SOFTWARE ENGINEERING

Dr.M.S.R LAKSHMI REDDY ASSOC.PROF CSE

Introduction to Software Engineering

Term **software engineering** is the product of two words, **software**, and **engineering**.

Software:

- The **software** is a collection of integrated programs.
- Software subsists of carefully-organized instructions and code written by developers on any of various particular computer languages.
- Computer programs and related documentation such as requirements, design models and user manuals.

Software is classified into two types

1. Generic
2. Custom

Generic:

That means developed to be sold to a range of different customers.

Eg: Firefox, Excel, Word, Power point etc.,

Custom:

That means developed for a single customer according to their specification.

Eg: Custom E-Commerce websites, Custom Applications.

Engineering:

Engineering is the application of science and maths to solve problems.

(or)

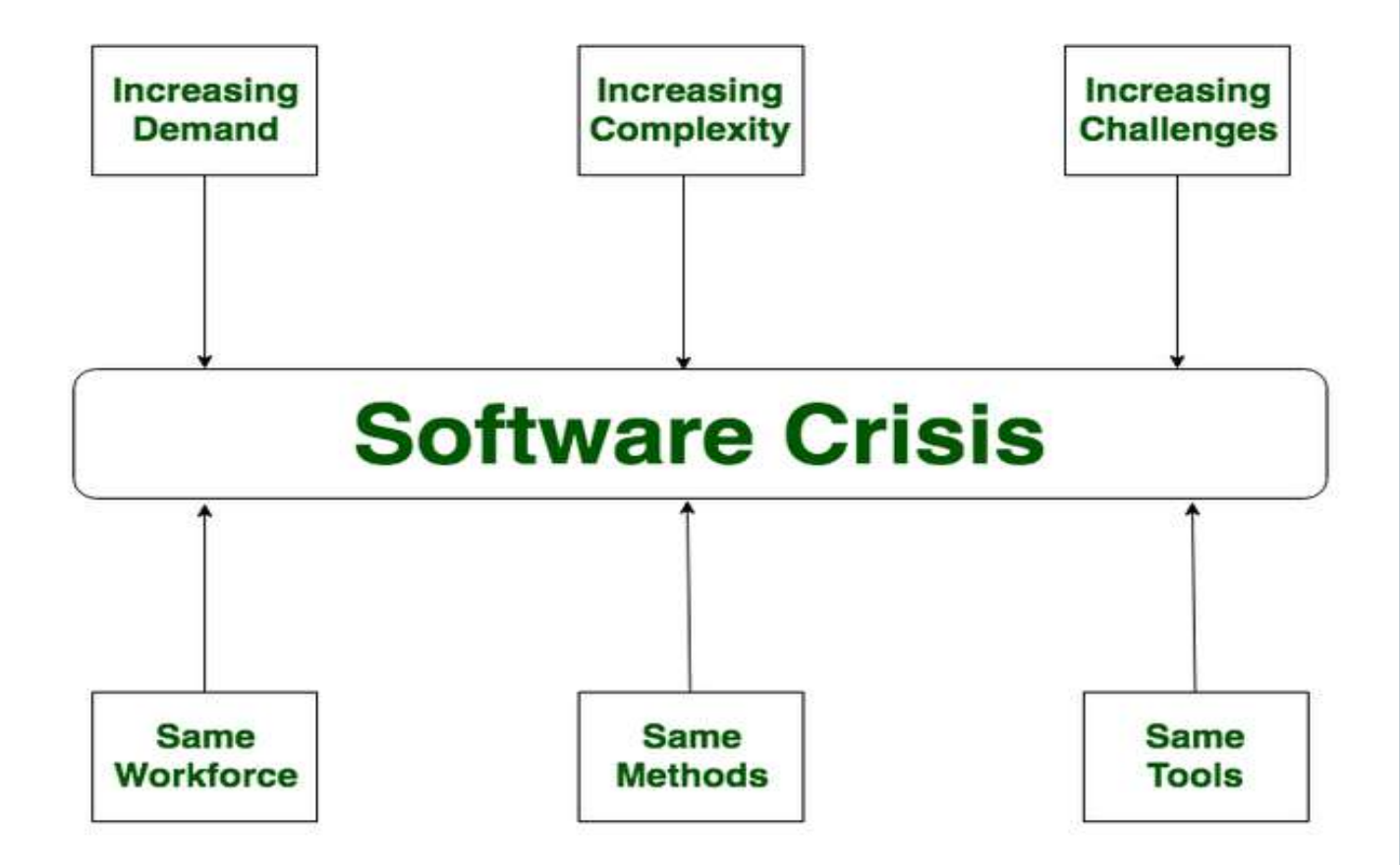
It is the process of designing and building something that serves a particular purpose and finds a cost effective solution to problems.

Software Crisis

It is a term used in early days of computing science for the difficulty of writing useful & efficient program in required time.

Causes of Software crisis:

- Project running over budget
- Project running overtime(Missied dead lines)
- Software was very inefficient
- Software was low quality
- Mismatch between user needs and final product.
- Lack of skilled developers and standard practices


Increasing Demand
Increasing Complexity
Increasing Challenges
Software Crisis
Same Workforce
Same Methods
Same Tools

Main solution of software crisis is **software Engineering.**

Software Engineering

It is defined as systematic, disciplined, quantifiable approach to the development, operation and maintenance of software. That is, the application of engineering to software

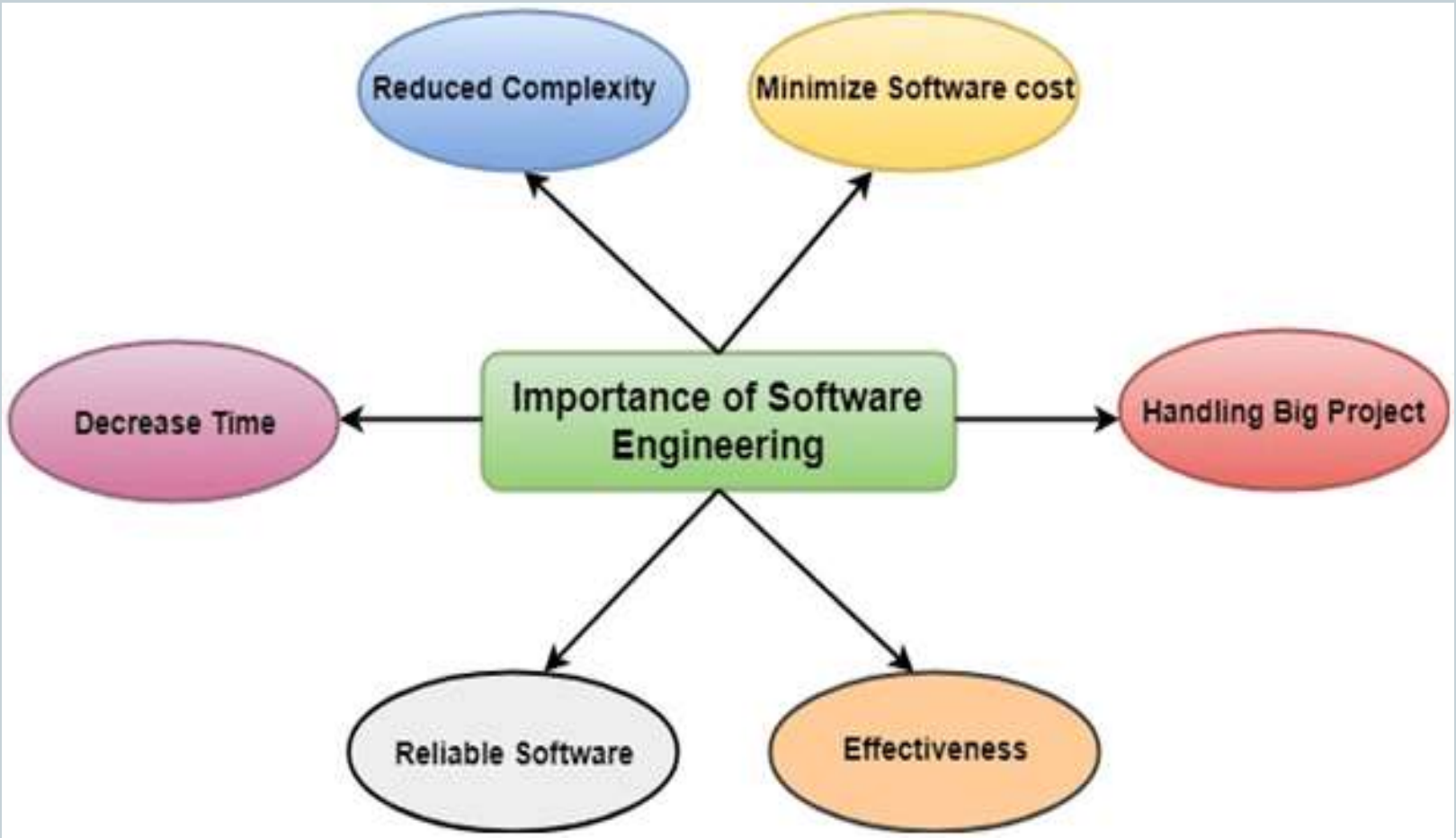
"Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software."

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

Importance of Software Engineering



The Evolving Role of Software Engineering

- The evolving role of software means changing role of software.
- Basically software plays a dual role
  - Software as a product
  - Software as a process

Software as a Product

- Being a product the role of software can be recognized by its *computing potentials, hardware capabilities and accessibility of network computers by the local hardware*.
- Being a product it also acts as an information transformer i.e. *producing, managing, modifying and displaying the source of information*.

  Ex: Traditional desktop applications like Microsoft Office, Adobe Photoshop, and video games sold on discs are examples of software products.

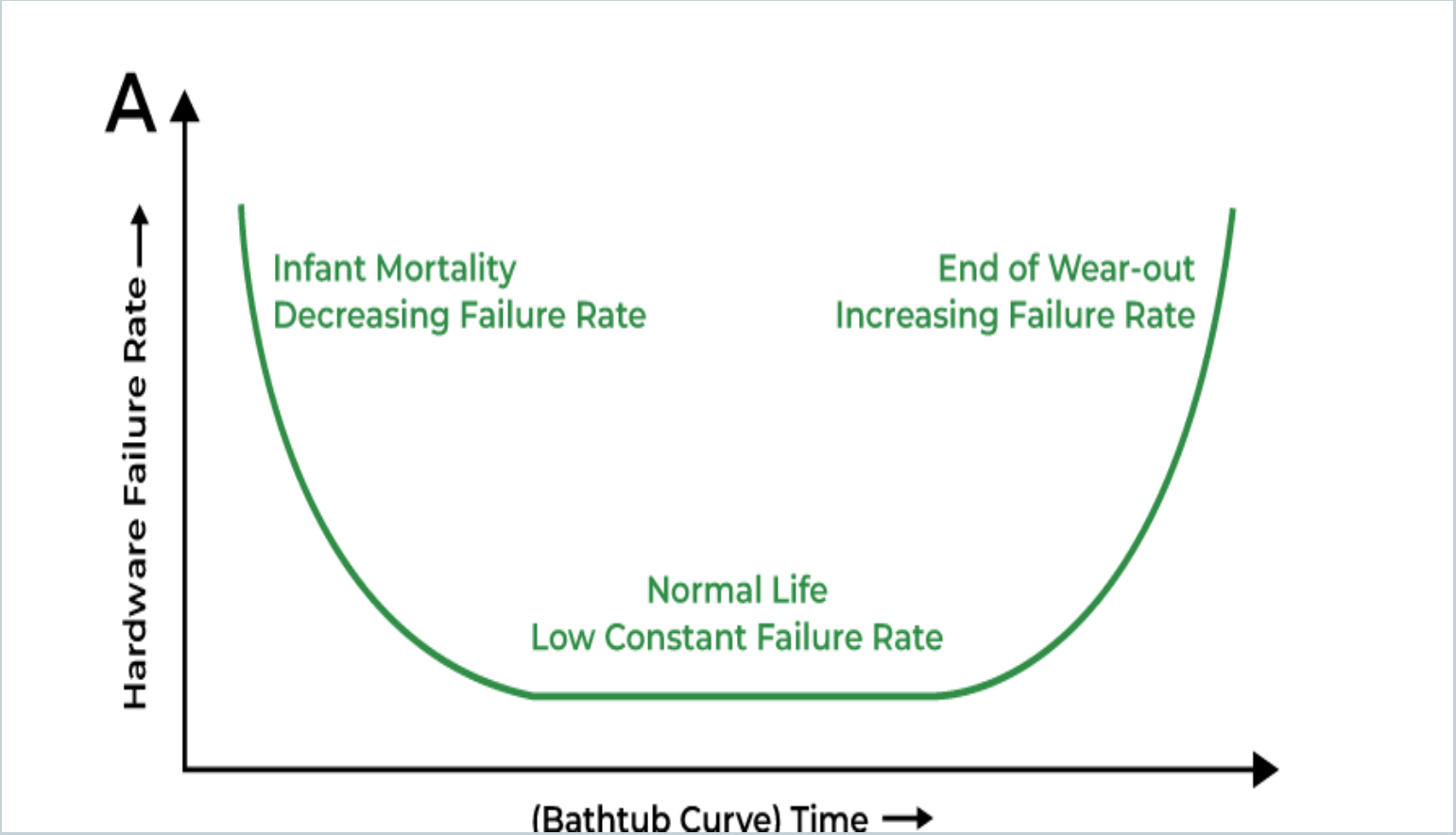
Software as a Process(Vehicle)

- Being a process the software acts as a *vehicle* for delivering the *product*.
- It is a information transformer.
- It transform in *single bit or Multimedia*
- It supports or directly provides system functions
- In this role the duty of software is to *control the computer Ex: os or to establish communications between the computers. Ex: networking*
- It helps in building other software's *Ex: s/w tools*

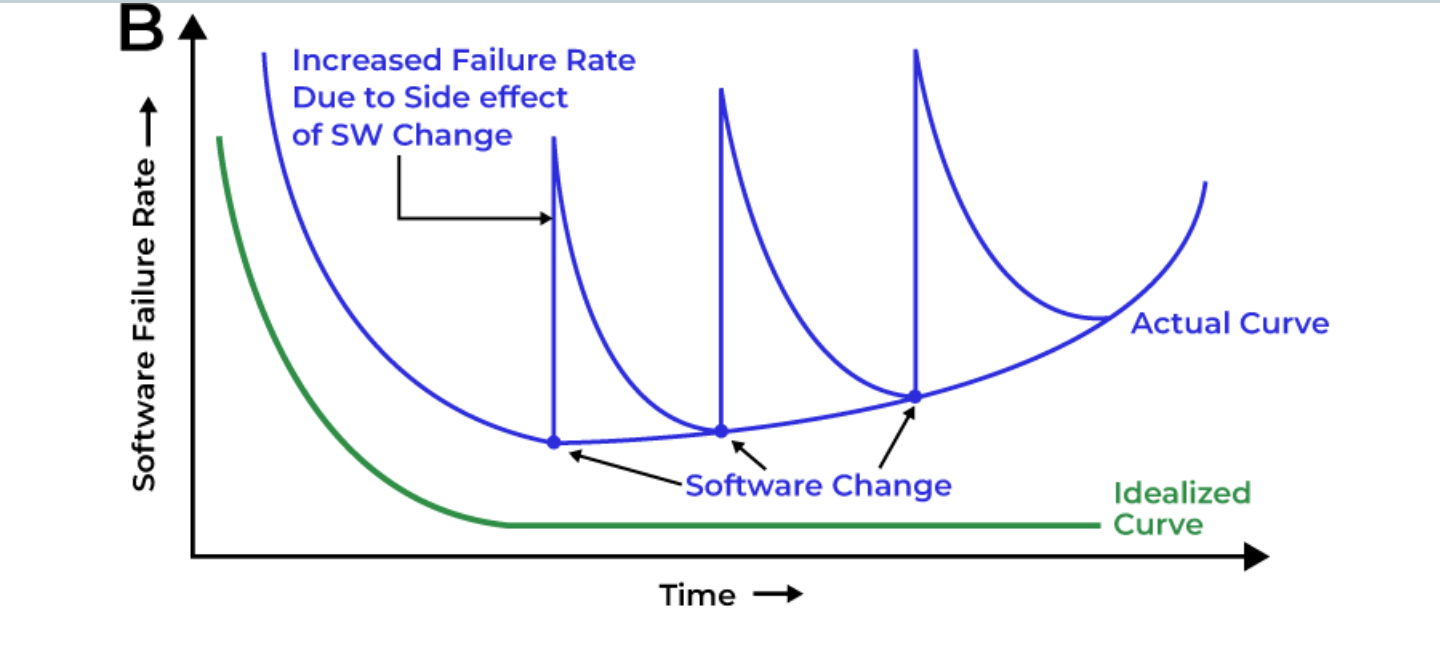
Characteristics Of Software

- ✓ Software is developed or engineered, not manufactured
- ✓ Software does not “wear out”
- ✓ Most software is custom built rather than being assembled from components

FAILURE CURVE FOR HARDWARE



FAILURE CURVE FOR SOFTWARE



The Changing Nature of Software

- Based on the complex growth of software it can be classified into following categories.

- The seven broad categories of software are challenges for software engineers.

Software Application Domains

1. System Software
2. Application Software
3. Engineering & Scientific Software
4. Embedded Software
5. Product-Line Software
6. Web Application
7. Artificial Intelligence software

System Software

It is a collection of programs written to service other program.

The purpose of the system software is to establish a communication with the hardware

Eg: OS (Operating Systems)

Application Software

- It is consists of standalone programs that are developed for specific business need. This software may be supported by database systems.

- It is a program (or) group of programs designed for end users.

  Eg: Email Apps, Spread sheets.

Engineering & Scientific Software

- Scientific and engineering software satisfies the needs of a scientific or engineering user to perform enterprise-specific tasks.

- Such software is written for specific applications using principles, techniques, and formulae particular to that field. Examples are software like MATLAB, AUTOCAD, PSPICE, ORCAD, etc.

Embedded software

- This category consists of program that can reside within a product or system.
- Such software can be used to implement and control features and functions for the end-user and for the system itself.

  Eg: GPS Device, Calculators, Modern Smart watches.

Product-Line software

- A set of application programs that are built from a common set of software modules.
- A software product line (SPL) implies a formal procedure for designing the modules based on predicting how they can be reused to solve a variety of problems.

Eg: Computer graphics, DBMS.

Web Application Software

- It consists of various web pages that can be retrieved by a browser .
- The web pages can be developed using programming languages like JAVA,PERL, CGI,HTML,DHTML.

Eg: Online auctions, Web mails…..

Artificial Intelligence software

- AI software is a computer program capable of high-complexity tasks, like learning, decision-making, and solving problems.
- AI software uses machine learning.
- Machine learning includes algorithm that are developed to tell computer how to respond to something by example
- It uses a structure as close as possible to human brain.

  Eg: Robotics, image & voice recognition.

Legacy Software

- The Legacy systems are the older, Large complex computer based systems which may be using the obsolete technology.
- These systems may include the older hardware, software, process and procedures.
- The Legacy systems are business critical systems and are used for a long period because it is too risky to replace them.

  Eg: Air Traffic Control (ATC) system which may be used from a long period.

Characteristics of Legacy software

- Life Time :Very high
- Business Criticality : High
- Poor Quality
- Convoluted i.e., complex and difficult code
- Poor or nonexistent documents and test cases(IC)

Software Engineering Layered Technology

- Software Engineering is a layered technology.
- Any software can be developed using these layered approaches.
- Various layers on which the technology is based are quality focus layer, process layer, methods layer, tools layer.


Tools
Method
Process
A quality focus

- **Quality focus:**

- A disciplined **quality** management is a backbone of software engineering technology
- Software engineering must rest on an *organizational commitment to quality.*
- Total quality management(TQM),Six Sigma,ISO 9001,ISO 9000-3, Capability Maturity Model Integration(CMMI) & similar approaches encourages a continuous process improvement ***culture.***
- This **culture** ultimately leads to the development of increasingly more mature approaches to software engineering.
- The ***bedrock*** that supports software engineering is a quality focus.

- **Process :**

- The ***foundation*** for software engineering is the process layer.
- Software engineering process is the *glue* that ***holds*** the technology layers together and enables rational and timely development of computer software.
- Process defines a framework for a set of Key Process Areas (KPAs) it must be established for effective delivery of software engineering technology.
- This establishes the context in which technical methods are applied,workproducts(models,documents,data, reports,forms,etc.)are produced,milestones are established, quality is ensured, and change is properly managed.

- Methods:

- Methods provide the technical ***how-to's*** for building software.
- Methods will include *communication,requirements analysis, design modeling , program construction, testing, and support.*
- Software engineering methods *rely on a set of basic principles* that govern each area of the technology and include modeling activities and other descriptive techniques.

- Tools:
- It provide *automated or semi-automated* support for the *process and the methods.*
- When tools are integrated so that information created by one tool can be used by another.
- CASE combines software,hardware, and a software engineering database to create a software engineering environment analogous to CAD/CAE (computer-aided design/engineering) for hardware.

- A Generic View of Software Engineering:

- Engineering is the analysis, design, construction, verification, and management of technical (or social) entities.
- What is the problem to be solved?
- What characteristics of the entity are used to solve the problem?
- How will the entity (and the solution) be realized?
- How will the entity be constructed?
- What approach will be used to uncover errors that were made in the design and construction of the entity?
- How will the entity be supported over the long term, when corrections, adap-tations, and enhancements are requested by users of the entity.

Process Framework

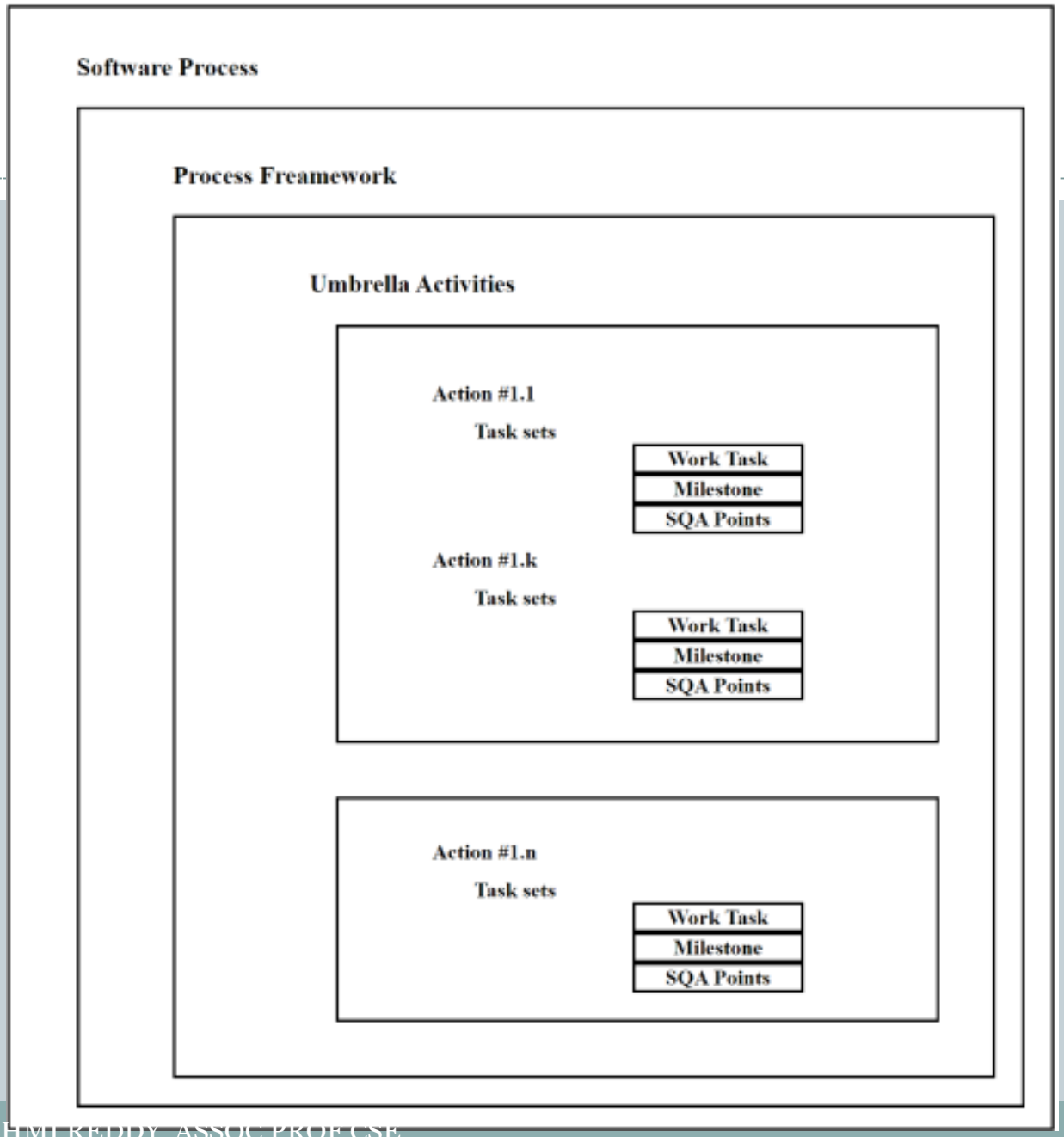
- Software process can be defined as the structured set of activities that are required to develop the software system.

- The fundamental activities are:
  - >Specifications
  - >Design and implementation
  - >validation
  - >Evolution

Common Process Framework

The process framework is required for representing the common process activities.

The software process is characterized by process framework activities, task sets and umbrella activities.


Software Process
Process Framework
Umbrella Activities
Action #1.1
Task sets
Work Task
Milestone
SQA Points
Action #1.k
Task sets
Work Task
Milestone
SQA Points
Action #1.n
Task sets
Work Task
Milestone
SQA Points

Process framework activities

The process framework is required for representing common process activities. Five framework activities are described in a process framework for software engineering.

Those are

Communication

Planning

Modeling

Construction

Deployment

- **Communication:** By communication, customer requirement gathering is done. Communication with consumers and stakeholders to determine the system's objectives and the software's requirements.

- **Planning:** Establish engineering work plan, describes technical risk, lists resources requirements, work produced and defines work schedule.

- **Modeling:** Architectural models and design to better understand the problem and for work towards the best solution. The software model is prepared by:
  - o Analysis of requirements
  - o Design

- **Construction:** Creating code, testing the system, fixing bugs, and confirming that all criteria are met. The software design is mapped into a code by:
  - o Code generation
  - o Testing

- **Deployment:** In this activity, a complete or non-complete product or software is represented to the customers to evaluate and give feedback. On the basis of their feedback, we modify the product for the supply of better products.

Task Sets

- The task set defines the actual work done in order to achieve the software objective.
- The task set is used to adopt the framework activities and project team requirements using:
    - Collection of software engineering work tasks
    - Project milestones
    - Software quality assurance points

Umbrella activities

Activities that occur throughout a software process for better management and tracking of the project.

- **Software project tracking and control** – Compare the progress of the project with the plan and take steps to maintain a planned schedule.
- **Risk management** – Evaluate risks that can affect the outcome and quality of the software product.
- **Software quality assurance (SQA)** – Conduct activities to ensure the quality of the product.
- **Technical reviews** – Assessment of errors and correction done at each stage of activity.
- **Measurement** – All the measurements of the project and product features.
- **Software configuration management (SCM)** – Controlling and tracking changes in the software.
- **Reusability management** – Back up work products for reuse and apply the mechanism to achieve reusable software components.
- **Work product preparation and production** – Project planning and other activities used to create work product are documented.

Capability Maturity Model Integration

- CMMI stands for Capability Maturity Model Integration.
- The Capability Maturity Model Integration(CMMI) is a framework used in software engineering *to assess and improve an organization's software development processes.*
- CMMI is a set of best practices & guidelines for developing high quality software.

There are two representations for CMMI

1. Staged(Maturity Levels)
2. Continuous(Capability Levels)

Staged (Maturity Levels):

Representation is an approach which focuses on the predefined set of process areas.

Organizations use predefined and proven improvement path. This improvement path is described by a set of *components called maturity levels.*

The stage representation is concerned with selecting multiple process areas to improve within maturity level, whether individual processes are performed or incomplete is not primary focus.

Hence the name initial is given to the starting point of staged representation.

The staged representation allows organization to make improvements based on predefine improvement path.

Continuous(Capability Levels)

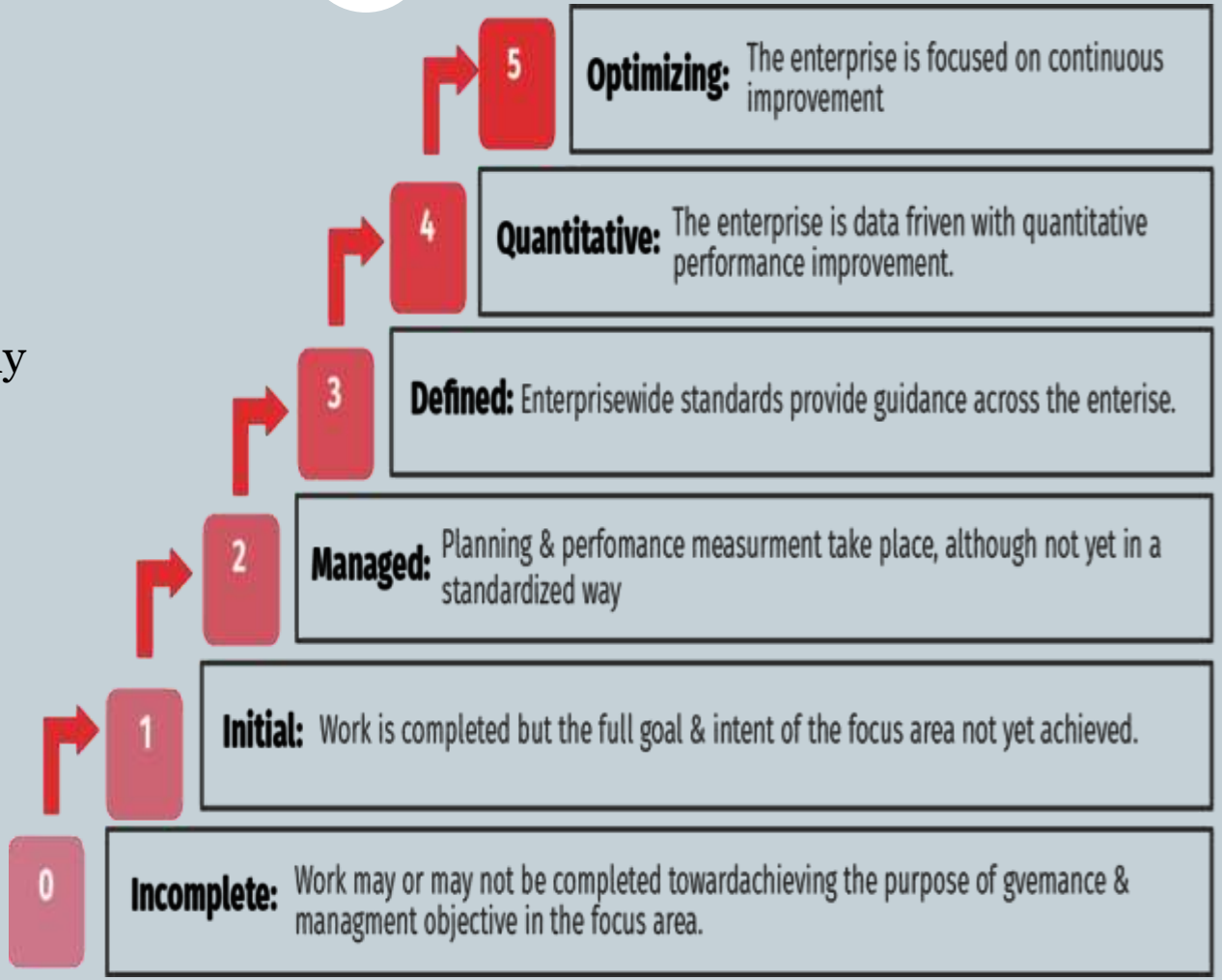
- Continuous representation focuses on specific process area.
- It uses Capability levels to assess and improve the development of a particular process area.
- Hence there is a *incomplete* which is given to the starting point of continuous presentation.
- *Continuous representation provides flexibility to organizations to choose which processes need the improvement as well as how much improvement is needed.*

Continuous(Capability Levels)

- The basic building block of CMMI is process area.
- The process areas represent the phases of software process development such as project planning, requirement management, softwareanalysis, configuration management and so on.
- The process area defines the goal and practices of a process.
- Following is a process capability levels for the processes.

Capability Levels

- Level 0: Incomplete
- Level 1: Performed
- Level 2: Managed
- Level 3: Defined
- Level 4: Quantitatively managed
- Level 5: Optimizing



Capability Levels

- **Level 0: Incomplete:**
  - At the level, the process areas is not performed.
  - This is a level at which the goals and objectives of CMMI level1 are not defined completely.
- **Level 1: intial(Performed):**
  - The goals and objectives of process are satisfied.
  - The work tasks that are required to perform are for development of the product are defined.

- **Level 2: Managed(REPAETABLE)**

- The criteria for level 1 are satisfied at this stage.
- The work associated with the process area is conducted as per the business or organization policy.
- All the required resources are easily available at this stage. Stakeholders take active participation in project evelopment.
- The work tasks and work products are monitored, reviewed and evaluated.

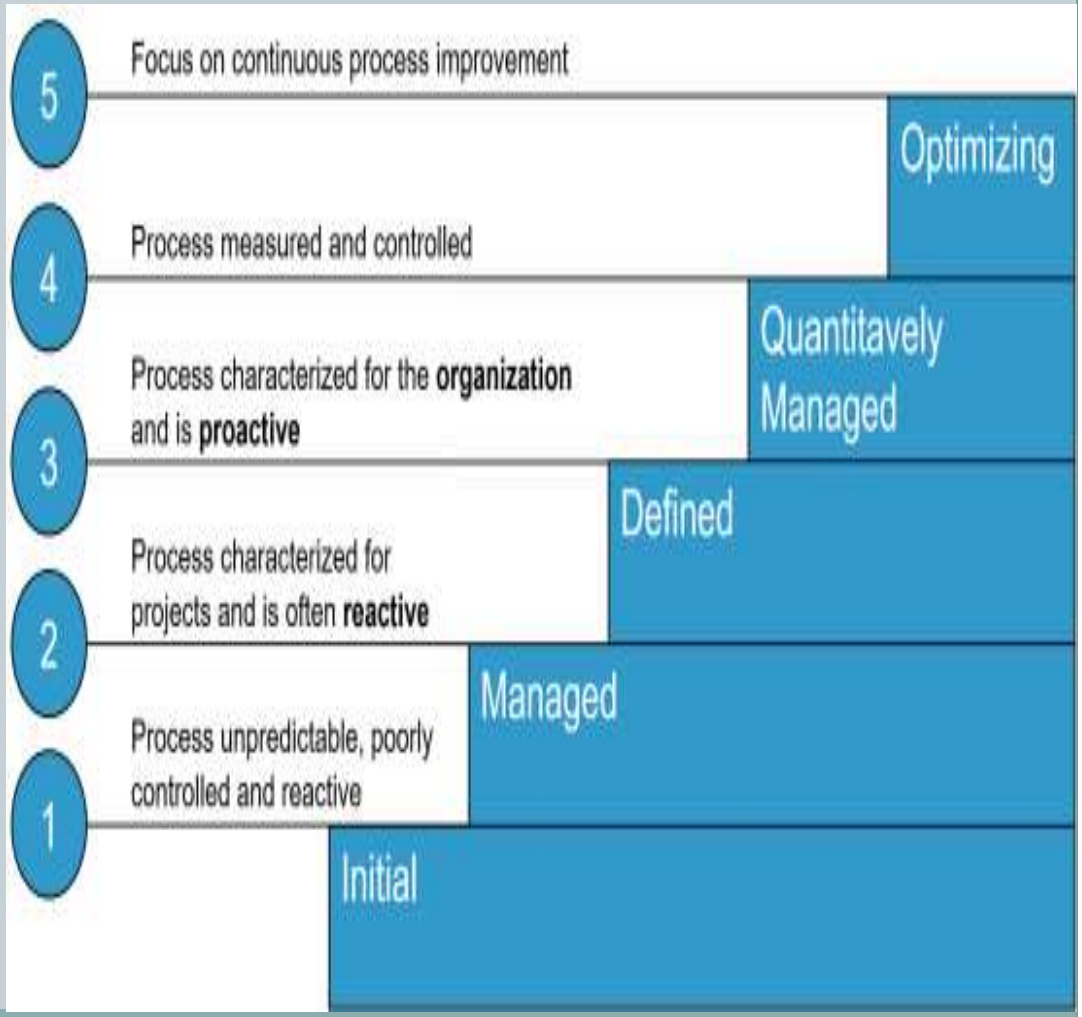
- **Level 3: Defined:**

- The criteria for level 2 are satisfied at this stage.
- The process is standardized, documented and followed.
- All the projects use documented and approved version of software process which is useful in developing and supporting software.

- **Level 4: Quantitatively Managed:**
  - The criteria for level 3 are satisfied at this stage.
  - Both the software process and product are quantitatively understood and controlled using detailed measures.
- **Level 5: Optimizing:**
  - The Criteria for level 4 are satisfied at this stage
  - Establish mechanisms to plan and implement change.
  - Innovative ideas and technologies can be tested.
  - The modifications in process area can be made to meet changing customer needs and continually improve the process area under consideration.

Staged(Maturity Levels)

- Level 1: Initial
- Level 2: Managed
- Level 3: Defined
- Level 4: Quantitatively managed
- Level 5: Optimizing



- **Level 1:** ***Initial*** ***(process is unpredictable and*** *poorly controlled)*

- In this level the processes are *very basic, poorly controlled and they are reactive to the situation.*
- In this level processes are immature & not well defined.
- No engineering management, everything is *adhoc basis.*
- Software process is unpredictable with respect to *time and cost.*
- It depends on the current staff, as staff changes so does the process

- **Level 2: Managed (Repeatable)**

- Basic project management policies are done in this level.
- Projects are planned implemented, measured and monitored at this level.
- *Planning and managing of new projects based on the experience with similar projects*
- Realistic plans based on the performance based on the previous projects

- **Level 3: Defined (Process standardization)**

- The main focus of this level is defining standard of the project.
- Documentations are done in this level.
- Peer reviews, Intergroup communications, *training programs* are implemented to ensure that the staff have *skills and knowledge* required done in this level.
- Risk management

- **Level 4 : Managed(Quantitatively Managed)**

- At This level project is in controlled position, processes offer a *high quality results with minimum risk involved.*
- In this level software *quality management, qualitative management* are done.
- Software process is predictable with respect to *time and cost.*

- **Level 5: Optimized (Continuous process improvement)**

➢ Processes are steady and flexible at this level.

➢ Organizations analysis defects to determine their *causes and goals* is to preventing the occurrence of defects optimize resources at this stage for continuous improvement.

Difference between Capability and Maturity levels

Maturity Level

- These levels represent a staged approach to process improvement.
- There are 5 maturity levels in CMMI ranging from LEVEL 1 to LEVEL 5
- Each maturity level builds upon the previous one and organizations progress through them sequentially.
- Maturity levels provide a roadmap for improving processes and achieving a higher level of organizational maturity.

Capability Level

- These levels represent a continuous approach to process improvement.
- There are 6 Capability levels in CMMI ranging from LEVEL0 to LEVEL5.
- Organizations can focus on specific process areas or capabilities independently without the need to progress through a predefined sequence.
- Capability levels allow organizations to target and improve specific areas of their processes that are most critical for their business needs.

Process Patterns

- Process is defined as a series of activities in which one or more inputs are used to produce one or more outputs.
- The process pattern is a template which appears as general solution to a common problem.
- Software process is defined as collection of patterns. And the pattern gives a template and this template has the important characteristics and features of the process.

Process Pattern

- Process Template
  - Pattern Name
  - Intent
  - Types
    - Task Pattern
    - Stage Pattern
    - Phase Pattern
- Initial Context
- Problem
- Solution
- Resulting context
- Known uses/Related problems.

Process Patterns

- The description of process pattern is as follows:

➢ Process template

- Pattern Name:
  - The Pattern name should be a meaningful name given to the pattern. From pattern name one can guess its functionality.
- Intent:
  - The objective or the purpose of the pattern should be described here.

Process Patterns

- Type:
  - The type of pattern should be specified here
    - Task Pattern: It represents the software engineering action or a work task which is a part of process.

      Ex: Formal Technical review is a task pattern.
    - Stage Pattern: It defines the process framework activity. A framework activity has multiple work tasks, hence stage pattern consists of multiple task patterns.

      Ex: Coding phase is a stage pattern.
    - Phase Pattern: It defines the sequence of framework activities.

      Ex: The phase pattern can spiral model or prototype model.

Process Patterns

- Initial context:
  - Conditions under which the pattern applies are described by initial context. Prior to the initiation of the pattern :
    - What organizational or term-related activities have already occurred?
    - Entry state for the process?
    - Software engineering information or project information already exists?

For example, the Planning pattern requires that :

•Collaborative communication has been established between customers and software engineers.
•Successful completion of a number of task patterns for the communication pattern has occurred.
•The project constraints, basic requirements, and the project scope are known.

- **Problem:**

  Under this section the problem is mentioned for which the pattern is to be described.

  Ex: Insufficient requirements. That means customer are not sure about what they want exactly. They could not specify the requirements in proper manner.

- **Solution:**

  Every problem for which pattern has to be described should be accompanied with some solution.

  The problem of insufficient requirements ahs solution. i.e Establish effective communication with the customer. Ask questions in order to obtain meaningful requirements. This solution will help the software developer to get useful information before the actual work starts.

- Resulting Context:

It describes the results after successful implementation of pattern. The resulting context should have following type of information on successful completion of pattern.

The team related or organizational activities that must have occurred.

Exit state for the process.

The software engineering information or project information that has been developed.

- Known uses:

If any problems are identified in the pattern then those problems are the related problems of process pattern.

If any problems occurred then we have to rectify those problems.

Ex: Spiral model is useful for the large scale projects in which work products must be examined in several iterations.

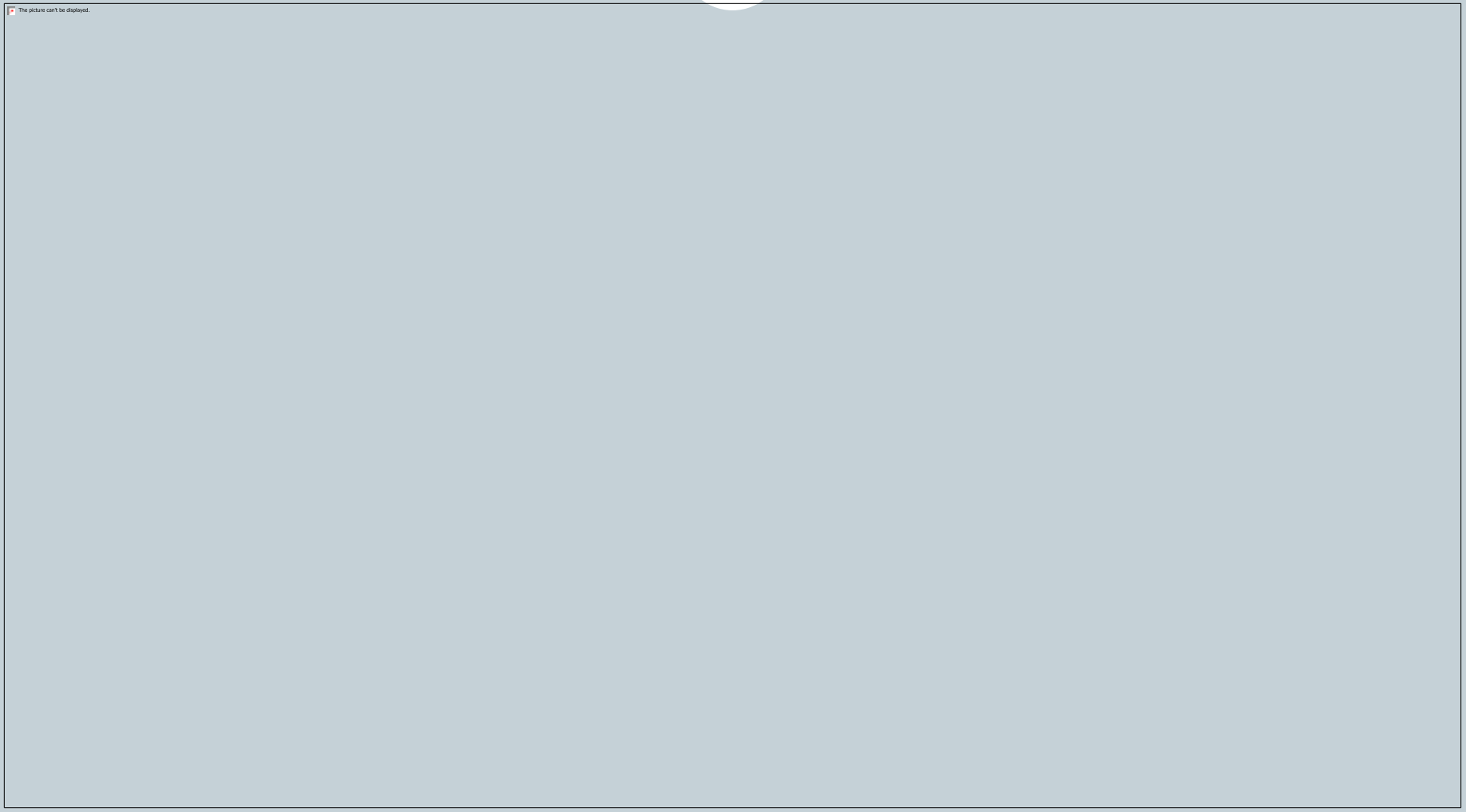
PROCESS ASSESSMENT

- Normally process is suffered by following problems:
  - *The software has to be delivered on time.*
  - *The software should satisfy customer needs and requirements.*
  - *The software should posses the long term quality characteristics*.

A software process assessment is a *disciplined examination* of the software processes used by an *organization* based on a *process model.*

The assessment includes the *identification and characterization* of current practice, identifying areas of *strengths and weaknesses*, the ability of current practices to *control or avoid significant causes of poor quality, cost and schedule*.

Software process Assessment



- Following approaches are used for software assessment:
  - Standard CMMI assessment method for process improvement.(SCAMPI)
  - CMM-based appraisal for internal process improvement(CBA IPI)
  - SPICE(ISO/IEC15504)
  - ISO 9001(2000 for Software)

Standard CMMI assessment method for process improvement

- It is five step process assessment model. These five steps are
  - *Initiating*
  - *Diagnosing*
  - *Establishing*
  - *Acting*
  - *Learning*

This model makes the use of SEI CMM as the base model.

CMM-based appraisal for internal process improvement(CBA IPI)

- This method provides the ***diagnostic technique*** *for assessing the relative maturity of a software organization*. it uses the SEI CMM as the basis for the assessment.
- It gives the organization an insight into its software development capability by assessing *the strength and weakness of the current process.*

Software Process Improvement and Capability Determination (SPICE)

- SPICE typically refers to a systematic approach for evaluating and enhancing software development processes.
- It involves two main goals:
  - Software Process Improvement (SPI): Enhancing the efficiency, effectiveness, and quality of software development processes.
  - Capability Determination: Assessing an organization's ability to perform specific software processes to achieve desired outcomes.
- Frameworks like **SPICE (ISO/IEC 15504)** and **CMMI (Capability Maturity Model Integration)** are commonly used
- This standard helps in doing an objective evaluation on efficiency of any process.

ISO 9001:2000

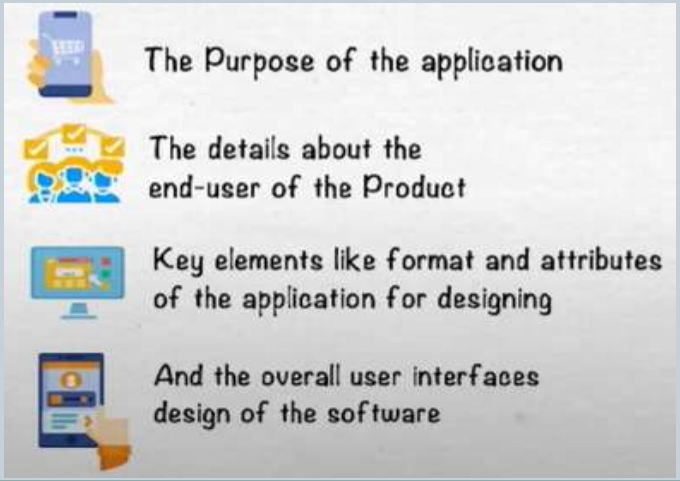
- This is a popularly used standard for improving the overall quality of the organization.
- It is applied to organizations aiming to improve the overall quality of product, process and services. They evaluate the ability of an organization to consistently provide products that meet customer requirements. Here the main aim of the organization should be ***enhancing customer satisfaction.***

SOFTWARE DEVELOPMENT LIFE CYCLE(SDLC)

- **SDLC** is a systematic process for building software that ensures the *quality and correctness* of the software built.
- SDLC process aims to produce high-quality software that meets *customer expectations.*
- The system development should be complete in the *pre-defined time frame and cost.*
- SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase.
- There are 7 phases in SDLC they are

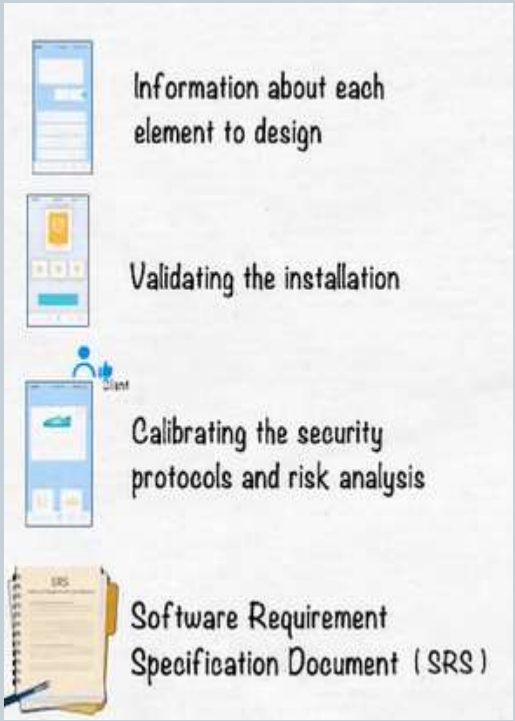
1. Requirement gathering
2. Analysis
3. Design
4. Implementation
5. Testing
6. Deployment
7. Maintenance

- # Requirements Gathering:
- The requirement is the first stage in the SDLC process. It is conducted by the *senior team members with inputs from all the stakeholders and domain experts in the industry.* Planning for the quality assurance requirements and recognization of the risks involved is also done at this stage.
- This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project.



- Analysis:

What is needed?
What is not needed?
Analyze each requirements.
Risks are predicted
Documentation is prepared as software requirements specifications(SRS)



- **Design:**
- In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.
- This design phase serves as input for the next phase of the model.
- There are two kinds of design documents developed in this phase:
- **High-Level Design (HLD):** *It gives the architecture of the software products.*
  - Brief description and name of each module
  - An outline about the functionality of every module
  - Interface relationship and dependencies between modules
  - Database tables identified along with their key elements
  - Complete architecture diagrams along with technology details
- **Low-Level Design (LLD):***It describes how each and every feature in the software product should work.*
  - Functional logic of the modules
  - Database tables, which include type and size
  - Complete detail of the interface
  - Addresses all types of dependency issues
  - Listing of error messages
  - Complete input and outputs for every module

- Development:

Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided *into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process*.

- In this phase, Developer needs to follow certain predefined coding guidelines. They also need to use programming tools like compiler, interpreters, debugger to generate and implement the code.

- Testing:

Testing begins when coding is done. The purpose of testing is to uncover errors, fix the bugs and meet the customer requirements.

- Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.
- During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

- Deployment:

It is the final step in the software development life cycle and delivers the final product to the customer in a live production environment. After the product deploys, the product is ready for customers to use.

- Maintenance:

Once the software is delivered to the user and the users start using the software, then actual problems will come. Those problems are solved and fixed in this phase.

- Once the system is deployed, and customers start using the developed system, following 3 activities occur
- Bug fixing – bugs are reported because of some scenarios which are not tested at all
- Upgrade – Upgrading the application to the newer versions of the Software
- Enhancement – Adding some new features into the existing software
- The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

PROCESS MODELS

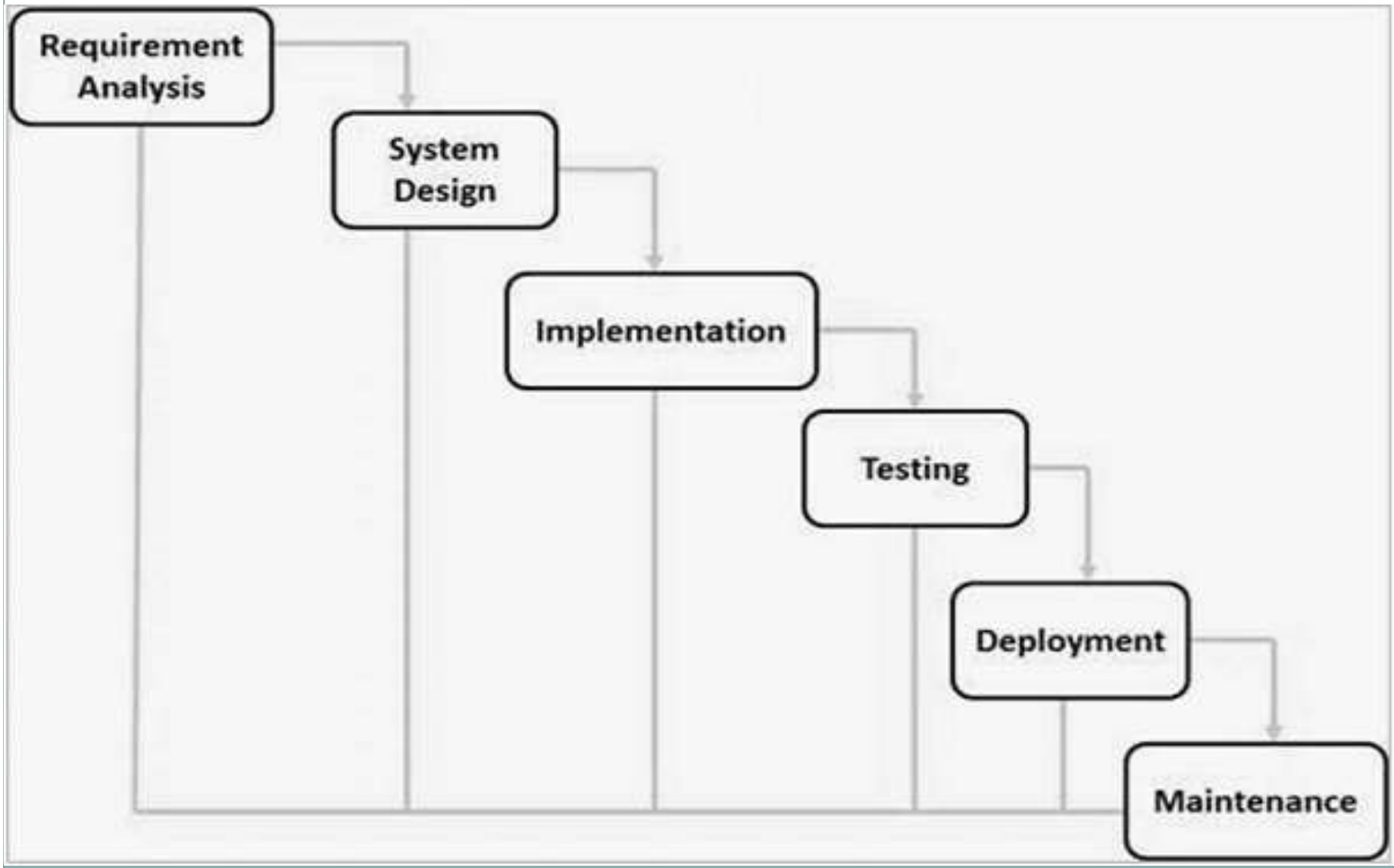
- A software process model is an ***abstraction*** of the software development process. The models specify the stages and order of a process.
- This is the Representation of the **order of activities** of the process and the **sequence** in which they are performed.
- **A model will define the following:**
- The tasks to be performed
- The input and output of each task
- The pre and post-conditions for each task
- The flow and sequence of each task

List of Process Models

- Water-fall model
- Incremental process models
  - The Incremental model
  - The RAD model
- Evolutionary process models
  - Proto typing
  - The Spiral model
  - The Concurrent Development model.
- The Unified Process

Waterfall Model

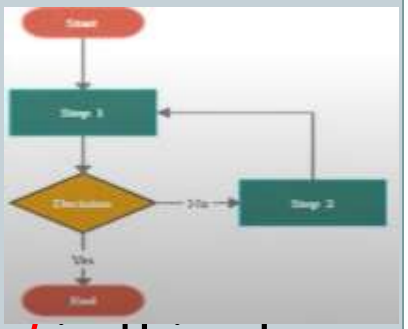
- The waterfall model is also called '**Linear sequential model**' or '**Classic life cycle model**'.
- It is the oldest model. This model suggests a systematic **sequential approach** to software development.
- The software development starts with **requirements gathering** phase. Then progresses through **analysis**, **design**, **coding**, **testing** , **deployment** and **maintenance**.


Requirement Analysis
System Design
Implementation
Testing
Deployment
Maintenance

Waterfall Model

- 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.



- Design:

The requirement specification 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*. *Each unit is developed and tested for its functionality, which is referred to as unit testing.*

- Integrating and testing:

All the units developed in the implementation phase are integrated 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.*

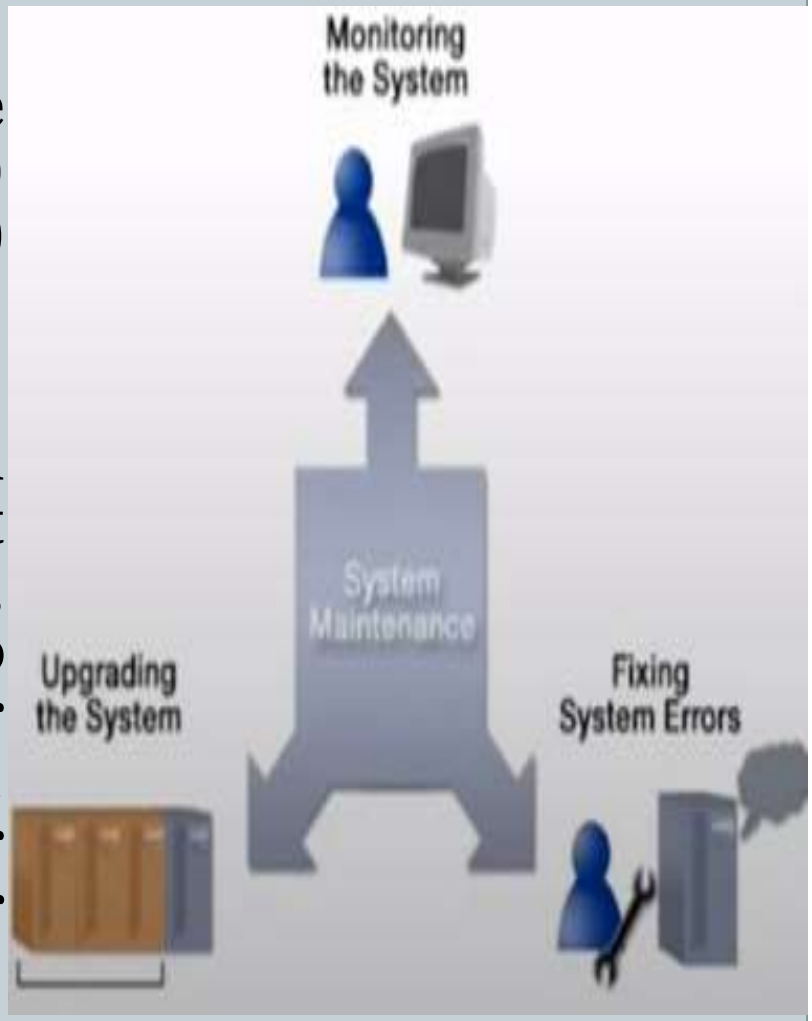
Waterfall Model

- Deployment:

After completion of testing phase the product is deployed (or) delivered to the customer (or) released into 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.



Waterfall Model

- **When to use the waterfall model**
- The waterfall model is appropriate for developing small-scale software which should have stable and, clear requirements. And also, requirements should not be constantly changing.

Advantages and Disadvantages OF Waterfall Model

- Advantages:
  - Simple and easy to understand and use.
  - Works well for smaller projects where requirements are very well understood.
  - Clearly defined stages.
  - Easy to arrange tasks.
  - Process and results are well documented.
- Disadvantages:
  - High amount of risk and uncertainty.
  - Not a good model for complex and object-oriented projects.
  - Cannot accommodate changing requirements.
  - It is difficult to measure progress within stages.

Incremental Model

- In this model the requirements are divided into multiple modules.
- In each module we have all the phases that are analysis, design, code & testing.
- This model used for software with less features and used in day to day applications.
- Ex : Banking
- For this model less manpower is required.
- In this we have 2 models

  1.Incremental Model

  2.RAD(Rapid Application Development) model

1. Incremental Model

- Incremental Model is a process of software development where *requirements are broken down into multiple standalone modules* of software development cycle.
- Incremental development is done in steps from Communication, Planning, Modeling (analysis, design), Construction (code, test) and Deployment (delivery, feedback).

- The incremental model combines elements of linear and parallel process flows.
- When an incremental model is used, the first increment is often a **core product.**
- *Core product* basic requirements are addressed but many supplementary features (some known, others unknown) remain undelivered.
- The *core product* is used by the customer (or undergoes detailed evaluation). As a result of use and/or evaluation, a plan is developed for the *next increment*

- The plan addresses the *modification of the core product to better meet the needs of the customer* and the delivery of additional features and functionality. This process is repeated following the *delivery of each increment*, until the complete product is produced.
- The incremental process model focuses on the delivery of an operational product with each increment. Early increments are *stripped-down versions of the final product*, but they do provide capability that serves the user and also provide a platform for evaluation by the user.

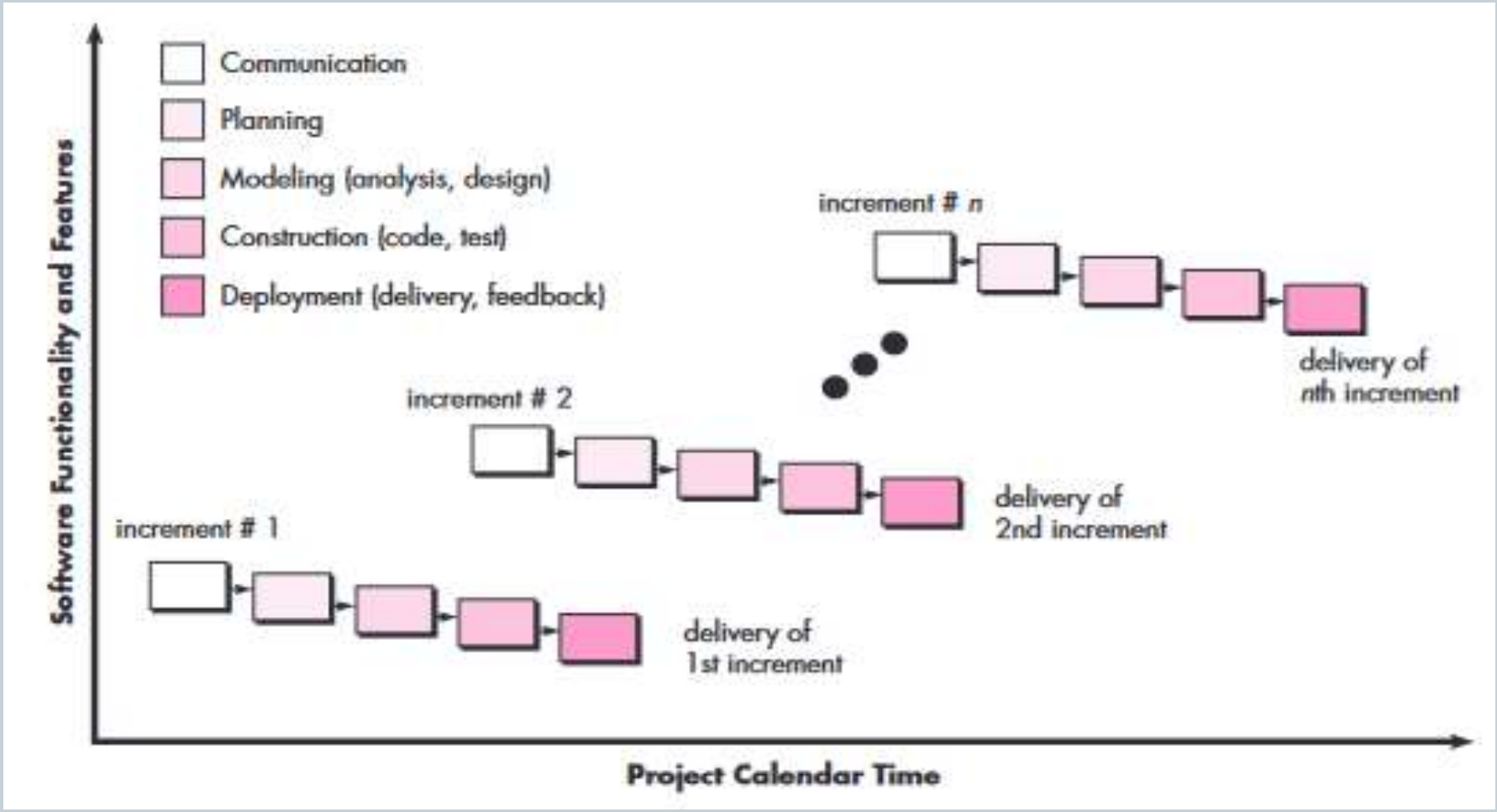
- Incremental development is *particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.*
- *Early increments* can be implemented with fewer people. If the core *product is well received,* then additional staff (if required) can be added to implement the *next increment.*
- In addition, increments can be planned to manage technical risks

Example

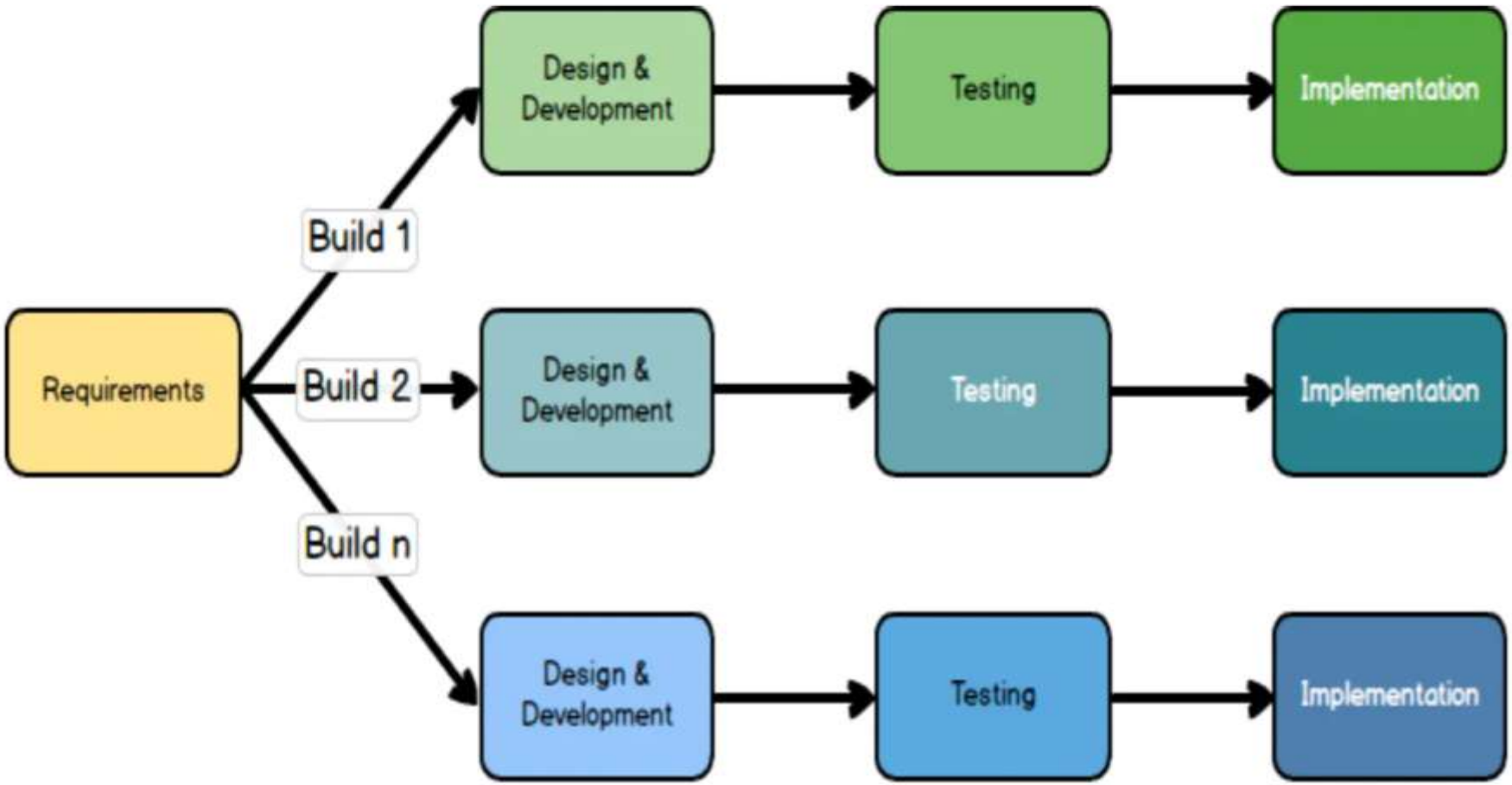
- word-processing software developed using the incremental paradigm might deliver *basic file management, editing, and document production functions in the first increment.*
- *More sophisticated editing and document production capabilities in the second increment.*
- *Spelling and grammar checking in the third increment.*
- *Advanced page layout capability in the fourth increment.*

It should be noted that the process flow for any increment can incorporate the prototyping paradigm.

Incremental model



Incremental model



When to use Incremental models?

- Requirements of the system are clearly understood
- When demand for an early release of a product arises
- When software engineering team are not very well skilled or trained
- When high-risk features and goals are involved
- Such methodology is more in use for web application and product based companies

Advantages and Disadvantages

Advantages of Incremental Model:

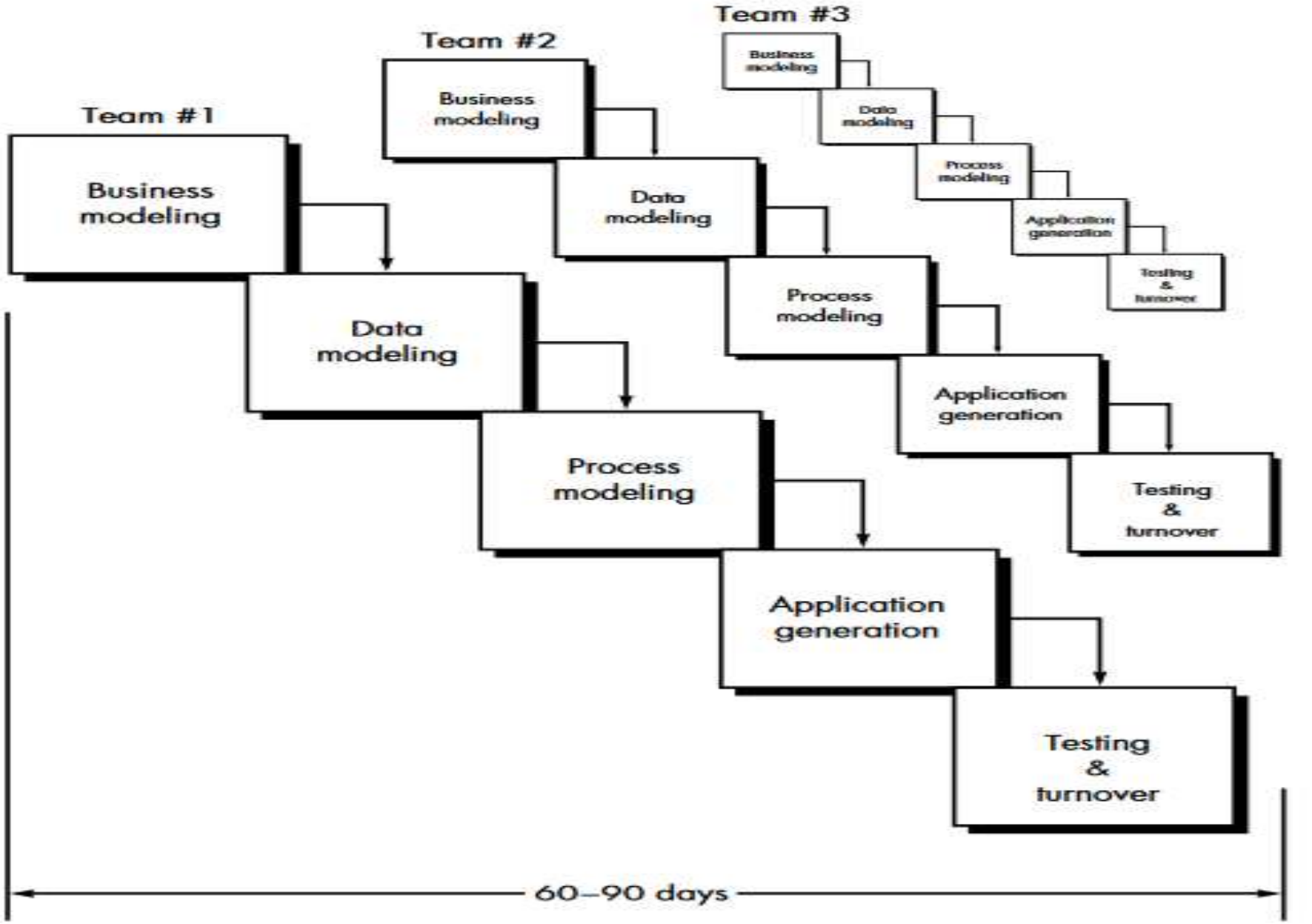
- **Early Delivery:** Allows for the early release of partial functionality to users.
- **Feedback Integration:** Incorporates user feedback iteratively, leading to improved product quality.
- **Risk Reduction:** Smaller, manageable increments reduce overall project risk.
- **Easier Testing:** Testing and debugging are more manageable due to smaller increments.

Disadvantages of Incremental Model:

- **Complex Management:** Requires careful coordination and tracking of multiple increments.
- **Potentially Higher Costs:** Initial investment in a flexible architecture can be higher.
- **Incomplete Features:** Some critical features may not be available in early increments.
- **Dependency Challenges:** Dependencies between increments can cause issues if not managed well.
- **Documentation Overhead:** Managing documentation for each increment can become cumbersome.

RAD Model

- Rapid application development (RAD) is an incremental software development process model that *emphasizes an extremely short development cycle.*
- The RAD model is a *"high-speed" adaptation* of the linear sequential model in which rapid development is achieved by using component-based construction. If requirements are *well under-stood and project scope is constrained*, the RAD process enables a development team
- To create a "fully functional system" within very short time periods (e.g., 60 to 90 days)
- Multiple teams work on developing the software system using RAD model parallelly.


Team #1
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
Team #2
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
Team #3
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
60–90 days

RAD Model

Business modeling: The information flow among business functions is modeled in a way that answers the following questions:

- What information drives the business process?
- What information is generated?
- Who generates it?
- Where does the information go?
- Who processes it?

- Data modeling:
- The information flow defined as part of the business modeling phase is refined into a set of data objects that are needed to support the business.
- The characteristics *(called attributes)* of *each object are identified and the relationships between these objects defined.*

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

- Application generation:
- RAD assumes the use of *fourth generation techniques* Instead of building software using conventional third generation programming languages the RAD process works to reuse existing program components (when possible) or create reusable components (when necessary).
- In all cases, automated tools are used to facilitate construction of the software.

- Testing and turnover:
- Since the RAD process emphasizes reuse, many of the program components have already been tested.
- This reduces overall testing time.
- However, new components 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.
- 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.
- 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.

Disadvantage of RAD Model

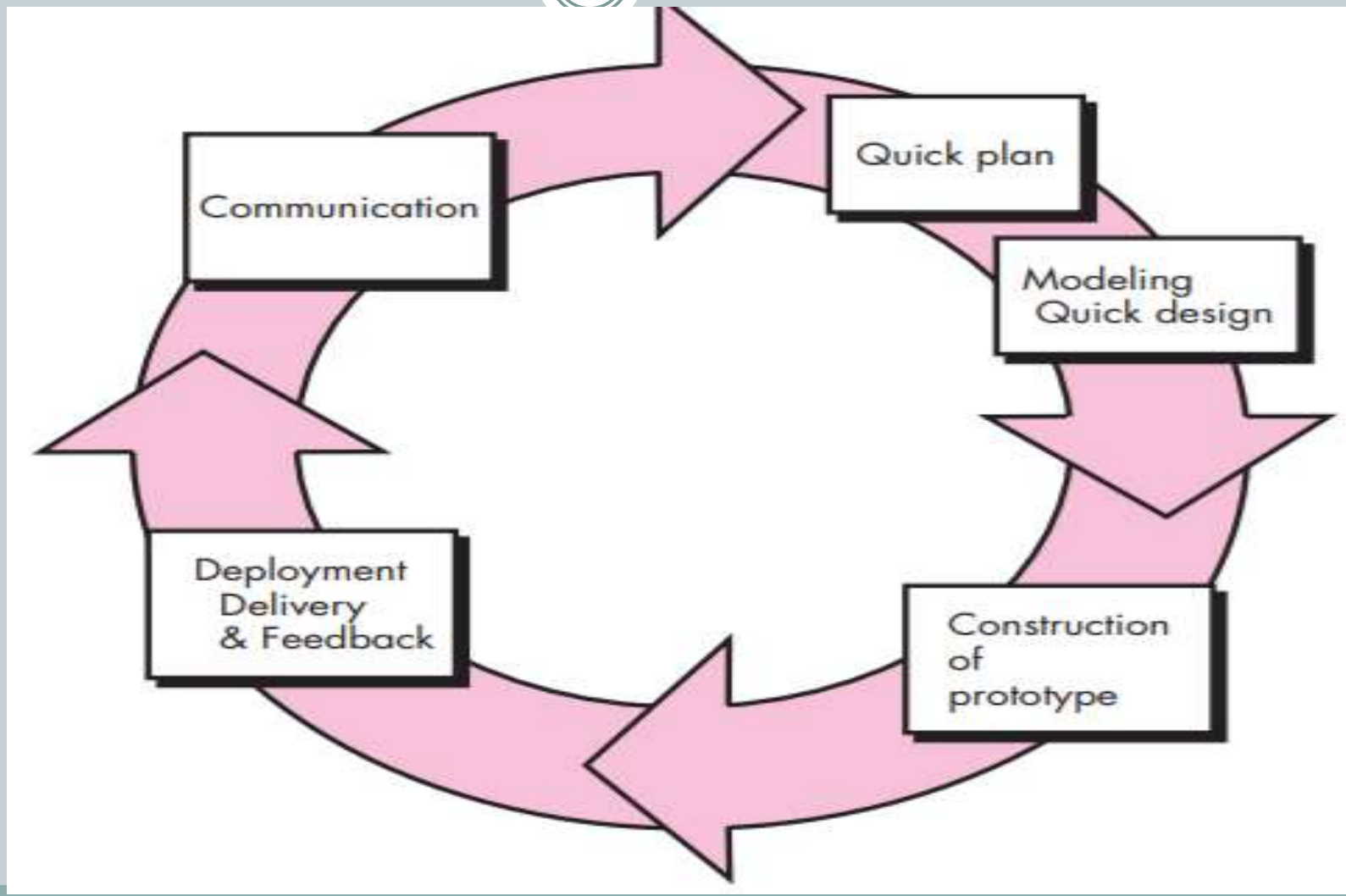
- It required highly skilled designers.
- For smaller projects, we cannot use the RAD model.
- On the high technical risk, it's not suitable.
- Required user involvement.

EVOLUTIONARY PROCESS MODEL

- Evolutionary model is a combination of *Iterative and Incremental model* of software development life cycle.
- Evolutionary model is also referred to as the successive versions model.
- **Following are the evolutionary process models.**
  1. The prototyping model
  2. The spiral model
  3. Iterative model
  4. Concurrent development model

PROTOTYPE MODEL

- Prototyping Model is one of the most popularly used (SDLC models).
- This model is used when the customer defines a set of *general objectives* for software, but does not identify detailed ***requirements for functions and features.***
- In this model, a ***prototype*** of the end product is first developed, tested, and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.


Communication
Quick plan
Modeling
Quick design
Construction
of
prototype
Deployment
Delivery
& Feedback

Phases of prototype model

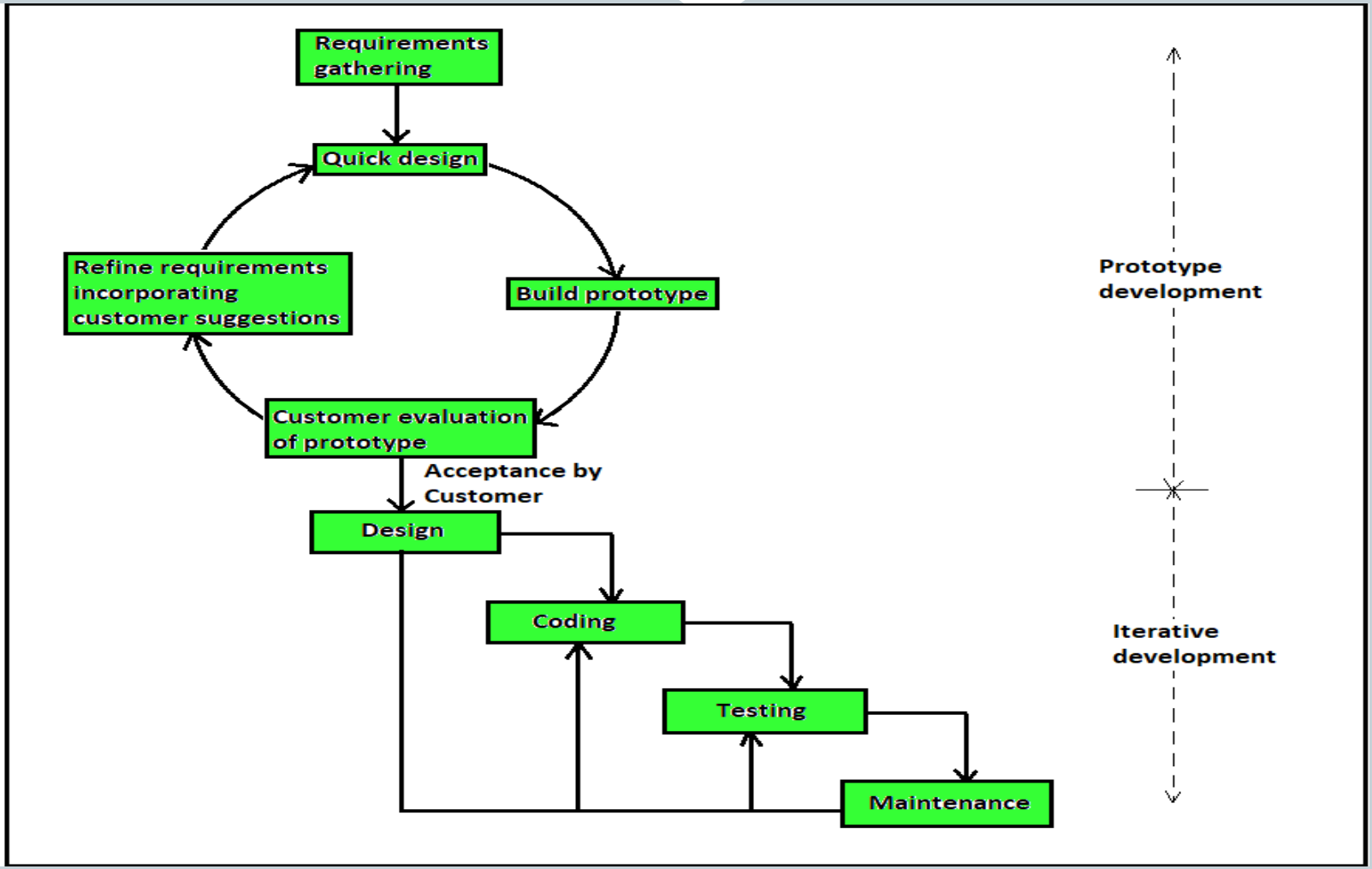
- **Initial Communication** – In this phase, business analysts and other individuals responsible for collecting the *requirements and discussing the need for the product, meet the stakeholders or clients.*

- **Quick Plan** – When we have the requirements known to the developer, then the developer can implement the *quick design.* The quick design includes implementing important aspects, such as the *input and output format of the software.* The main focus of this project is on the *things visible to the user rather than the detailed plan.* With the help of this, we can also create the prototype of the project..

- **Modeling Quick Design –** In this phase, we can get a clear idea about the software development because the software is now in the build state. Also, it allows the developer to understand the project's basic requirements.
- **Construction of prototype**

  The customer evaluates the prototype of the project.
- **Development of the Prototype –** If the client is unsatisfied with the project, then the developer can recreate the project according to the client's satisfaction. This process repeats until the client is fully satisfied. After the client is happy, the project will be developed based on the final prototype.

Work flow of prototype model



Types of Prototyping Models

- There are four types of Prototyping Models:
  - Rapid Throwaway Prototyping
  - Evolutionary Prototyping
  - Incremental Prototyping
  - Extreme Prototyping

Rapid Throwaway Prototyping:

- Rapid throwaway is based on the preliminary requirement. It is quickly developed to show how the requirement will look visually.
- The *customer's feedback helps drives changes* to the requirement, and the prototype is again created until the requirement is baselined.
- In this method, a *developed prototype will be discarded and will not be a part of the ultimately accepted prototype*. This technique is useful for exploring ideas and getting instant feedback for customer requirements

- **Evolutionary Prototyping:**

- In this method, the prototype developed initially is *incrementally* refined based on customer feedback till it finally gets accepted.
- In comparison to Rapid Throwaway Prototyping, it offers a better approach that saves time as well as effort.
- This is because developing a prototype from scratch for every iteration of the process can sometimes be *very frustrating for the developers.*

- **Incremental Prototyping:**

- In incremental Prototyping, the final product is decimated into *different small prototypes* and developed individually. Eventually, the different prototypes are *merged into a single product*. This method is helpful to reduce the feedback time between the user and the application development team.

- there might be the possibility that the pieces just do not fit together due to some lack of ness in the development phase – this can only be fixed by careful and complete plotting of the entire system before prototyping starts.

- **Extreme Prototyping:**

- This method is mainly used for web development. It consists of three sequential independent phases:
- In 1st phase, a basic prototype with all the existing static pages is presented in HTML format.
- In the 2nd phase, ***Functional screens are made with a simulated data process*** using a prototype services layer.
- This is the final step where all the services are implemented and associated with the final prototype.
- This Extreme Prototyping method makes the project cycling and delivery robust and fast and keeps the entire developer team focused and centralized on product deliveries rather than discovering all possible needs and specifications and adding necessitated features.

Advantages and disadvantages

Advantages of Prototyping Model

- Prototype model need not know the detailed input, output, processes, adaptability of operating system and full machine interaction.
- 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.
- Errors are detected much earlier.
- 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 a*nd 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 thrown away prototype when the users are confused with it.

Spiral model

- Spiral Model Originally proposed by Barry Boehm the spiral model is an *evolutionary* software process model that couples the iterative nature of prototyping with the controlled and *systematic aspects of the waterfall model.*
- It provides the potential for *rapid development* of increasingly more complete versions of the software.
- The spiral development model is a *risk-driven* process model generator that is used to guide multi-stakeholder concurrent engineering of software intensive systems.

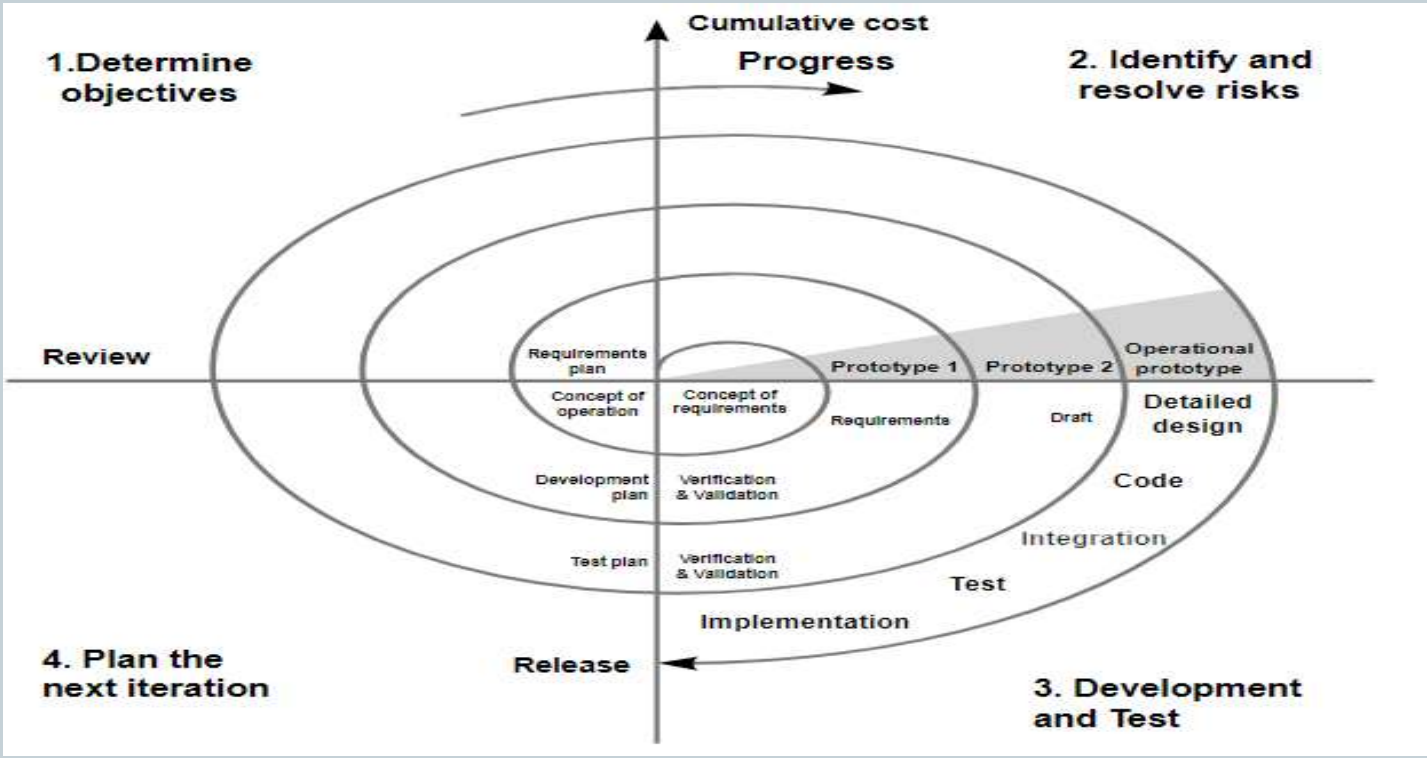
- It has two main distinguishing features.
  - One is a ***cyclic approach*** for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk.
  - The other is a set of ***anchor point milestones*** for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions.

Spiral model

- As this evolutionary process begins, the software team performs activities that are implied by a circuit around the spiral in a *clockwise direction, beginning at the center.*
- This model gives efficient development of incremental versions of software. The dramatic representation of this model appears like a spiral with many loops.
- The exact no. of loops in the spiral is not fixed.
- Each loop of the spiral represents a phase of the software process.
- The first circuit around the spiral might result in the development of a product specification subsequent passes around the spiral might be used to develop a prototype and then progressively more sophisticated versions of the software

Spiral Model

- Each phase in this model is split into four sector (or quadrants) as shown in figure.
- The following activities are carried out during each phase of a spiral model.



Phases of spiral Model

- **Spiral Model Phases**
- It has four stages or phases: The planning of objectives, risk analysis, engineering or development, and finally review. A project passes through all these stages repeatedly and the phases are known as a Spiral in the model.
- **Determine objectives and find alternate solutions –** This phase includes requirement gathering and analysis. Based on the requirements, objectives are defined and different alternate solutions are proposed.
- **Risk Analysis and resolving –** In this quadrant, all the proposed solutions are analyzed and any potential risk is identified, analyzed, and resolved.
- **Develop and test:** This phase includes the actual implementation of the different features. All the implemented features are then verified with thorough testing.
- **Review and planning of the next phase –** In this phase, the software is evaluated by the customer. It also includes risk identification and monitoring like cost overrun or schedule slippage and after that planning of the next phase is started.

Advantages and Disadvantages

Spiral Model Advantages

- The spiral model is perfect for projects that are **large and complex** in nature as continuous prototyping and evaluation help in mitigating any risk.
- Because of its **risk handling ability**, the model is best suited for projects which are very critical like software related to the health domain, space exploration, etc.
- This model supports the client feedback and **implementation of change requests** (CRs) which is not possible in conventional models like a waterfall.
- Since customer gets to see a prototype in each phase, so there are higher chances of customer satisfaction.

Spiral Model Disadvantages

- Because of the prototype development and risk analysis in each phase, it is very **expensive and time taking**.
- It is **not suitable for a simpler and smaller** project because of multiple phases.
- It requires **more documentation** as compared to other models.
- Project **deadlines can be missed** since the number of phases is unknown in the beginning and frequent prototyping and risk analysis can make things worse.

Iterative Model

- This Model, we can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.
- The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.

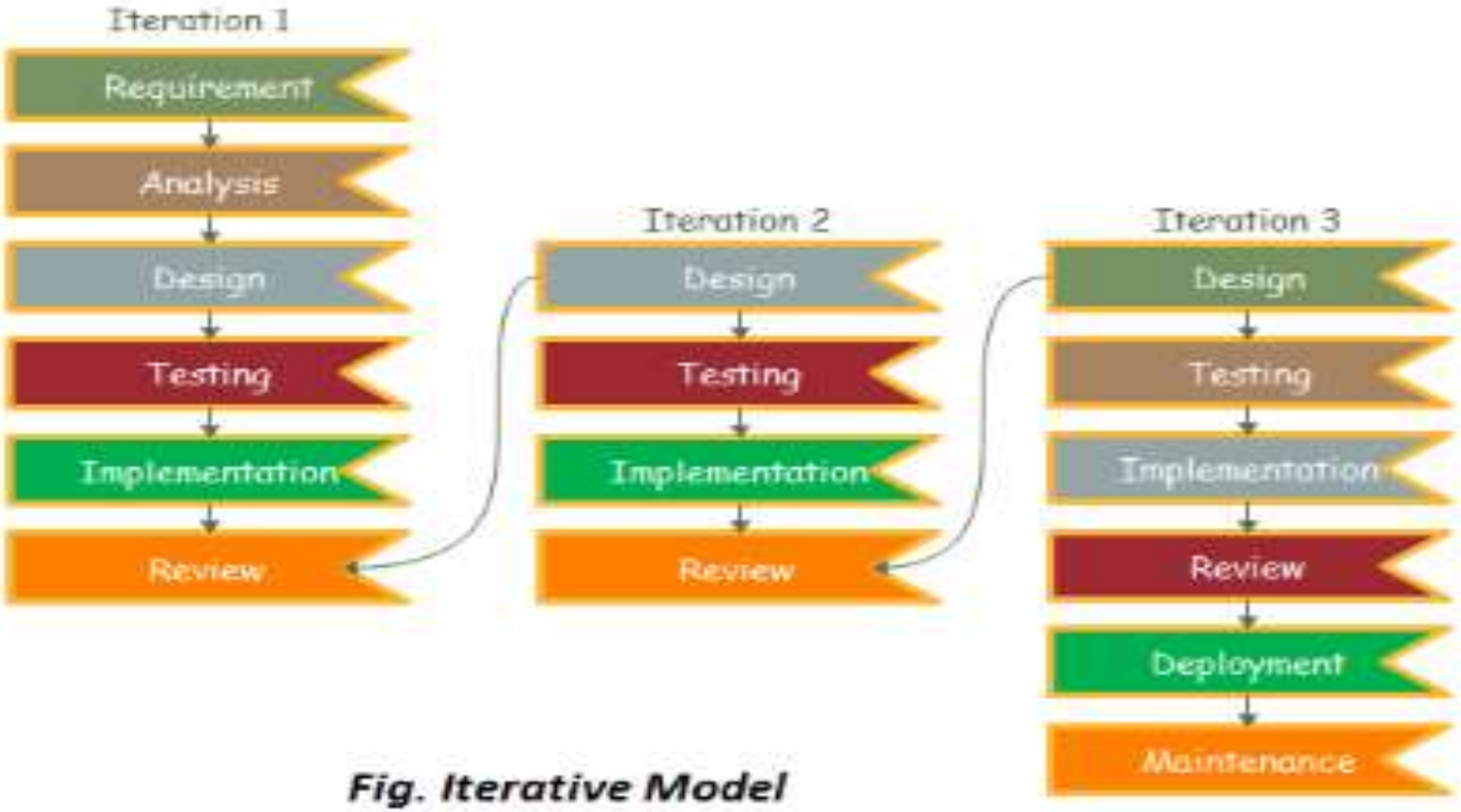


Fig. Iterative Model

- **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, team design the software by the different diagrams like Data Flow diagram, activity diagram, class diagram, state transition diagram, etc.

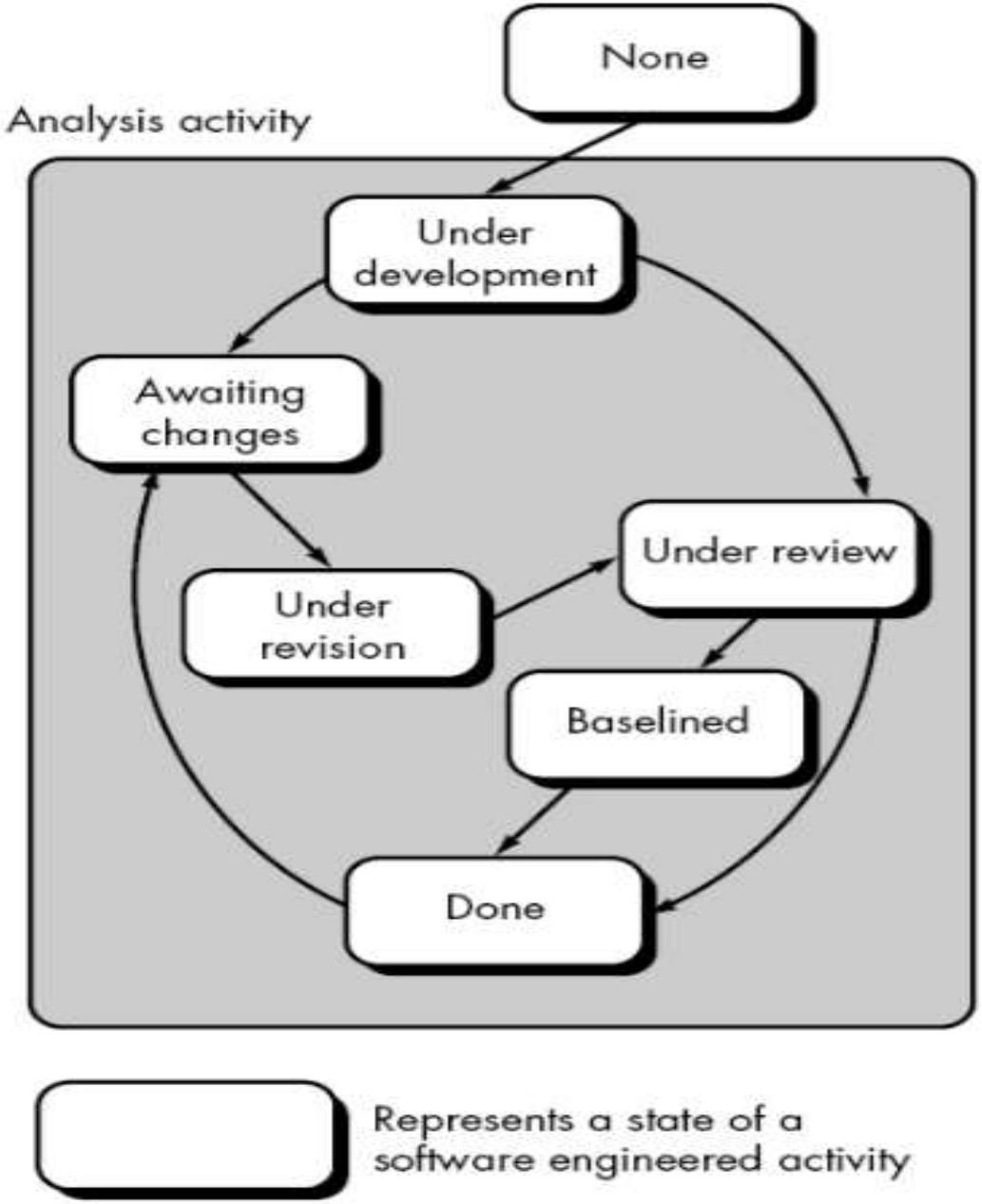
- **3. Implementation:** In the implementation, requirements are written in the coding language and transformed into computer programmes which are called Software.
- **4. Testing:** After completing the coding phase, software testing starts using different test methods. There are many test methods, but the most common are white box, black box, and grey box test methods.
- **5. Deployment:** After completing all the phases, software is deployed to its work environment.
- **6. Review:** In this phase, after the product deployment, review phase is performed to check the behaviour and validity of the developed product. And if there are any error found then the process starts again from the requirement gathering.
- **7. Maintenance:** In the maintenance phase, after deployment of the software in the working environment there may be some bugs, some errors or new updates are required. Maintenance involves debugging and new addition options.

- ***When to use the Iterative Model?***
  - When requirements are defined clearly and easy to understand.
  - When the software application is large.
  - When there is a requirement of changes in future.
- ***Advantage(Pros) of Iterative Model:***
  - Testing and debugging during smaller iteration is easy.
  - A Parallel development can plan.
  - It is easily acceptable to ever-changing needs of the project.
  - Risks are identified and resolved during iteration.
  - Limited time spent on documentation and extra time on designing.
- ***Disadvantage(Cons) of Iterative Model:***
- It is not suitable for smaller projects.
- More Resources may be required.
- Design can be changed again and again because of imperfect requirements.
- Requirement changes can cause over budget.
- Project completion date not confirmed because of changing requirements.

Concurrent development model

- The concurrent development model is called as ***concurrent model.***
- The *communication activity* has completed in the first iteration and exits in the ***awaiting changes state.***
- The modeling activity completed its initial communication and then go to the *underdevelopment state.*
- If the customer specifies the *change in the requirement,* then the modeling activity moves from the *under development state* into the awaiting change state.
- The concurrent process model activities moving from one state to another state.

- **Inactive:** No any activity / state performed.
- **Under Development:** Any activity performed.
- **Awaiting Changes:** If customer want any changes
- **Under review:** Testing activity start.
- **Under revision:** Do all required changes.
- **Baselined:** As per the SRS document
- **Done:** Project completed & Deployed.



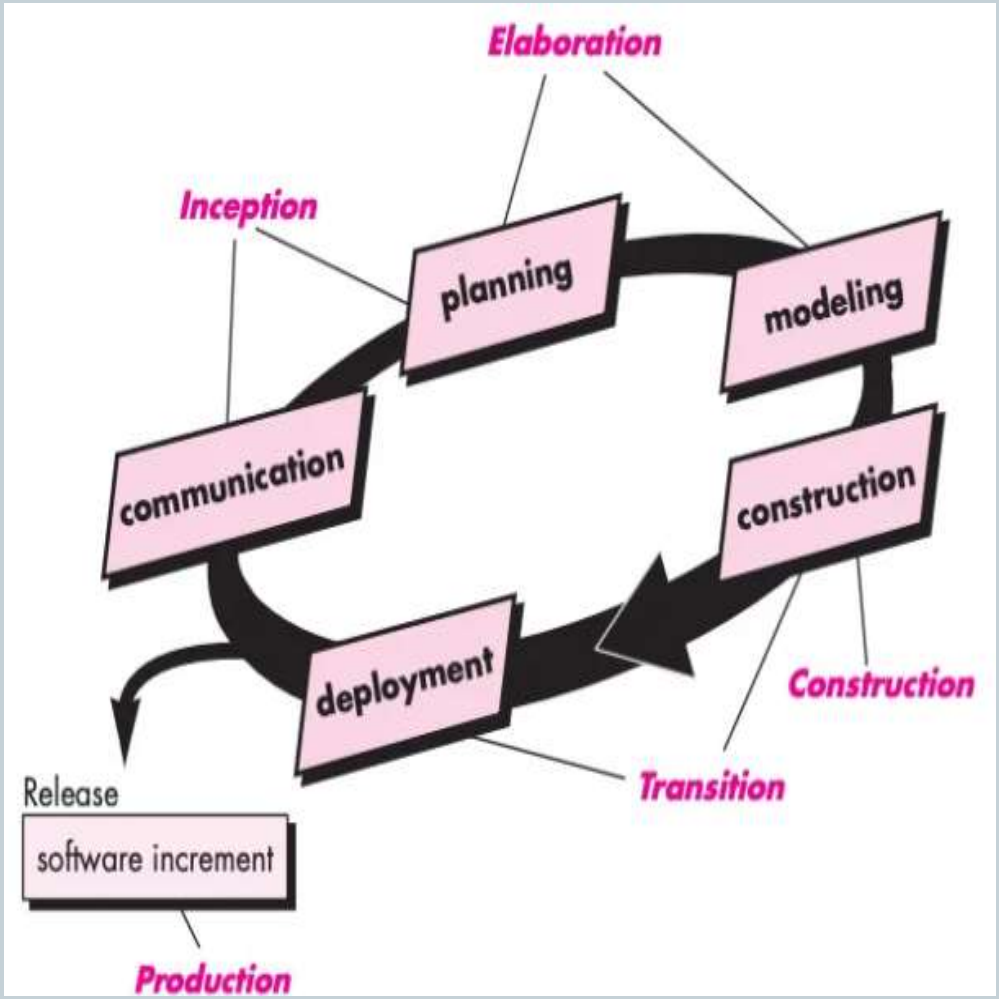
Concurrent Development Model

- **Advantages of the concurrent development model**
- This model is applicable to all types of software development processes.
- It is easy for understanding and use.
- It gives immediate feedback from testing.
- It provides an accurate picture of the current state of a project.
- **Disadvantages of the concurrent development model**
- It needs better communication between the team members. This may not be achieved all the time.
- It requires to remember the status of the different activities.

Unified Process

- The **Unified Process (UP)** is a software development framework ***used for object-oriented modeling***. The framework is also known as Rational Unified Process (RUP) and the Open Unified Process (Open UP). Some of the key features of this process include:
- It defines the order of phases.
- It is component-based, meaning a software system is built as a set of software components. There must be well-defined interfaces between the components for smooth communication.
- It follows an ***iterative, incremental, architecture-centric, and use-case driven approach***

- **Phases of RUP:** There is total of five phases of the life cycle of RUP
  1. Inception
  2. Elaboration
  3. Construction
  4. Transition
  5. Production


Elaboration
Inception
planning
modeling
communication
construction
deployment
Construction
Transition
Release
software increment
Production

- **Inception –**
  - The inception phase of the UP encompasses ***both customer communication and planning activities***
  - 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.***
    - What are the key features?
    - How does this benefit the customers?
    - Which methodology will we follow?
    - What are the risks involved in executing the project?
    - Schedule and cost estimates.

Elaboration: The elaboration phase encompasses the communication and modeling activities of the generic process model . ***Elaboration refines and expands the preliminary use cases that were developed as part of the inception phase*** and ***expands the architectural representation to include five different views of the software.*** *the usecase model, the requirements model, the design model, the implementation model, and the deployment mode .*

The architectural baseline demonstrates the viability of the architecture but does not provide all features and functions required to use the system.

Construction : The construction phase of the UP is identical to the construction activity defined for the generic software process. ***Using the architectural model as input***, the construction phase develops or acquires the software components that will make each use case operational for end users. As components are being implemented, unit tests are designed and executed for each.

- **Transition –**
- This phase involves the deployment, multiple iterations, beta releases, and improvements of the software. The users will test the software, which may raise potential issues. The development team will then fix those errors.
- **Production –**
- production phase of the UP coincides with the deployment activity of the generic process. During this phase, the ongoing use of the software is monitored, support for the operating environment (infrastructure) is provided, and defect reports and requests for changes are submitted and evaluated

Advantages and Disadvantages

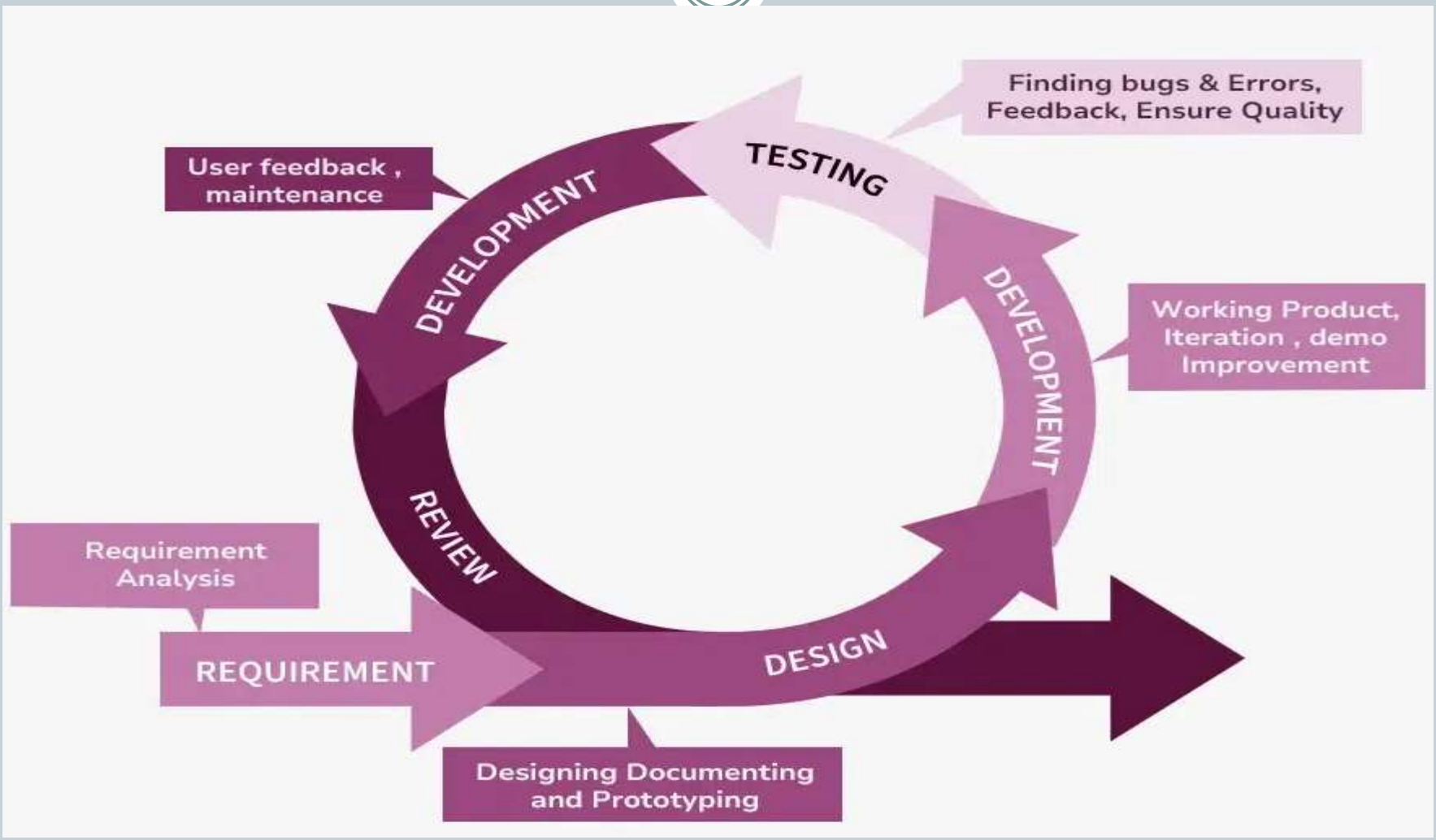
Advantages:

- It provides good documentation, it completes the process in itself.
- It provides risk-management support.
- It reuses the components, and hence total time duration is less.

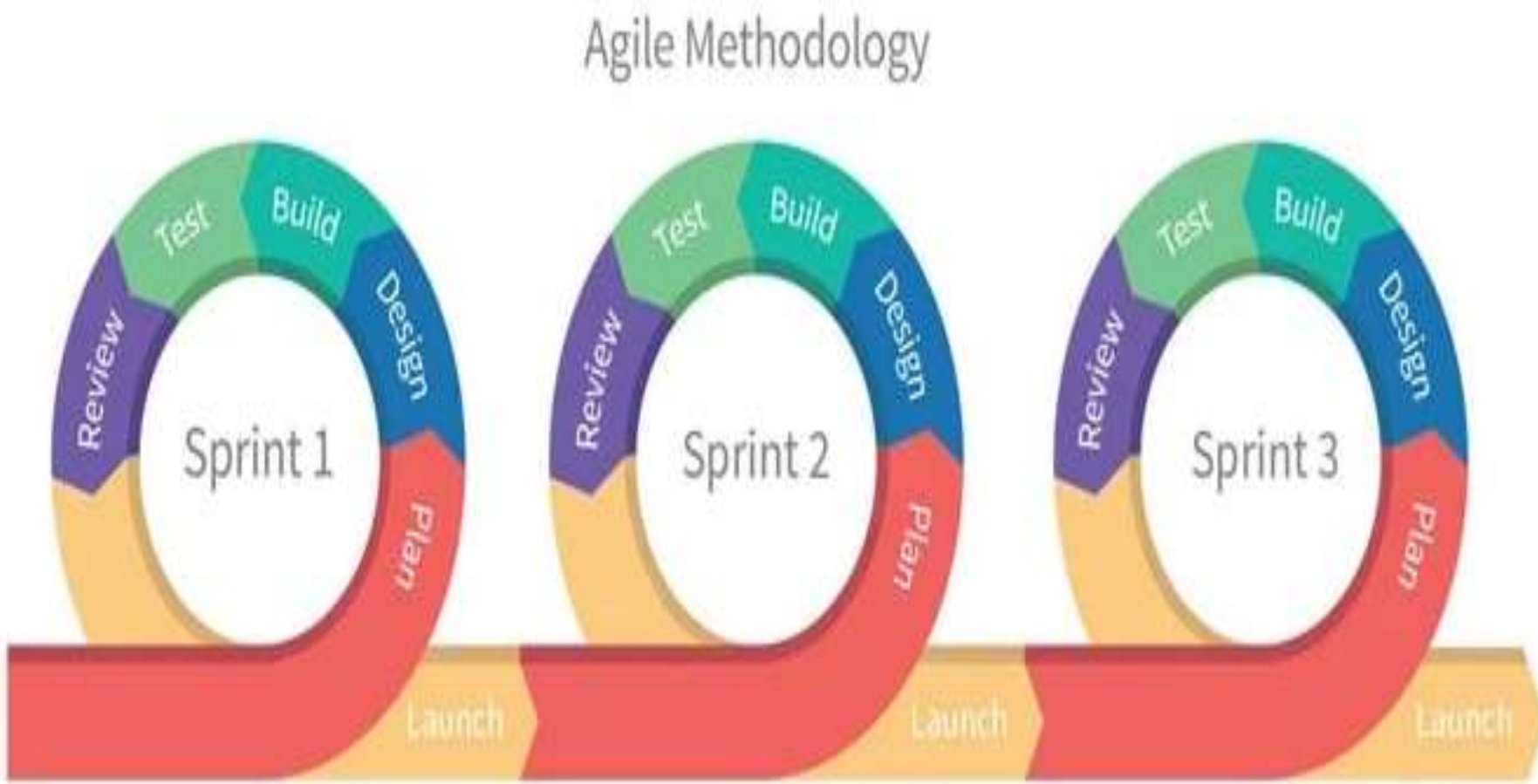
Disadvantages:

- Team of expert professional is required, as the process is complex.
- More dependency on risk management.
- Hard to integrate again and again.

AGILE MODEL


Finding bugs & Errors, Feedback, Ensure Quality
User feedback , maintenance
TESTING
DEVELOPMENT
DEVELOPMENT
Working Product, Iteration , demo Improvement
REVIEW
Requirement Analysis
REQUIREMENT
DESIGN
Designing Documenting and Prototyping

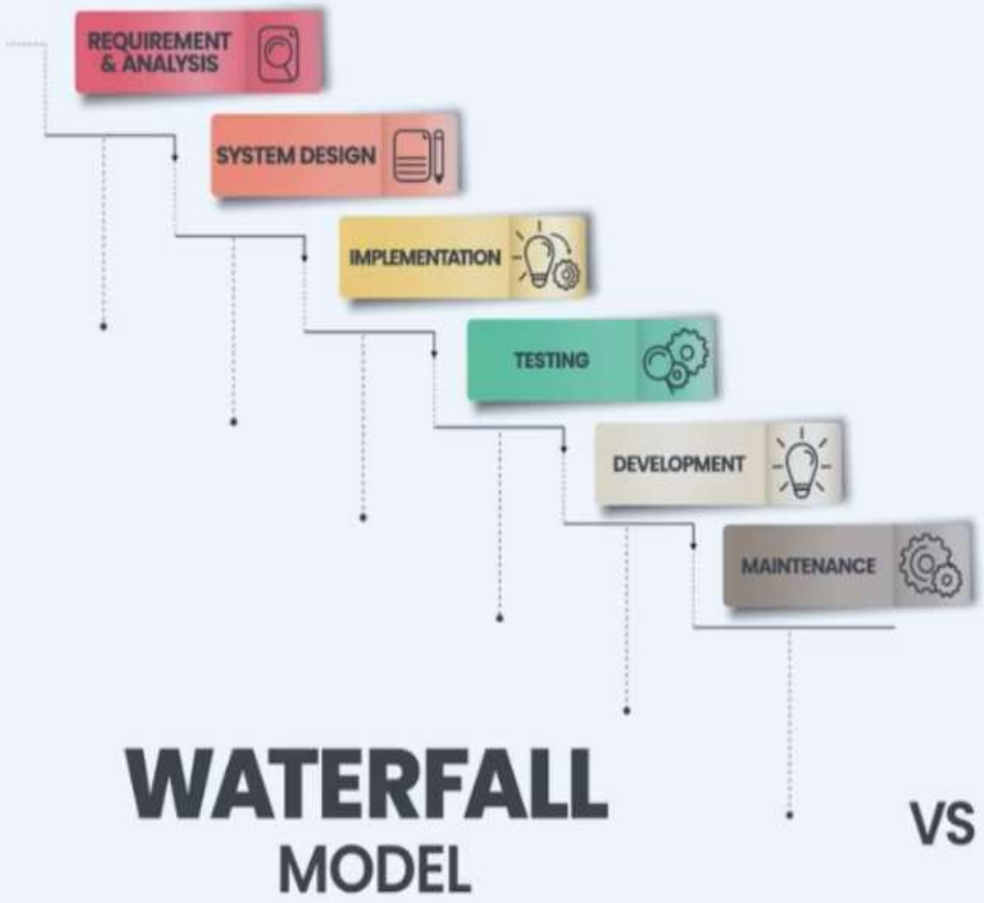
AGILE MODEL


Agile Methodology
Review
Test
Build
Design
Plan
Sprint 1
Launch
Review
Test
Build
Design
Plan
Sprint 2
Launch
Review
Test
Build
Design
Plan
Sprint 3
Launch

- Agile methodology is a project management approach that's all about adaptability, collaboration, and delivering value fast. Born from the 2001 Agile Manifesto for Software Development, it moves software development towards iterative "sprints" and enables rapid development and testing cycles that align with user needs.

- Agile breaks projects into short “sprints.” Multiple teams deliver products every 1 to 4 weeks. Agile enables fast progress and feedback with defined requirements. Regular feedback loops allow for changes based on user needs or market shifts.

Agile vs. Waterfall



VS