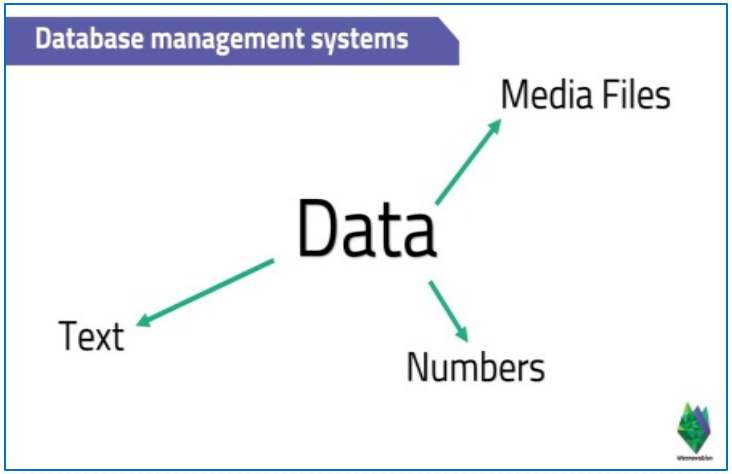
Unit 1: System Development Fundamental

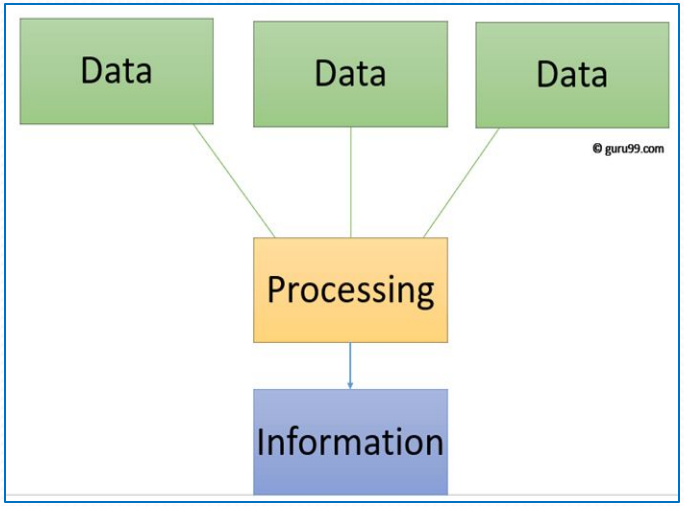
Data

- In simple words, **data** can be facts related to any object in consideration. For example, your name, age, height, weight, etc. are some data related to you. A picture, image, file, pdf, etc. can also be considered data
- Structured , Semi structured , Unstructured
  - Example : student , Ada, BCA, third, Semester



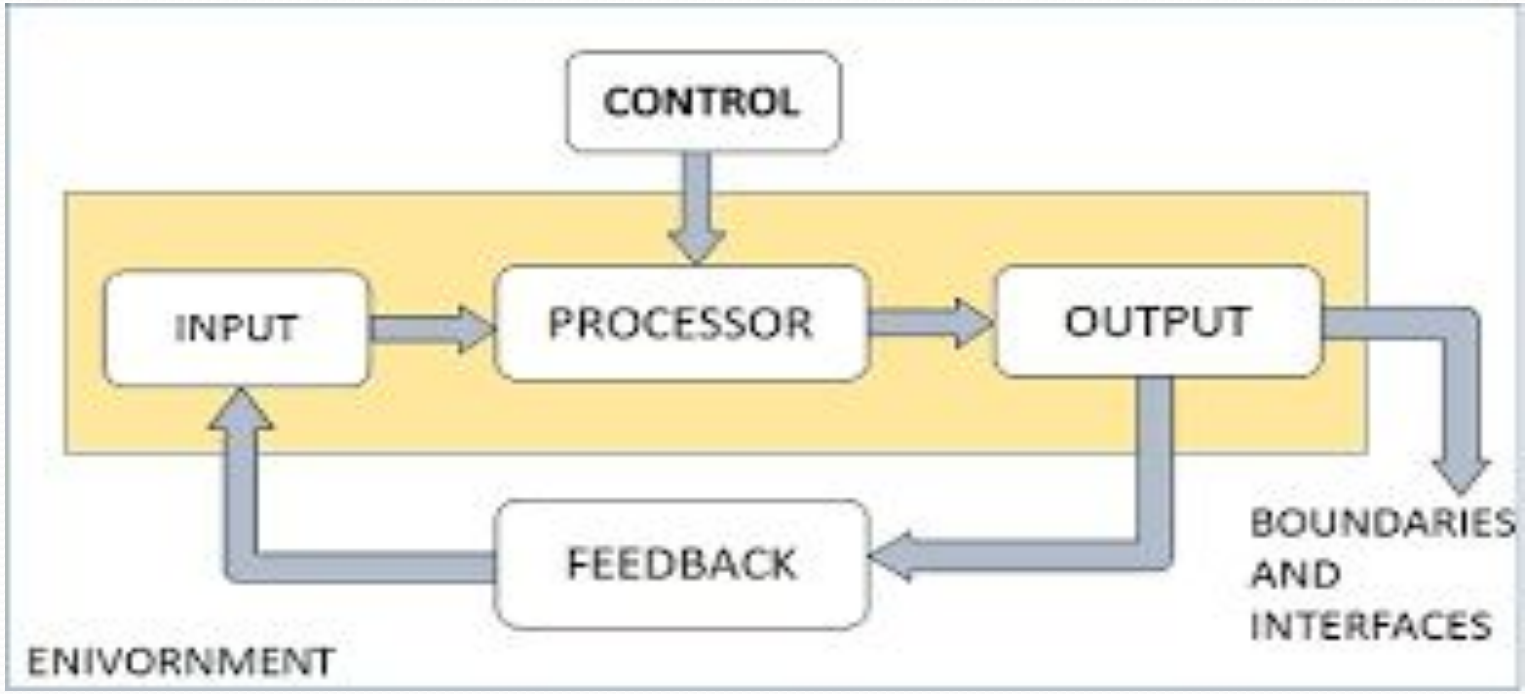
Information

- **Information** is a processed, organized data which gives logical meaning
  - **Ada** is a **student** of **BCA third semester**.



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.
- System is a collection of elements or components that are organized for a common purpose.



Example of system

- Information system
- Solar system
- Digestive system
- Public transportation system
- Electronic voting system

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.

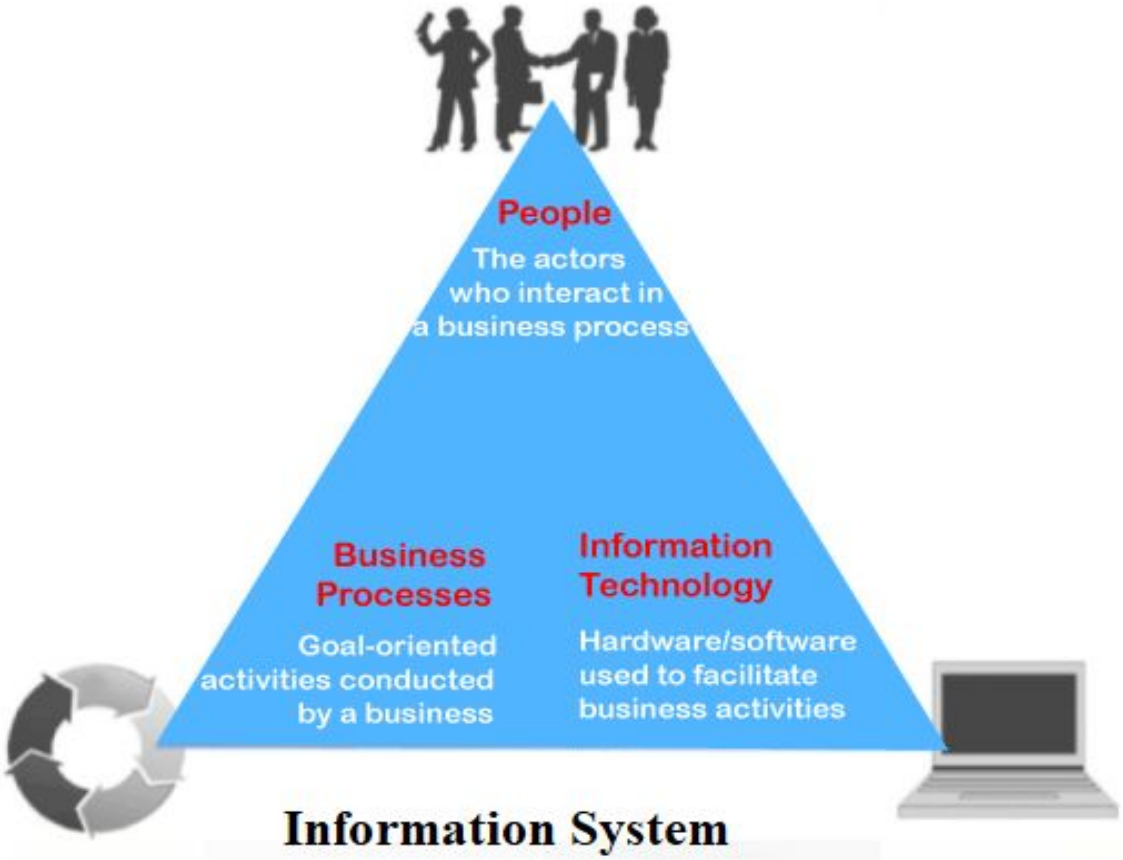
Properties of a System

- 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.

- **System analysis:** understanding and specifying in detail what an information system should do. Analysis specifies **what the system should do**.
- **System Design:** Specifying in detail how the parts of information system should be implemented. System Design focuses on **how to accomplish the objective of the system**.
- **Definition of SAD:** The complex organizational process whereby computer-based information system are developed and maintained.
  - SAD mainly focuses on −
    - Systems
    - Processes
    - Technology

Information systems

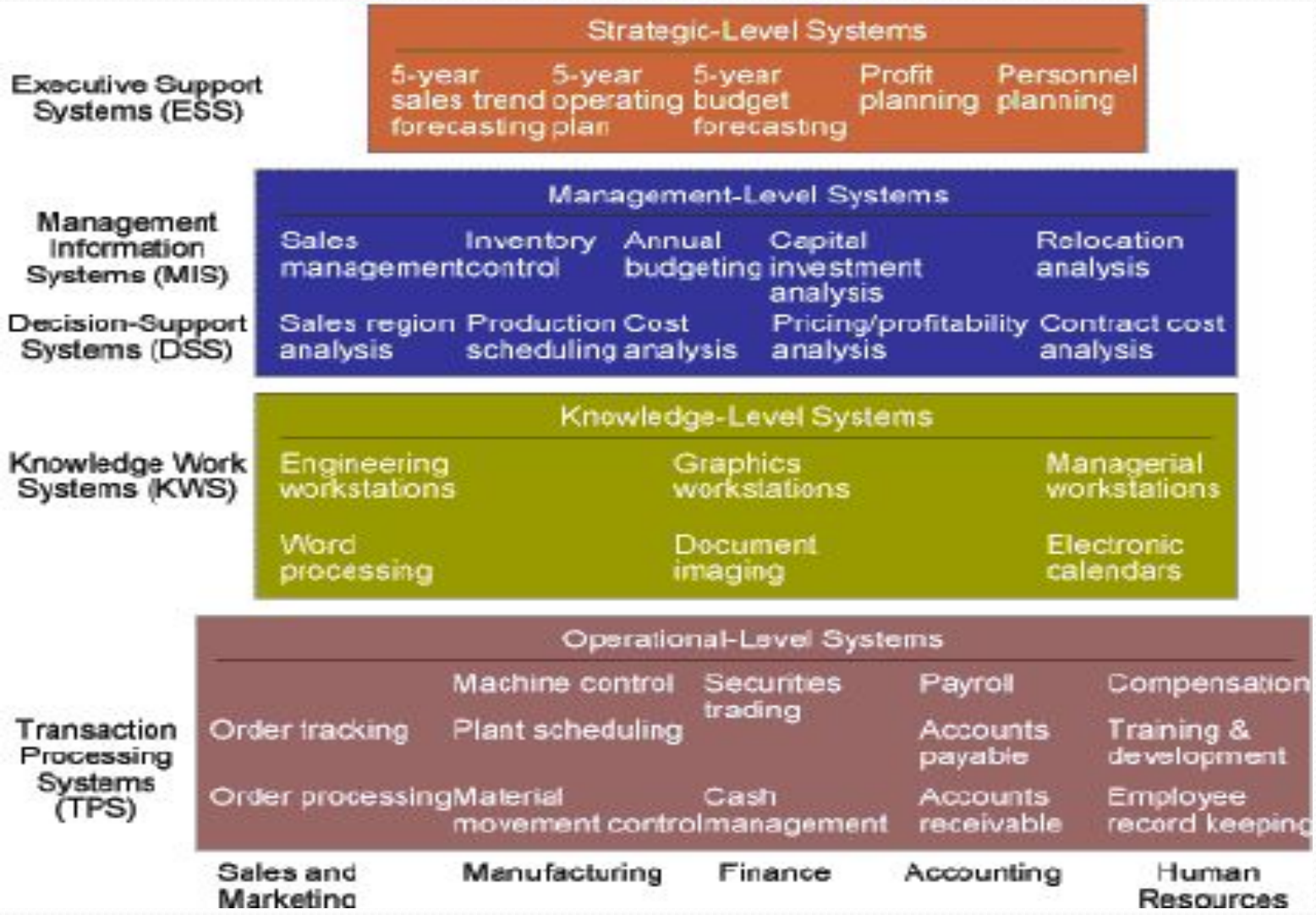
- Information systems are a set of interconnected elements working together to collect, process, store, and distribute information to help coordination, visualization in an organization, analysis, and decision-making.
- The Information system can be defined as a collection of **software, hardware**, and **telecommunications network** that people develop and use to **gather, create,** and **distribute useful data**, mainly in organizational settings.

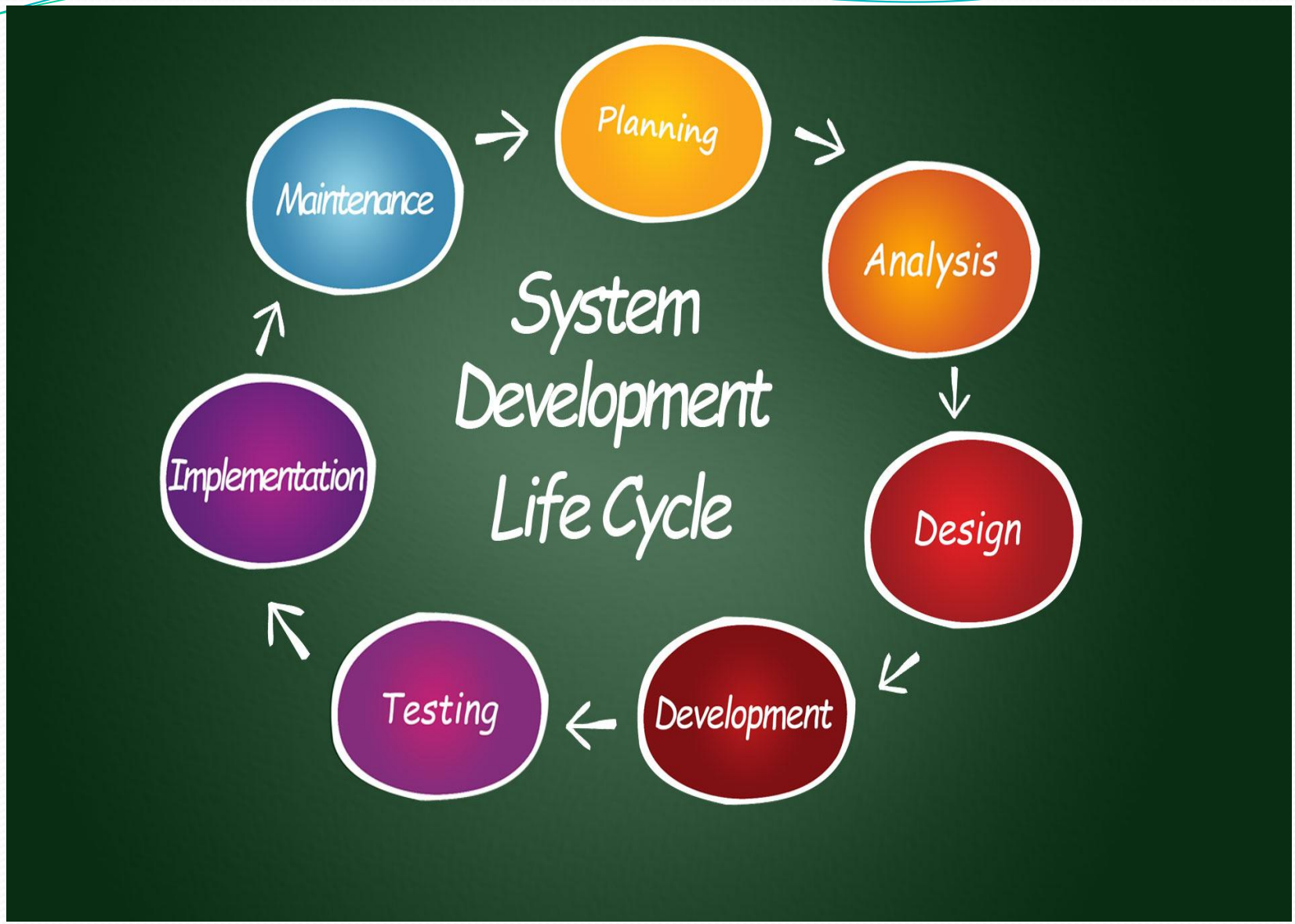

People
The actors who interact in a business process
Business Processes
Goal-oriented activities conducted by a business
Information Technology
Hardware/software used to facilitate business activities
Information System

Components of Information System

Components of Information System

Types of Information System




System Development Life Cycle
Planning
Analysis
Design
Development
Testing
Implementation
Maintenance

Software

- Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
- There are different types of software that can run on a computer: system software, utility software, and application software.
  - **System software**
    - If you think of software as being in layers, the system software is the bottom layer: it sits between the hardware and the application software.
    - Operating systems like Windows, macOS, Android and iOS are examples of system software. Operating systems are loaded into RAM when the device starts up, and have access to the hard drive.

Utility software

- Utility software is part of the system software and performs specific tasks to keep the computer running. Utility software is always running in the background. Examples of utility software are security and optimization programs.
- Security programs include anti-virus software that scans and removes viruses. Most computers will include some sort of anti-virus software, but you can add your own.
- Optimization programs can include tools for system clean-up, disk defragmentation, and file compression. These tools are typically installed as part of the operating system. They have access to the hard drive to keep it tidy.

- **Application Software**
  - Application software is a software program or group of programs designed for end-users. There are many types of application software.
  - Application Software and Examples
    - Word Processing Software: Google Docs, Microsoft Word, WordPad and Notepad
    - Database Software: MySQL, Microsoft SQL Server, Microsoft Access, Oracle, IBM DB2 and FoxPro
    - Spreadsheet Software: Google Sheets, Apple Numbers and Microsoft Excel
    - Multimedia Software: Media Player, VLC Media Player
    - Presentation Software: Google Slides, Microsoft Powerpoint
    - Enterprise Software: customer relationship management (CRM) software (HubSpot, Microsoft Dynamic 365)), project management tools (Jira)

Software Development Process

- A software development process (also knows as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

Four Activities of Software Process Framework

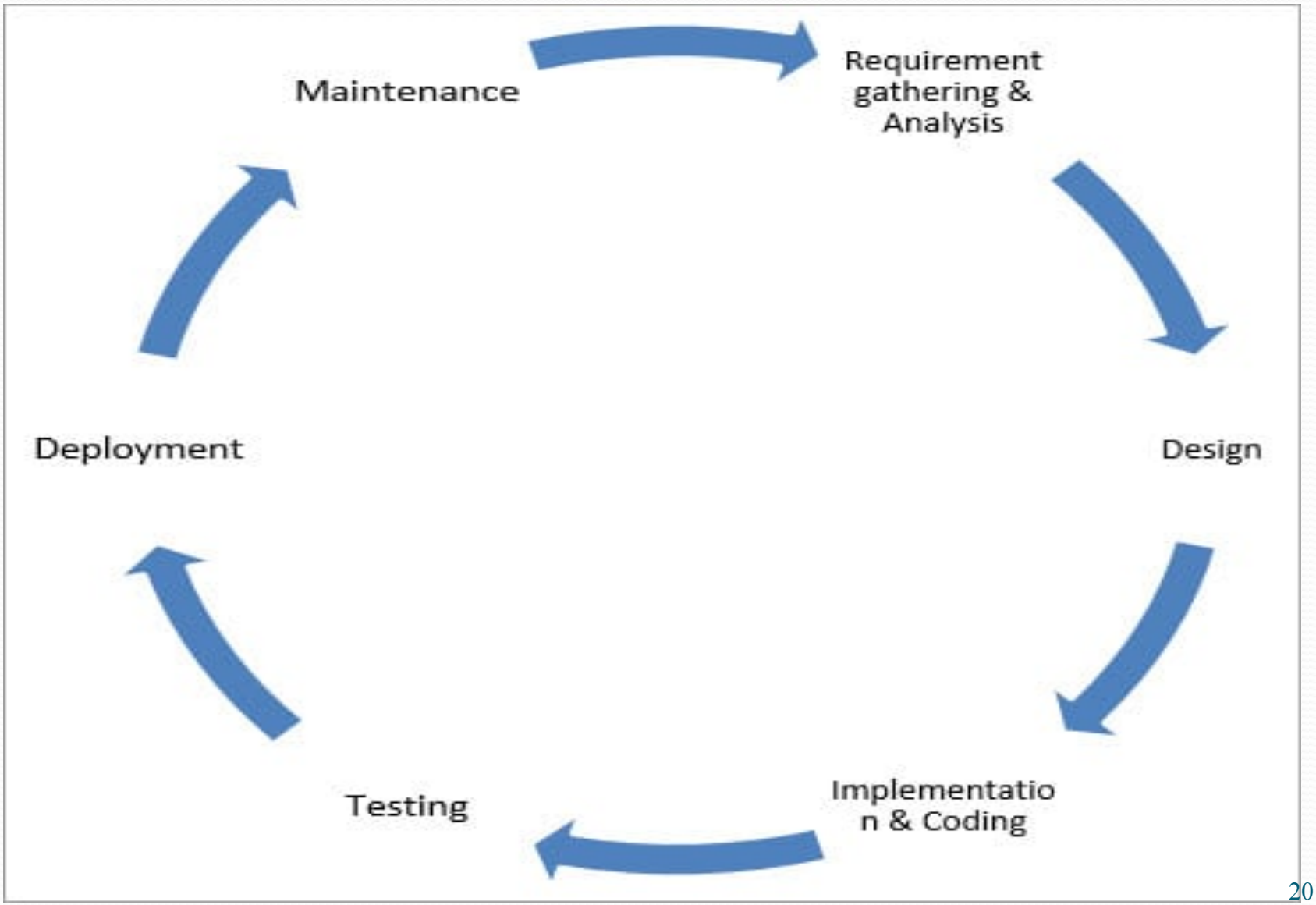
Software Development Process

- **Software Specifications:**
  In this process, detailed description of a software system to be developed with its functional and non-functional requirements.
- **Software Development:**
  In this process, designing, programming, documenting, testing, and bug fixing is done.
- **Software Validation:**
  In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs.
- **Software Evolution:**
  It is a process of developing software initially, then timely updating it for various reasons.

Software Development Life Cycle

- Software development cycle describes phases of the software cycle and the order in which those phases are executed.
- Each phase produces deliverables required by the next phase in the life cycle.
- Requirements are translated into design. Code is produced according to the design which is called development phase.
- After coding and development the testing verifies the deliverable of the implementation phase against requirements.
- The testing team follows Software Testing Life Cycle (STLC)

Software Development Life Cycle



Software Development Life Cycle

- **Requirement gathering and analysis:**
  - Business requirements are gathered in this phase.
  - This phase is the main focus of the project managers and stake holders.
  - Meetings with managers, stake holders and users are held in order to determine the requirements like; Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase.
  - After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.
  - Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model. The testing team follows the Software Testing Life Cycle and starts the Test Planning phase after the requirements analysis is completed.

Software Development Life Cycle

- **Design**:
  - In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase.
  - System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
  - The system design specifications serve as input for the next phase of the model.
  - In this phase the testers comes up with the Test strategy, where they mention what to test, how to test.
- **Implementation / Coding:**
  - On receiving system design documents, the work is divided in modules/units and actual coding is started.
  - Since, in this phase the code is produced so it is the main focus for the developer.
  - This is the longest phase of the software development life cycle.

Software Development Life Cycle

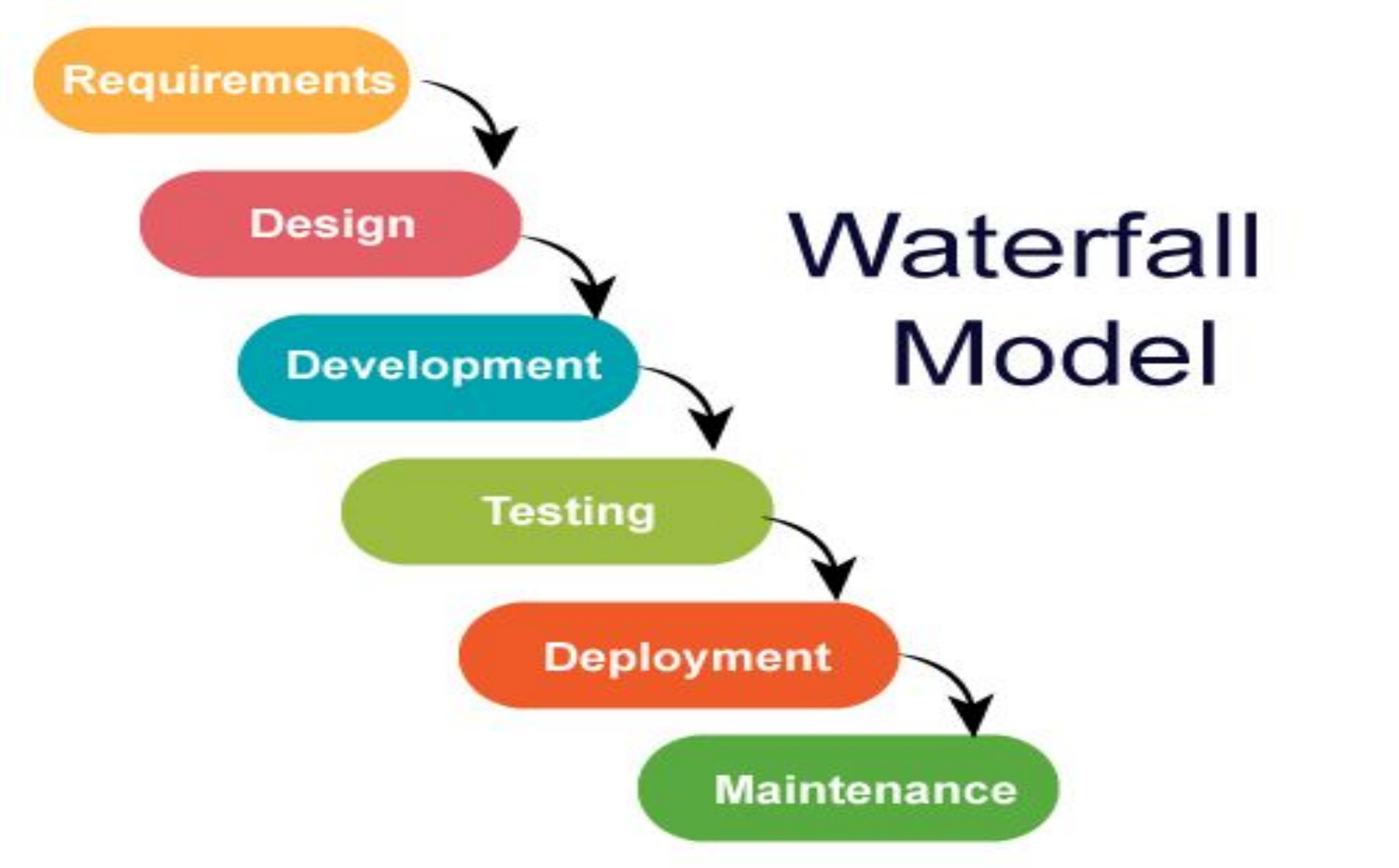
- **Testing**:
  - After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase.
  - During this phase all types of functional testing like unit testing, integration testing, system testing, acceptance testing are done as well as non-functional testing are also done.
- **Deployment:**
  - After successful testing the product is delivered / deployed to the customer for their use.
  - As soon as the product is given to the customers they will first do the beta testing. If any changes are required or if any bugs are caught, then they will report it to the engineering team. Once those changes are made or the bugs are fixed then the final deployment will happen.

Software Development Life Cycle

- **Maintenance:**
  - Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

Software Development Process Model

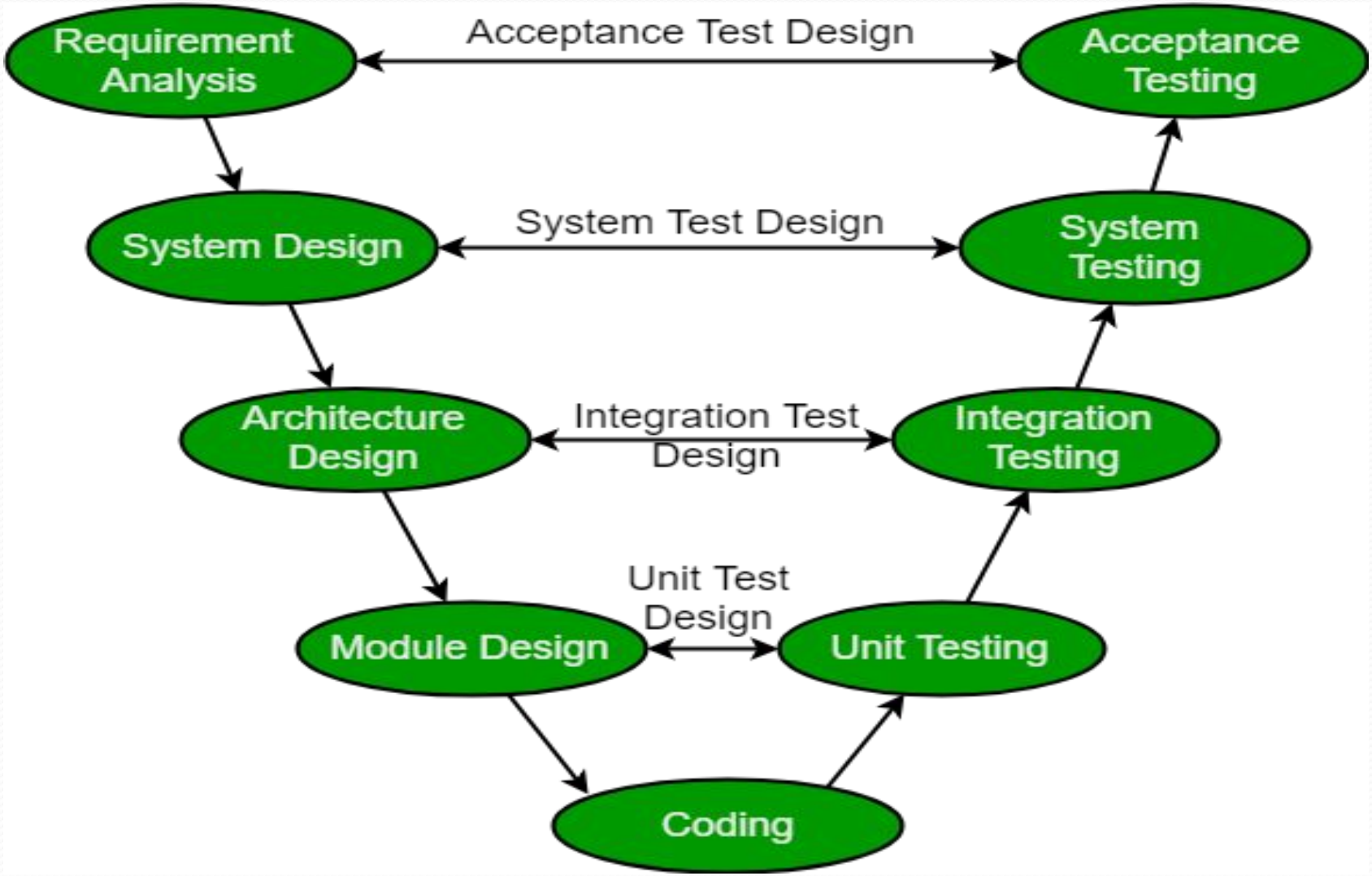
- A software process model is an abstraction of the software development process. The models specify the stages and order of a process. So, think of this as a representation of the **order of activities** of the process and the **sequence** in which they are performed.
- The most used, popular and important SDLC models are given below:
  - Waterfall model
  - V model
  - Incremental model
  - Iterative model
  - Spiral model
  - Prototype model
  - The Rational Unified Process
  - Microsoft Solution Framework
  - Agile methodologies


Requirements
Design
Development
Testing
Deployment
Maintenance
Waterfall Model

Waterfall Model

- The waterfall model is a **sequential, plan driven-process** where you must plan and schedule all your activities before starting the project. Each activity in the waterfall model is represented as a separate phase arranged in linear order.
- Each of these phases produces one or more documents that need to be approved before the next phase begins. However, in practice, these phases are very likely to overlap and may feed information to one another.
- The waterfall model is easy to understand and follow. It doesn't require a lot of customer involvement after the specification is done. Since it's inflexible, it can't adapt to changes. There is no way to see or try the software until the last phase.
- The waterfall model has a rigid structure, so it should be used in cases where the requirements are understood completely and unlikely to radically change.

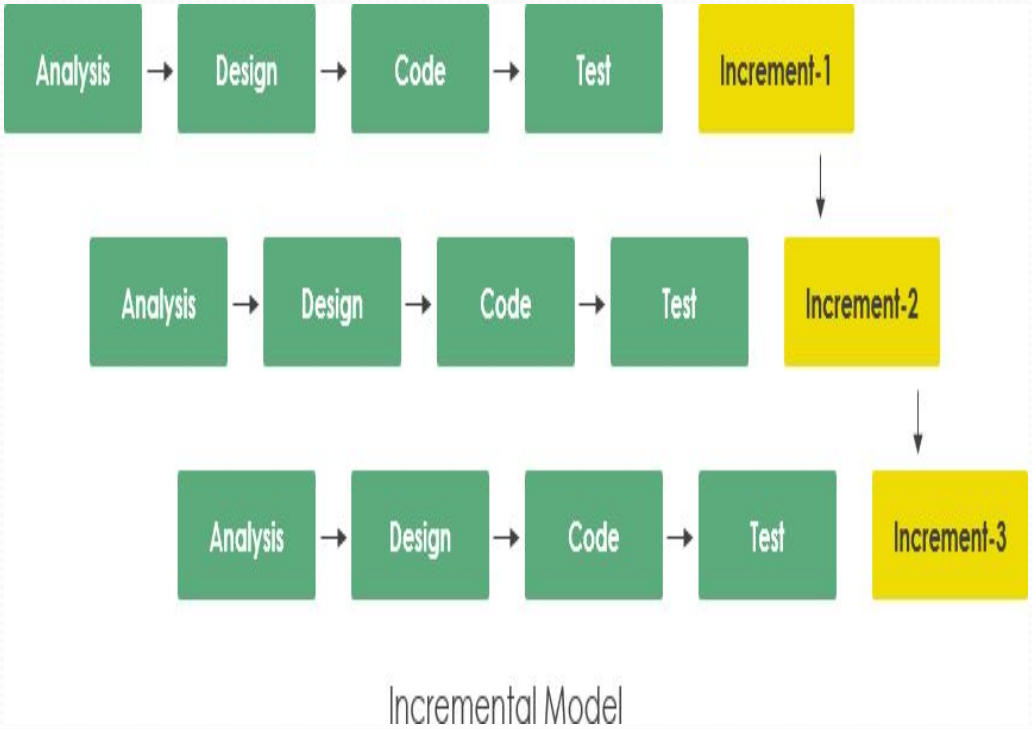
V-model


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

V-Model

- The V-model represents a development process that may be considered an extension of the waterfall model and is an example of the more general V-model.
- Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape.
- The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.
- The horizontal and vertical axes represent time or project completeness (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively.

Incremental model



Incremental Model

Incremental model

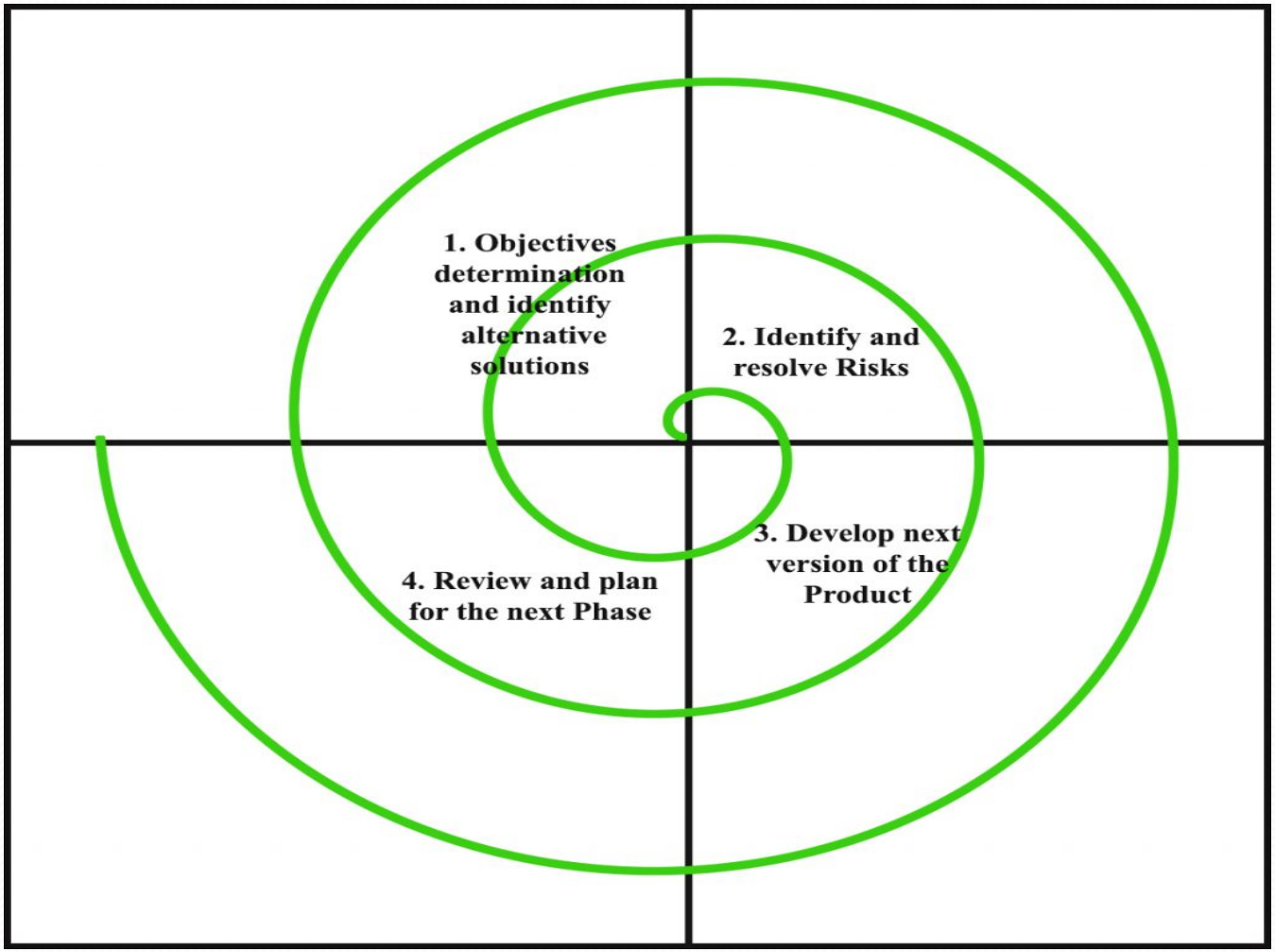
- The incremental build model is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished.
- It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. Each iteration passes through the requirements, design, coding and testing phases. And each subsequent release of the system adds function to the previous release until all designed functionally has been implemented.
- This model combines the elements of the waterfall model with the iterative philosophy of prototyping.

Iterative Model

Iterative Model

- An iterative life cycle model does not attempt to start with a full specification of requirements by first focusing on an initial, simplified set user features, which then progressively gains more complexity and a broader set of features until the targeted system is complete. When adopting the iterative approach, the philosophy of incremental development will also often be used liberally and interchangeably.
- In other words, the iterative approach begins by specifying and implementing just part of the software, which can then be reviewed and prioritized in order to identify further requirements. This iterative process is then repeated by delivering a new version of the software for each iteration. In a light-weight iterative project the code may represent the major source of documentation of the system; however, in a critical iterative project a formal software specification may also be required.

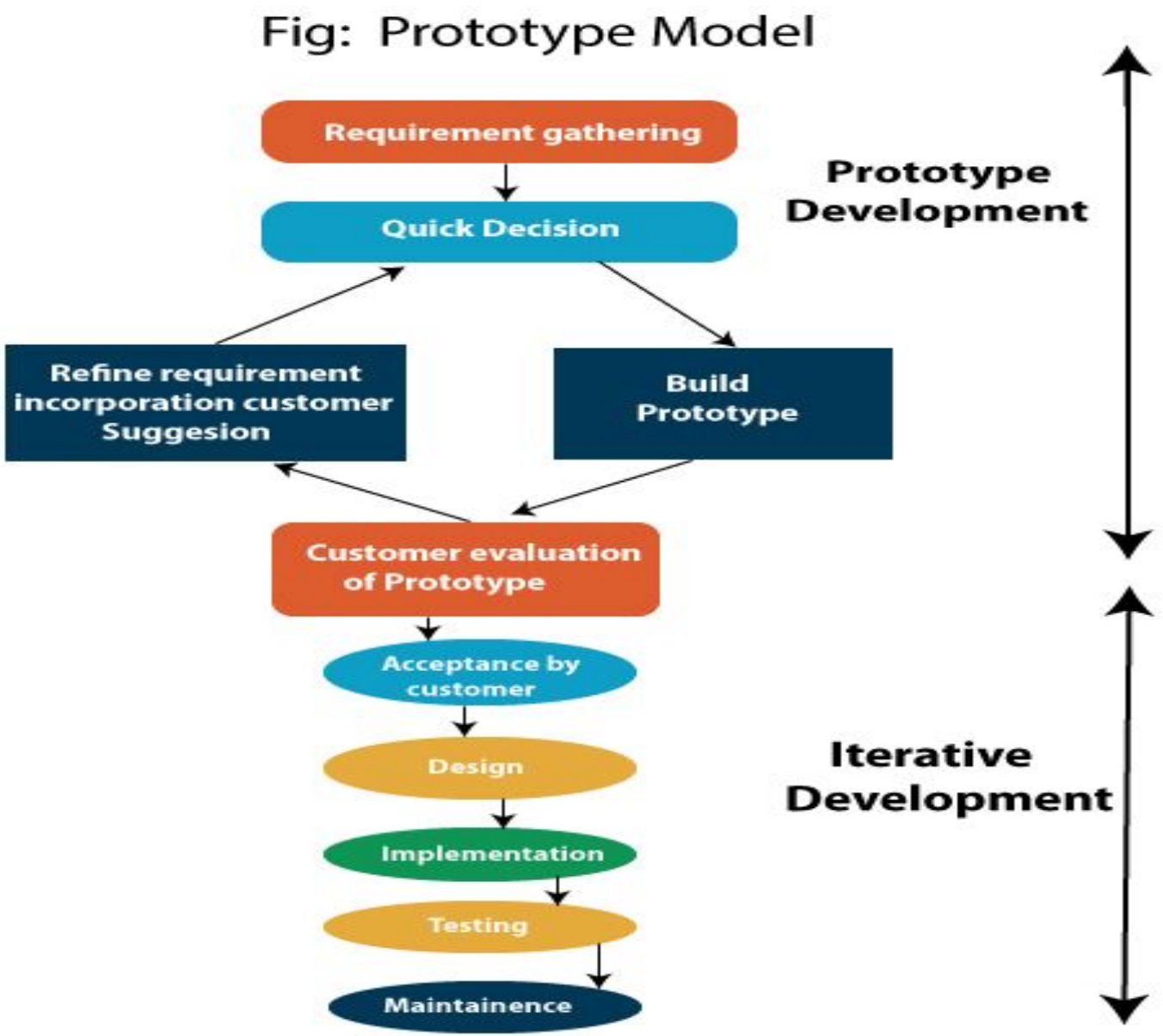
Spiral model



Spiral model

- **Spiral model** is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact 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.** The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.

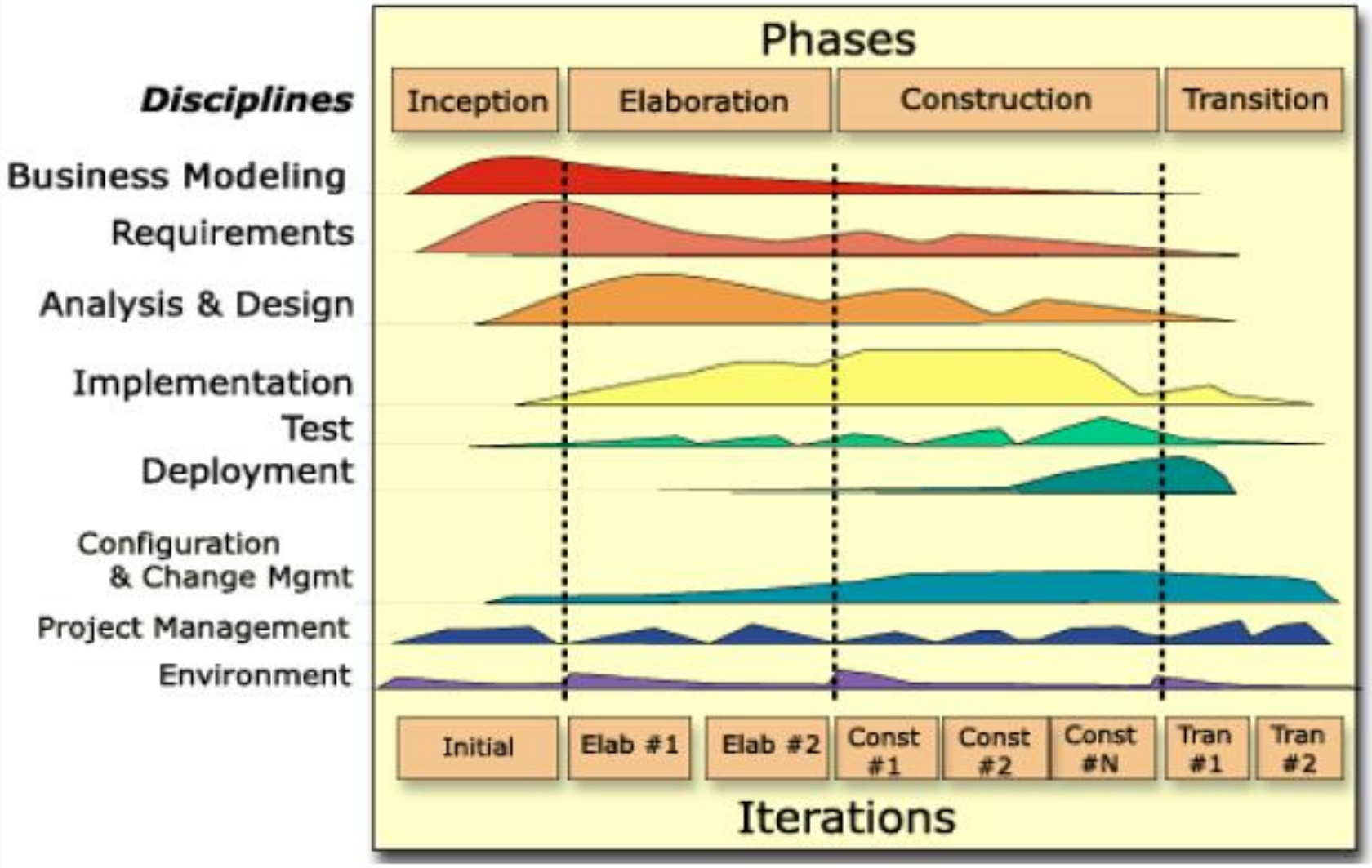
Prototype model



Prototype model

- A prototyping model suggest that before carrying out the development of the actual software, a working prototype of the system should be built.
- A prototype is a toy implementation of the system.
- Prototype is a working model of software with some limited functionality.
- Prototyping is used to allow the users evaluate the developer proposals and try them out before implementation.
- By using this prototype, customer can understand the requirements of desired system and also the customer can get an “actual feel” of the system. It is an attractive idea for complex and bigger systems.

The Rational Unified Process



The Rational Unified Process

- **Rational Unified Process(RUP)** is a software development process from Rational, a division of IBM.
- It divides the development process into four distinct phases that each involve business modeling, analysis and design, implementation, testing, and deployment. The four phases are:
  - **Inception** - The idea for the project is stated. The development team determines if the project is worth pursuing and what resources will be needed.
  - **Elaboration** - The project's architecture and required resources are further evaluated. Developers consider possible applications of the software and costs associated with the development.
  - **Construction** - The project is developed and completed. The software is designed, written, and tested.
  - **Transition** - The software is released to the public. Final adjustments or updates are made based on feedback from end users.

Microsoft Solution Framework

Microsoft Solution Framework

- Microsoft Solutions Framework (MSF) is a set of principles, models, disciplines, concepts, and guidelines for delivering information technology solutions from Microsoft. MSF is not limited to developing applications only, it is also applicable to other IT projects like deployment, networking or infrastructure projects. MSF does not force the developer to use a specific methodology (Waterfall, Agile) but lets them decide what methodology to use.