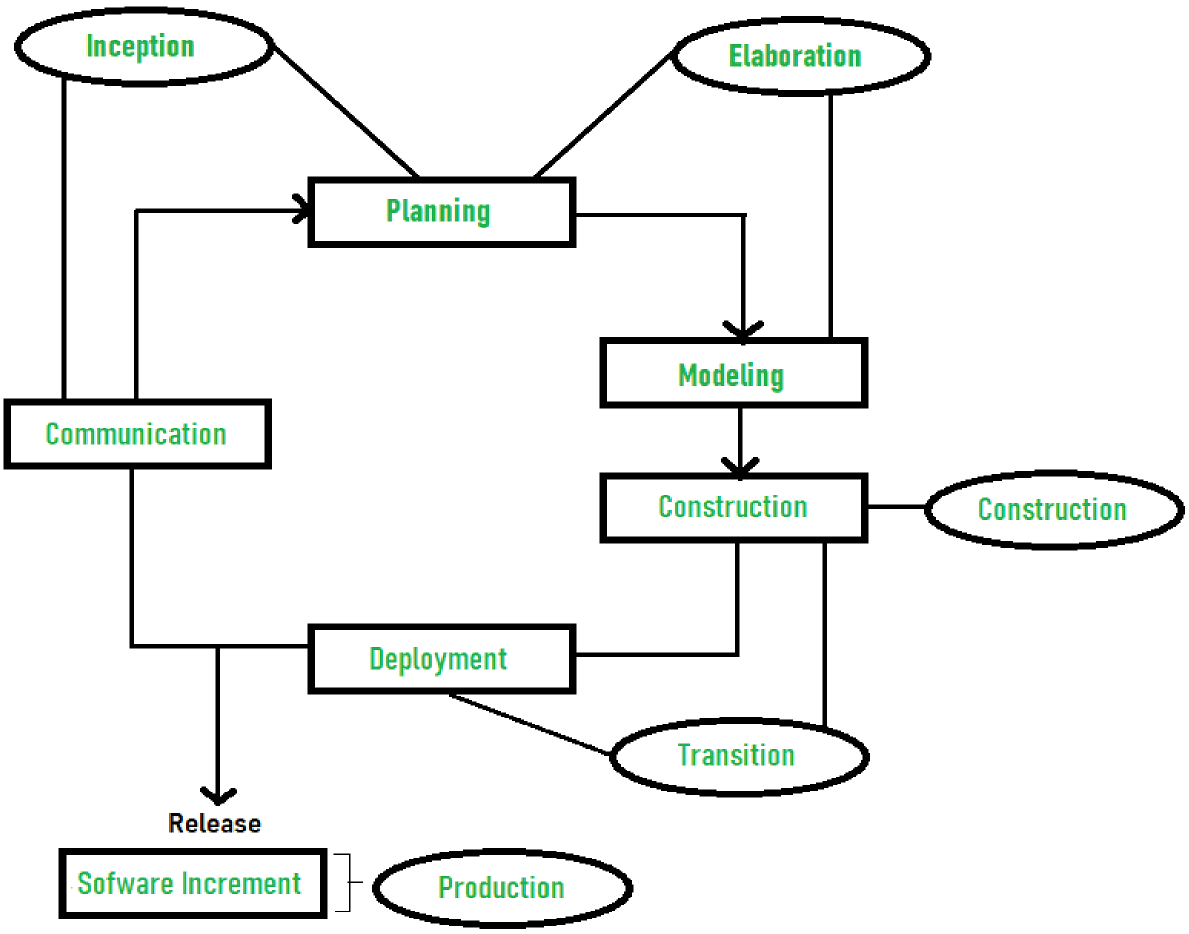
- This model is flexible for change.
- Each phase in RAD brings highest priority functionality to the customer.
- It reduced development time.
- It increases the reusability of features.

Disadvantage of RAD Model

- It required highly skilled designers.
- All application is not compatible with RAD.
- For smaller projects, we cannot use the RAD model.
- On the high technical risk, it's not suitable.
- Required user involvement.

Rational Unified Process (RUP) is a software development process for object-oriented models. It is also known as the Unified Process Model. It is created by Rational corporation and is designed and documented using UML (Unified Modeling Language). This process is included in IBM Rational Method Composer (RMC) product. IBM (International Business Machine Corporation) allows us to customize, design, and personalize the unified process. RUP is proposed by Ivar Jacobson, Grady Bootch, and James Rambaugh. Some characteristics of RUP include use-case driven, Iterative (repetition of the process), and Incremental (increase in value) by nature, delivered online using web technology, can be customized or tailored in modular and electronic form, etc. RUP reduces unexpected development costs and prevents wastage of resources.

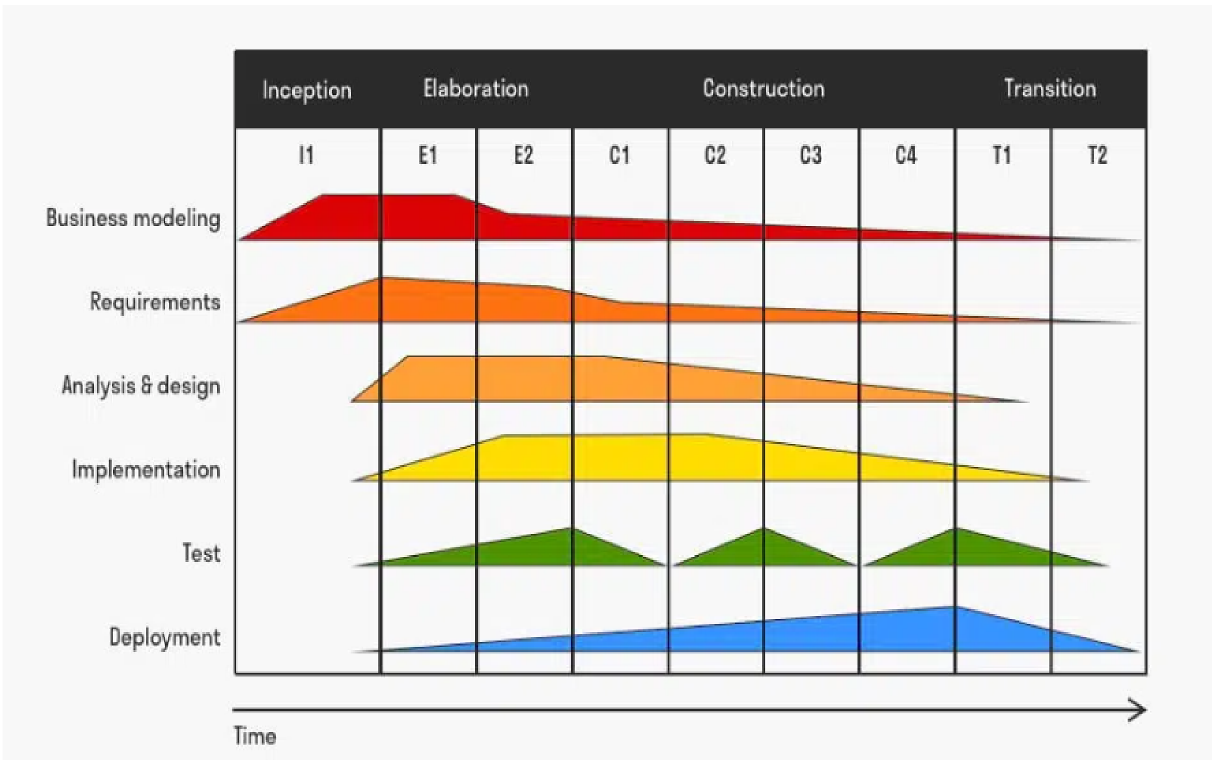
Phases of RUP: There is total of five phases of the life cycle of RUP:



1. **Inception –**
- Communication and planning are the main ones.
- Identifies the scope of the project using a use-case model allowing managers to estimate costs and time required.
- Customers' requirements are identified and then it becomes easy to make a plan for the project.
- The project plan, Project goal, risks, use-case model, and Project description, are made.

- The project is checked against the milestone criteria and if it couldn't pass these criteria then the project can be either canceled or redesigned.

2. **Elaboration –**
   - Planning and modeling are the main ones.
   - A detailed evaluation and development plan is carried out and diminishes the risks.
   - Revise or redefine the use-case model (approx. 80%), business case, and risks.
   - Again, checked against milestone criteria and if it couldn't pass these criteria then again project can be canceled or redesigned.
   - Executable architecture baseline.
3. **Construction –**
   - The project is developed and completed.
   - System or source code is created and then testing is done.
   - Coding takes place.
4. **Transition –**
   - The final project is released to the public.
   - Transit the project from development into production.
   - Update project documentation.
   - Beta testing is conducted.
   - Defects are removed from the project based on feedback from the public.
5. **Production –**
   - The final phase of the model.
   - The project is maintained and updated accordingly.



Advantages:
1. It provides good documentation; it completes the process in itself.
2. It provides risk-management support.
3. It reuses the components, and hence total time duration is less.
4. Good online support is available in the form of tutorials and training.

Disadvantages:
1. Team of expert professionals is required, as the process is complex.
2. Complex and not properly organized process.
3. More dependency on risk management.
4. Hard to integrate again and again.

Chapter 2

Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirement

Non-functional requirements

Here, are some examples of Non functional requirements:

1. Users must change the initially assigned login password immediately after the first successful login. Moreover, the initial should never be reused.
2. Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
3. Every unsuccessful attempt by a user to access an item of data shall be recorded on an audit trail.
4. A website should be capable enough to handle 20 million users with affecting its performance
5. The software should be portable. So moving from one OS to other OS does not create any problem.
6. Privacy of information, the export of restricted technologies, intellectual property rights, etc. should be audited.

Functional vs Non Functional Requirements

Following is the main difference between Functional and Non functional requirements:

Parameters	Functional Requirement	Non-Functional Requirement
What is it?	Verb	Attributes
Requirement	It is mandatory	It is non-mandatory
Capturing type	It is captured in use case.	It is captured as a quality attribute.
End-result	Product feature	Product properties
Capturing	Easy to capture	Hard to capture

Objective	Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Area of focus	Focus on user requirement	Concentrates on the user's expectation.
Documentation	Describe what the product does	Describes how the product works
Type of Testing	Functional Testing like System, Integration, End to End, API testing, etc.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc.
Test Execution	Test Execution is done before non-functional testing.	After the functional testing
Product Info	Product Features	Product Properties

Advantages of Non-Functional Requirement

Benefits/pros of Non-functional testing are:

- The nonfunctional requirements ensure the software system follow legal and compliance rules.
- They ensure the reliability, availability, and performance of the software system
- They ensure good user experience and ease of operating the software.
- They help in formulating security policy of the software system.

Disadvantages of Non-functional requirement

Cons/drawbacks of Non-function requirement are:

- None functional requirement may affect the various high-level software subsystem
- They require special consideration during the software architecture/high-level design phase which increases costs.
- Their implementation does not usually map to the specific software sub-system,
- It is tough to modify non-functional once you pass the architecture phase.

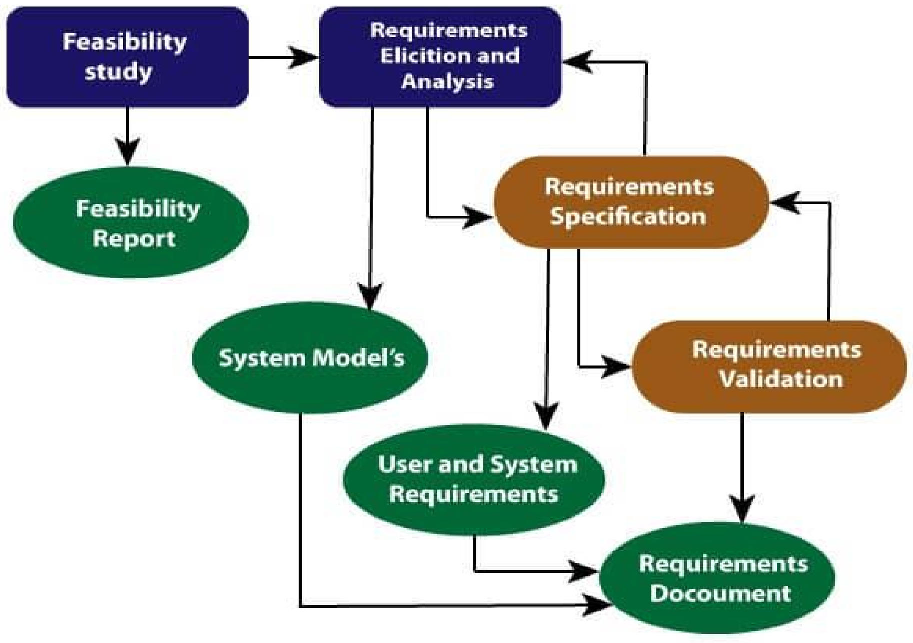
Four Phases of Requirement Engineering

Requirements engineering (RE) refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution,

Requirement Engineering Process

It is a four-step process, which includes -

1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification
4. Software Requirement Validation
5. Software Requirement Management



Requirement Engineering Process

1. Feasibility Study:

The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.

Types of Feasibility:

- **Technical Feasibility** - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.
- **Operational Feasibility** - Operational feasibility is **the measure of how well a proposed system solves the problems**, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.
- **Economic Feasibility** - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

2. Requirement Elicitation and Analysis:

This is also known as the **gathering of requirements**. Here, requirements are identified with the help of customers and existing systems processes, if available.

Analysis of requirements starts with requirement elicitation. The requirements are analyzed to identify inconsistencies, defects, omission, etc. We describe requirements in terms of relationships and also resolve conflicts if any.

Problems of Elicitation and Analysis

- Getting all, and only, the right people involved.
- Stakeholders often don't know what they want
- Stakeholders express requirements in their terms.
- Stakeholders may have conflicting requirements.
- Requirement change during the analysis process.
- Organizational and political factors may influence system requirements.

3. Software Requirement Specification:

Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language. It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.

The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.

- **Data Flow Diagrams:** Data Flow Diagrams (DFDs) are used widely for modeling the requirements. DFD shows the flow of data through a system. The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding. The DFD is also known as a data flow graph or bubble chart.
- **Data Dictionaries:** Data Dictionaries are simply repositories to store information about all data items defined in DFDs. At the requirements stage, the data dictionary should at least define customer data items, to ensure that the customer and developers use the same definition and terminologies.
- **Entity-Relationship Diagrams:** Another tool for requirement specification is the entity-relationship diagram, often called an "***E-R diagram***." It is a detailed logical representation of the data for the organization and uses three main constructs i.e. data entities, relationships, and their associated attributes.

4. Software Requirement Validation:

After requirement specifications are developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs. Requirements can be the check against the following conditions -

- If they can practically implement
- If they are correct and as per the functionality and specially of software
- If there are any ambiguities
- If they are full
- If they can describe

Requirements Validation Techniques

- **Requirements reviews/inspections:** systematic manual analysis of the requirements.
- **Prototyping:** Using an executable model of the system to check requirements.
- **Test-case generation:** Developing tests for requirements to check testability.
- **Automated consistency analysis:** checking for the consistency of structured requirements descriptions.

Software Requirement Management:

Requirement management is the process of managing changing requirements during the requirements engineering process and system development.

New requirements emerge during the process as business needs a change, and a better understanding of the system is developed.

The priority of requirements from different viewpoints changes during the development process.

The business and technical environment of the system changes during the development.