Unit 1: Systems Development Environment (6 LHs)

By: Kabita Dhital

BIM(5TH Semester)

Course Contains

- Teaching Strategies
  - *Interactive* Class Discussion
  - PowerPoint /printed notes
  - Team exercises

What is a System?

- The word System is derived from Greek word Systema, which means an organized relationship between any set of components to achieve some common cause or objective.
- A system is “an orderly grouping of interdependent components linked together according to a plan to achieve a specific goal.
- System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives.

Constraints of a System

- **Constraints of a System** are the **limitations or restrictions** under which a system must operate.

A system must have three basic constraints:-

1. A system must have some **structure and behavior** which is designed to achieve a predefined objective. Every system must have a **well-defined structure** and **specific behavior**. **Structure** refers to the components or parts of the system and how they are organized. **Behavior** refers to how the system functions, responds to inputs, and produces outputs. **Example:** In a computer system, hardware components (CPU, memory, input/output devices) form the structure, while data processing is the behavior. The objective is to process information efficiently.
2. **Interconnectivity** and **interdependence** must exist among the system components. A system is not just a collection of independent parts. All components must be **connected** and **dependent on one another**. **Interconnectivity** means components are linked and can communicate. **Interdependence** means the functioning of one component affects others. **Example:** In a banking system, customer service, database, and transaction modules are interconnected. If the database fails, transactions cannot be completed.
3. The **objectives of the organization** have a **higher priority** than the objectives of its subsystems. **Example:** In an organization, individual departments (finance, HR, IT) may have their own goals, but they must prioritize the organization's overall mission and strategy.

Properties of a System

- A system has the following properties −

Organization

- Organization implies structure and order. It is the arrangement of components that helps to achieve predetermined objectives.

Interaction

- It is defined by the manner in which the components operate with each other.
- For example, in an organization, purchasing department must interact with production department and payroll with personnel department.

Interdependence

- Interdependence means how the components of a system depend on one another. For proper functioning, the components are coordinated and linked together according to a specified plan. The output of one subsystem is the required by other subsystem as input.

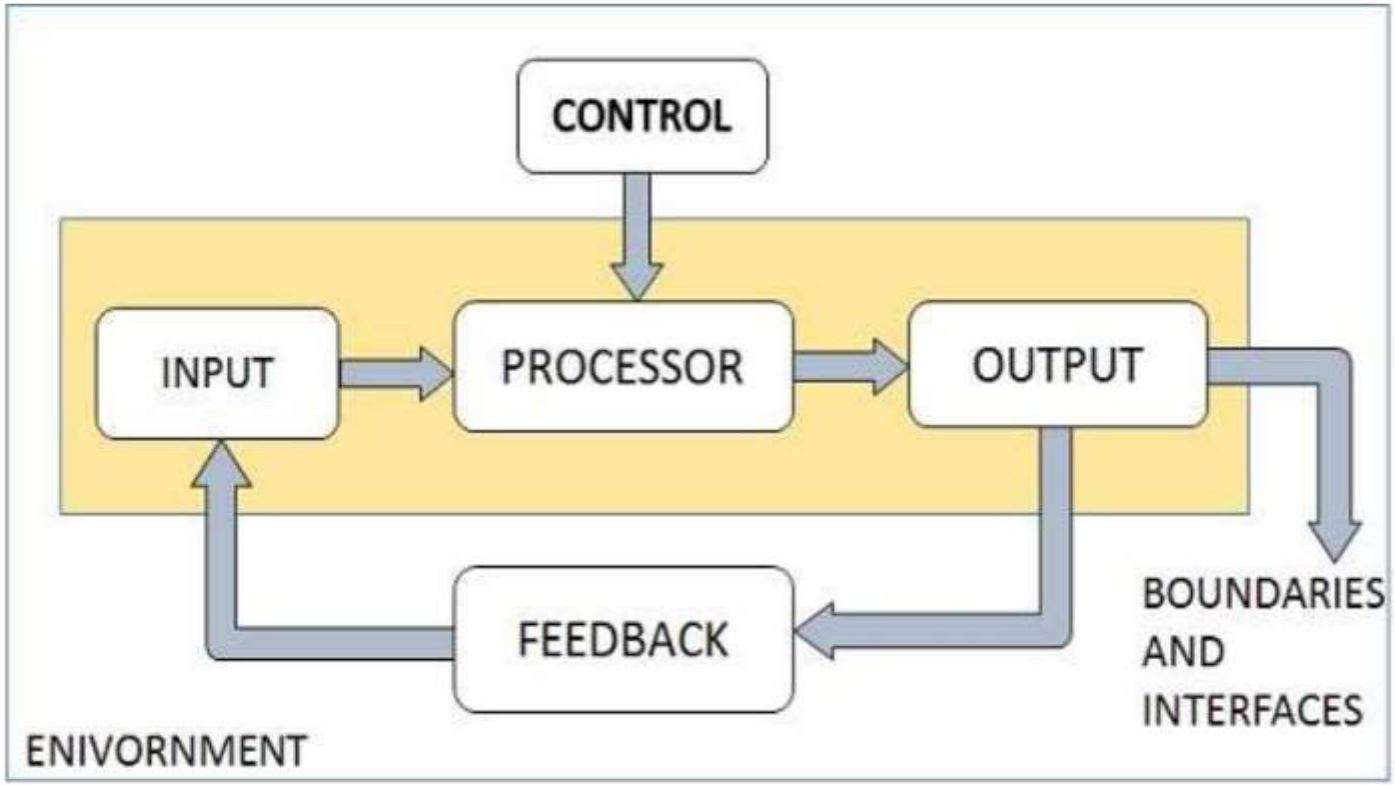
Integration

- Integration is concerned with how a system components are connected together. It means that the parts of the system work together within the system even if each part performs a unique function.

Central Objective

- The objective of system must be central. It may be real or stated. It is not uncommon for an organization to state an objective and operate to achieve another.
- The users must know the main objective of a computer application early in the analysis for a successful design and conversion.

Elements of a System



1.Outputs and Inputs

- The main aim of a system is to produce an output which is useful for its user.
- Inputs are the information that enters into the system for processing.
- Output is the outcome of processing.

2.Processor(s)

- The processor is the element of a system that involves the actual transformation of input into output.
- It is the operational component of a system. Processors may modify the input either totally or partially, depending on the output specification.
- As the output specifications change, so does the processing. In some cases, input is also modified to enable the processor for handling the transformation.

3.Control

- The control element guides the system.
- It is the decision–making subsystem that controls the pattern of activities governing input, processing, and output.
- The behavior of a computer System is controlled by the Operating System and software. In order to keep system in balance, what and how much input is needed is determined by Output Specifications.

4.Feedback

- Feedback provides the control in a dynamic system.
- Positive feedback is routine in nature that encourages the performance of the system.
- Negative feedback is informational in nature that provides the controller with information for action.

5.Environment

- The environment is the “supersystem” within which an organization operates.
- It is the source of external elements that strike on the system.
- It determines how a system must function. For example, vendors and competitors of organization’s environment, may provide constraints that affect the actual performance of the business.

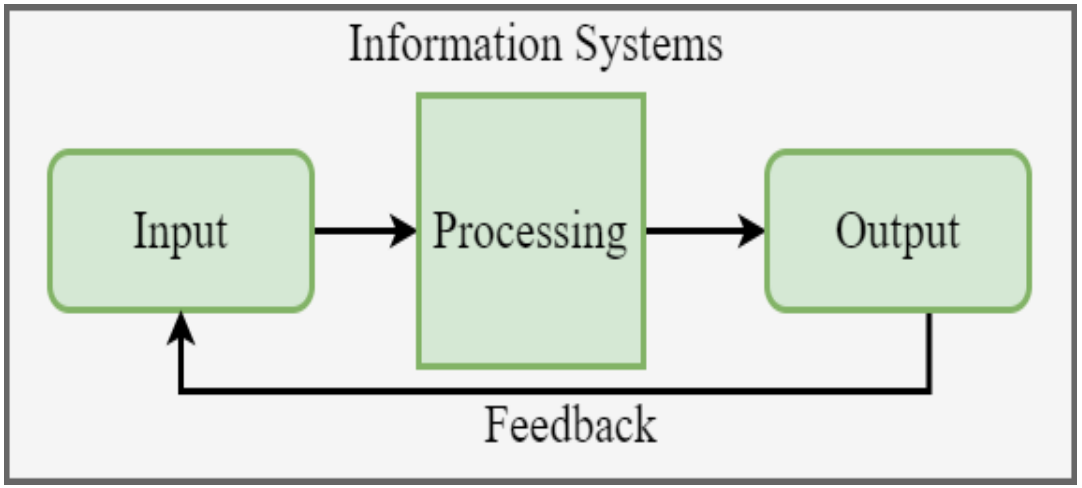
6.Boundaries and Interface

- A system should be defined by its boundaries. Boundaries are the limits that identify its components, processes, and interrelationship when it interfaces with another system.
- Each system has boundaries that determine its sphere of influence and control.
- The knowledge of the boundaries of a given system is crucial in determining the nature of its interface with other systems for successful design.

DEFINITION OF INFORMATION SYSTEM

" An information system is a set of interrelated components that works together to collect, process, store and breakdown the information to support decision making."

- Information system is a collection of hardware, software, infrastructure and trained personnel which are going to do easy planning to make reliable infrastructure, control, coordination between software and hardware and decision making in an organization



- An information system is a way to work with information using computers and other technology.
- It combines different parts like computer programs, physical devices, and networks.
- Businesses use information systems to collect important data. They use this data to run their operations smoothly.
- Information systems also help businesses talk to their customers.
- Using information systems makes businesses work better than their competitors.
- **Core Components of an IS**
- **Hardware:** Physical devices like computers, servers, phones, and networks.
- **Software:** Applications and programs that process data (e.g., operating systems, databases).
- **Data:** Raw facts and figures collected and stored.
- **People:** Users, managers, and IT professionals who interact with the system.
- **Processes/Procedures:** Rules, methods, and workflows for data handling.

Characteristics of Information system

1. Organization

- Organization implies structure and order. It is the arrangement of components that helps to achieve predetermined objectives.

2. Interaction

- It is defined by the manner in which the components operate with each other.
- For example, in an organization, purchasing department must interact with production department and payroll with personnel department.

3. Interdependence

- Interdependence means how the components of a system depend on one another. For proper functioning, the components are coordinated and linked together according to a specified plan. The output of one subsystem is the required by other subsystem as input.

4. Integration

- Integration is concerned with how a system components are connected together. It means that the parts of the system work together within the system even if each part performs a unique function.

5. Central Objective

- The objective of system must be central. It may be real or stated. It is not uncommon for an organization to state an objective and operate to achieve another.
- The users must know the main objective of a computer application early in the analysis for a successful design and conversion.

Types of information system:

- Transaction Processing Systems (TPS)

✓ Transaction processing systems are used to record day to day business transactions of the organization. They are used by users at the operational management level.

✓ Automate handling of data about business activities (transactions)

✓ Process orientation

✓ Example: Point of sales System, Payroll System etc.

- Management Information Systems (MIS)

✓ Converts raw data from transaction processing system into meaningful form

✓ used by tactical managers to monitor the organization's current performance status.

✓ Data orientation

✓ Example: HRMS,Sales Management System

Types of information system:

- **Decision Support Systems (DSS)**

✓ Decision support systems are used by senior management to make non-routine decisions. Decision support systems use input from internal systems (transaction processing systems and management information systems) and external systems.

✓ Designed to help decision makers

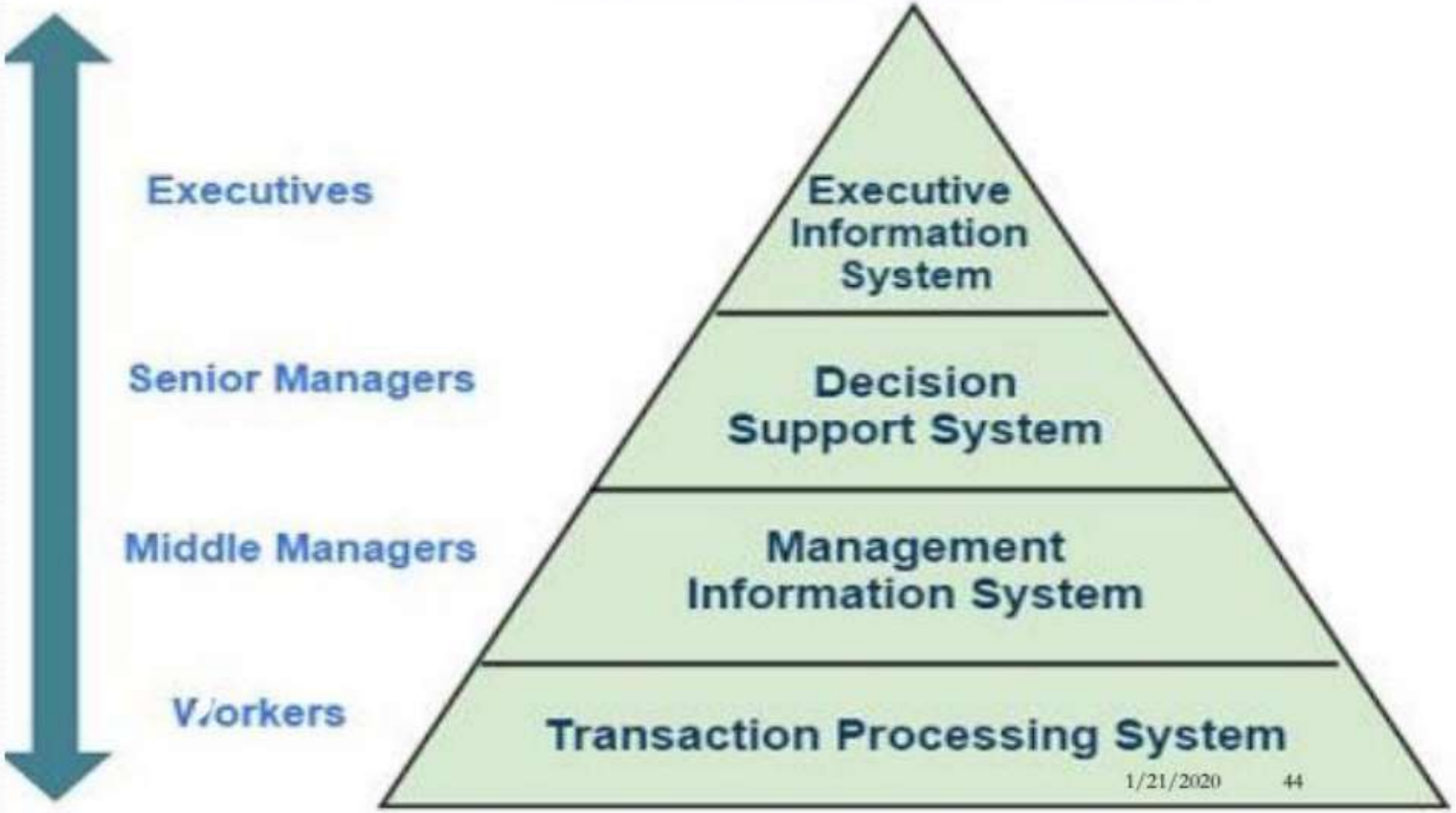
✓ Provides interactive environment for decision making

✓ Involves data warehouses, executive information systems (EIS)

✓ Database, model base, user dialogue

✓ Example: Financial Planning System, Bank Loan Management System

Information Systems



Heart of Systems Development Process

- The heart of the system development process (SDLC) involves the core activities of Analysis, Design, and Implementation (Coding), transforming a concept into a working system, but modern views emphasize that Testing is equally crucial, ensuring quality throughout, while User Involvement (understanding needs) is the guiding principle, making the entire lifecycle—Planning, Analysis, Design, Development, Testing, Implementation, Maintenance.
- They bridge the gap from idea (analysis) to reality (implementation).
- Implementation turns design into a tangible product
- **Analysis:** Understanding user needs, problems, and requirements to define what the system must do (Feasibility & Requirements).
- **Design:** Creating the blueprint, defining system architecture, interfaces, and data structures.
- **Implementation (Development/Coding):** Writing the actual code, building the system, and integrating modules.

The Systems Development Environment Introduction

Information Systems Analysis and Design (ISAD)

- Information Systems Analysis and Design (ISAD) is the structured process of understanding organizational needs, defining requirements, and creating/improving IT systems to solve problems, streamline operations, and meet business goals.
- A structured approach to information system development is a systematic and organized process of developing an information system using a defined set of steps or phases.
- Used to develop and maintain computer-based information systems.
- Used by a team of business and systems professionals.
- A structured approach to information system development is a systematic and organized process of developing an information system using a defined set of steps or phases.
- A structured approach to information system development is designed to ensure that the development process is efficient, effective, and produces a system that meets the needs of the organization

The Systems Development Environment Introduction

Information Systems Analysis and Design (ISAD)

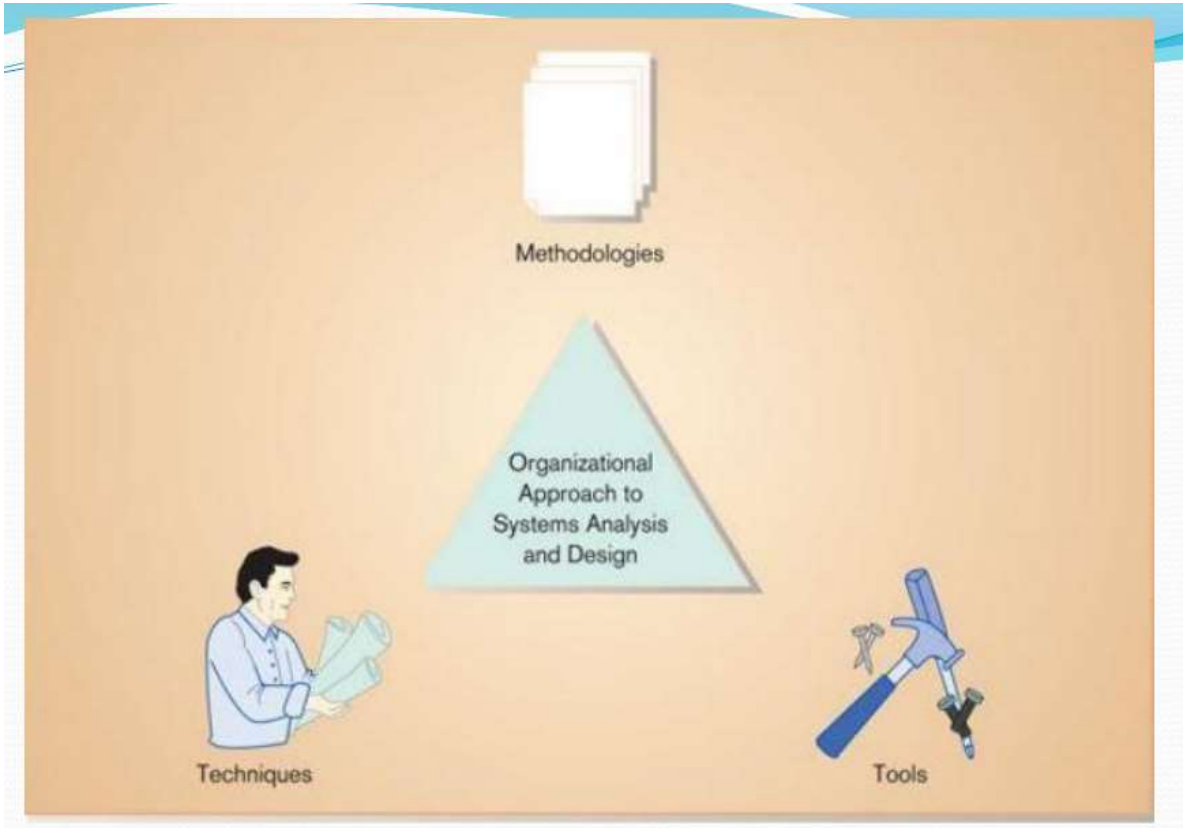


Figure 1-1 An organizational approach to systems analysis and design is driven by methodologies, techniques, and tools

- An output of system analysis and design is application software.
- Methodologies are comprehensive, multiple-step approaches to system development.
- It involves following a specific methodology or framework that outlines the activities, deliverables, and outcomes at each stage of the development process.
- Methodologies uses different techniques.
- Techniques are particular processes that you as an analyst, will follow to ensure your work is well thought out, complete, and comprehensive to others on your project team. Eg: conducting interviews to determine what your system should do, diagramming the system's logic, designing the reports your system will generate.
- Tools are computer programs that make it easy to use and benefit from techniques.

Modern Approach to Systems Analysis and Design

- 1950s: All applications had to be developed in machine language or assembly language. They had to be developed from scratch because due to the absence of software industry
- 1960s: Smaller, faster, less expensive computers, beginning of the software industry, use in-house development.
- 1970s: Realized how expensive to develop customized information system for every application , started development of database management system.
- 1980s: The software industry expended greatly, CASE(computer aided software engineering) tools.
- Started writing application software in oop languages, graphics were used, developed less software in-house and bought more from software vendors.
- 1990s: focus on system integration, GUI(Graphical user interface) applications, client/server platforms, Internet.
- The new century: Web application development, wireless PDAs (personal digital assistants, eg pocket PCs), ASP(application service provider).

- Systems Analyst -A system analyst is a person responsible for the development of software and hardware solution to the efficient working of the organization. Analysts study the environment and problems of an organization to determine whether a new information method can provide solution to the problem. The main job of system analyst is to provide right type of information, in right quantity at the right time in post effective manner to the management or the end user.

Roles of System Analyst:

- Defining IT requirements of organization
- Gathering Data/Facts
- Analyzing the problem
- Setting priority amongst requirements
- Problem solving
- Drawing Specification
- Designing System
- Evaluating System

Systems Development Life Cycle (SDLC)

Phases in SDLC:

- Planning
- Analysis
- Design
- Implementation
- Maintenance

Feasibility Study or Planning

- Define the problem and scope of existing system.
- Overview the new system and determine its objectives.
- Confirm project feasibility and produce the project Schedule.
- During this phase, threats, constraints, integration and security of system are also considered.
- A feasibility report for the entire project is created at the end of this phase.

Analysis and Specification

- Gather, analyze, and validate the information.
- Define the requirements and prototypes for new system.
- Evaluate the alternatives and prioritize the requirements.
- Examine the information needs of end-user and enhances the system goal.
- A Software Requirement Specification (SRS) document, which specifies the software, hardware, functional, and network requirements of the system is prepared at the end of this phase.

Feasibility Study or Planning

- Define the problem and scope of existing system.
- Overview the new system and determine its objectives.
- Confirm project feasibility and produce the project Schedule.
- During this phase, threats, constraints, integration and security of system are also considered.
- A feasibility report for the entire project is created at the end of this phase.

Analysis and Specification

- Gather, analyze, and validate the information.
- Define the requirements and prototypes for new system.
- Evaluate the alternatives and prioritize the requirements.
- Examine the information needs of end-user and enhances the system goal.
- A Software Requirement Specification (SRS) document, which specifies the software, hardware, functional, and network requirements of the system is prepared at the end of this phase.

System Design

- ✓ Includes the design of application, network, databases, user interfaces, and system interfaces.
- ✓ Transform the SRS document into logical structure, which contains detailed and complete set of specifications that can be implemented in a programming language.
- ✓ Create a contingency, training, maintenance, and operation plan.
- ✓ Review the proposed design. Ensure that the final.

Implementation

- ✓ Implement the design into source code through coding.
- ✓ Combine all the modules together into training environment that detects errors and defects.
- ✓ A test report which contains errors is prepared through test plan that includes test related tasks such as test case generation, testing criteria, and resource allocation for testing.
- ✓ Integrate the information system into its environment and install the new system..

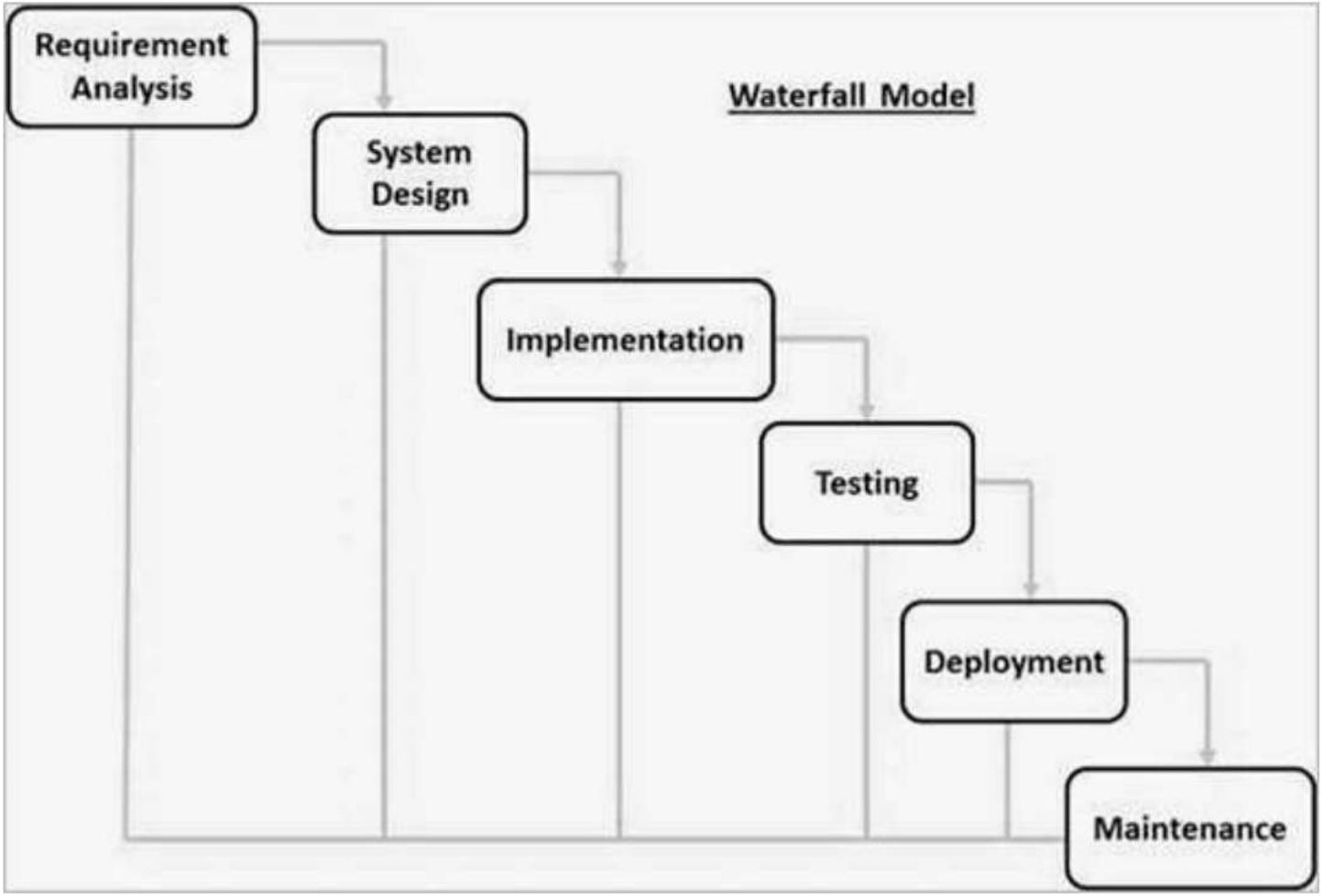
Maintenance/Support

- ✓ Include all the activities such as phone support or physical on-site support for users that is required once the system is installing.
- ✓ Implement the changes that software might undergo over a period of time, or implement any new requirements after the software is deployed at the customer location.
- ✓ It also includes handling the residual errors and resolve any issues that may exist in the system even after the testing phase.
- ✓ Maintenance and support may be needed for a longer time for large systems and for a short time for smaller systems.

Waterfall Model/Traditional Model

- Waterfall is the oldest and most straightforward of the structured SDLC methodologies — finish one phase, then move on to the next.
- No going back. Each stage relies on information from the previous stage and has its own project plan.
- Waterfall is easy to understand and simple to manage.
- But early delays can throw off the entire project timeline.
- And since there is little room for revisions once a stage is completed, problems can't be fixed until you get to the maintenance stage.
- This model doesn't work well if flexibility is needed or if the project is long term and ongoing.

Waterfall Model



Sequential phases in 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.
- **System Design** – The requirement specifications 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, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed 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.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the 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.

- The Waterfall Model is the simple and classical model of all the model we have.
- This model is also known as linear sequential model.
- This model is theoretically model not a practical model.
- In this Model each and every phase must be completed before moving to the next phase.
- This model is suitable for the small project where the technical issues are very clear.

Advantage

- Some of the major advantages of the Waterfall Model are as follows
  - Simple and easy to understand and use
  - Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
  - Phases are processed and completed one at a time.
  - Works well for smaller projects where requirements are very well understood.
  - Clearly defined stages.
  - Well understood milestones.
  - Easy to arrange tasks.
  - Process and results are well documented

Disadvantages

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

Different Approaches to Improving Development

1. CASE TOOLS
2. Rapid Application Development
3. Service-Oriented Architecture
4. Agile Methodologies
5. extreme Programming
6. Object-Oriented Analysis And Design

CASE TOOLS

- CASE stands for Computer Aided Software Engineering. It means, development and maintenance of software projects with help of various automated software tools.
- CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.
- CASE tools can be used to help in multiple phases of the SDLC: Project Identification and Selection, analysis, design, and implementation and maintenance.
- There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools
- Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

The general types of CASE tools are listed below:

- Central Repository - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.
- Upper Case Tools - Upper CASE tools are used in planning, analysis and design stages of SDLC.
- Lower Case Tools - Lower CASE tools are used in implementation, testing and maintenance.
- Integrated Case Tools - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation. CASE tools can be grouped together if they have similar functionality, process activities and capability of getting integrated with other tools.

- Diagram tools : These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.

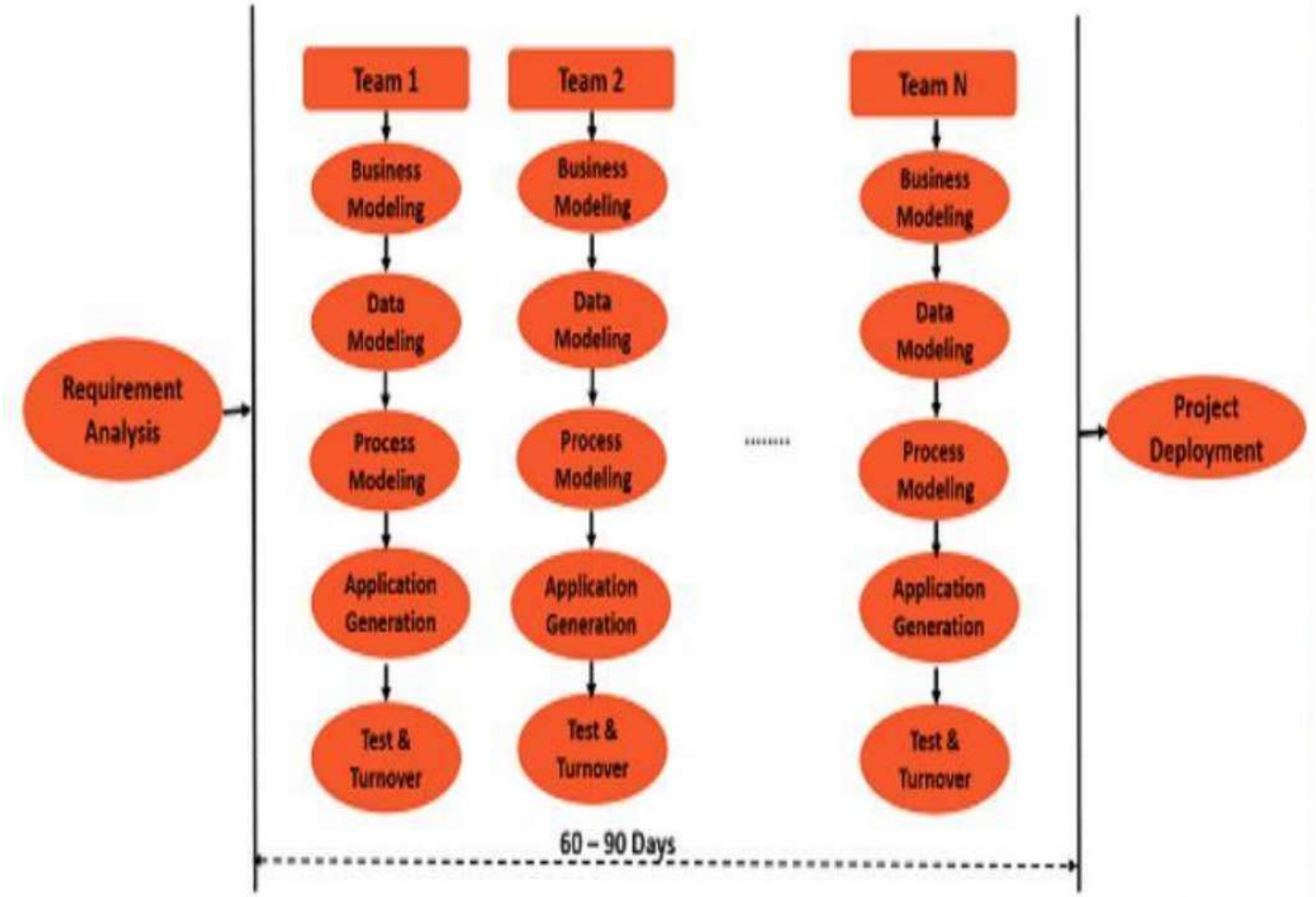
- Design Tools : These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design

TABLE 1-2 Examples of CASE Usage within the SDLC

SDLC Phase	Key Activities	CASE Tool Usage
Project identification and selection	Display and structure high-level organizational information	Diagramming and matrix tools to create and structure information
Project initiation and planning	Develop project scope and feasibility	Repository and documentation generators to develop project plans
Analysis	Determine and structure system requirements	Diagramming to create process, logic, and data models
Logical and physical design	Create new system designs	Form and report generators to prototype designs; analysis and documentation generators to define specifications
Implementation	Translate designs into an information system	Code generators and analysis, form and report generators to develop system; documentation generators to develop system and user documentation
Maintenance	Evolve information system	All tools are used (repeat life cycle)

RAD(Rapid Application Development)

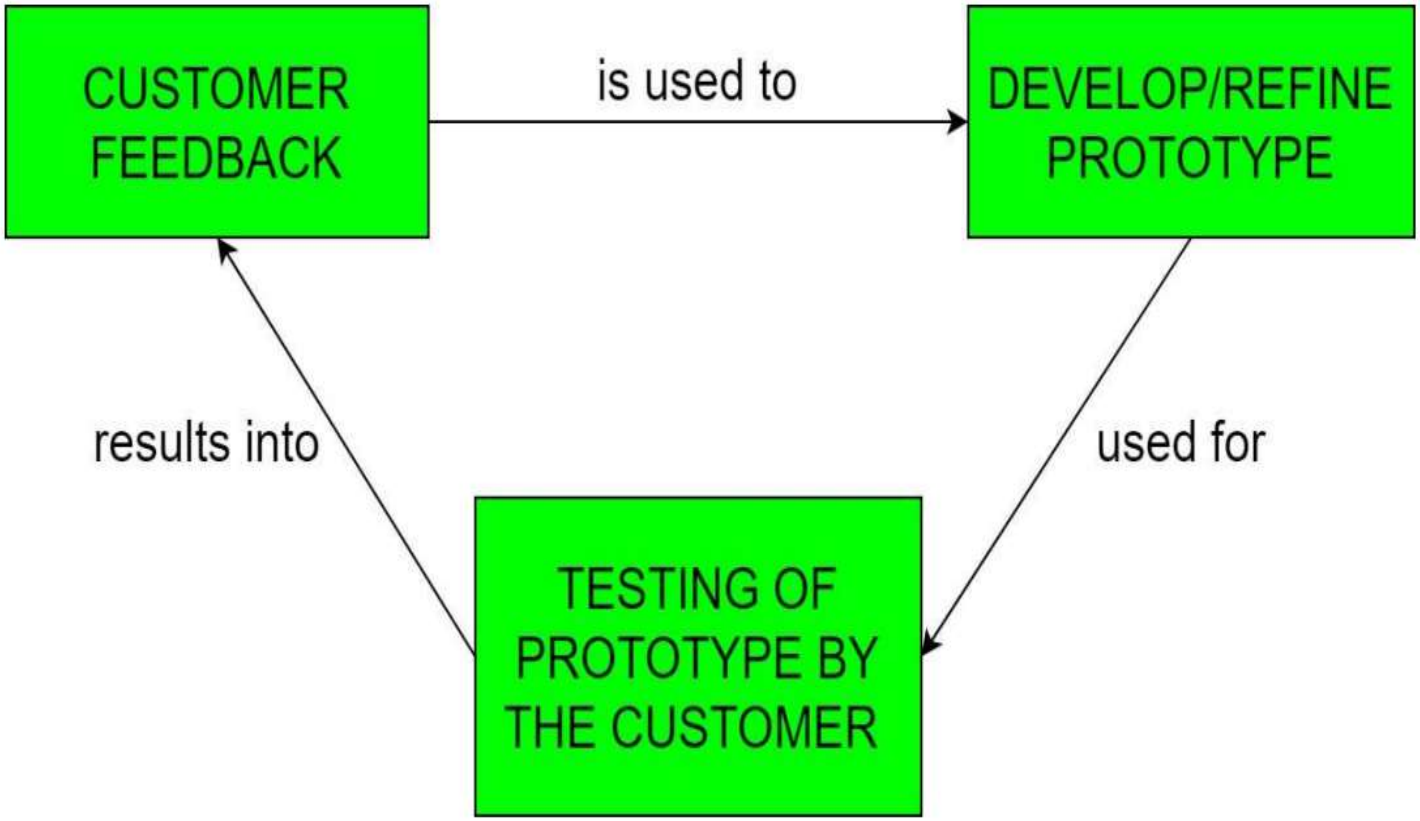
- Rapid application development is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.
- In the RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process.
- The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable.


Requirement Analysis
Team 1
Business Modeling
Data Modeling
Process Modeling
Application Generation
Test & Turnover
Team 2
Business Modeling
Data Modeling
Process Modeling
Application Generation
Test & Turnover
.....
Team N
Business Modeling
Data Modeling
Process Modeling
Application Generation
Test & Turnover
Project Deployment
60 – 90 Days

Prototype or Transformation Model

- Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered.
- This model is used when the customers do not know the exact project requirements beforehand.
- 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
- It offers a small scale facsimile of the end product and is used for obtaining customer feedback as described below:

Prototype or Transformation Model



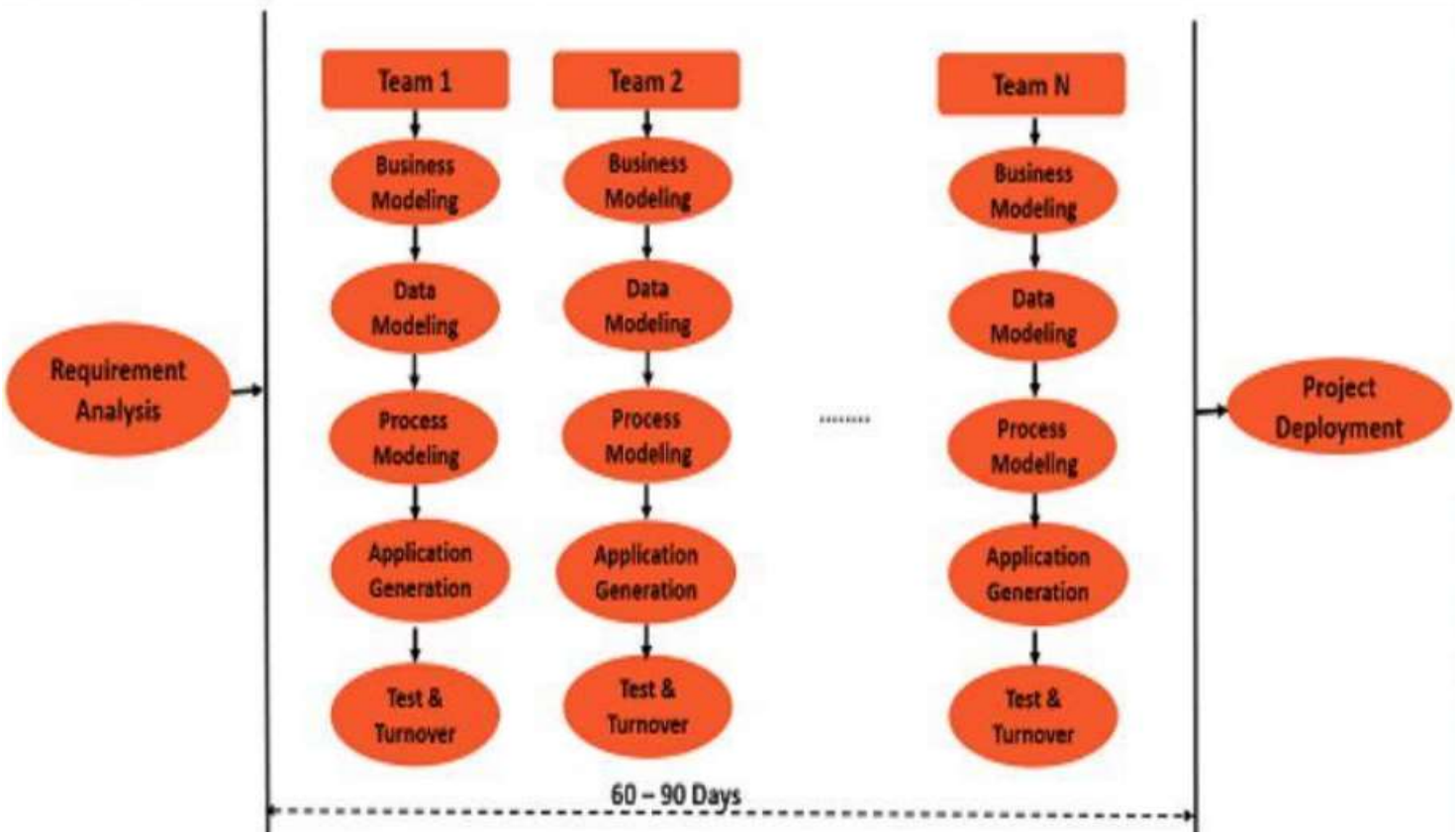
- The time interval between the project begin and final delivery is substantially reduced because all parts of the system are prototyped and tested simultaneously.
- Of course, there might be the possibility that the pieces just not fit together due to some lackness in the development phase – this can only be fixed by careful and complete plotting of the entire system before prototyping starts.

A) Rapid Throwaway Prototyping

- This technique offers a useful method of exploring ideas and getting customer feedback for each of them.
- In this method, a developed prototype need not necessarily be a part of the ultimately accepted prototype.
- Customer feedback helps in preventing unnecessary design faults and hence, the final prototype developed is of better quality.

B) Evolutionary Prototyping

- In this method, the prototype developed initially is incrementally refined on the basis of customer feedback till it finally gets accepted.
- In comparison to Rapid Throwaway Prototyping, it offers a better approach which 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



- **When to use RAD model:**

✓ RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.

✓ It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.

✓ RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

Advantages of the RAD model:

- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.

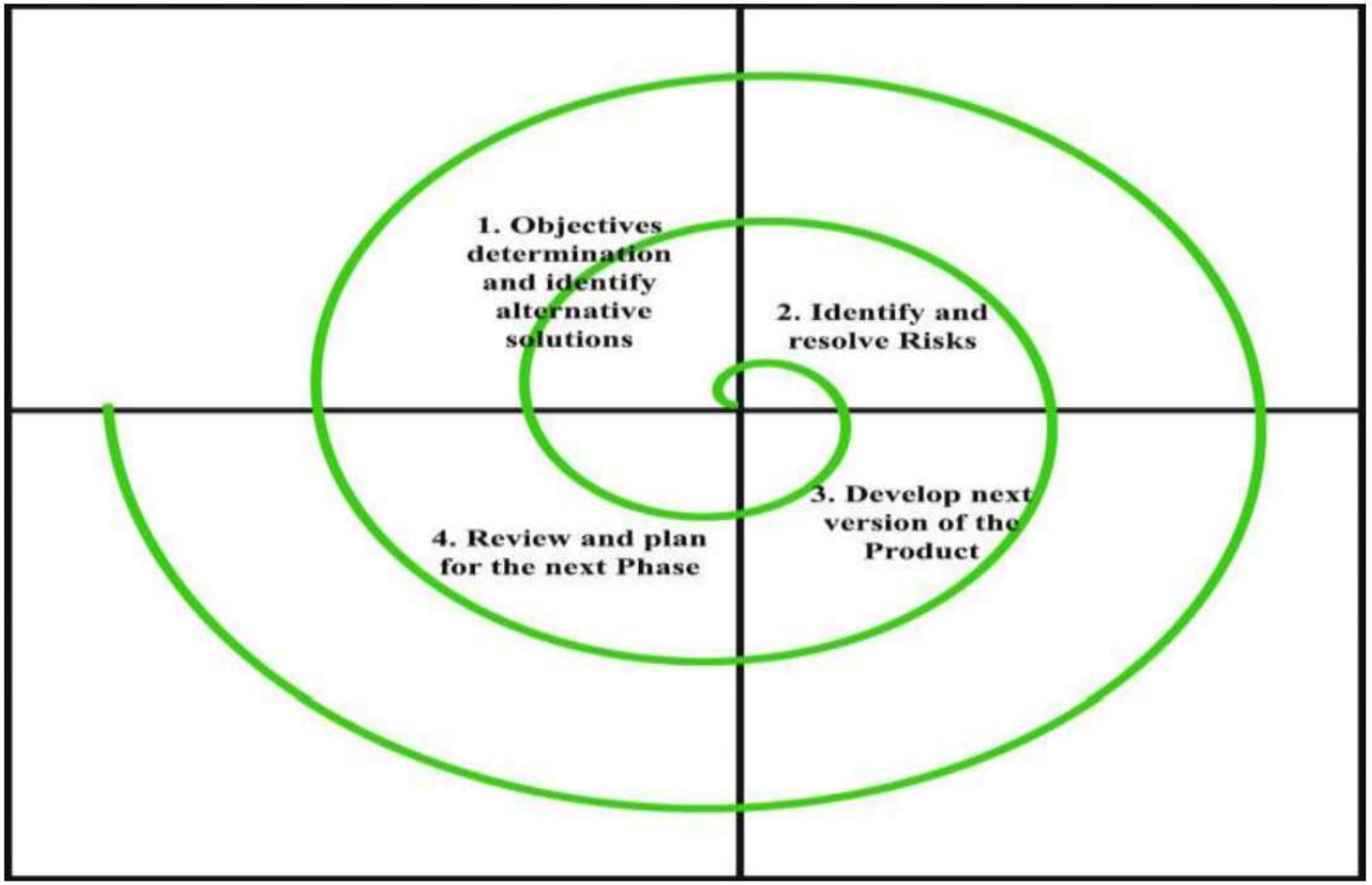
Disadvantages of RAD model:

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

When to use RAD model:

- ✓ RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- ✓ It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- ✓ RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

Spiral model



Spiral Model

- Looks like a spiral with many loops.
- Also called risk handling model
- Number of loops of the spiral is unknown and can vary from project to project.
- Each loop of the spiral is called a Phase of the software development process.

Phases of spiral model

Objectives determination and identify alternative solutions:

- Requirements are gathered from the customers
- Objectives are identified, elaborated, and analyzed at the start of every phase.
- Then alternative solutions possible for the phase are proposed in this quadrant.

Phases of spiral model

Identify and resolve Risks:

- During the second quadrant, all the possible solutions are evaluated to select the best possible solution.
- Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy.
- At the end of this quadrant, the Prototype is built for the best possible solution.

Develop next version of the Product:

- During the third quadrant, the identified features are developed and verified through testing.
- At the end of the third quadrant, the next version of the software is available.

Review and plan for the next Phase:

- In the fourth quadrant, the Customers evaluate the so far developed version of the software.
- In the end, planning for the next phase is started.

Advantages of spiral model

- **Risk Management:** In-built risk analysis in every loop helps identify and mitigate risks early, making it ideal for high-risk or mission-critical projects.
- **Flexibility & Change Handling:** Easily accommodates changing requirements and additions at later stages, unlike traditional models.
- **Customer Satisfaction:** Continuous customer involvement through prototypes and feedback ensures the final product meets expectations.
- **Suitability for Large Projects:** Breaks down complex projects into smaller, manageable iterations, simplifying development.
- **Early Software Delivery:** Produces working prototypes and software early in the lifecycle.
- **Strong Documentation:** Emphasizes detailed documentation and control at each stage, aiding project transparency.
- **Effective Monitoring:** Easy to monitor progress and budget as each loop includes review and assessment.
- **Customization:** Allows for highly customized products as requirements can be refined iteratively.

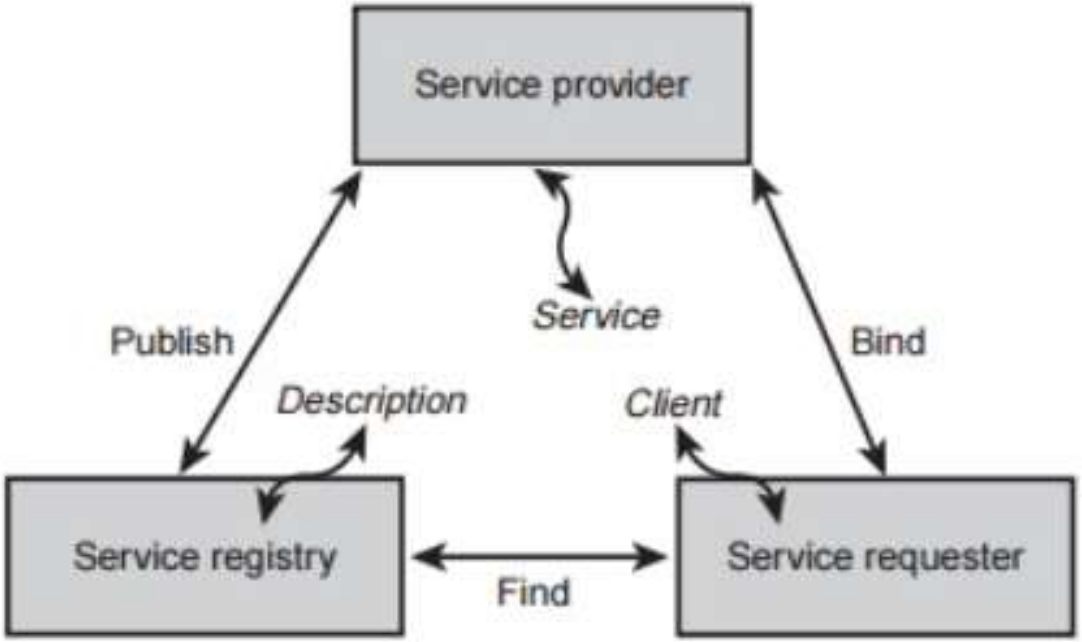
Disadvantages of Spiral Model:

- **Complex:** The Spiral Model is much more complex than other SDLC models.
- **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
- **Too much dependability on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.
- **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.

Service-Oriented Architecture:

- It is modern and new concept of software development . It make individual SOA services are unassociated or loosely coupled to another. Each service executes one action. Each service can be used in other application within the organization or even in other organizations.

- We can say that service-oriented architecture is simply a group of services that can be called upon to provide specific functions. Rather than including calls to other services, a service can use certain defined protocols so that it can communicate with other services.


Service provider
Service
Publish
Bind
Description
Client
Service registry
Service requester
Find

- Service providers publish (and unpublish) their services to a service registry.
- Then, service requesters can find the desired Web Services by searching for their descriptions at the service registry.
- Once the requester locates the desired service, its client binds with the service at the service provider and then invokes the service.
- The SOA is responsible for describing and organizing the mechanisms and practices for each of these actions.
- In addition, the SOA is responsible for describing how Web Services can be combined into larger services.
- **Key Functional Components**

The SOA has four key functional components: service implementation, publication, discovery, and invocation.

- **Service Implementation**

There are two primary ways to build a Web Service:

i. Build from scratch

ii. Wrap an existing application

There are also two ways to handle the service interface:

i. Create a new interface

ii. Reuse an existing interface

- **Publication**

Once the service is created, it needs to be made available to others. Publication means registering the service in a service registry or directory so that users or other applications know it exists.

- **Service Discovery**

This is the process of finding available services. Clients or applications search the service registry to locate a suitable service that matches their requirements.

- **Service Invocation**

After a service is discovered, it is used or executed. Invocation means calling the service to perform its task, typically through a standard protocol like HTTP, SOAP, or REST.

Advantages of Service-oriented Architecture

- SOA allows reuse the service of an existing system alternately building the new system.
- It allows plugging in new services or upgrading existing services to place the new business requirements.
- It can enhance the performance, functionality of a service and easily makes the system upgrade.
- SOA has capability to adjust or modify the different external environments and large applications can be managed easily.
- The companies can develop applications without replacing the existing applications.
- It provides reliable applications in which you can test and debug the independent services easily as compared to large number of code.

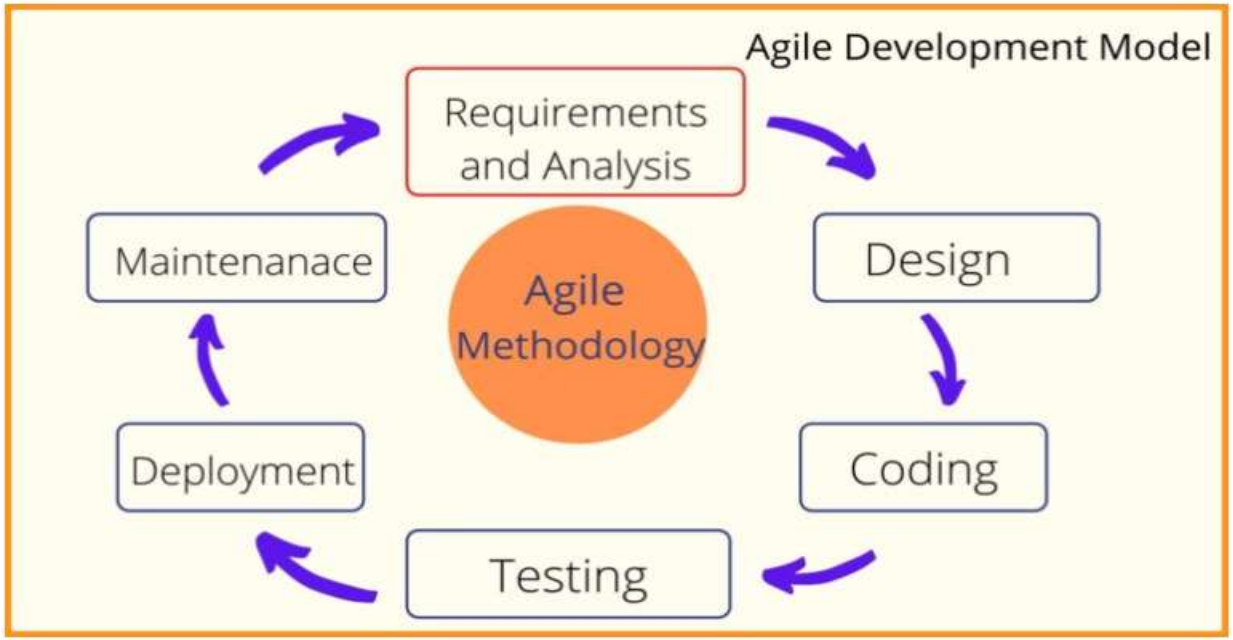
Disadvantages of Service-oriented Architecture

- SOA requires high investment cost (means large investment on technology, development and human resource).
- There is greater overhead when a service interacts with another service which increases the response time and machine load while validating the input parameters.
- SOA is not suitable for GUI (graphical user interface) applications which will become more complex when the SOA requires the heavy data exchange.

Agile Methodologies

- The word 'agile' means – Able to move your body quickly and easily.
- In software development, the term 'agile' is adapted to mean 'the ability to respond to changes – changes from Requirements, Technology and People.
- According to Fowler (2003), the Agile Methodologies share three key principles: (1) a focus on adaptive rather than predictive methodologies, (2) a focus on people rather than roles, and (3) a focus on self-adaptive processes.
- Agile Methodologies share iterative development (Martin, 1999).Iterative development focuses on the frequent production of working versions of a system that have a subset of the total number of required features. Iterative development provides feedback to customers and developers alike.

- The **Agile model** is an iterative and incremental approach to software development that prioritizes flexibility, collaboration, and continuous delivery.
- Agile methodology emphasizes the importance of delivering working software frequently, rather than focusing solely on comprehensive documentation or following a strict plan.
- Agile teams work closely with customers and stakeholders to ensure that their needs are being met, and they prioritize responding to change over following a rigid plan.
- Some popular Agile methodologies include Scrum, Kanban, and Extreme Programming (XP).



Agile model phases

- **The Agile Model** was primarily designed to help a project adapt quickly to change requests.
- So, the main aim of the Agile model is to facilitate quick project completion. To accomplish this task, it's important that agility is required.
- Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project.
- Also, anything that is a waste of time and effort is avoided. The Agile Model refers to a group of development processes.

When To Use the Agile Model?

- When new changes need to be implemented. The freedom that playfulness gives for change is very important.
- New changes can be implemented at a very low cost due to the frequency of new increments.
- Implementing a new feature requires developers to lose only a few days or even just hours of work to get it back and implemented.
- In the Agile model, unlike the Waterfall model, very limited planning is required to start a project.
- Agile believes that the needs of end users are always changing in the dynamic business and IT world.
- Changes can be discussed and features can be redesigned or removed based on feedback.

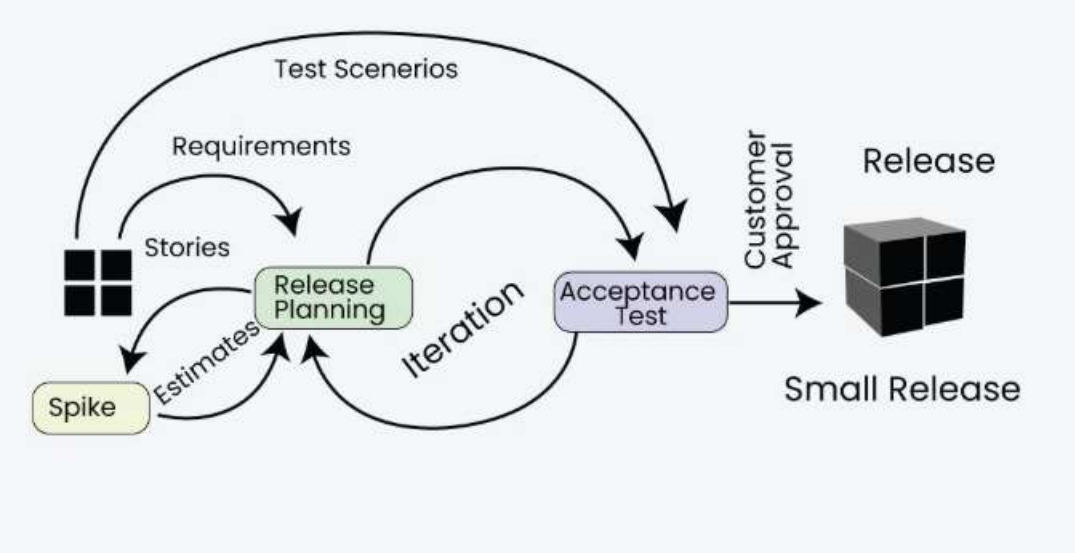
- **Advantages of the Agile Model**
- It reduces the total development time of the whole project.
- Agile development emphasizes face-to-face communication among team members, leading to better collaboration and understanding of project goals.
- Customer representatives get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.
- Agile development puts the customer at the center of the development process, ensuring that the end product meets their needs.
- **Disadvantages of the Agile Model**
- The lack of formal documents creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
- It is not suitable for handling complex dependencies.
- The agile model depends highly on customer interactions so if the customer is not clear, then the development team can be driven in the wrong direction.
- Due to the absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.
- Agile development models require a high degree of expertise from team members, as they need to be able to adapt to changing requirements and work in an iterative environment. This can be challenging for teams that are not experienced in agile development practices and can lead to delays and difficulties in the project.

Extreme Programming

- Extreme Programming is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop a software.
- eXtreme Programming (XP) was conceived and developed to address the specific needs of software development by small teams in the face of vague(indefinite) and changing requirements.
- Extreme Programming is one of the Agile software development methodologies. It provides values and principles to guide the team behavior.
- Why is it called "Extreme?"
  - Extreme Programming takes the effective principles and practices to extreme levels.
  - Code reviews are effective as the code is reviewed all the time.
  - Testing is effective as there is continuous regression and testing.
  - Design is effective as everybody needs to do refactoring daily.
  - Integration testing is important as integrate and test several times a day.

Extreme programming (XP)

- Extreme programming (XP) is one of the most important software development frameworks of Agile models. It is used to improve software quality and responsiveness to customer requirements.
- The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.
- Extreme Programming (XP) is an Agile software development methodology that focuses on delivering high-quality software through frequent and continuous feedback, collaboration, and adaptation.
- XP emphasizes a close working relationship between the development team, the customer, and stakeholders, with an emphasis on rapid, iterative development and deployment.



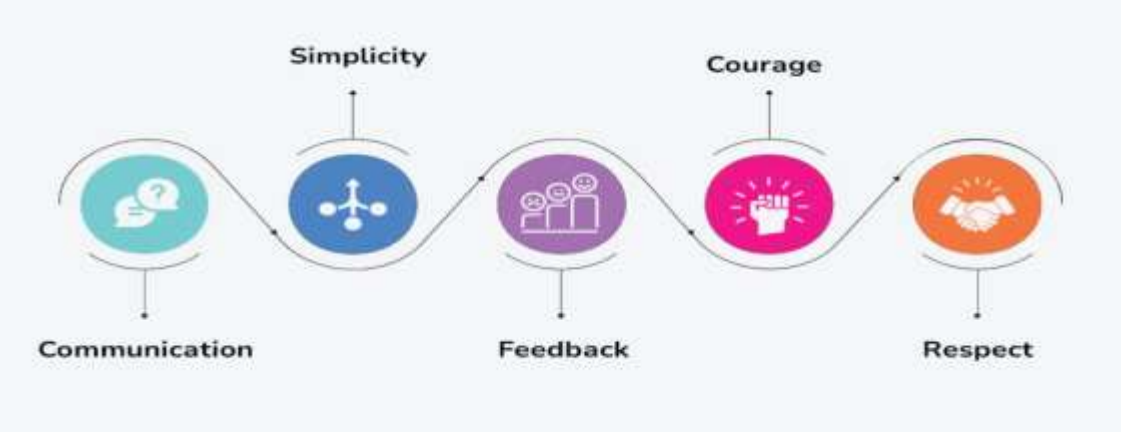
Extreme programming (XP)

- XP is based on the frequent iteration through which the developers implement User Stories.
- User stories are simple and informal statements of the customer about the functionalities needed.
- A User Story is a conventional description by the user of a feature of the required system.
- It does not mention finer details such as the different scenarios that can occur. Based on User stories, the project team proposes Metaphors.
- Metaphors are a common vision of how the system would work. The development team may decide to build a Spike for some features.
- A Spike is a very simple program that is constructed to explore the suitability of a solution being proposed. It can be considered similar to a prototype.
- Some of the basic activities that are followed during software development by using the XP model are given below:
- **Coding:** The concept of coding which is used in the XP model is slightly different from traditional coding. Here, the coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system, and choosing among several alternative solutions.

- **Testing:** The XP model gives high importance to testing and considers it to be the primary factor in developing fault-free software.
- **Listening:** The developers need to carefully listen to the customers if they have to develop good quality software. Sometimes programmers may not have the depth knowledge of the system to be developed. So, the programmers should understand properly the functionality of the system and they have to listen to the customers.
- **Designing:** Without a proper design, a system implementation becomes too complex, and very difficult to understand the solution, thus making maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.
- **Feedback:** One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.
- **Simplicity:** The main principle of the XP model is to develop a simple system that will work efficiently in the present time, rather than trying to build something that would take time and may never be used. It focuses on some specific features that are immediately needed, rather than engaging time and effort on speculations of future requirements.
- **Pair Programming:** XP encourages pair programming where two developers work together at the same workstation. This approach helps in knowledge sharing, reduces errors, and improves code quality.
- **Continuous Integration:** In XP, developers integrate their code into a shared repository several times a day. This helps to detect and resolve integration issues early on in the development process.

Values of Extreme Programming (XP)

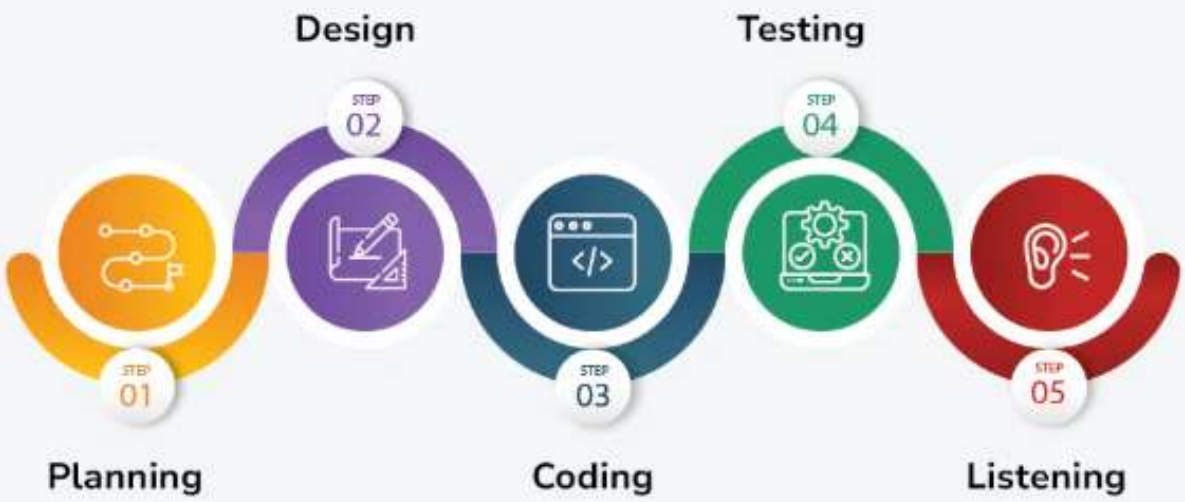
- **Communication:** The essence of communication is for information and ideas to be exchanged amongst development team members so that everyone has an understanding of the system requirements and goals. Extreme Programming (XP) supports this by allowing open and frequent communication between members of a team.
- **Simplicity:** Keeping things as simple as possible helps reduce complexity and makes it easier to understand and maintain the code.
- **Feedback:** Feedback loops which are constant are among testing as well as customer involvements which helps in detecting problems earlier during development.
- **Courage:** Team members are encouraged to take risks, speak up about problems, and adapt to change without fear of repercussions.
- **Respect**: Every member's input or opinion is appreciated which promotes a collective way of working among people who are supportive within a certain group.



Applications of Extreme Programming (XP)

- Some of the projects that are suitable to develop using the XP model are given below:
- **Small projects:** The XP model is very useful in small projects consisting of small teams as face-to-face meeting is easier to achieve.
- **Projects involving new technology or Research projects:** This type of project faces changing requirements rapidly and technical problems. So XP model is used to complete this type of project.
- **Web development projects:** The XP model is well-suited for web development projects as the development process is iterative and requires frequent testing to ensure the system meets the requirements.
- **Collaborative projects:** The XP model is useful for collaborative projects that require close collaboration between the development team and the customer.
- **Projects with tight deadlines:** The XP model can be used in projects that have a tight deadline, as it emphasizes simplicity and iterative development.
- **Projects with rapidly changing requirements: The** XP model is designed to handle rapidly changing requirements, making it suitable for projects where requirements may change frequently.
- **Projects where quality is a high priority:** The XP model places a strong emphasis on testing and quality assurance, making it a suitable approach for projects where quality is a high priority

Life Cycle of Extreme Programming (XP)



Object-Oriented Design (OOD) - System Design

- Object-Oriented Analysis and Design (OOAD) is a way to design software by thinking of everything as objects similar to real-life things.
- In OOAD, we first understand what the system needs to do, then identify key objects, and finally decide how these objects will work together.
- This approach helps make software easier to manage, reuse, and grow.
- Object-oriented design (OOD) is a programming technique that solves software problems by building a system of interrelated objects.
- It makes use of the concepts of classes and objects, encapsulation, inheritance, and polymorphism to model real-world entities and their interactions.
- A system architecture that is modular, adaptable, and simple to understand and maintain is produced using OOD.
- **For example:** OOA helps you figure out all the things you need to know about the game world - the characters, their features, and how they interact. It's like making a map of everything important. OOA also helps you understand what your game characters will do. It's like writing down a script for each character.
- Every program has specific tasks or jobs it needs to do. OOA helps you list and describe these jobs.

Object-Oriented Analysis And Design

- The object-oriented approach combines data and processes (called methods) into single entities called objects. Objects usually correspond to the real things an information system deals with, such as customers, suppliers, contracts, and rental agreements.
- The goal of OOAD is to make systems elements more reusable, thus improving system quality and the productivity of systems analysis and design.
- Another key idea behind object orientation is inheritance. Inheritance allows the creation of new classes that share some of the characteristics of existing classes.
- For example, from a class of objects called “person,” you can use inheritance to define another class of objects called “customer. "Objects of the class “customer” would share certain characteristics with objects of the class “person”: They would both have names, addresses, phone numbers, and so on. Because “person” is the more general class and “customer” is more specific, every customer is a person but not every person is a customer.

- The object-oriented approach to systems development shares the iterative development approach of the Agile Methodologies.
- One of the most popular realizations of the iterative approach for object-oriented development is the Rational Unified Process (RUP), which is based on an iterative, incremental approach to systems development. RUP has four phases: inception, elaboration, construction, and transition (see Figure).
- In the inception phase, analysts define the scope, determine the feasibility of the project, understand user requirements, and prepare a software development plan.
- In the elaboration phase, analysts detail user requirements and develop a baseline architecture. Analysis and design activities constitute the bulk of the elaboration phase. In the construction phase, the software is actually coded, tested, and documented.
- In the transition phase, the system is deployed, and the users are trained and supported.

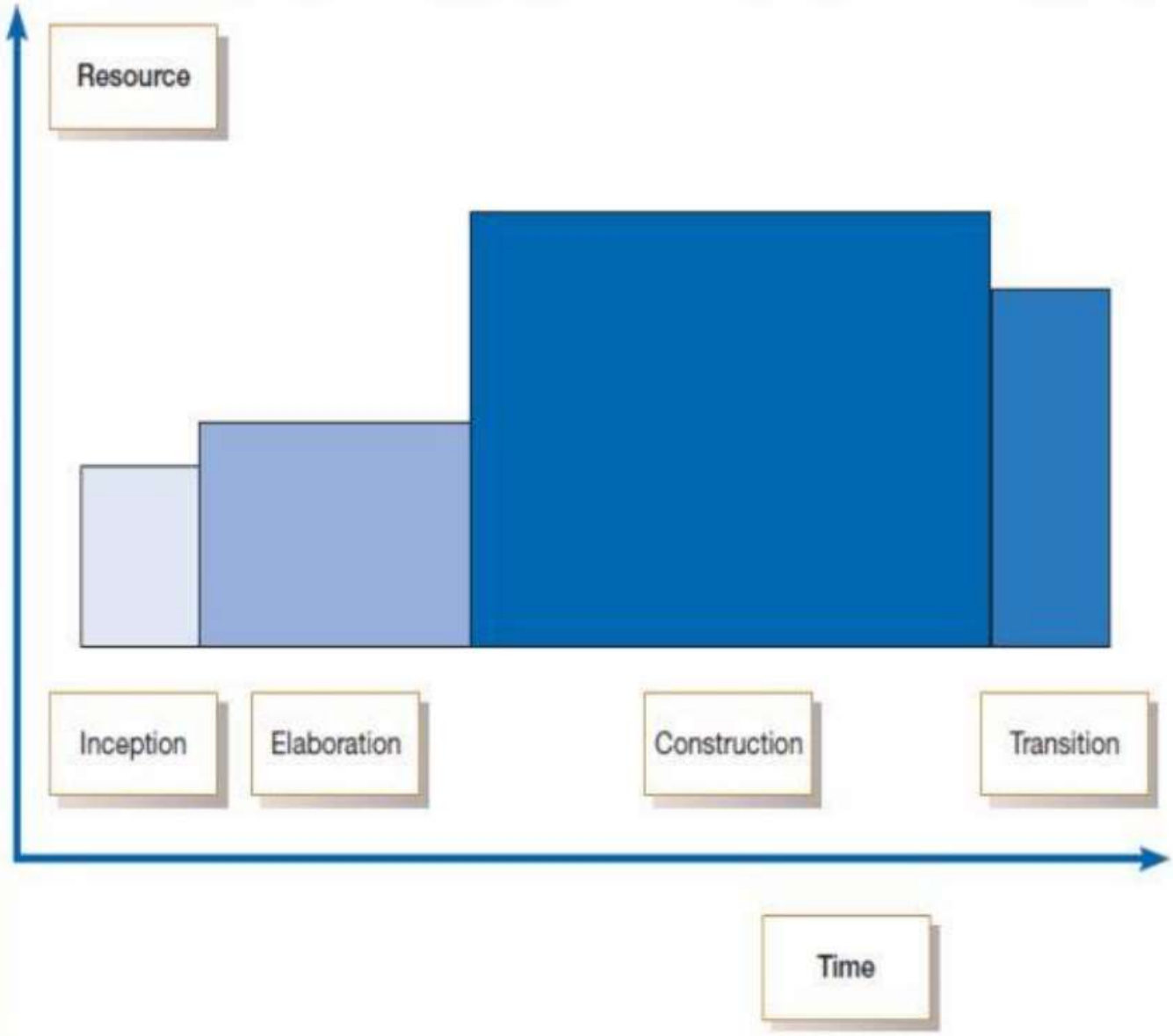


FIGURE 1-11
Phases of OOAD-based development

- The construction phase is generally the longest and the most resource intensive. The elaboration phase is also long, but less resource intensive. The transition phase is resource intensive but short. The inception phase is short and the least resource intensive. The areas of the rectangles in Figure 1-11 provide an estimate of the overall resources allocated to each phase.

Challenges of Object-Oriented Analysis and Design(OOAD)

- Because objects and their interactions need to be carefully explained and handled, it might complicate a software system.
- Because objects must be instantiated, managed, and interacted with, this may result in additional overhead and reduce the software's speed.
- For beginner software engineers, OOAD might have a challenging learning curve since it requires a solid grasp of OOP principles and methods.
- It can be a time-consuming process that involves significant upfront planning and documentation. This can lead to longer development times and higher costs.
- OOAD can be more expensive than other software engineering methodologies due to the upfront planning and documentation required.

Real world applications of Object-Oriented Analysis and Design(OOAD)

- **Banking Software:** In banking systems, OOAD is frequently used to simulate complex financial transactions, structures, and customer interactions. Designing adaptable and reliable financial apps is made easier by OOAD's modular and scalable architecture.
- **Online Shopping Platforms:** E-commerce system development frequently makes use of OOAD. Product catalogs, user profiles, shopping carts, and payment procedures are all modeled, which facilitates platform maintenance and functionality expansion.
- **Flight Control Systems:** OOAD is crucial in designing flight control systems for aircraft. It helps model the interactions between different components such as navigation systems, sensors, and control surfaces, ensuring safety and reliability.
- **Electronic Health Record (EHR) Systems:** Patient data, medical records, and healthcare workflows are all modeled using OOAD.


Thank
you