Software Engineering

The term software engineering is the product of two words, software and engineering.

- The **software** is a collection of integrated programs.
- **Engineering** is the application of scientific and practical knowledge to invent, design, build, maintain, and improve frameworks, processes, etc.
- **Software Engineering** is an engineering branch related to the evolution of software product using well-defined scientific principles, techniques, and procedures. The result of software engineering is an effective and reliable software product.

Why is Software Engineering required

Software Engineering is required due to the following reasons:

- To manage Large software
- For more Scalability
- Cost Management
- To manage the dynamic nature of software
- For better quality Management

Types of Software

1. Generic
2. Customized

Characteristics of a good software Engineer

- Exposure to systematic methods, i.e., familiarity with software engineering principles.
- Good technical knowledge of the project range (Domain knowledge).
- Good programming abilities.
- Good communication skills. These skills comprise of oral, written, and interpersonal skills.
- High motivation.
- Sound knowledge of fundamentals of computer science.
- Intelligence.
- Ability to work in a team, etc.

Software Processes

A software process is the set of activities and associated outcome that produce a software product.

Software specifications: The functionality of the software and constraints on its operation must be defined.

Software development: The software to meet the requirement must be produced.

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

Software evolution: The software must evolve to meet changing client needs.

Software Crisis

- **Size:** Software is becoming more expensive and more complex with the growing complexity and expectation out of software

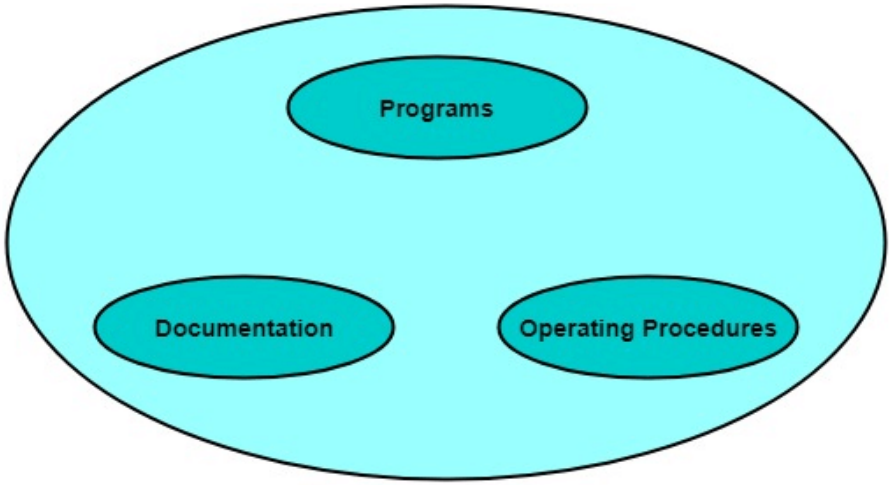
- **Quality:** Many software products have poor quality

- **Cost:** Software development is costly

- **Delayed Delivery:** Very often the software takes longer than the estimated time to develop, which in turn leads to cost shooting up. For example, one in four large-scale development projects is never completed.

Program vs. Software

- Software is more than programs. Any program is a subset of software, and it becomes software only if documentation & operating procedures manuals are prepared.
- There are three components of the software as shown below:



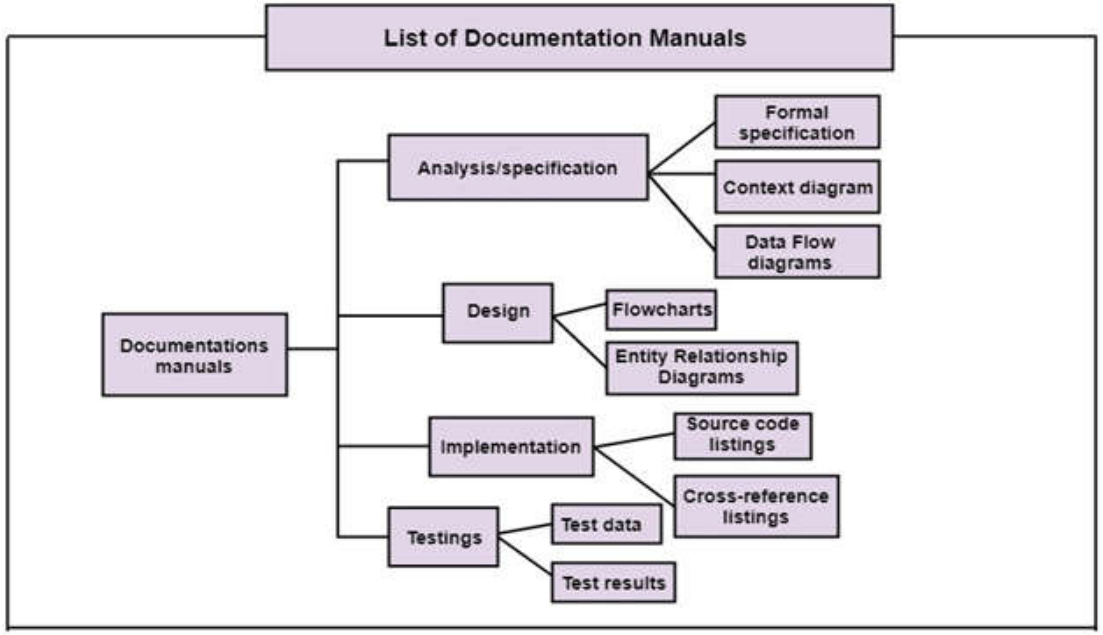
Software= Program + Documentation + Operating Procedures

Fig:Components of Software

Program vs. Software (cont.)

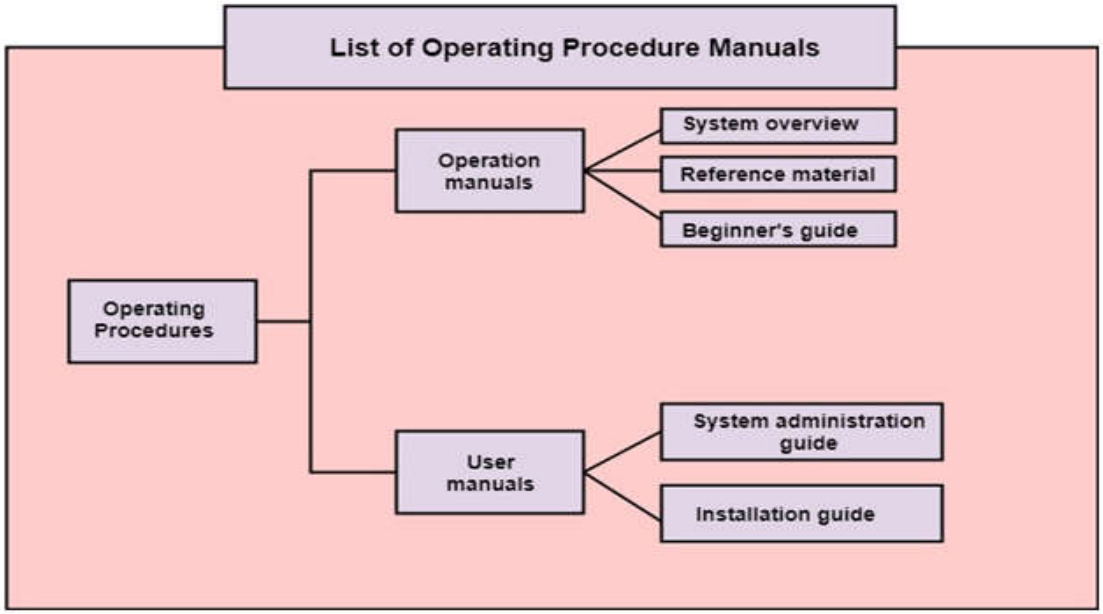
Program: Program is a combination of source code & object code.

Documentation: Documentation consists of different types of manuals. Examples of documentation manuals are: Data Flow Diagram. Flow Charts, ER diagrams, etc.



Program vs. Software (cont.)

Operating Procedures: Operating Procedures consist of instructions to set up and use the software system and instructions on how to react to the system failure. Example: installation guide, Beginner's guide, reference guide, system administration guide, etc.



Software Development Life Cycle (SDLC)

- A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle.

- A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.

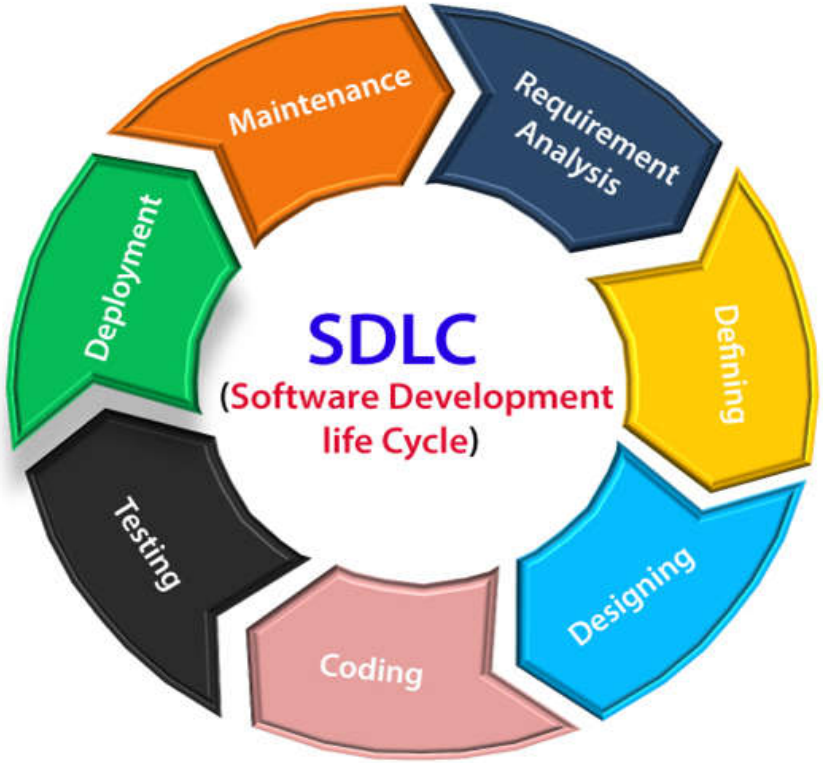
- a life cycle model maps the various activities performed on a software product from its inception to retirement.

Need of SDLC

- The development team must determine a suitable life cycle model for a particular plan and then observe to it.
- Without using an exact life cycle model, the development of a software product would not be in a systematic and disciplined manner.
- When a team is developing a software product, there must be a clear understanding among team representative about when and what to do.
- A software life cycle model describes entry and exit criteria for each phase. A phase can begin only if its stage-entry criteria have been fulfilled. So without a software life cycle model, the entry and exit criteria for a stage cannot be recognized. Without software life cycle models, it becomes tough for software project managers to monitor the progress of the project.

SDLC Cycle

- SDLC Cycle represents the process of developing software. SDLC framework includes the following steps:



The stages of SDLC

Stage1: Planning and requirement analysis

- Requirement Analysis is the most important and necessary stage in SDLC.

- The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry.

- Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.

The stages of SDLC (cont.)

Stage2: Defining Requirements

- Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stakeholders.
- This is accomplished through "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project life cycle.

Stage3: Designing the Software

- The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.

The stages of SDLC (cont.)

Stage4: Developing the project

- In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.

Stage5: Testing

- After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.
- During this stage, unit testing, integration testing, system testing, acceptance testing are done.

The stages of SDLC (cont.)

Stage6: Deployment

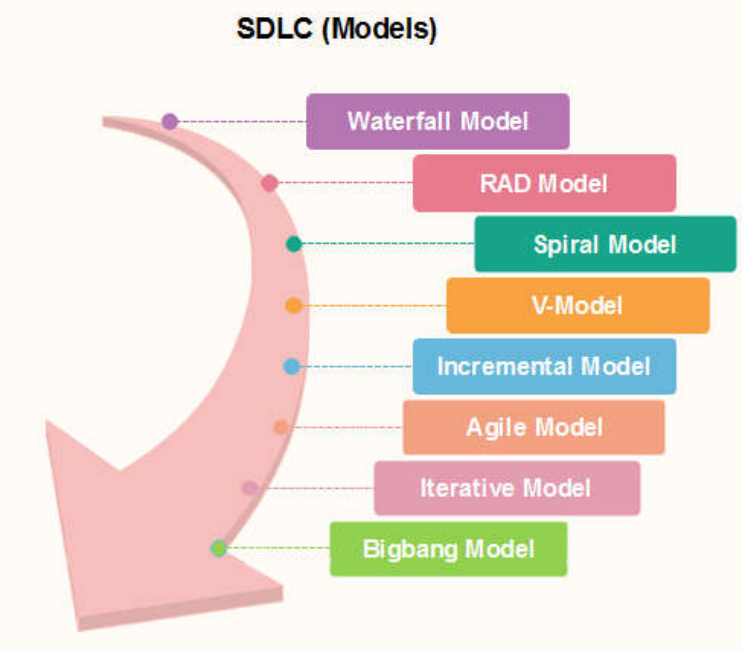
- Once the software is certified, and no bugs or errors are stated, then it is deployed.
- Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment.
- After the software is deployed, then its maintenance begins.

Stage7: Maintenance

- Once the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time.
- This procedure where the care is taken for the developed product is known as maintenance.

SDLC Models

There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "**Software Development Process Models**."



SDLC Models

Waterfall Model

- In this method, the whole process of software development is divided into various phases.
- The waterfall model is a continuous software development model in which development is seen as flowing steadily downwards (like a waterfall) through the steps of requirements analysis, design, implementation, testing (validation), integration, and maintenance.

V-Model

- In this type of SDLC model testing and the development, the step is planned in parallel. So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase.

SDLC Models (cont.)

RAD Model

- RAD or Rapid Application Development process is an adoption of the waterfall model; it targets developing software in a short period. The RAD model is based on the concept that a better system can be developed in lesser time by using focus groups to gather system requirements.
- Business Modeling
- Data Modeling
- Process Modeling
- Application Generation
- Testing and Turnover

SDLC Models (cont.)

Spiral Model

- The spiral model is a **risk-driven process model**. This SDLC model helps the group to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc. The spiral technique is a combination of rapid prototyping and concurrency in design and development activities.
- Each cycle in the spiral begins with the identification of objectives for that cycle, the different alternatives that are possible for achieving the goals, and the constraints that exist. This is the first quadrant of the cycle (upper-left quadrant).
- The next step in the cycle is to evaluate these different alternatives based on the objectives and constraints. The focus of evaluation in this step is based on the risk perception for the project.
- The next step is to develop strategies that solve uncertainties and risks. This step may involve activities such as benchmarking, simulation, and prototyping.

SDLC Models (cont.)

Incremental Model

- The incremental model is not a separate model. It is necessarily a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC process is repeated, with each release adding more functionality until all requirements are met. In this method, each cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.

SDLC Models (cont.)

Agile Model

- Agile methodology is a practice which promotes continues interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.
- Analysis, design, development, and testing are not as predictable (from a planning point of view) as we might like.

Iterative Model

- It is a particular implementation of a software development life cycle that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete. In short, iterative development is a way of breaking down the software development of a large application into smaller pieces.

SDLC Models (cont.)

Big bang model

- Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come.
- This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements are either unknown or final release date is not given.

SDLC Models (cont.)

Prototype Model

- The prototyping model starts with the requirements gathering. The developer and the user meet and define the purpose of the software, identify the needs, etc.
- A '**quick design**' is then created. This design focuses on those aspects of the software that will be visible to the user. It then leads to the development of a prototype. The customer then checks the prototype, and any modifications or changes that are needed are made to the prototype. Looping takes place in this step, and better versions of the prototype are created.

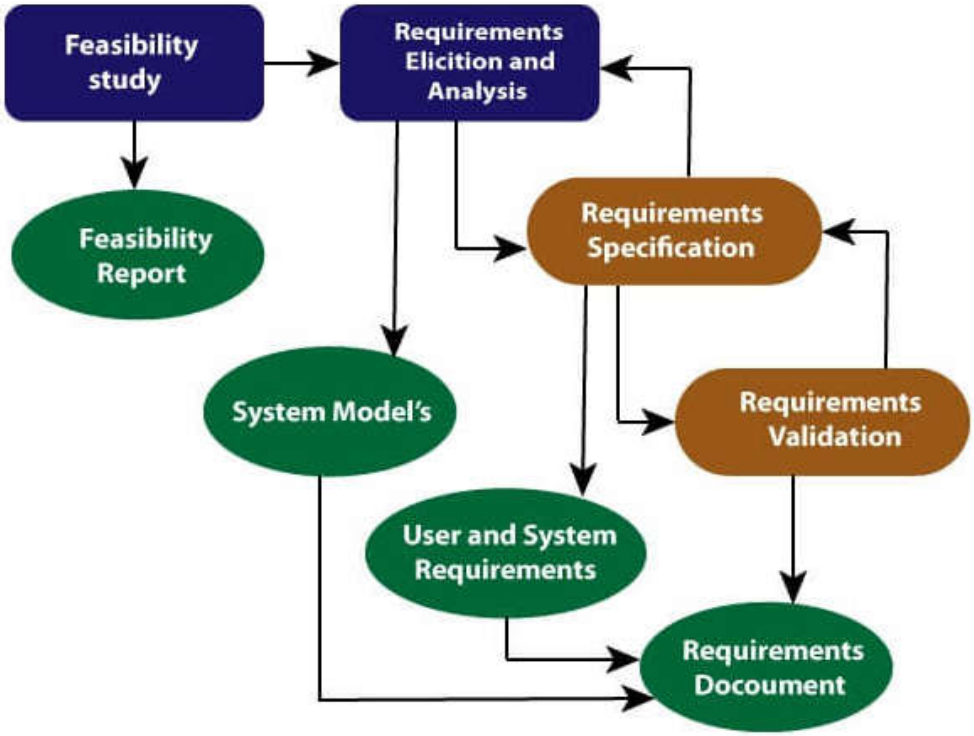
Requirement Engineering

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, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system.

Requirement Engineering Process

- Feasibility Study
- Requirement Elicitation and Analysis
- Software Requirement Specification
- Software Requirement Validation
- Software Requirement Management

Requirement Engineering (Cont.)



Requirement Engineering Process

Requirement Engineering (Cont.)

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 assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.
- Economic Feasibility - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

Requirement Engineering (Cont.)

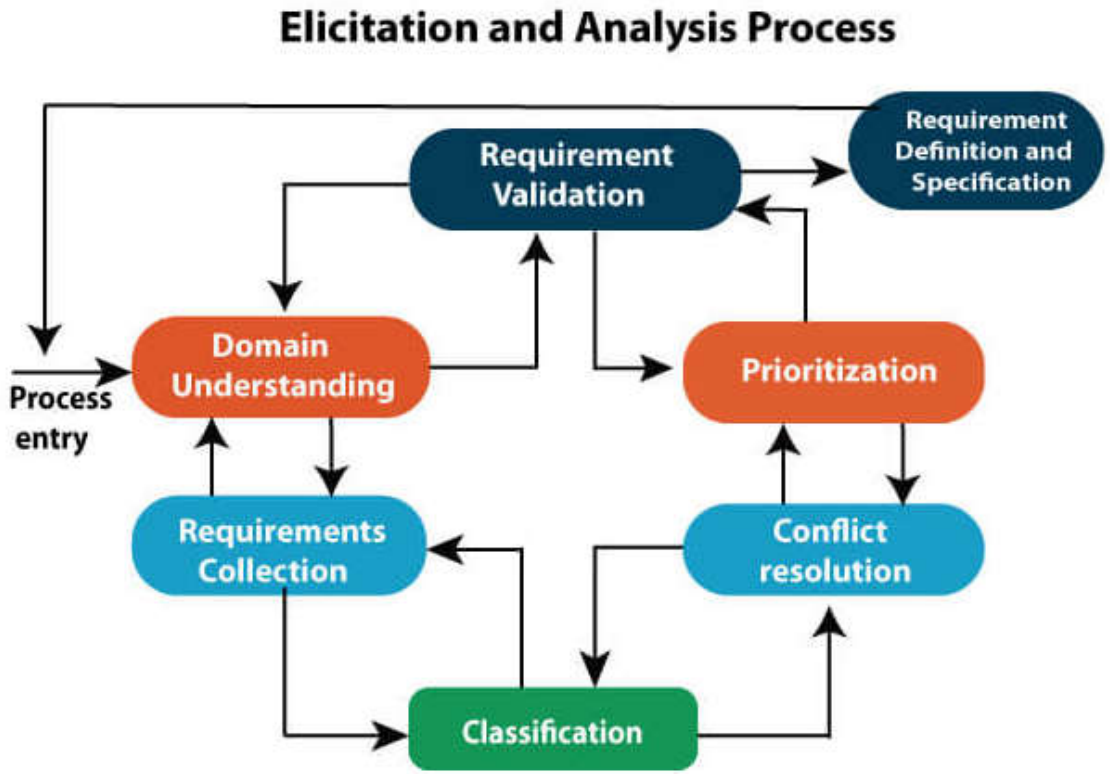
Requirement Elicitation and Analysis:

Requirements are identified with the help of customers and existing systems processes, if available. The requirements are analyzed to identify inconsistencies, defects, omission, etc.

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.

Requirement Engineering (Cont.)



Requirement Engineering (Cont.)

Software Requirement Specification:

- 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:** DFD shows the flow of data through a system. 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.
- **Entity-Relationship Diagrams:** 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.

Requirement Engineering (Cont.)

Software Requirement Validation:

- After requirement specifications 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 are correct and as per the functionality and specialty of software
- If there are any ambiguities
- 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.

Requirement Engineering (Cont.)

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

Prerequisite of Software requirements

Software Requirements: Largely software requirements must be categorized into two categories:

Functional Requirements: Functional requirements define a function that a system or system element must be qualified to perform and must be documented in different forms. The functional requirements are describing the behavior of the system as it correlates to the system's functionality.

Non-functional Requirements: This can be the necessities that specify the criteria that can be used to decide the operation instead of specific behaviors of the system.

- Non-functional requirements are divided into two main categories:
- Execution qualities like security and usability, which are observable at run time.
- Evolution qualities like testability, maintainability, extensibility, and scalability that embodied in the static structure of the software system.

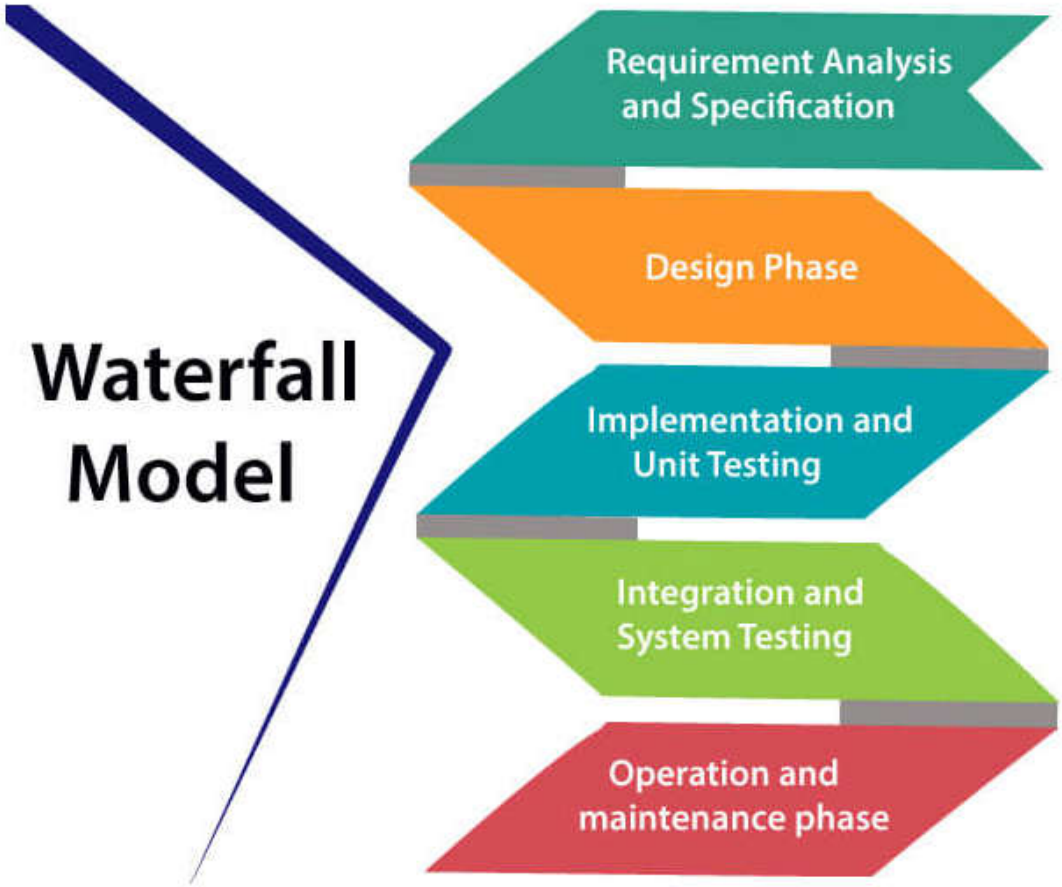
Prerequisite of Software requirements

Collection of software requirements is the basis of the entire software development project. Hence they should be clear, correct, and well-defined. A complete Software Requirement Specifications should be:

- Clear
- Correct
- Consistent
- Coherent
- Comprehensible
- Modifiable
- Verifiable
- Prioritized
- Unambiguous
- Traceable
- Credible source

Software Engineering Models

Model-Waterfall



Model-Waterfall

When to use Water Fall

- When the requirements are constant and not changed regularly.
- A project is short
- The situation is calm
- Where the tools and technology used is consistent and is not changing
- When resources are well prepared and are available to use.

Model-Waterfall

Advantages of Waterfall model

- This model is simple to implement also the number of resources that are required for it is minimal.
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
- The start and end points for each phase is fixed, which makes it easy to cover progress.
- The release date for the complete product, as well as its final cost, can be determined before development.
- It gives easy to control and clarity for the customer due to a strict reporting system.

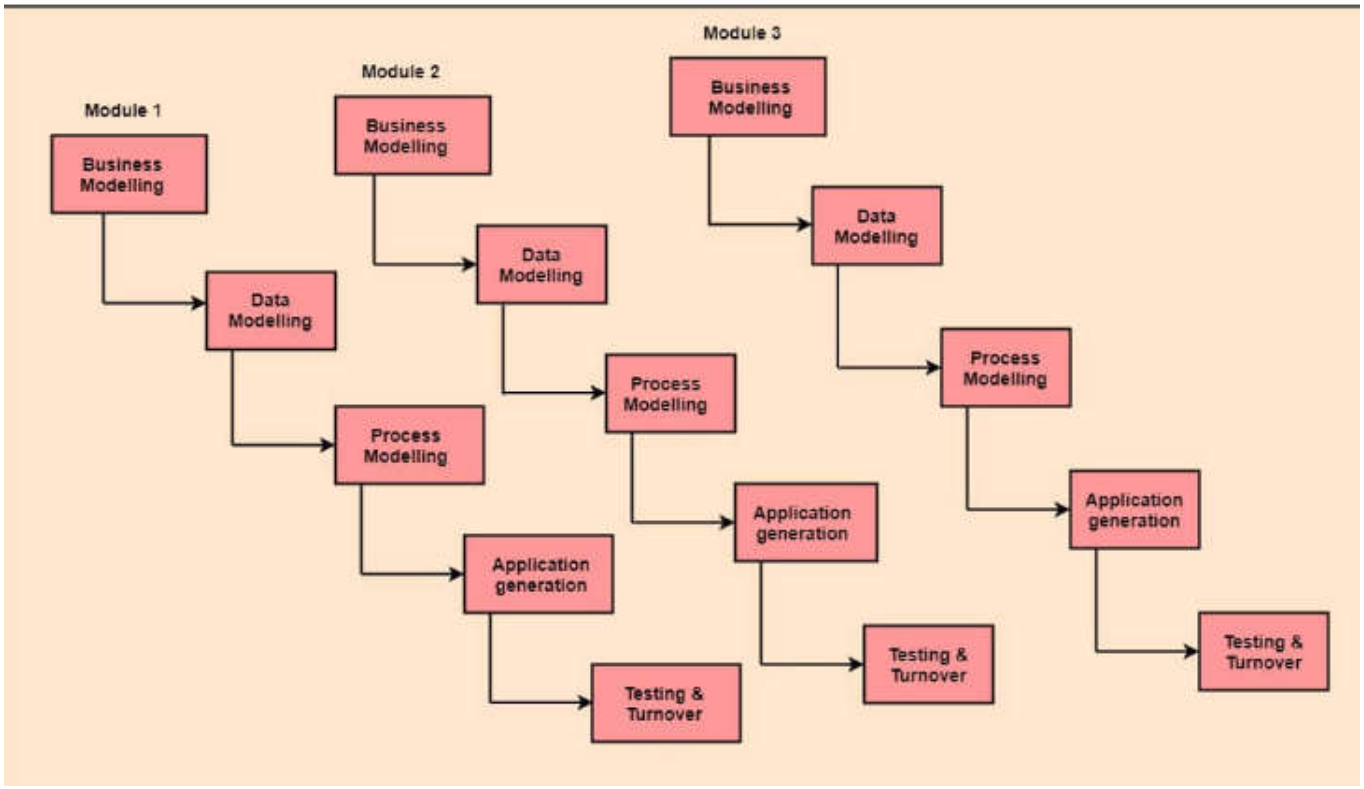
Model-Waterfall

Disadvantages of Waterfall model

- In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
- This model cannot accept the changes in requirements during development.
- It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement, It becomes tough to go back and change it.
- Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare.

Model-Rapid Application Development

RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element based construction approach.



Model-Rapid Application Development

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.

Model-Rapid Application Development

Advantage of RAD Model

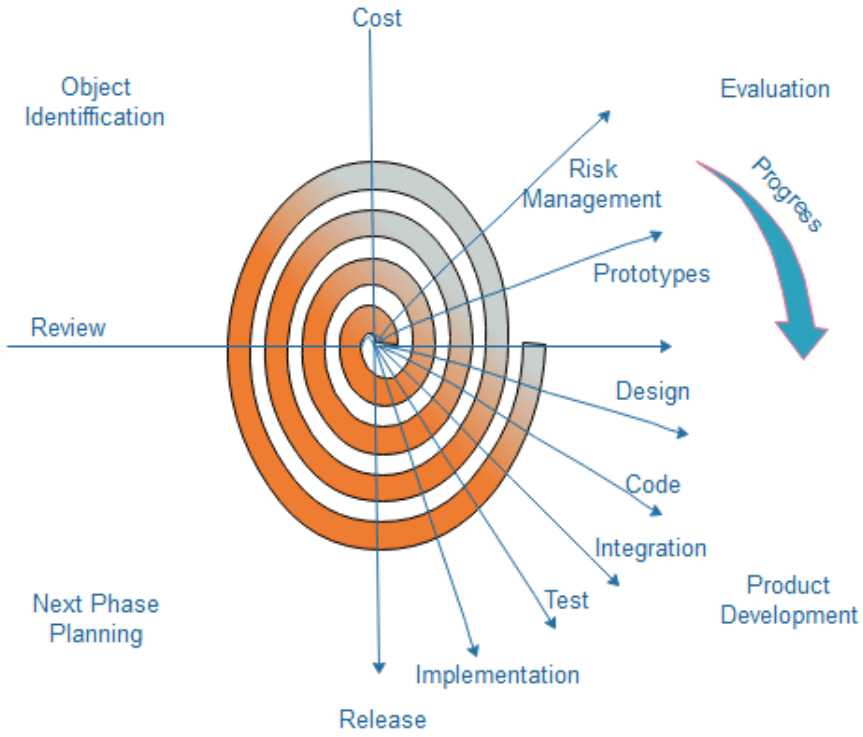
- This model is flexible for change.
- In this model, changes are adoptable.
- Each phase in RAD brings highest priority functionality to the customer.
- It reduced development time.
- It increases the reusability of features.

Model-Rapid Application Development

- 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.

Model-Spiral Model

- couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model.



Model-Spiral Model

When to use Spiral Model?

- When deliverance is required to be frequent.
- When the project is large
- When requirements are unclear and complex
- When changes may require at any time
- Large and high budget projects

Model-Spiral Model

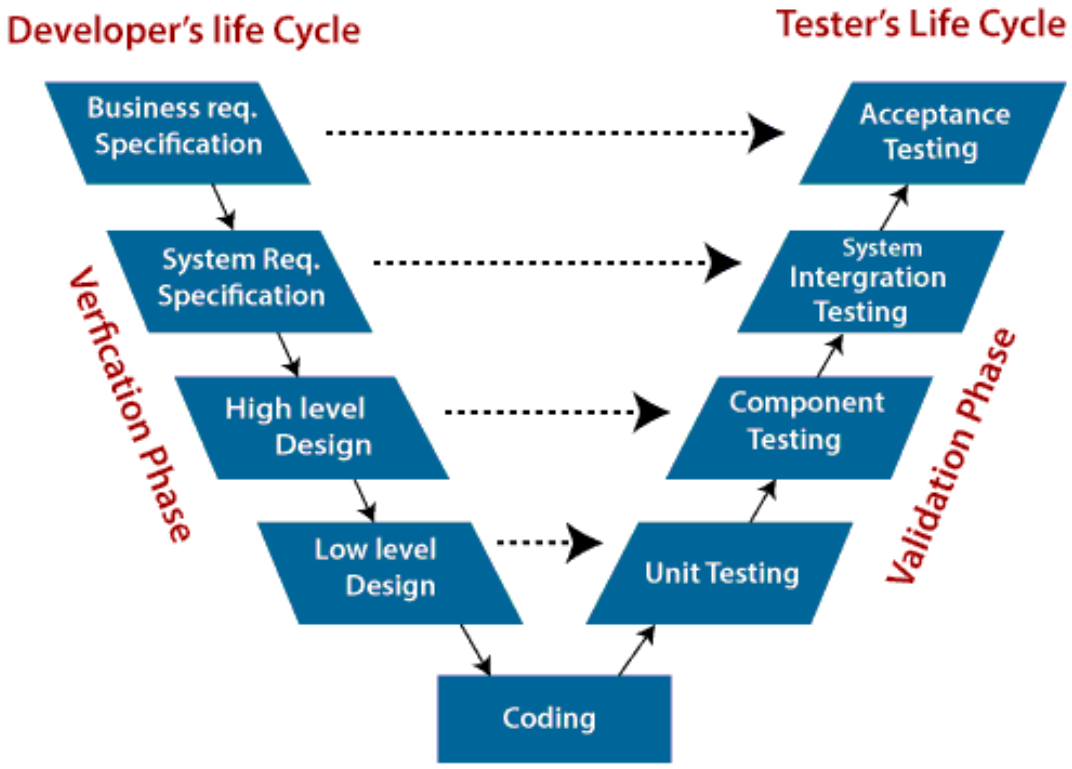
Advantages

- High amount of risk analysis
- Useful for large and mission-critical projects.

Disadvantages

- Can be a costly model to use.
- Risk analysis needed highly particular expertise
- Doesn't work well for smaller projects.

Model-Verification and Validation Model



Model-Verification and Validation Model

When to use V-Model?

- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

Disadvantage (Cons) of V-Model:

1. Very rigid and least flexible.
2. Not a good for a complex project.
3. Software is developed during the implementation stage, so no early prototypes of the software are produced.
4. If any changes happen in the midway, then the test documents along with the required documents, has to be updated.

Model-Verification and Validation Model

Advantage (Pros) of V-Model:

1. Easy to Understand.
2. Testing Methods like planning, test designing happens well before coding.
3. This saves a lot of time. Hence a higher chance of success over the waterfall model.
4. Avoids the downward flow of the defects.
5. Works well for small plans where requirements are easily understood.

Model-Agile Model

Model-Agile Model

Agile Testing Methods:

- Scrum - Crystal -Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD) -Lean Software Development
- eXtreme Programming(XP)

Scrum

- SCRUM is an agile development process focused primarily on ways to manage tasks in team-based development conditions.
- There are three roles in it, and their responsibilities are:
- **Scrum Master:** The scrum can set up the master team, arrange the meeting and remove obstacles for the process
- **Product owner:** The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
- **Scrum Team:** The team manages its work and organizes the work to complete the sprint or cycle.

Model-Agile Model

Agile Testing Methods:

eXtreme Programming(XP)

- This type of methodology is used when customers are constantly changing demands or requirements, or when they are not sure about the system's performance.

Crystal:

- There are three concepts of this method-

1. Chartering: Multi activities are involved in this phase such as making a development team, performing feasibility analysis, developing plans, etc.
2. Cyclic delivery: under this, two more cycles consist, these are:
    1. Team updates the release plan.
    2. Integrated product delivers to the users.
3. Wrap up: According to the user environment, this phase performs deployment, post-deployment.

Model-Agile Model

Agile Testing Methods:

Dynamic Software Development Method(DSDM):

The essential features of DSDM are that users must be actively connected, and teams have been given the right to make decisions. The techniques used in DSDM are:

1. Time Boxing
2. MoSCoW Rules
3. Prototyping

The DSDM project contains seven stages:

1. Pre-project
2. Feasibility Study
3. Business Study
4. Functional Model Iteration
5. Design and build Iteration
6. Implementation
7. Post-project

Model-Agile Model

Agile Testing Methods:

Feature Driven Development(FDD):

- This method focuses on "Designing and Building" features. In contrast to other smart methods, FDD describes the small steps of the work that should be obtained separately per function.

Lean Software Development:

- Lean software development methodology follows the principle "just in time production." The lean method indicates the increasing speed of software development and reducing costs. Lean development can be summarized in seven phases.
.

Model-Agile Model

Agile Testing Methods:

- When to use the Agile Model?
- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

Model-Agile Model

- Advantage(Pros) of Agile Method:

1.Frequent Delivery

2.Face-to-Face Communication with clients.

3.Efficient design and fulfils the business requirement.

4.Anytime changes are acceptable.

5.It reduces total development time.

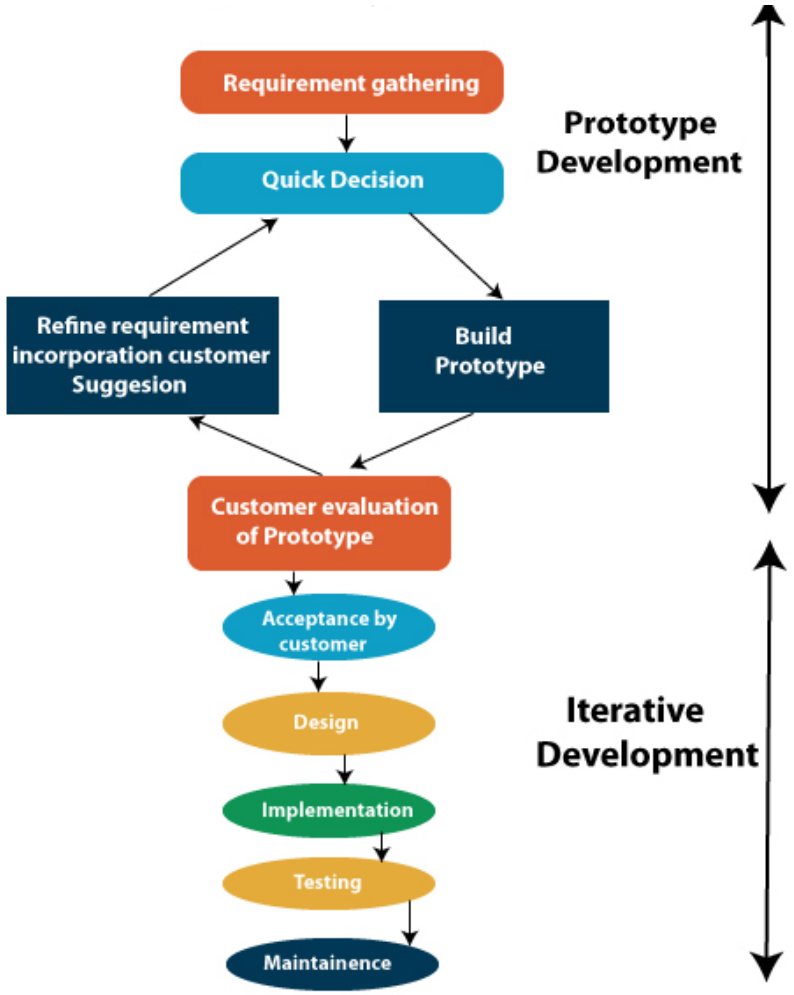
- Disadvantages(Cons) of Agile Model:

1.Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.

2.Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

.

Model- Prototype Model



Model- Prototype Model

Advantage of Prototype Model

1. Reduce the risk of incorrect user requirement
2. Good where requirement are changing/uncommitted
3. Regular visible process aids management
4. Support early product marketing
5. Reduce Maintenance cost.
6. Errors can be detected much earlier as the system is made side by side.

Model- Prototype Model

Disadvantage of Prototype Model

1. An unstable/badly implemented prototype often becomes the final product.
2. Require extensive customer collaboration
    1. Costs customer money
    2. Needs committed customer
    3. Difficult to finish if customer withdraw
    4. May be too customer specific, no broad market
3. Difficult to know how long the project will last.
4. Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
5. Prototyping tools are expensive.
6. Special tools & techniques are required to build a prototype.
7. It is a time-consuming process.

Model- Iterative Model

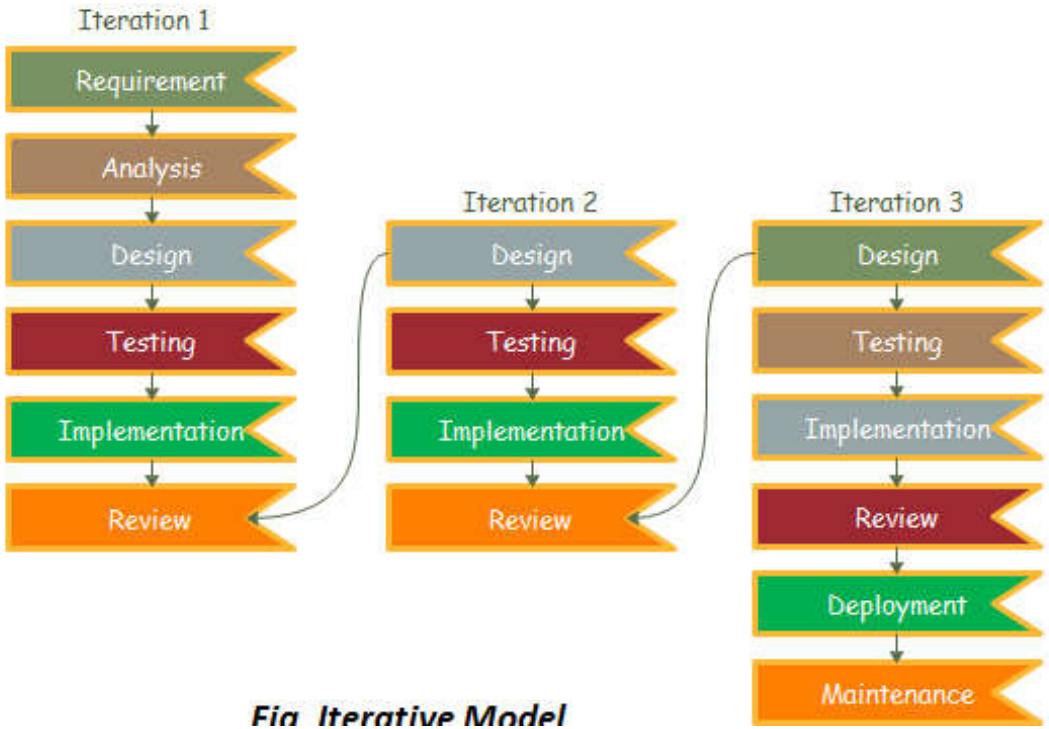


Fig. Iterative Model

Model- Iterative Model

When to use the Iterative Model?

1. When requirements are defined clearly and easy to understand.
2. When the software application is large.
3. When there is a requirement of changes in future.

Advantage(Pros) of Iterative Model:

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.

Model- Iterative Model

Disadvantage(Cons) of Iterative Model:

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.