CHAPTER ONE

Introduction to System Analysis and Design

Overview of Systems and Their Components

A system is an organized collection of components that work together to achieve a specific goal or set of goals. Systems can be found in various domains, including natural, social, and technological environments. In the context of information systems, a system typically refers to a set of interrelated components that collect, process, store, and distribute information to support decision making, coordination, control, analysis, and visualization within an organization.

Key Components of a System:

1. **Inputs:**
    - **Definition:** Inputs are the resources, data, or materials that are put into a system to be processed.
    - **Examples:** Raw data, user commands, energy, materials.
2. **Processes:**
    - **Definition:** Processes are the activities or operations that transform inputs into outputs.
    - **Examples:** Data processing, computation, manufacturing steps, transformation of raw materials.
3. **Outputs:**
    - **Definition:** Outputs are the results produced by the system after processing the inputs.
    - **Examples:** Processed information, finished products, reports, decisions.
4. **Feedback:**
    - **Definition:** Feedback is information about the output of a system that is used to make adjustments or improvements to the inputs or processes.
    - **Examples:** Performance reports, customer feedback, error messages.
5. **Control:**
    - **Definition:** Control involves the mechanisms that monitor and regulate the operation of a system to ensure it achieves its goals.

- **Examples:** Quality control processes, management oversight, automated control systems.

6. **Environment:**
   - **Definition:** The environment encompasses everything outside the system that can influence its operation and performance.
   - **Examples:** External data sources, regulatory constraints, economic conditions.
7. **Boundaries:**
   - **Definition:** Boundaries define the limits of the system and differentiate it from its environment.
   - **Examples:** Organizational boundaries, scope of a project, the perimeter of a manufacturing plant.

Types of Systems:

1. **Open Systems:**
   - **Characteristics:** Interact with their environment by receiving inputs and producing outputs.
   - **Examples:** Businesses, ecosystems, computer systems connected to the internet.
2. **Closed Systems:**
   - **Characteristics:** Do not interact with their environment; all inputs and outputs are contained within the system.
   - **Examples:** A clock, a sealed laboratory experiment.

Information Systems:

Information systems specifically refer to systems designed to manage and process data to provide useful information for decision-making within an organization. They consist of several critical components:

1. **Hardware:**
   - **Definition:** Physical devices and equipment that perform input, processing, storage, and output activities.
   - **Examples:** Computers, servers, peripherals.
2. **Software:**

- **Definition:** Programs and applications that control hardware and process data.
- **Examples:** Operating systems, database management systems, enterprise applications.

3. **Data:**
   - **Definition:** Raw facts and figures that are processed into meaningful information.
   - **Examples:** Customer records, sales transactions, inventory levels.
4. **People:**
   - **Definition:** Individuals who use and manage information systems.
   - **Examples:** IT professionals, end-users, system analysts.
5. **Processes:**
   - **Definition:** Procedures and rules that define how data is collected, processed, and distributed.
   - **Examples:** Business processes, data entry protocols, reporting standards.
6. **Networks:**
   - **Definition:** Communication systems that connect hardware components and allow data sharing.
   - **Examples:** Local Area Networks (LAN), Wide Area Networks (WAN), the internet.

Introduction to System Development Life Cycle (SDLC)

The System Development Life Cycle (SDLC) is a structured approach used for developing information systems. It provides a systematic process for planning, creating, testing, and deploying information systems, ensuring that high-quality systems are delivered that meet or exceed customer expectations. The SDLC consists of distinct phases, each with specific tasks and deliverables. Understanding these phases helps in managing the complexity of system development projects and improving project success rates.

Phases of the SDLC

1. **Planning:**
   - **Purpose:** To define the scope, objectives, and feasibility of the project.
   - **Activities:** Project initiation, feasibility analysis, project scheduling, resource allocation.

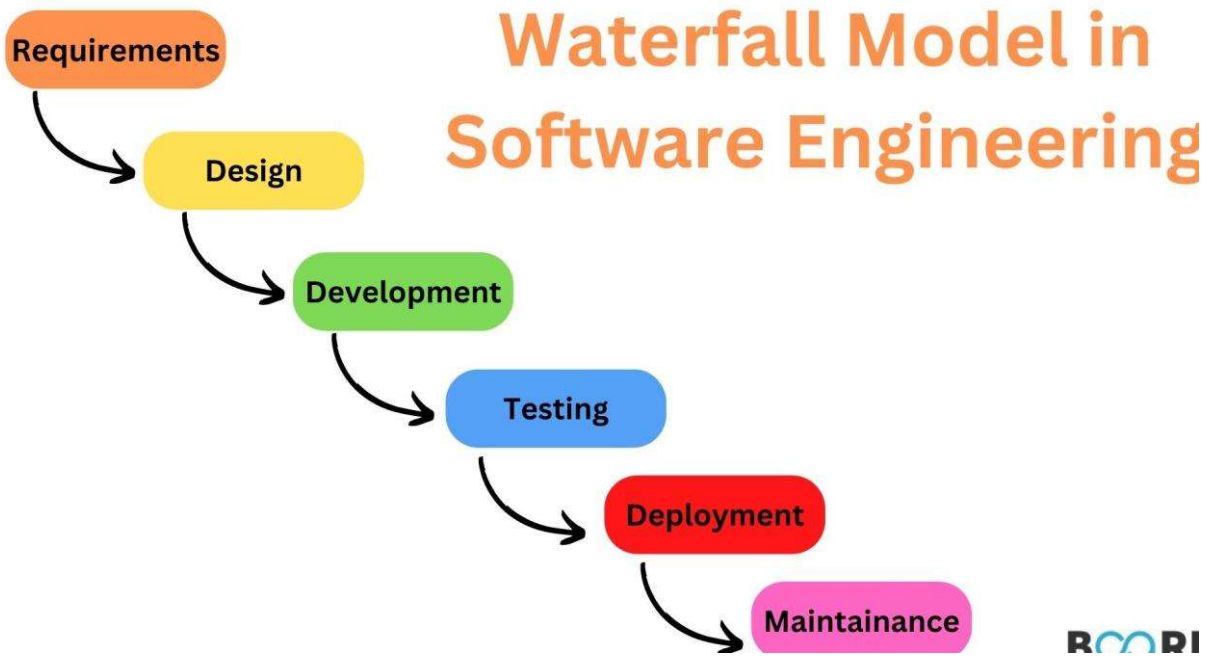
- **Deliverables:** Project charter, feasibility study report, project plan.

2. **Analysis:**
   - **Purpose:** To gather detailed requirements and analyze business needs.
   - **Activities:** Requirement gathering (interviews, surveys, document analysis), requirement analysis, requirement documentation.
   - **Deliverables:** System requirements specification (SRS), use cases, process diagrams.
3. **Design:**
   - **Purpose:** To create detailed system designs based on requirements gathered.
   - **Activities:** System architecture design, database design, interface design, specification of system components.
   - **Deliverables:** Design documents, data models, UI/UX prototypes.
4. **Implementation (Development):**
   - **Purpose:** To build and develop the system based on design specifications.
   - **Activities:** Coding, integration of system components, development of databases, creation of system interfaces.
   - **Deliverables:** Source code, database schema, developed system modules.
5. **Testing:**
   - **Purpose:** To ensure the system works as intended and is free of defects.
   - **Activities:** Unit testing, integration testing, system testing, user acceptance testing (UAT).
   - **Deliverables:** Test plans, test cases, test scripts, defect reports.
6. **Deployment:**
   - **Purpose:** To deliver the system to the users and make it operational.
   - **Activities:** System installation, data migration, user training, deployment planning.
   - **Deliverables:** Deployed system, user manuals, training materials.
7. **Maintenance:**
   - **Purpose:** To monitor, support, and enhance the system post-deployment.
   - **Activities:** Bug fixing, system updates, performance tuning, new feature integration.
   - **Deliverables:** Maintenance reports, system updates, enhancement specifications.

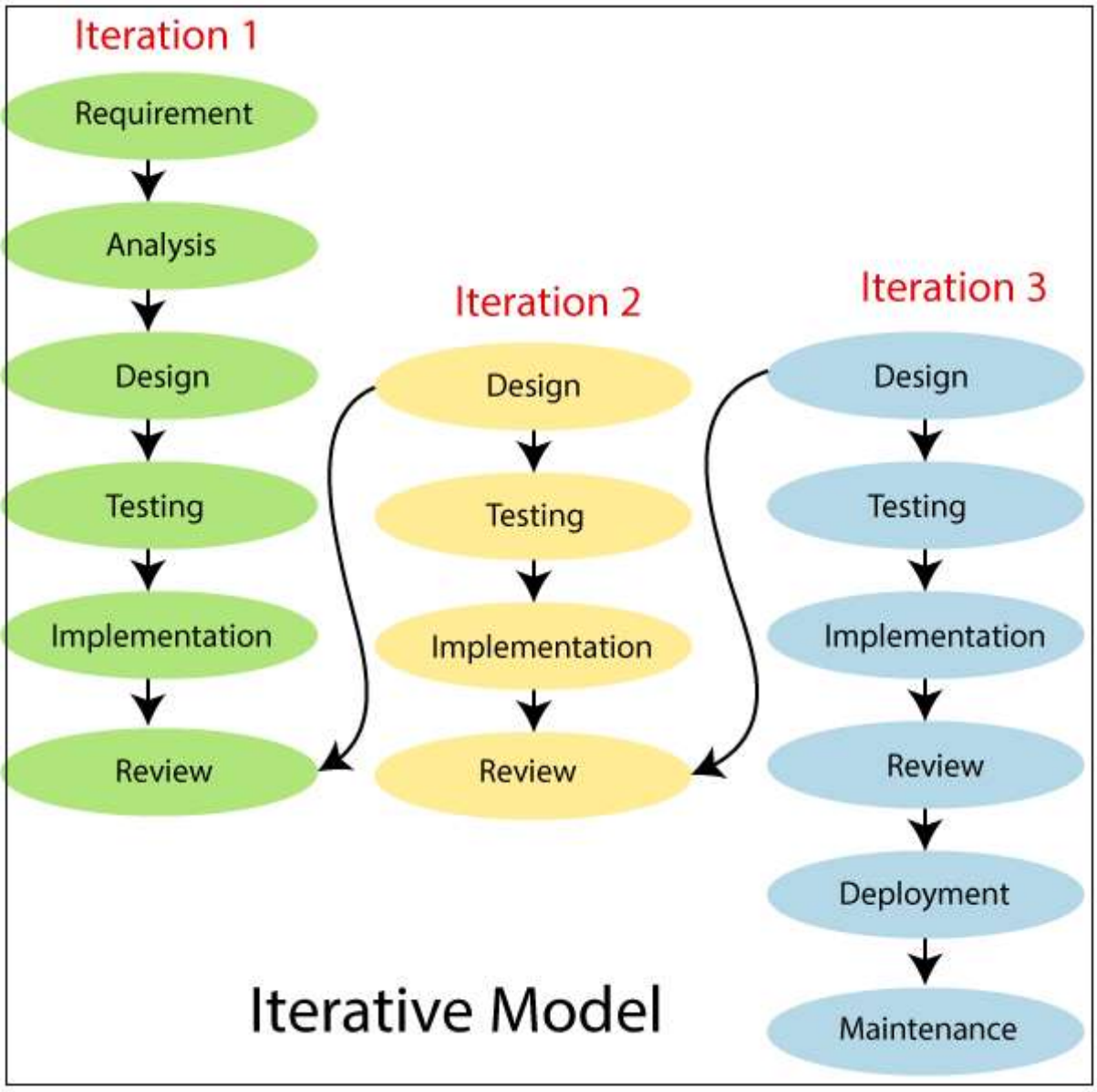
Types of SDLC Models

Different SDLC models have been developed to address various project requirements and constraints. Each model has its strengths and weaknesses, making them suitable for different types of projects.

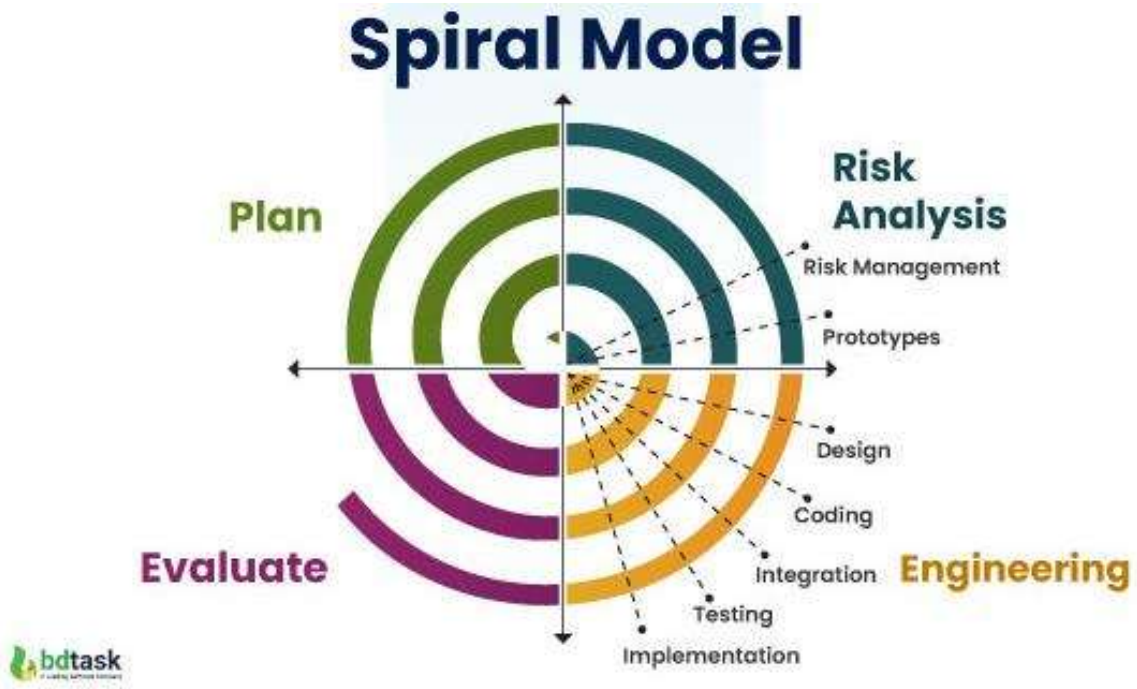
1. **Waterfall Model:**
    - **Description:** A linear and sequential approach where each phase must be completed before the next one begins.
    - **Strengths:** Simple, easy to understand, well-suited for projects with clear requirements.
    - **Weaknesses:** Inflexible, difficult to accommodate changes, high risk if initial requirements are not well-understood.
    - **Use Case:** Suitable for projects with well-defined requirements and low uncertainty.



2. **V-Model (Validation and Verification Model):**
    - **Description:** An extension of the Waterfall model that includes corresponding testing phases for each development stage.



4. **Spiral Model:**
    - **Description:** Combines iterative development with risk assessment. Each iteration involves planning, risk analysis, engineering, and evaluation.
    - **Strengths:** Focuses on risk management, iterative refinement, and client feedback.
    - **Weaknesses:** Can be complex to manage, requires significant risk assessment expertise.
    - **Use Case:** Suitable for large, high-risk projects where risk management is a priority.



5. **Agile Model:**
    - **Description:** Emphasizes flexibility, collaboration, and customer feedback. Uses iterative cycles called sprints to develop system increments.
    - **Strengths:** Highly flexible, adaptive to changes, focuses on customer satisfaction.
    - **Weaknesses:** Requires strong team collaboration, can be challenging in fixed-budget projects.
    - **Use Case:** Suitable for dynamic projects where requirements are expected to change frequently.

Prototyping Model

Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered. It offers a small-scale facsimile of the end product and is used for obtaining customer feedback.

In this process model, the system is partially implemented before or during the analysis phase thereby allowing the customers to see the product early in the life cycle. The process starts by interviewing the customers and developing the incomplete high-level paper model. This document is used to build the initial prototype supporting only the basic functionality as desired by the customer. Once the customer figures out the problems, the prototype is further refined to eliminate them. The process continues until the user approves the prototype and finds the working model to be satisfactory.

Steps of Prototyping Model

Step 1: Requirement Gathering and Analysis: This is the initial step in designing a prototype model. In this phase, users are asked about what they expect or what they want from the system.

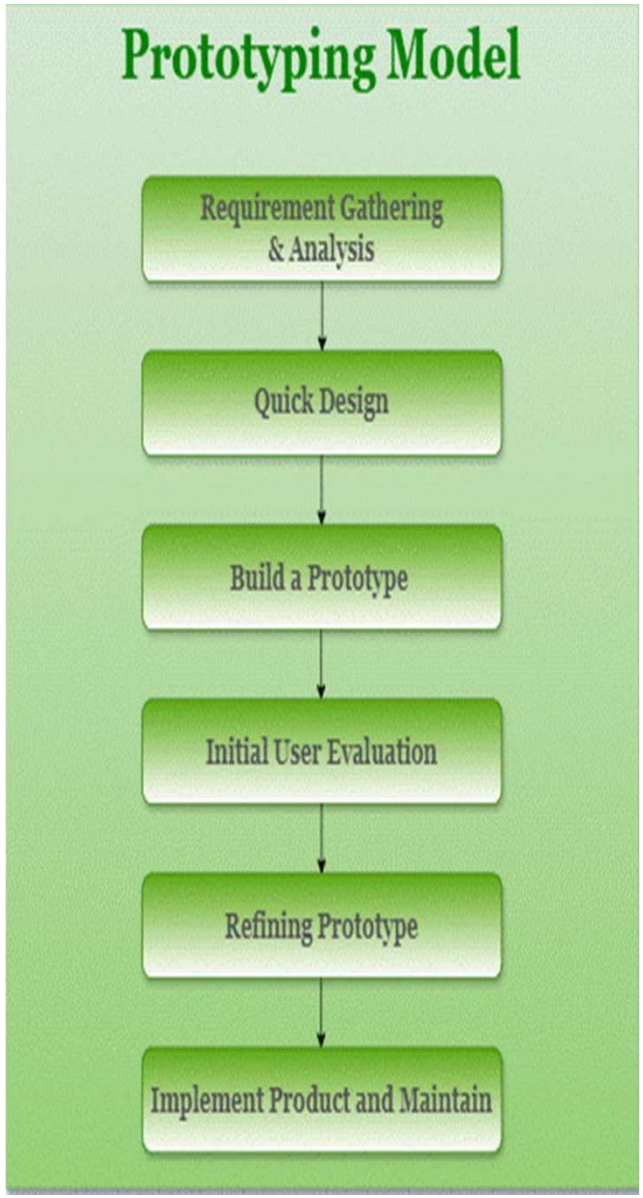
Step 2: Quick Design: This is the second step in the Prototyping Model. This model covers the basic design of the requirement through which a quick overview can be easily described.

Step 3: Build a Prototype: This step helps in building an actual prototype from the knowledge gained from prototype design.

Step 4: Initial User Evaluation: This step describes the preliminary testing where the investigation of the performance model occurs, as the customer will tell the strengths and weaknesses of the design, which was sent to the developer.

Step 5: Refining Prototype: If any feedback is given by the user, then improving the client's response to feedback and suggestions, the final system is approved.

Step 6: Implement Product and Maintain: This is the final step in the phase of the Prototyping Model where the final system is tested and distributed to production, here the program is run regularly to prevent failures.


Prototyping Model
Requirement Gathering & Analysis
Quick Design
Build a Prototype
Initial User Evaluation
Refining Prototype
Implement Product and Maintain

Advantages of Prototyping Model

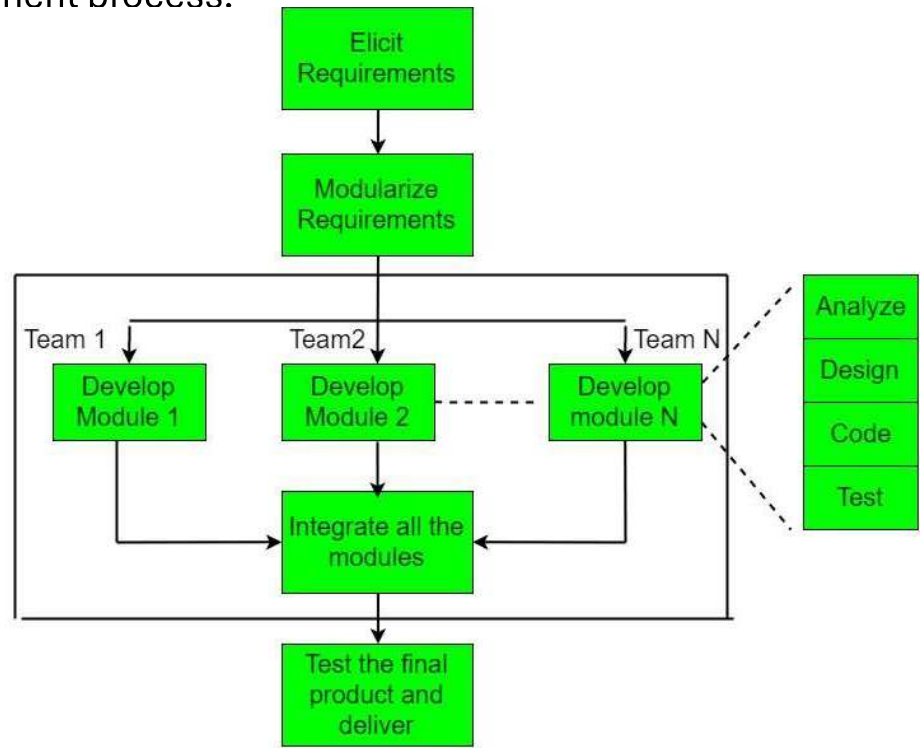
•The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
•New requirements can be easily accommodated as there is scope for refinement.
•Missing functionalities can be easily figured out.
•Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
•The developed prototype can be reused by the developer for more complicated projects in the future.
•Flexibility in design.

Disadvantages of the Prototyping Model

•Costly concerning time as well as money.
•There may be too much variation in requirements each time the prototype is evaluated by the customer.
•Poor Documentation due to continuously changing customer requirements.
•It is very difficult for developers to accommodate all the changes demanded by the customer.
•There is uncertainty in determining the number of iterations that would be required before the prototype is finally accepted by the customer.
•After seeing an early prototype, the customers sometimes demand the actual product to be delivered soon.

Rapid Application Development Model (RAD)

The RAD model or Rapid Application Development model is a type of software development methodology that emphasizes quick and iterative release cycles, primarily focusing on delivering working software in shorter timelines. Unlike traditional models such as the Waterfall model, RAD is designed to be more flexible and responsive to user feedback and changing requirements throughout the development process.



This model consists of 4 basic phases:

1.Requirements Planning – This involves the use of various techniques used in requirements elicitation like brainstorming, task analysis, form analysis, user scenarios, FAST (Facilitated Application Development Technique), etc. It also consists of the entire structured plan describing the critical data, methods to obtain it, and then processing it to form a final refined model.

2.User Description – This phase consists of taking user feedback and building the prototype using developer tools. In other words, it includes re-examination and validation of the data collected in the first phase. The dataset attributes are also identified and elucidated in this phase.

3.Construction – In this phase, refinement of the prototype and delivery takes place. It includes the actual use of powerful automated tools to transform processes and data models into the final working product. All the required modifications and enhancements are to be done in this phase.

4.Cutover – All the interfaces between the independent modules developed by separate teams have to be tested properly. The use of powerfully automated tools and subparts makes testing easier. This is followed by acceptance testing by the user.

Advantages of Rapid Application Development Model (RAD)

- The use of reusable components helps to reduce the cycle time of the project.
- Feedback from the customer is available at the initial stages.
- Reduced costs as fewer developers are required.
- The use of powerful development tools results in better quality products in comparatively shorter periods.
- The progress and development of the project can be measured through the various stages.
- It is easier to accommodate changing requirements due to the short iteration time spans.

Disadvantages of Rapid application development model (RAD)

- The use of powerful and efficient tools requires highly skilled professionals.
- The absence of reusable components can lead to the failure of the project.
- The team leader must work closely with the developers and customers to close the project on time.
- The systems which cannot be modularized suitably cannot use this model.
- Customer involvement is required throughout the life cycle.

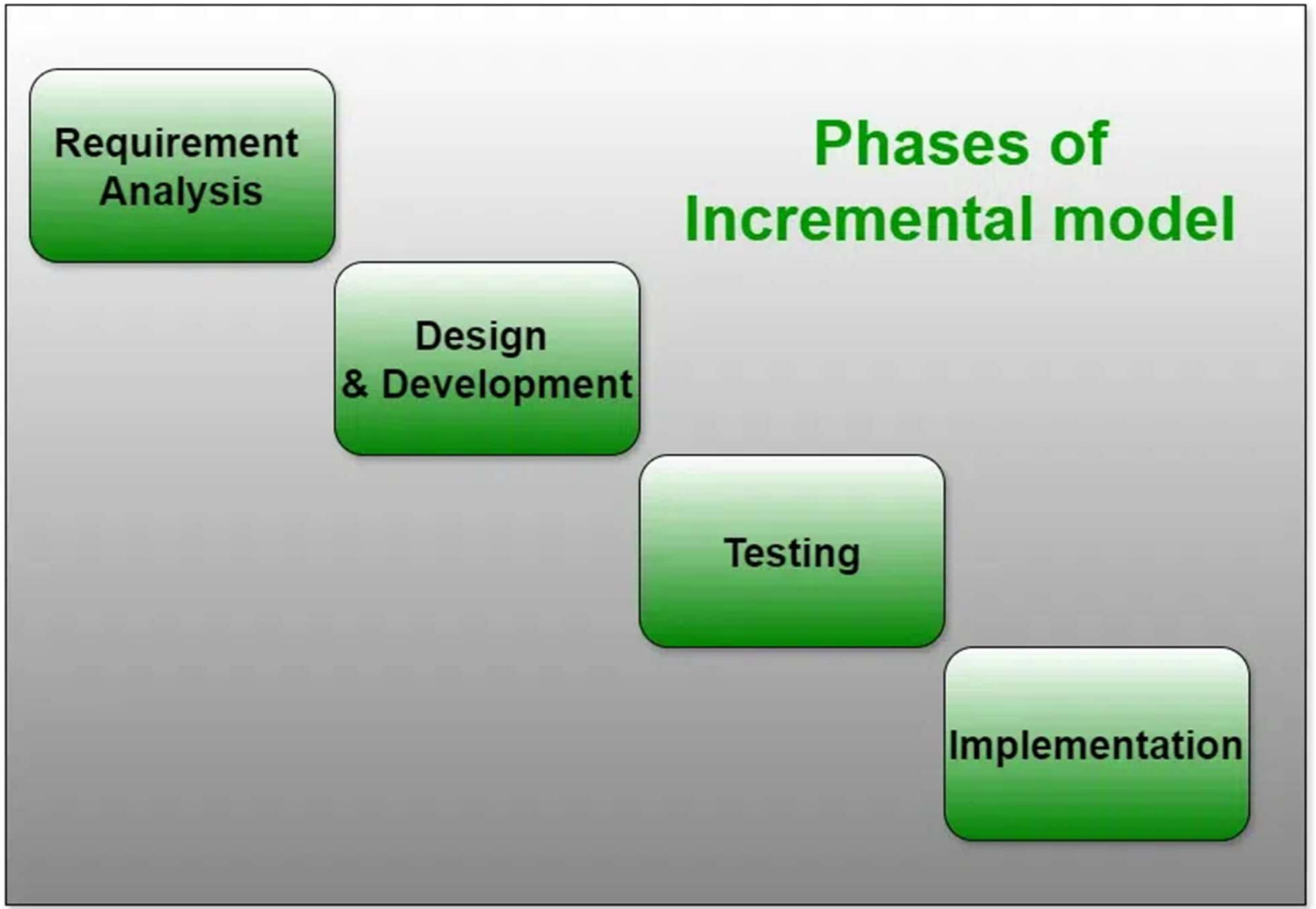
Incremental Process Model

The **Incremental Process Model** is also known as the Successive version model. This article focuses on discussing the Incremental Process Model in detail.

First, a simple working system implementing only a few basic features is built and then that is delivered to the customer. Then thereafter many successive iterations/ versions are implemented and delivered to the customer until the desired system is released.

Phases of incremental model

Requirements of Software are first broken down into several modules that can be incrementally constructed and delivered.


Requirement Analysis
Phases of Incremental model
Design & Development
Testing
Implementation

1.Requirement analysis: In Requirement Analysis At any time, the plan is made just for the next increment and not for any kind of long-term plan. Therefore, it is easier to modify the version as per the needs of the customer.
2.Design & Development: At any time, the plan is made just for the next increment and not for any kind of long-term plan. Therefore, it is easier to modify the version as per the needs of the customer. The Development Team first undertakes to develop core features (these do not need services from other features) of the system. Once the core features are fully developed, then these are refined to increase levels of capabilities by adding new functions in Successive versions. Each incremental version is usually developed using an iterative waterfall model of development.
3.Deployment and Testing: After Requirements gathering and specification, requirements are then split into several different versions starting with version 1, in each successive increment, the next version is constructed and then deployed at the customer site. in development and Testing the product is checked and tested for the actual process of the model.
4.Implementation: In implementation After the last version (version n), it is now deployed at the client site.

Advantages of the Incremental Process Model

1.Prepares the software fast.
2.Clients have a clear idea of the project.
3.Changes are easy to implement.
4.Provides risk handling support, because of its iterations.
5.Adjusting the criteria and scope is flexible and less costly.
6.Comparing this model to others, it is less expensive.
7.The identification of errors is simple.

Disadvantages of the Incremental Process Model

1.A good team and proper planned execution are required.
2.Because of its continuous iterations the cost increases.
3.Issues may arise from the system design if all needs are not gathered upfront throughout the program lifecycle.
4.Every iteration step is distinct and does not flow into the next.
5.It takes a lot of time and effort to fix an issue in one unit if it needs to be corrected in all the units.

Concurrent development model

The concurrent Development Model is a type of software development that enables the different phases of the development life cycle including designing, coding, and testing to be worked on at the same time. This model helps teams to stay fluid and responsive, they can solve problems and adapt to changes with one planning phase needing to finish before the next begins. It is best suited to projects in which delivery to stakeholders, review, and redelivery are frequent, which makes it a crucial approach in contemporary software development.

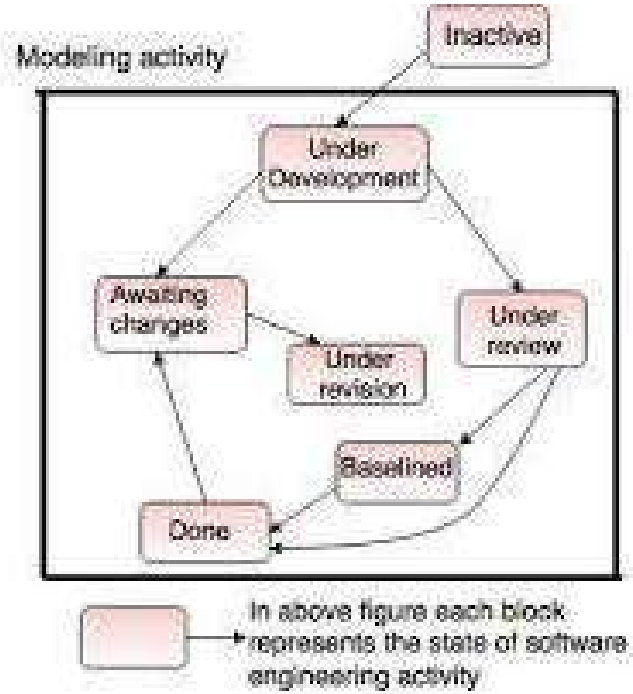


Fig. - One element of the concurrent process model

Pros

•**Improves Understanding of Requirements:** Prototyping stimulates the definition of user needs and expectations in the early stage.

•**User Feedback Driven**: People can touch and give a sort of live input from the design, which makes for a better product at the end of the day.

•**Flexibility:** Modifications are possible at every stage of development if necessary.

•**Risk Reduction:** Lead time identification of the concerns minimizes the chances of failing the project.

•**Enhances Communication:** Prototypes are useful mainly on two fronts; communication among the developers and the user hence, helping to keep the desired goals and objectives in a similar line.

Cons

•**Time-Consuming:** Continual rehearsals and feedback seem to prolong the time of the activity.

•**Increased Costs**: Cumulative alterations and modifications in the documentation can become expensive for the implementors.

•**Scope Creep**: This means that there will be constant changes that stakeholders may ask for hence causing more scope creep.

•**Limited Scalability:** A prototype actuality may not contain a necessary system's scalability, which may result in necessary work redoing.

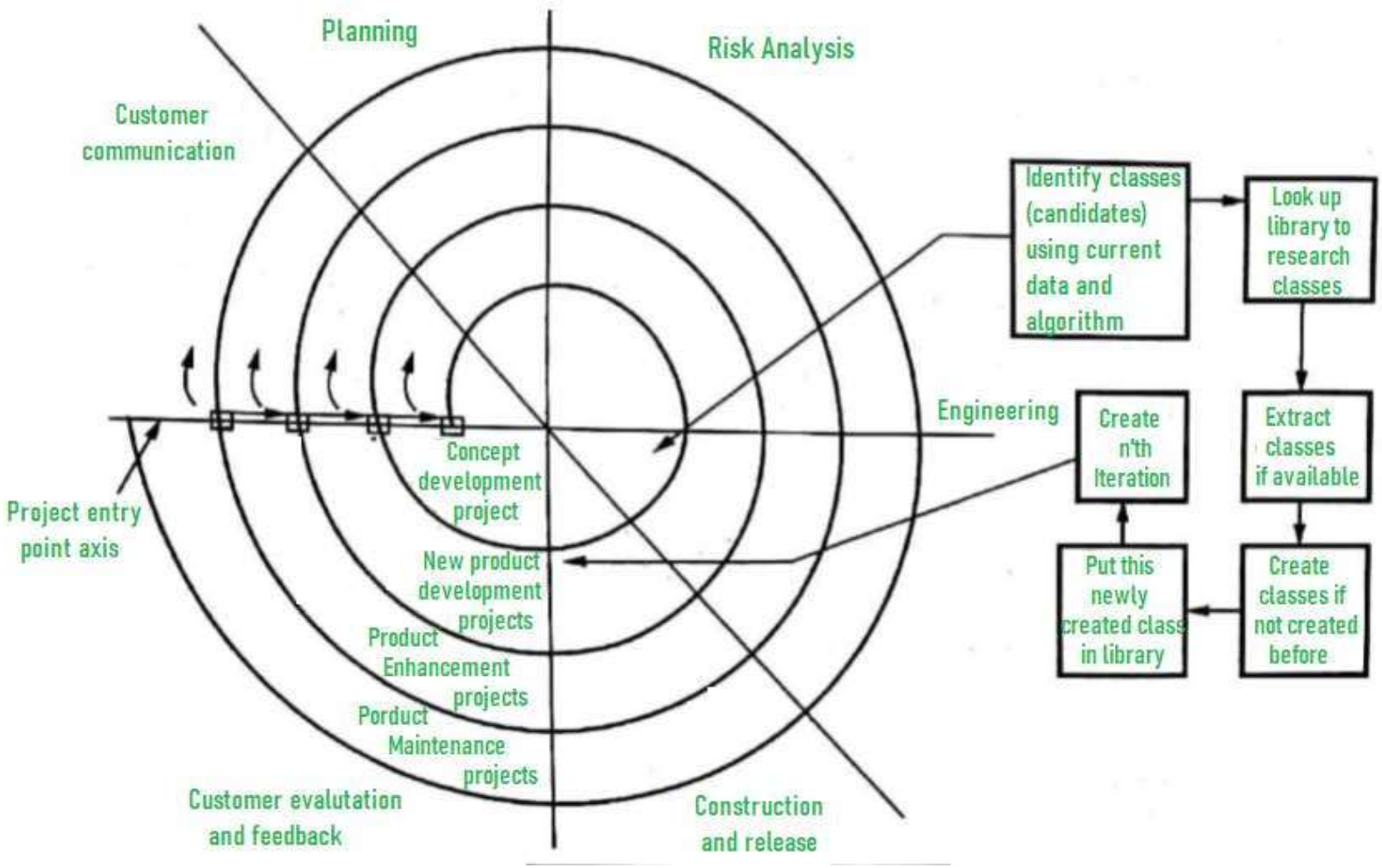
•**Inadequate Documentation:** Emphasis on the physical construction of the prototypes may result in inadequate documentation to support the subsequent maintenance.

Component Based Model (CBM)

The component-based assembly model uses object-oriented technologies. In object-oriented technologies, the emphasis is on the creation of classes. Classes are the entities that encapsulate data and algorithms. In component-based architecture, classes (i.e., components required to build application) can be uses as reusable components. This model uses various characteristics of spiral model. This model is evolutionary by nature.

Characteristics of Component Assembly Model:

- Uses object-oriented technology.
- Components and classes encapsulate both data and algorithms.
- Components are developed to be reusable.
- Paradigm similar to spiral model, but engineering activity involves components.
- The system produced by assembling the correct components.


Planning
Risk Analysis
Customer communication
Engineering
Concept development project
New product development projects
Product Enhancement projects
Porduct Maintenance projects
Project entry point axis
Customer evalutation and feedback
Construction and release
Identify classes (candidates) using current data and algorithm
Look up library to research classes
Extract classes if available
Create classes if not created before
Put this newly created class in library
Create n'th Iteration