Software Process Models (or Software Engineering Paradigm)

It is descriptive and diagrammatic model of software life cycle.
- Identifies all the activities required for product development.
- Captures the order in which these activities are to be undertaken.
- Divides the life cycle into phases,where several different activities may be carried out in each phase.
- To solve real life problems in industry settings, Software Engineers or a team of engineers must incorporate development strategy that covers the process, methods and tools.
- This strategy is called a software process model or software Engineering Paradigm, which is selected on the basis of the nature of the project and the applications, development methods and tools to be used, the controls and the deliverables that are required.

WHY USE A LIFE CYCLE MODEL?
- **Primary advantage is it helps in development of software in a systematic and disciplined manner.**
- **When a program is developed by a single programmer,he has the freedom to decide his exact step.**
- **When the product is being developed by a team,there must be a precise understanding among team member as to "when to do what" otherwise it would lead to chaos and project failure.**

Types of software process models :

- Linear Sequential Model (Water Fall Model)
- Prototyping Model
- RAD Model
- Evolutionary Model
- Incremental Model
- Spiral Model
- V-Model
- Big-bang Model

Water fall or linear sequential model or traditional life cycle model

This model assumes that everything is carried out and taken place perfectly as planned in the previous stage and there is no need to think about past issues that may arise in next phase .

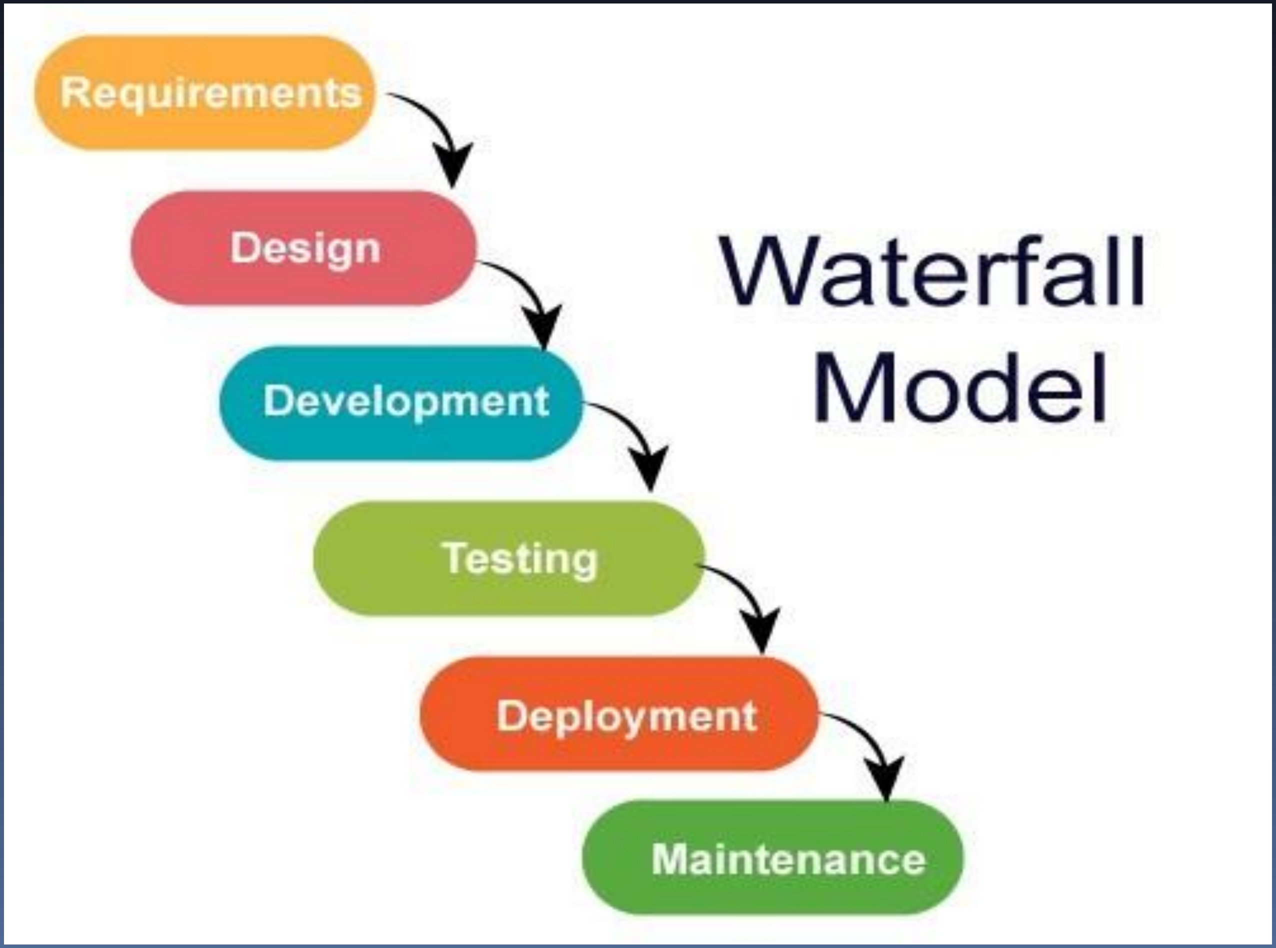
This model doesn't work smoothly if there are some issues left at the previous step.

The sequential nature of model doesn't allow us to go back and undo or redo our actions.

This model is best suited when developers already have designed and developed similar software in the past and are aware of all its domains.

Waterfall model phases

(Linear Sequential Model)



Waterfall Model Phases

A. Software Requirement analysis

This is the most crucial phase for the whole project, here project team along with the customer makes a detailed list of user requirements. The project team explains out the functionality and limitations (if there are any) of the software they are developing,in detail. The document which contains all this information is called SRS and it clearly and unambiguously indicates the requirements. A small amount of top-level analysis and design is also documented. This document is verified and endorsed by the customer before starting the project. SRS serves as the input for further phases.

Waterfall Model Phases

b. Design

- **It is a multi step process to address various aspects to be implemented such as** data structures , software architecture, interface representation , procedural (algorithmic details) **etc**
- **It** translates requirements into representation **of the software which can be assessed before the code generation**
- **Design document is also a part of the** software configuration

Waterfall Model Phases

c. Code generation

- Design translated into a machine readable form

d. Testing

- It focuses on the logical intervals of the software (all statements) and functional externals of the software , tests to uncover errors
- **Basic objective:** Defined input should produce desired output

Waterfall Model Phases

e. Installation Phase: -

In the installation phase of the waterfall model, the process will continue until the application is bug-free/ stable/ and works according to customer needs.

After that, the stable application is installed into the customer's environment for customer use.

After receiving the product, the customer will do one round of testing for their satisfaction. While using a product if the customer faces any defect, it will be informed to the development team of that particular software to solve the issue. When all the bugs are resolved, the software finally deployed to the end-user.

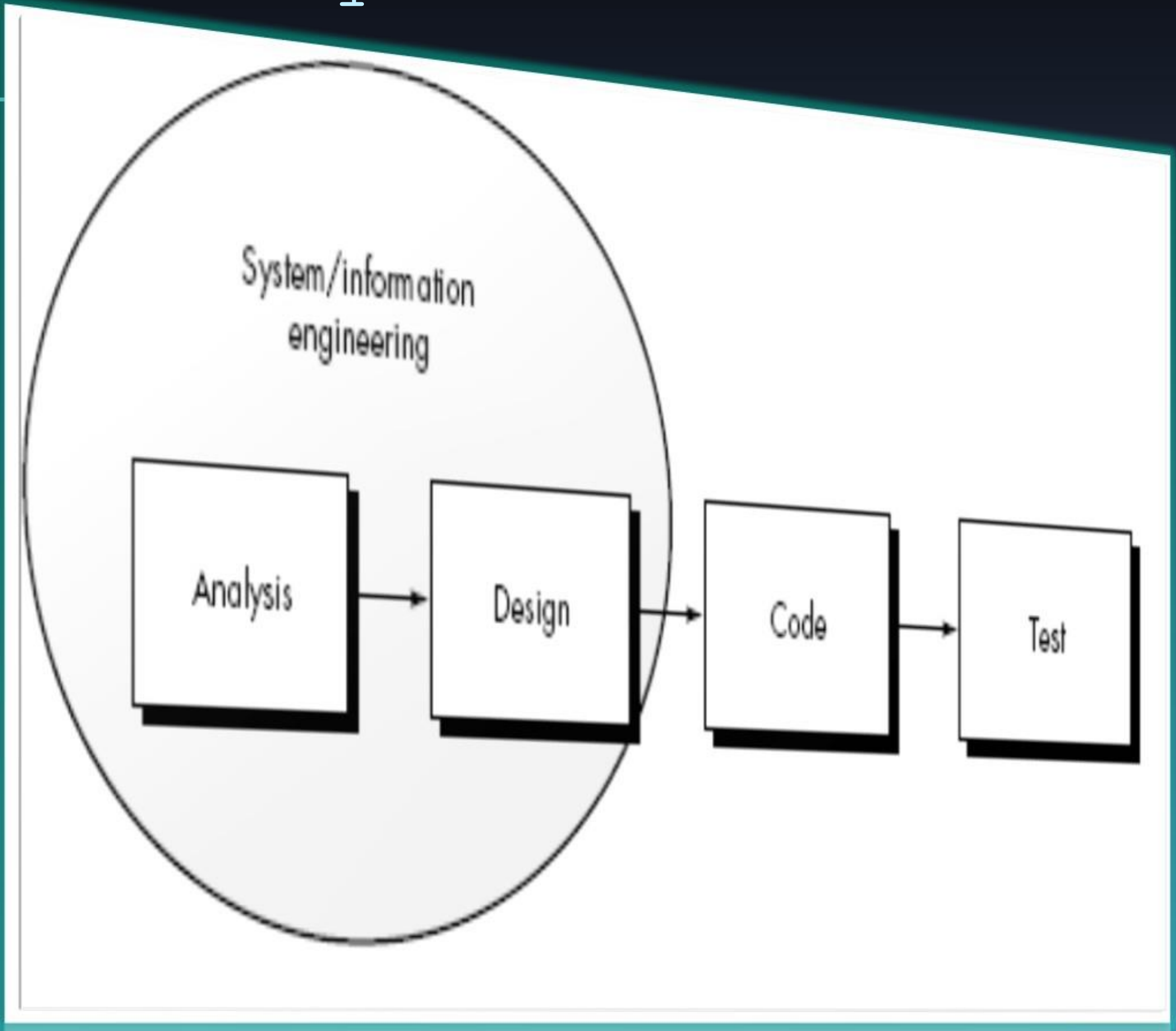
Waterfall Model Phases

f.Maintenance Phase: -

The Last and long-lasting phase of the waterfall model is the maintenance phase. This process will continue until the application comes to an end. When a customer starts using the Software, then they may have some issues which need to be in-detail tested & fixed.

Maintenance phase also includes changes in software and hardware to maintain its operational effectiveness and improve its performance. The process of taking care of product time to time is called maintenance.

Waterfall Model (Linear Sequential Model



Waterfall Model Phases

Drawback : Difficulty of accommodating change **after the process is underway**

Appropriate when requirements are well understood

Waterfall Model Phases

Advantages of Waterfall Model

Important benefits of the waterfall model are as follows-

Simple and easy to understand and use

It represents all the tasks that you want to do in real life. For example, you need the requirements of a client. It contains different phases, and each phase is started only when the previous phases get completed.

Specific deliverable and review process

Each phase has a specific deliverable and review process. After the requirement phase, we have all the requirements of what the customer needs. Once the software is developed, we have its deliverable.

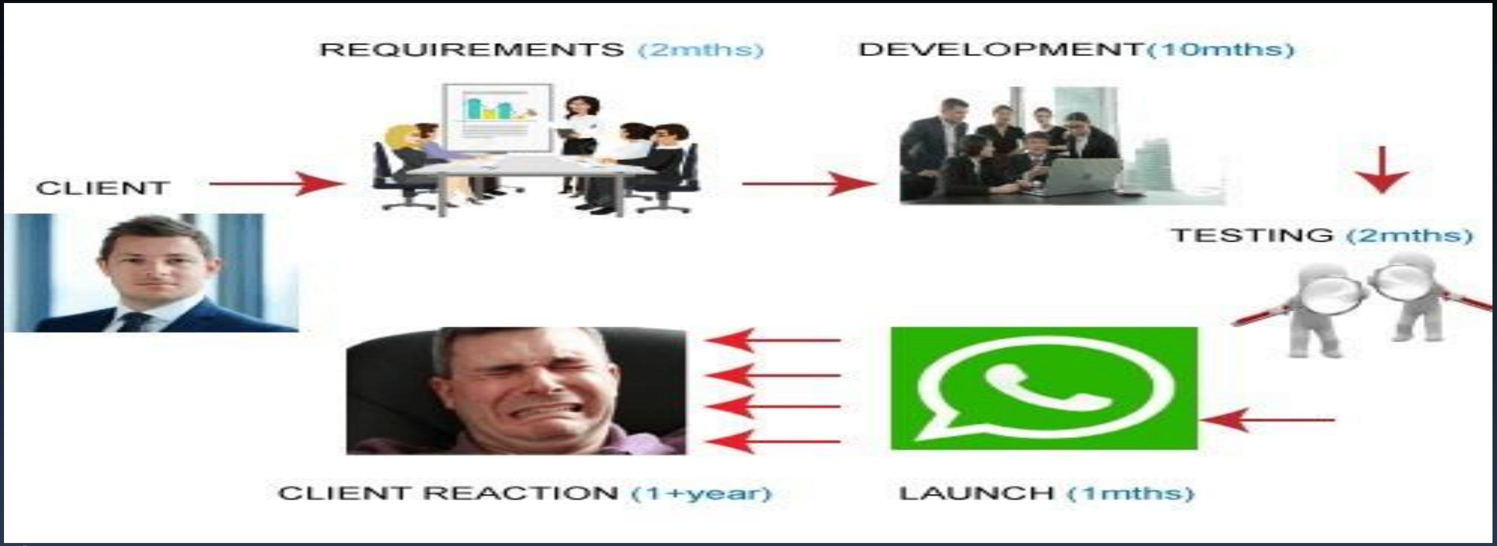
Phases do not overlap

In this model, phases do not overlap, i.e., they are completed once at a time. Once the previous phase is completed, then only the next phase gets started. For example, the Development phase will start only when the design phase is completed.

Waterfall Model Phases

Problems of waterfall model

1. It is difficult to define all requirements at the beginning of a project
2. This model is not suitable for accommodating any change
3. A working version of the system is not seen until late in the project's life
4. It does not scale up well to large projects.
5. Real projects are rarely sequential



Suppose the client wants an app like a WhatsApp, so he reaches to the company where both the company and the client had a discussion for 2 months. The company made the documentation of all the requirements in 2 months. Now, the development team starts developing the software and suppose it took around 10 months to develop the software. It means that 12 months have been used, i.e., 2 months in requirement phase and 10 months in a development phase, but still the client does not have the idea about the internal phases. Once the development is completed, testing is done, and it will take around 2 months for software quality testing. Once the testing is done, it goes to the integration and launch so that WhatsApp will become live. However, when it reaches to the client, then the client says that it has taken more than a year and the software that I received was not what I expected. This happened because the client had only verbal communication with the software team. If the client wants some changes in the software, then the whole process will be executed again.

Prototyping Model

Customer

What is to be done or what to explain
(objective of software?)
Input? Output?
?

Developer

Efficiency of algorithm, adaptability , interfaces ?
?

Prototyping Model

Requirements Gathering :

- Define overall system objective
- Track all known requirements
- Outline areas of further development

Quick Design Approach:

- Results in all user visible aspects (input approach, output formats etc)

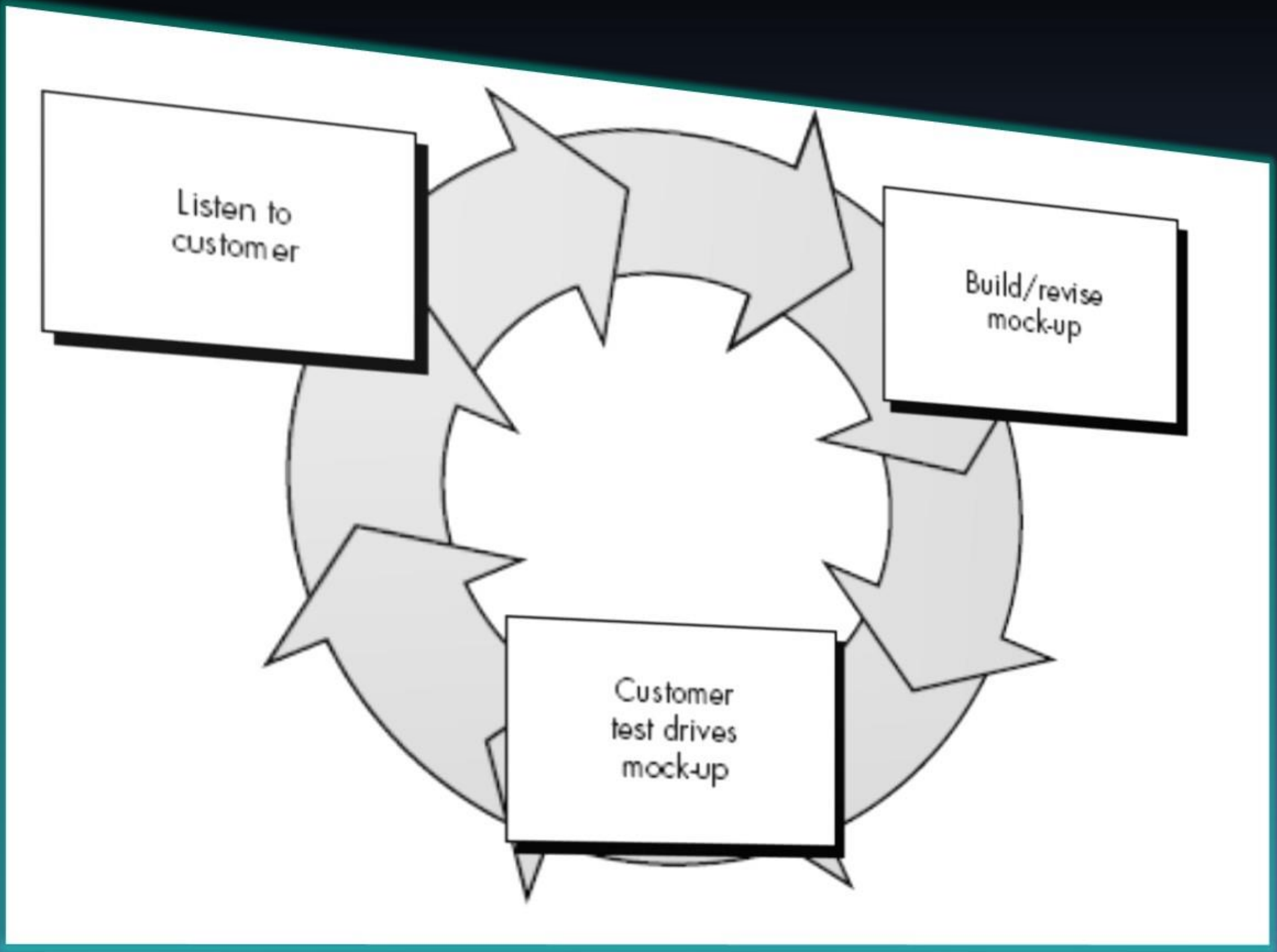
Construction of a Prototype :

- Evaluated by the customer/user
- Specify refinements in requirements

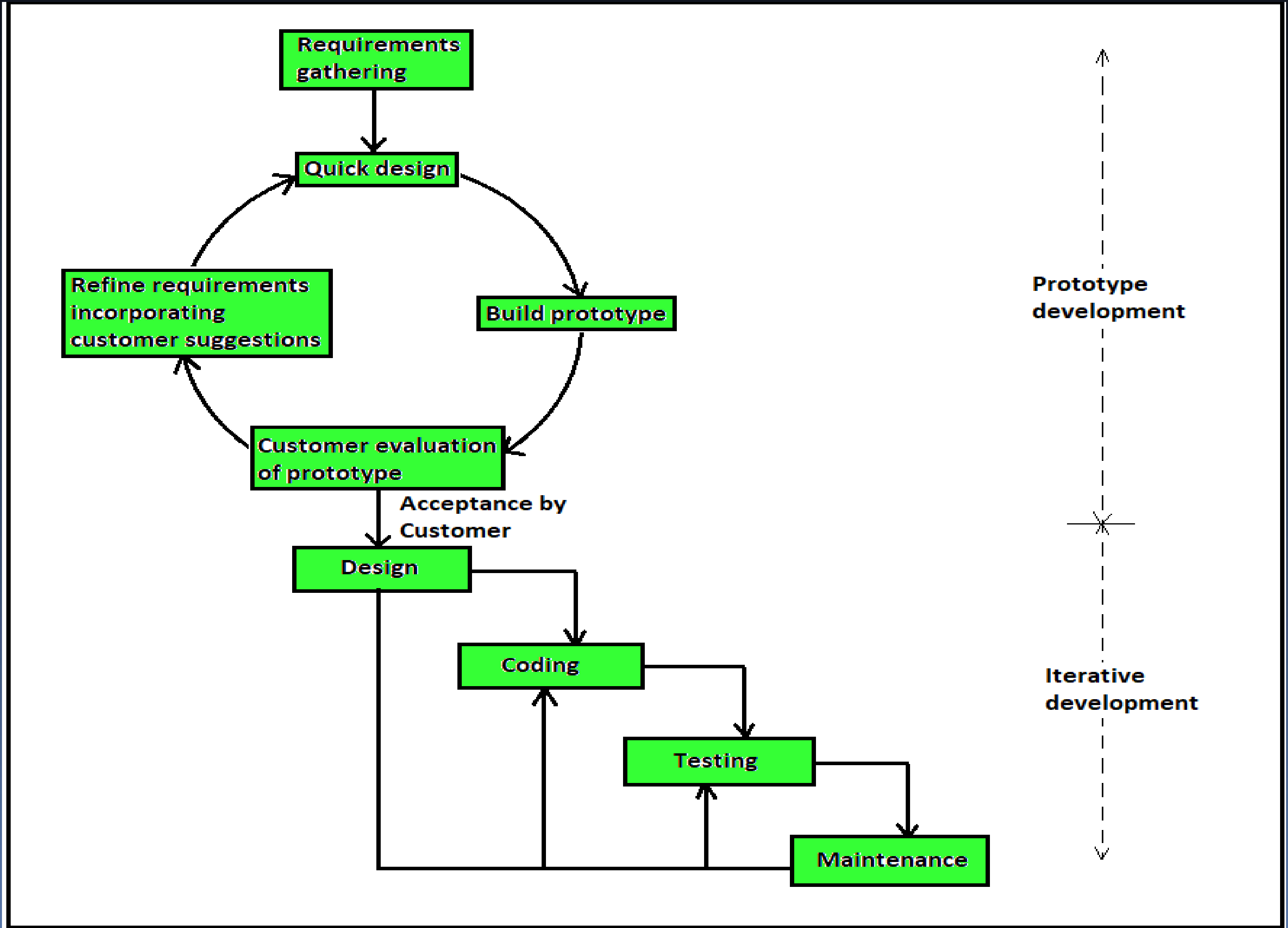
Software prototyping refers to the activity of creating prototypes of software applications i.e incomplete versions of the software program being developed.

- The prototyping model is a systems development method in which a prototype is built,tested and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed.
- Customer sets general objective for software but doesn't properly identify detailed system behavior. Prototyping models tries to capture requirements of customer in detail through a series of quick design and evolution
- This model beings with requirement gathering,then a quick design occurs which leads to development of prototype. Customer evaluates the prototype and uses it to refine requirements. Then iteration occurs as a prototype is modified to satisfy customers need.

Prototyping Model



Prototype model



Prototyping Model

* Prototype serves as a tool to identify software requirements

* Working prototypes are built from existing program fragments or tools , libraries etc

Problematic Aspects of Prototyping

I want a working version of my software immediately!!

Custom er

I am not done yet! I only have a prototype ready!!

Develop er

Problematic Aspects of Prototyping

OK!! Just apply a few fixes to the prototype to make it a working product!!

I'll have to make some implementation compromise to get the prototype work quickly

Customer

Developer

Problematic aspects of Prototyping

Inappropriate OS **or** programming language **may be used because it is available and known to the developer**

Inefficient algorithms **may be implemented**

The less than ideal choice **may become an integral part of the system!!**

Limitations:

Customer cries foul and demands some “few fixes” be applied to make the prototype a working product

Developers may make a prototype for small database , use algorithm for few and for specific operating system ,so it may create problem in future.

Advantages:

Users are actively involved in development.

 Since in this methodology a working model of the system is provided, the user get a better understanding of the system being developed.

Errors can be detected much earlier as the system is made side by side.

Quicker user feedback is available leading to better solution.

The RAD Model (Rapid Application Development)

Primarily used for information system **applications**

If requirements are well understood , **and** modularized , **RAD process develops fully functional system within** 60 **to** 90 days

RAD model

- The RAD (Rapid Application Development) model is a software engineering methodology that **emphasizes quick iterations, user involvement, and rapid prototyping to deliver working software faster**.

- It is an iterative and incremental approach that prioritizes speed and flexibility over extensive initial planning, making it well-suited for projects where requirements may change.

Phases of RAD process

Business Modeling: Models information flow among business functions, various input process, output aspects of information

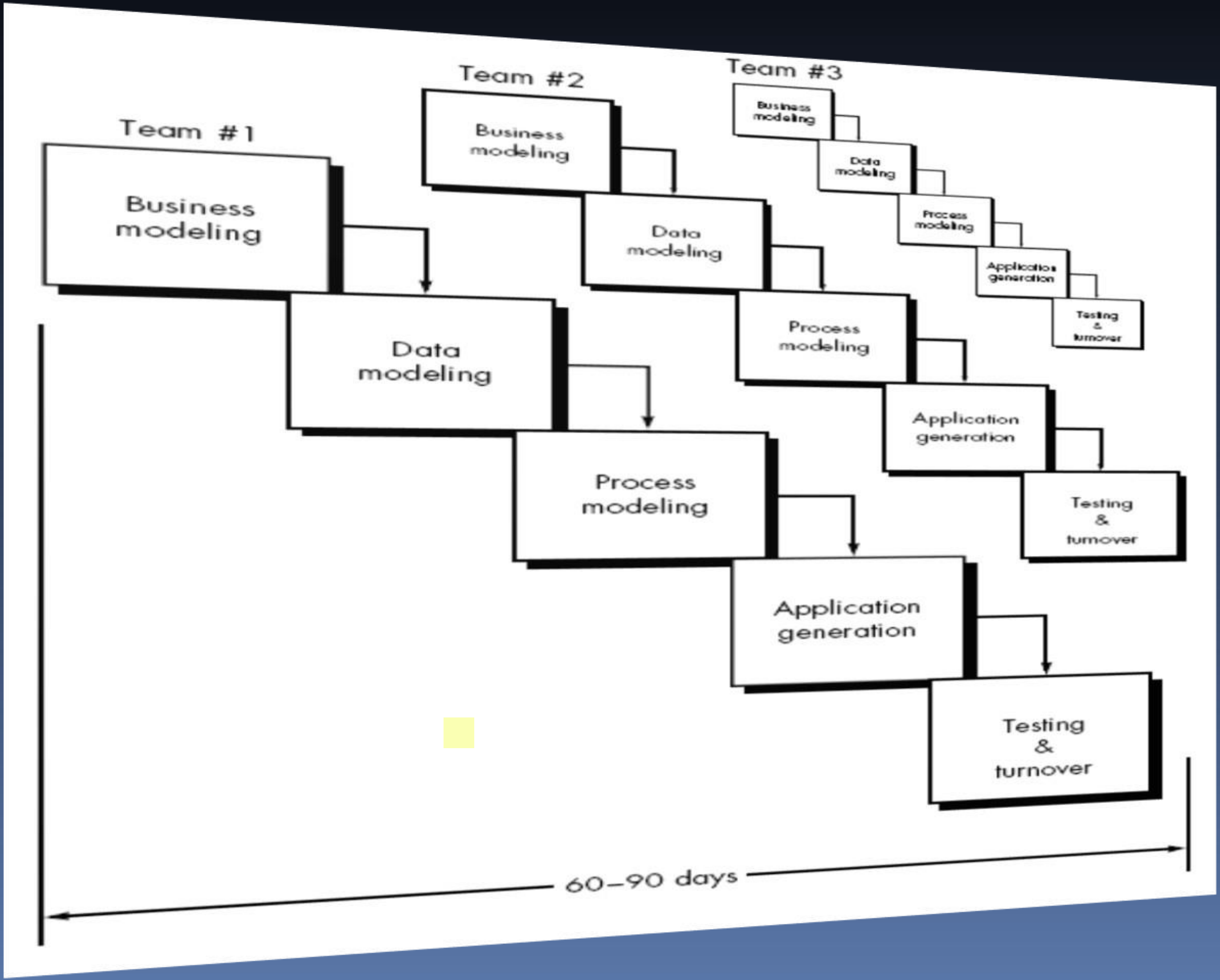
Data Modeling: Phase one is redefined into a set of data objects that takes part in business activity (information process)

Process Modeling: The data objects are transformed thru processing to achieve the information flow

Application Generation : RAD model makes use of fourth generation techniques reuses existing a program comp0nents wherever possible any other automated tools to speed up the development process

Testing and turn over: Testing overhead is reduced due to re-usability only the new components need testing

Phases of RAD process



Drawbacks of RAD Model:

Very High Human resource requirement **overhead**

Customer and developer both should be committed

All types of application are not **appropriate for development under RAD strategy**

Evolutionary Software Process Models:

Linear sequential model is meant for straight line (linear) **development approach (delivers a complete product)**

Prototyping helps the customer to understand **the system requirements**

Evolutionary Software Process Models:

Evolutionary models are inherently iterative **in nature**

It helps to develop increasingly more complete versions **of the target software**

Evolutionary model

- The evolutionary model is a software development approach that emphasizes iterative and incremental development, where the software is built in small, successive releases based on user feedback.

- This model is highly flexible and adaptable, allowing requirements to evolve and new features to be added over time, making it suitable for large or complex projects where requirements are not fully defined at the start

Key aspects of the evolutionary model

- **Iterative and incremental**: The development process is divided into smaller, repeating cycles, with each cycle adding new functionality to the previous version.

- **User feedback**: Customer or user feedback is gathered at the end of each cycle to plan and guide the next iteration, which allows the software to adapt to changing needs.

- **Early delivery**: A working, though basic, version of the software is delivered early in the development cycle, providing early value and a basis for feedback.
- **Adaptability**: It is well-suited for projects with ambiguous or evolving requirements, as it allows for continuous improvement and changes.
- **Risk reduction**: Issues and errors can be identified and addressed in smaller chunks during each iteration, reducing the risk of major problems later in the project.

Advantages

- **Flexibility:** High adaptability to changes and evolving requirements.
- **Early returns:** Working software is delivered early, which can provide a faster return on investment.
- **Risk mitigation:** Risks are addressed and managed in each iteration, rather than all at once at the end.
- **Customer involvement:** High customer involvement throughout the development process.

Disadvantages

- **Communication challenges:** Requires constant and clear communication between team members and stakeholders.
- **Complexity:** Can be more complex to manage, especially for larger projects, due to multiple increments and changing requirements.
- **Initial cost:** May have higher initial costs due to continuous testing, feedback, and prototyping.
- **Potential for poor structure:** If not managed properly, systems can become poorly structured.

Iterative model

- An iterative model is a software development approach where a system is built in repeated cycles (iterations), with each cycle adding to and refining the product until the full requirements are met.

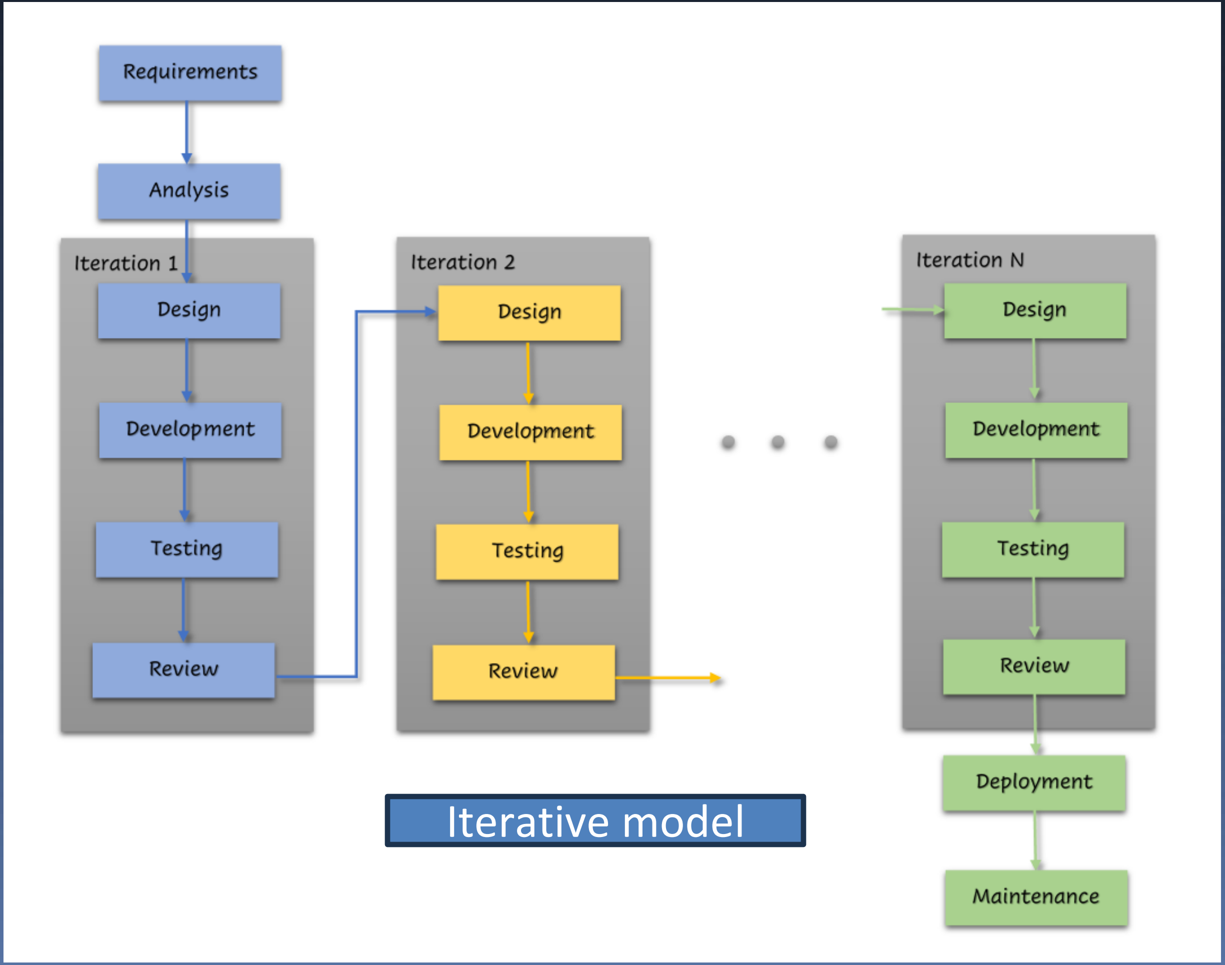
- This model involves a continuous loop of planning, designing, coding, and testing in smaller segments, making it ideal for complex projects where requirements may change.

- Eg: Microsoft Windows operating system

Iterative model

Consider an iterative life cycle model which consists of repeating the following phases in sequence:

- Requirements
- Analysis
- Design
- Testing
- Implementation
- Review
- Maintenance



- **Requirements** : A Requirements phase, in which the requirements for the software are gathered and analysed. Iteration should eventually result in a requirements phase that produces a complete and final specification of requirements.

- **Analysis Phases:** 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.

- **Design phase**: A Design phase, in which a software solution to meet the requirements is designed. This may be a new design, or an extension of an earlier design.

- **Development Phases:** where the software is coded, integrated and tested.
- **Implementation Phases:** In the implementation, requirements are written in the coding language and transformed into computer programmes which are called Software.
- **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.

- **Review:** A Review phase, in which the software is evaluated, the current requirements are reviewed, and changes and additions to requirements proposed.

How it works?

- Initial Version: Development begins with a simple, working version of the software that includes a small set of requirements.
- Each Iteration: A complete development cycle of analysis, design, implementation, and testing is performed for a specific component or functionality.
- Feedback and Refinement: After each iteration, the results are evaluated. Stakeholders provide feedback, which is used to make improvements and adjustments in the next cycle.
- Continuous Improvement: The process repeats, with each subsequent iteration building upon the previous one to add new features or enhance existing ones.
- Final Product: The final product is achieved after the last iteration, once all requirements have been fully implemented and refined.

When to use 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.

Advantages of Iterative Model

- Generates working software quickly and early during the software life cycle.
- More flexible - less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Easier to manage risk because risks are identified and handled during its iteration.
- Each iteration is an easily managed milestone.

Disadvantages

- **Complexity:** The process can become complex to manage due to frequent changes and the need for continuous communication.
- **Costly:** It can require more resources and may be more costly than traditional models due to frequent testing and rework.
- **Time-Consuming:** Iterations and rework can potentially make the project time-consuming.
- **Resource Intensive:** Requires a skilled and experienced development team to manage the iterative process effectively.

Types of Evolutionary Software Process Models:

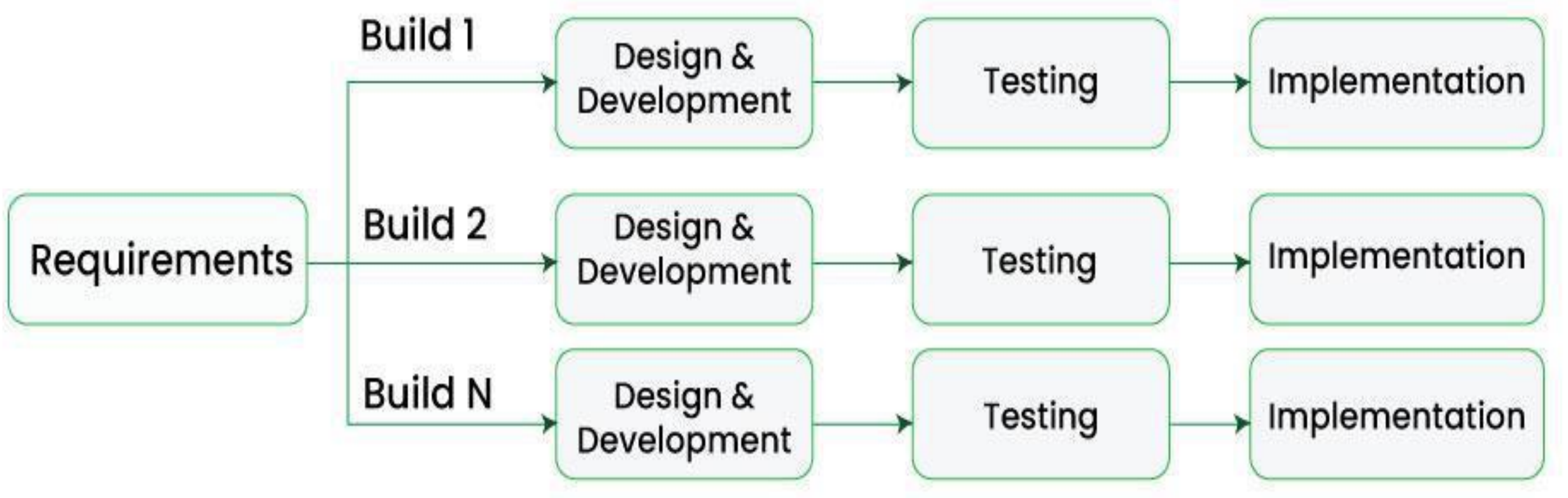
Iterative + incremental model

Spiral model

Incremental Model

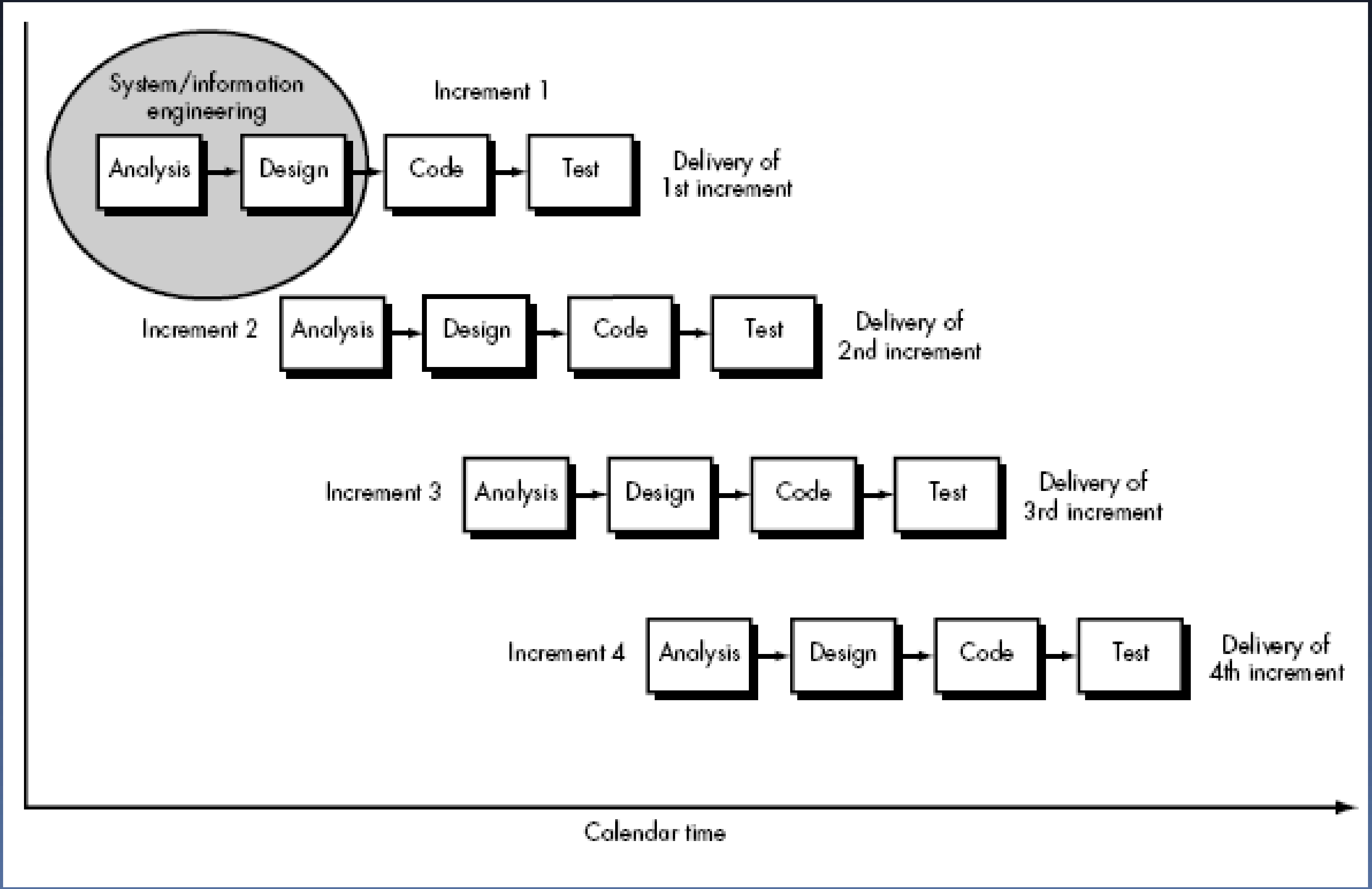
It is a combination of linear sequential model philosophy with the iterative philosophy of prototyping paradigm
The model is outlined pictorially as below
Example of Incremental Model: Word processing Softrware
First Increment: the core product (Basic Word Processing Application Software
Basic WP requirements are addressed
Supplementary features (Known + unknown) are not delivered
Core is used by the customer (undergoes detailed review)
Helps to plan next development in order to better meet customers need and delivery of additional features and functionality

Incremental model



- Subsequent increments: This process is repeated each time until the complete product is produced (final version)
- Each increment is a stripped down (refined)version of the final product
- provides a platform for evaluation by the user and hence for their development

- Benefits:
- Low manpower requirement
- Early increments can be implemented with fewer people
- Increments can be planned to manage various technical risk (change in hardware platform , OS features etc)

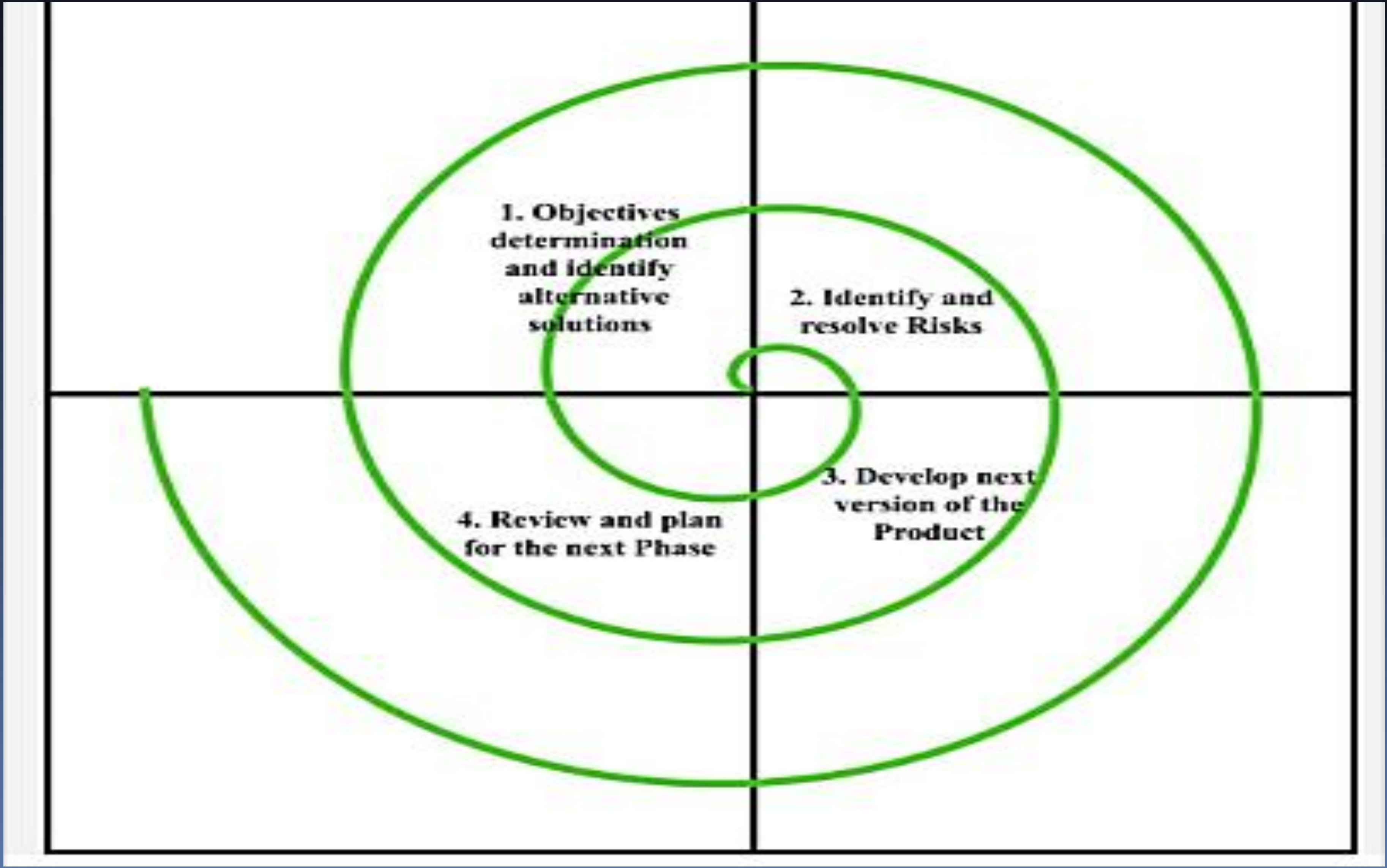

System/information engineering
Increment 1
Analysis
Design
Code
Test
Delivery of 1st increment
Increment 2
Analysis
Design
Code
Test
Delivery of 2nd increment
Increment 3
Analysis
Design
Code
Test
Delivery of 3rd increment
Increment 4
Analysis
Design
Code
Test
Delivery of 4th increment
Calendar time

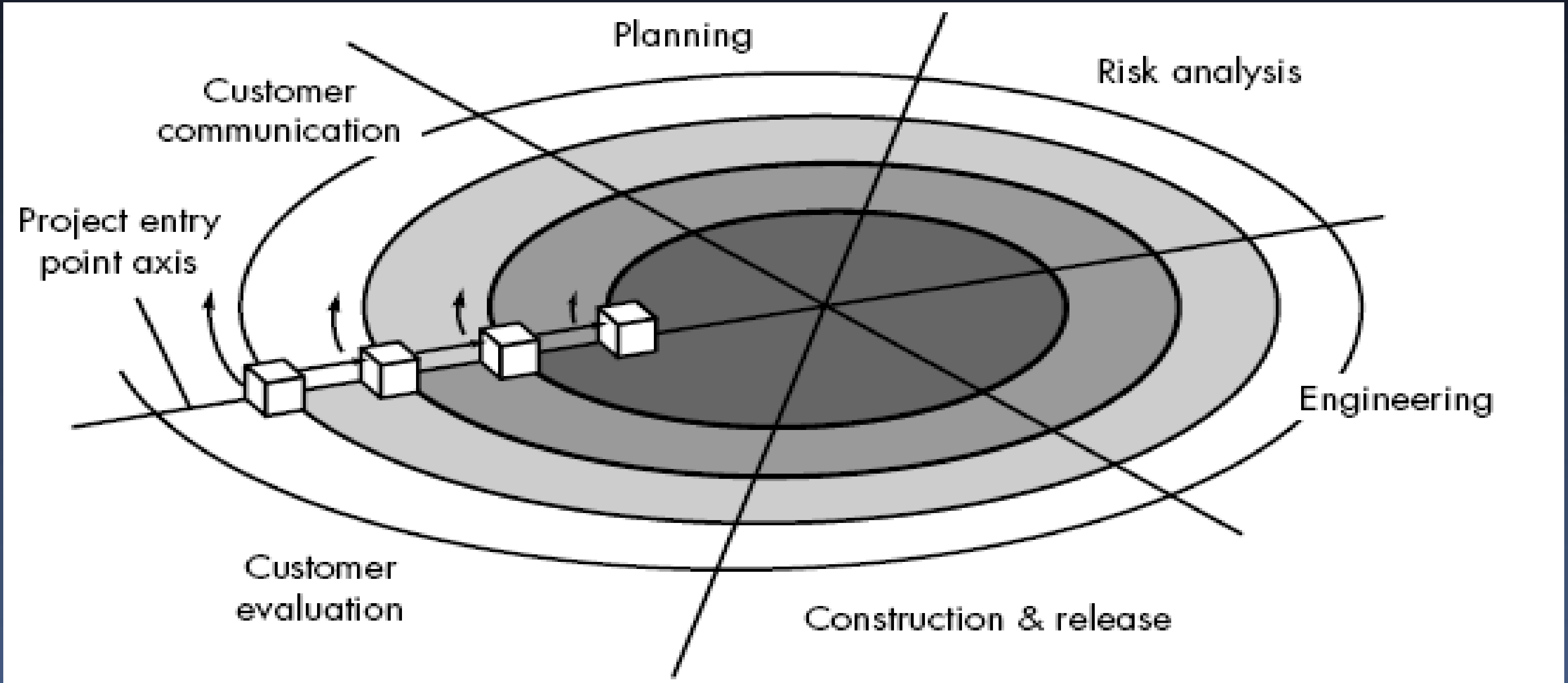
4 b. Spiral Model (Proposed by Bohem)

- This evolutionary software process model combines the iterative nature of prototyping model , the control and systematic aspect of linear sequential model
- It has potential for rapid development of incremental versions of the software
- Software is developed in a series of incremental releases
  - Early stage increments: paper model or prototype
  - Subsequent stage releases: more complete version of required software

- Spiral model handles the software development process in phase manner, each phase being treated as a project work
- Spiral model divides the development process into four projects:

Spiral model


1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase



☐	Product maintenance projects
░	Product enhancement projects
▒	New product development projects
▓	Concept development projects

Spiral model

1. Objectives Determination and Identification of Alternative Solutions

- This phase involves gathering business requirements and defining the project's objectives, constraints, and alternative solutions.

- Continuous communication between the customer and the system analyst is essential here.

Spiral model

2. Risk Analysis and Resolution

- The most critical phase, where potential risks (technical, management, etc.) associated with the project are identified, analyzed, and evaluated.

- Risk mitigation strategies are planned, and a prototype may be built to help resolve uncertain or high-risk elements.

Spiral model

3. Develop the next version of the Product

- In this phase, the actual software product is developed based on the requirements and risk analysis from the previous quadrants.
- This includes activities like architectural design, coding, and testing the software

Spiral model

4. Review and plan for the next Phase

- The customer evaluates the developed software version and provides feedback.
- Based on this evaluation and the remaining risks, planning for the next iteration of the spiral begins, with the goal of adding more functionality or refining existing features.

- Spiral Model is divided into a number of Frame Work Activities or Task Regions .
- There are six task regions or frame work activities:
- i. Customer Communication: Task for effective communication between customer and developer
- ii. Planning : Task is to define resources, timeline and other project related information
- iii. Risk analysis: the task is to assess technical and management risks
- iv. Engineering: task required to build one or more representations of the application
- v. Construction and release: task to construct, test, install and provide support (documentation, training etc)
- vi. Customer evaluation: task is to obtain customer feedback or evaluation (engineering stage versus implementation stage)

Advantages of the Spiral Model

1. Risk Handling
2. Good for large projects
3. Flexibility in Requirements
4. Customer Satisfaction
5. Iterative and Incremental Approach
6. Improved Quality

Disadvantages of the Spiral Model

1. Complex
2. Expensive
3. Too much risk analysis
4. Time consuming

- Spiral model is a realistic approach to development of large scale projects
- Spiral model uses Prototyping as a Risk Reduction mechanism. Prototyping is applied at any stage of the product
- It incorporated systematic approach as suggested by classical life cycle of software in an iterative way (frame-work)
- It demands direct consideration of technical risk

Discussions:

- Difficult to convince customer that evolutionary approach is controllable
- High expertise is required to assess considerable risk
- This is a new model not used widely as linear sequential development approach
- It will take number of years before the effectiveness of this model is known

V-model

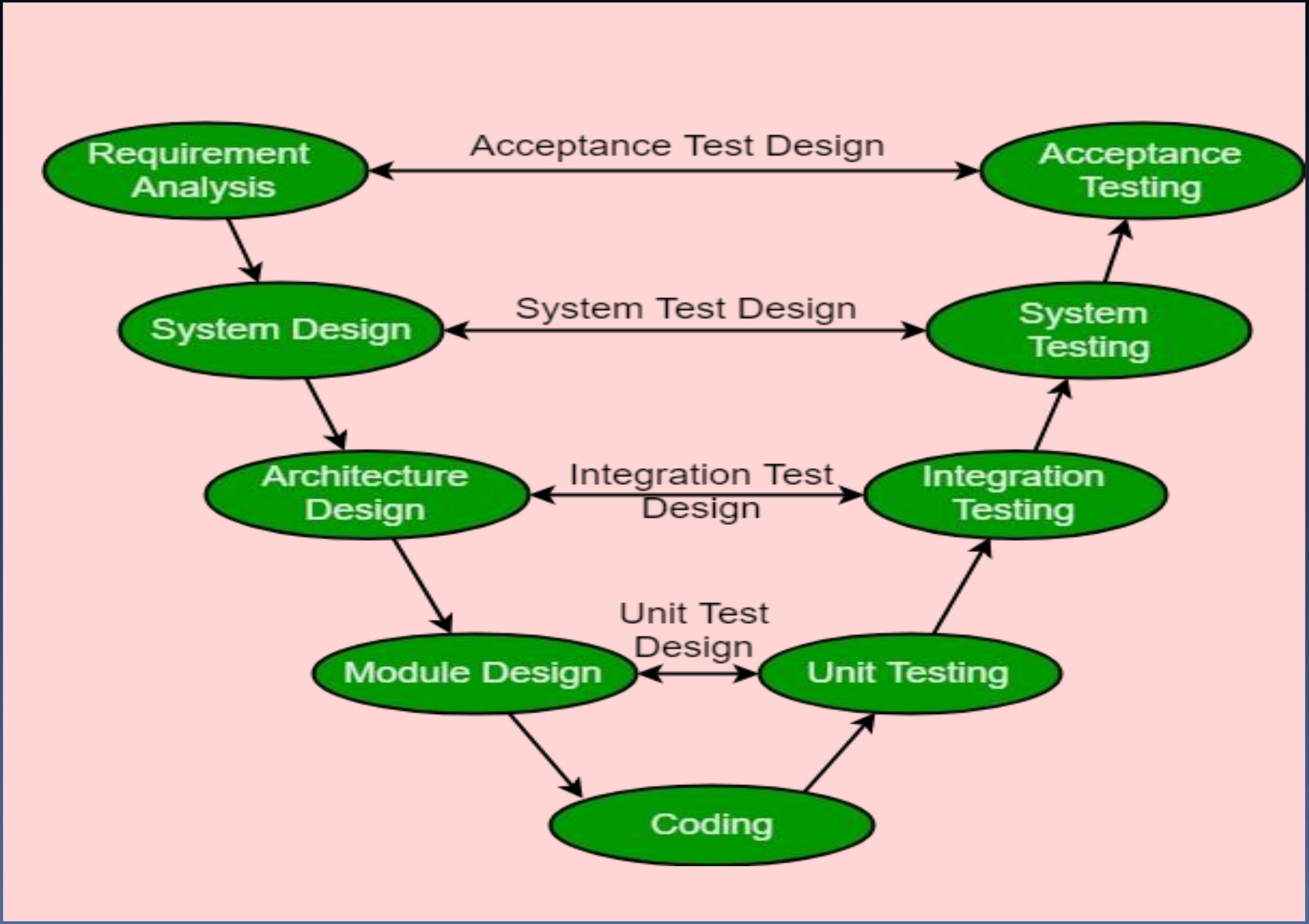
The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as Verification and Validation model.

The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

Verification: It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.

Validation: It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expectations and requirements.

So V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation phases are joined by coding phase in V-shape. Thus it is called V-Model.


Requirement Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding

Design Phase

Requirement Analysis: This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.

System Design: This phase contains the system design and the complete hardware and communication setup for developing product.

Architectural Design: System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.

Module Design: In this phase the system breaks dowm into small modules. The detailed design of modules is specified, also known as Low-Level Design (LLD).

When to use?

Where requirements are clearly defined and fixed.

The V-Model is used when sample technical resources are available with technical expertise.

Advantages:

This is a highly disciplined model and Phases are completed one at a time.

V-Model is used for small projects where project requirements are clear.

Simple and easy to understand and use.

This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product.

It enables project management to track progress accurately.

Disadvantages:

High risk and uncertainty.

It is not a good for complex and object-oriented projects.

It is not suitable for projects where requirements are not clear and contains high risk of changing.

This model does not support iteration of phases.

Big Bang Model

In this model, developers do not follow any specific process. Development begins with the necessary funds and efforts in the form of inputs. And the result may or may not be as per the customer's requirement, because in this model, even the customer requirements are not defined.

This model is ideal for small projects like academic projects or practical projects. One or two developers can work together on this model.

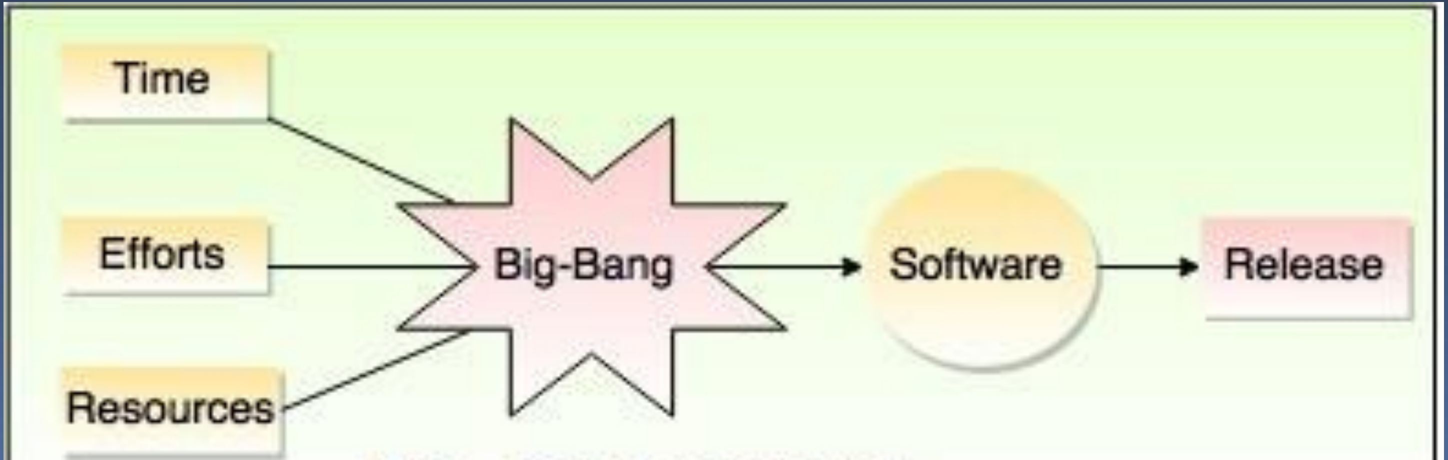


Fig. - Big-Bang Model

When to use Big Bang Model?

As we discussed above, this model is required when this project is small like an academic project or a practical project. This method is also used when the size of the developer team is small and when requirements are not defined, and the release date is not confirmed or given by the customer.

Advantage(Pros) of Big Bang Model:

- There is no planning required.
- Simple Model.
- Few resources required. Easy to manage.
- Flexible for developers.
- It is a good learning aid for new comers or students.

Disadvantage(Cons) of Big Bang Model:

- There are high risk and uncertainty.
- Not acceptable for a large project.
- If requirements are not clear that can cause very expensive.