Software Engineering(303105253)

Computer Science & Engineering

Module-1

Introduction:

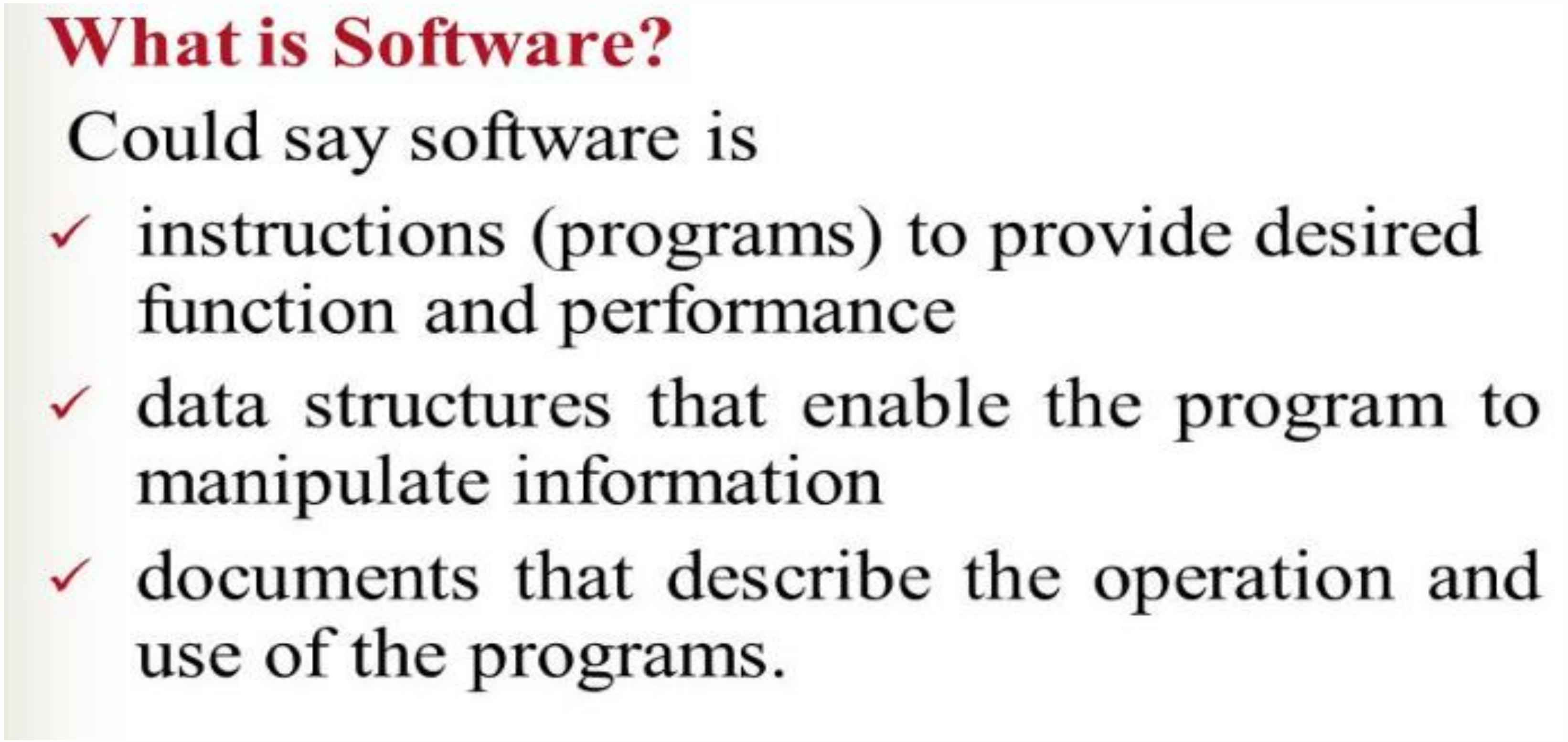
- ✓ Study of Different Models,
- ✓ Software Characteristics,
- ✓ Components,
- ✓ Applications,
- ✓ Layered Technologies,
- ✓ Processes, Methods and Tools,
- ✓ Generic View Of Software Engineering,
- ✓ Process Models- Waterfall model, Incremental,
- ✓ Evolutionary process models- Prototype, Spiral And Concurrent Development Model
- ✓ Agile Development : Agility and Agile Process model,
- ✓ Extreme Programming,
- ✓ Other process models of Agile Development and Tools

INTRODUCTION: STUDY OF DIFFERENT MODELS

Definition: Software Engineering is a framework for building software and is an engineering approach to software development.

Software is defined as: Instructions + Data Structures + Documents

SOFTWARE?



What is Software?

Could say software is

- ✓ instructions (programs) to provide desired function and performance
- ✓ data structures that enable the program to manipulate information
- ✓ documents that describe the operation and use of the programs.

Software Engineering?

What is Engineering?
Application of science, tools& methods to find cost effective solution to the problem.

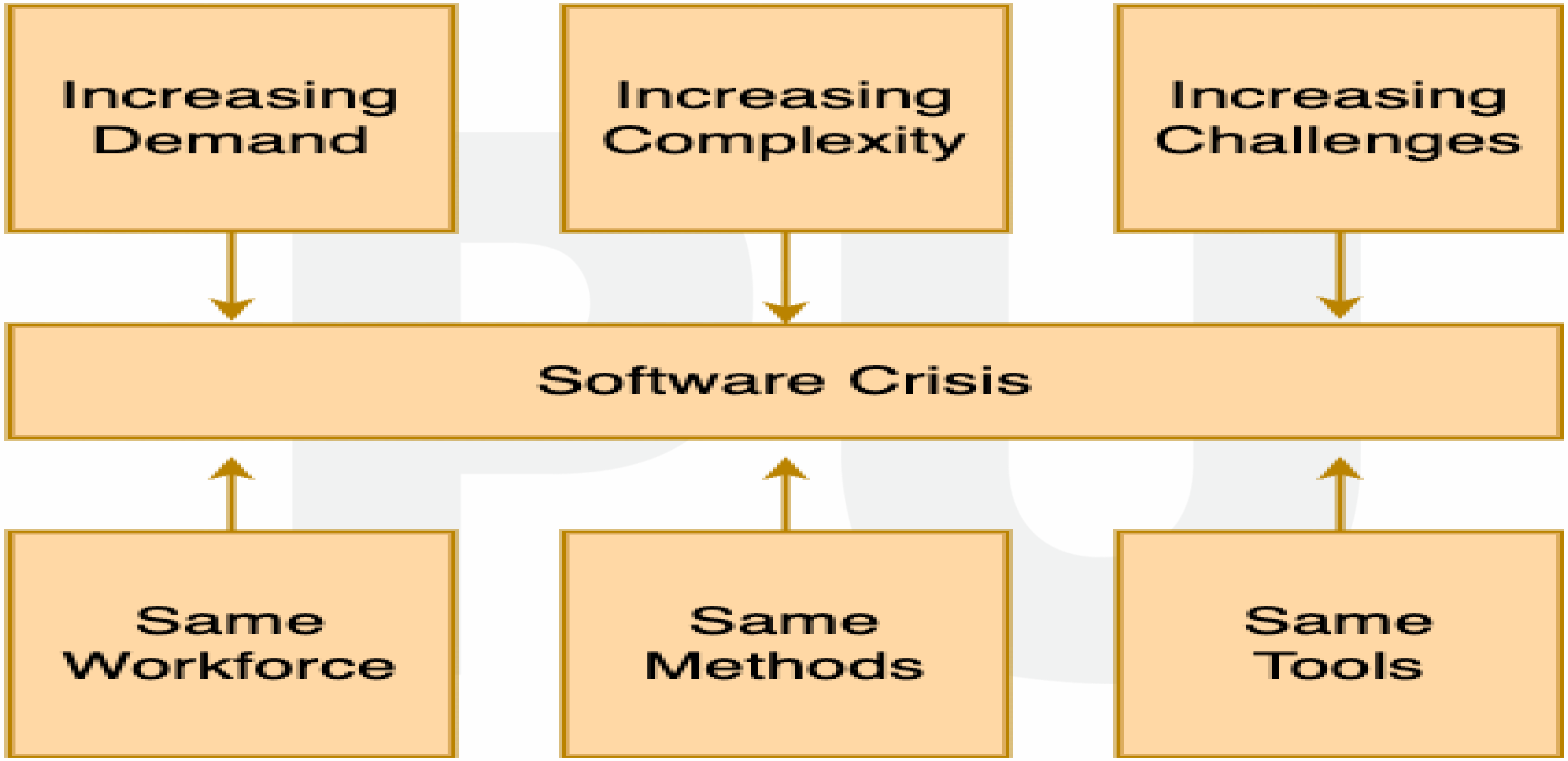
Software Engineering?
It is a systematic, discipline & quantifiable approach for the development, operation and maintenance of software.

Software Crisis

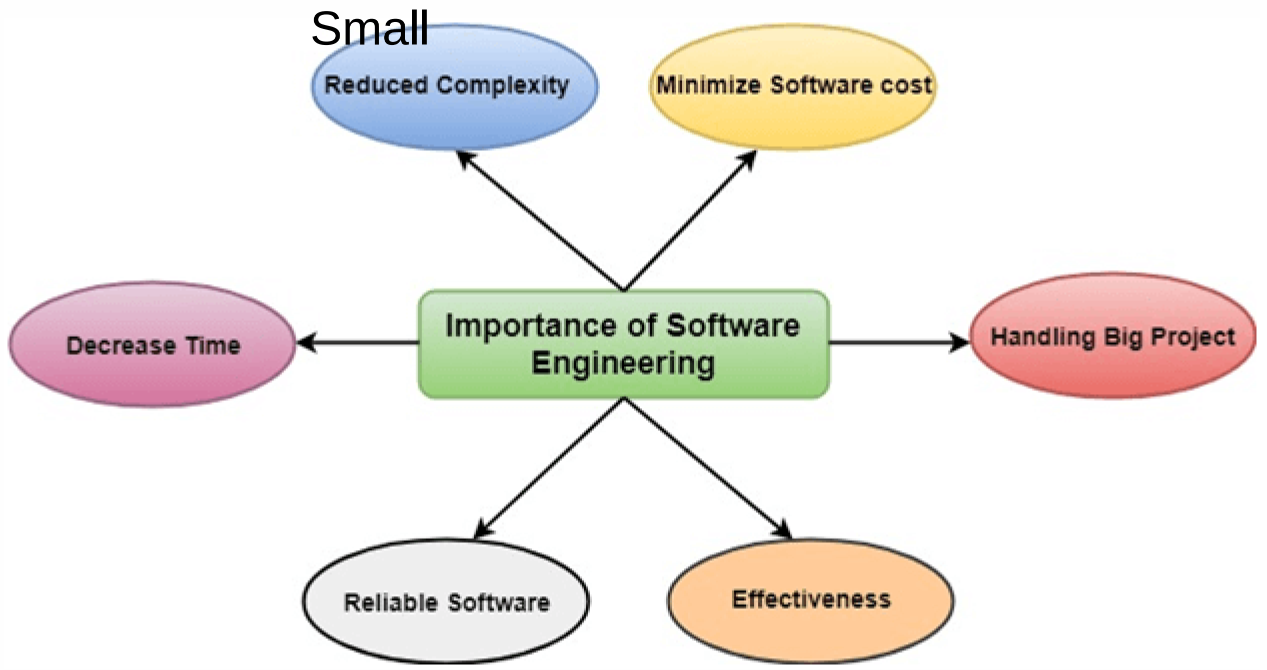
It is used in early days computing science for the difficulty of writing useful & efficient programme in efficient time.

Causes:

Project running over budget
Project running over time
S/w is inefficient
s/w low quality


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

Importance of Software Engineering



Cont.,

- **Reduces complexity:** Big software is always complicated and challenging to progress. Software engineering has a great solution to reduce the complication of any project. **Software engineering divides big problems into various small issues**. And then start solving each small issue one by one. All these small problems are solved independently to each other.
- **To minimize software cost:** Software needs a lot of hardwork and software engineers are highly paid experts. A **lot of manpower** is required to develop software with a large number of codes. But in software engineering, programmers project everything and decrease all those things that are not needed. In turn, the cost for software productions becomes less as compared to any software that does not use software engineering method.
- **Handling big projects:** Big projects are not done in a couple of days, and they need lots of patience, planning, and management. And to invest six and seven months of any company, it requires heaps of **planning, direction, testing, and maintenance**. No one can say that he has given four months of a company to the task, and the project is still in its first stage. Because the company has provided many resources to the plan and it should be completed. So to handle a big project without any problem, the company has to go for a software engineering method.
- **Reliable software: Software should be secure**, means if you have delivered the software, then it should work for at least its given time or subscription. And if any bugs come in the software, the company is responsible for solving all these bugs. Because in software engineering, testing and maintenance are given, so there is no worry of its reliability.

Types of Software engineers

- ✓ **Front-End Engineer** : HTML, CSS, and JavaScript
- ✓ **Back-End Engineer** : Java, Python, PHP, Ruby, or Node.js, DB(SQL)
- ✓ **Full Stack Engineer** : Java, Python, PHP, Ruby, or Node.js
- ✓ **Software Engineer in Test (QA Engineer)**: Selenium, Appium, Cypress, Playwright, Puppeteer, WebdriverIO
- ✓ **Software Development Engineer in Test (SDET):** Automation Frameworks
- ✓ **DevOps Engineer** : Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP).
- ✓ **Security Engineer** : AWS Security, Azure Security, or Google Cloud Security
- ✓ **Data Engineer**: Data Warehousing, Database management, and Data Mining(big data technologies such as Hadoop, Spark, or Kafka)
- ✓ **Cloud Architect** : AWS, Azure, or Google Cloud Platform

APPLICATIONS

1.Mobile App Development

mobile app development is one of the most prominent applications of software engineering. from social networking and entertainment to productivity and e-commerce, mobile apps have become integral to our daily lives. software engineers leverage their expertise to design, develop, and deploy user-friendly and feature-rich mobile applications across platforms like ios and android.

2. Web Development

Web development encompasses the creation of websites and web applications using programming languages, frameworks, and tools. Software engineers utilize their skills to build interactive websites, e-commerce platforms, content management systems (CMS), and more. They focus on optimizing performance, security, and user experience to ensure seamless functionality across different devices and browsers.

Cont.,

4. Artificial Intelligence and Machine Learning

Artificial intelligence (AI) and machine learning (ML) are revolutionizing industries by enabling computers to perform tasks that traditionally required human intelligence. Software engineers play a crucial role in developing AI-powered applications and systems, including chatbots, recommendation engines, autonomous vehicles, and predictive analytics tools. They leverage algorithms, data analysis techniques, and programming languages to create intelligent solutions that drive innovation and efficiency.

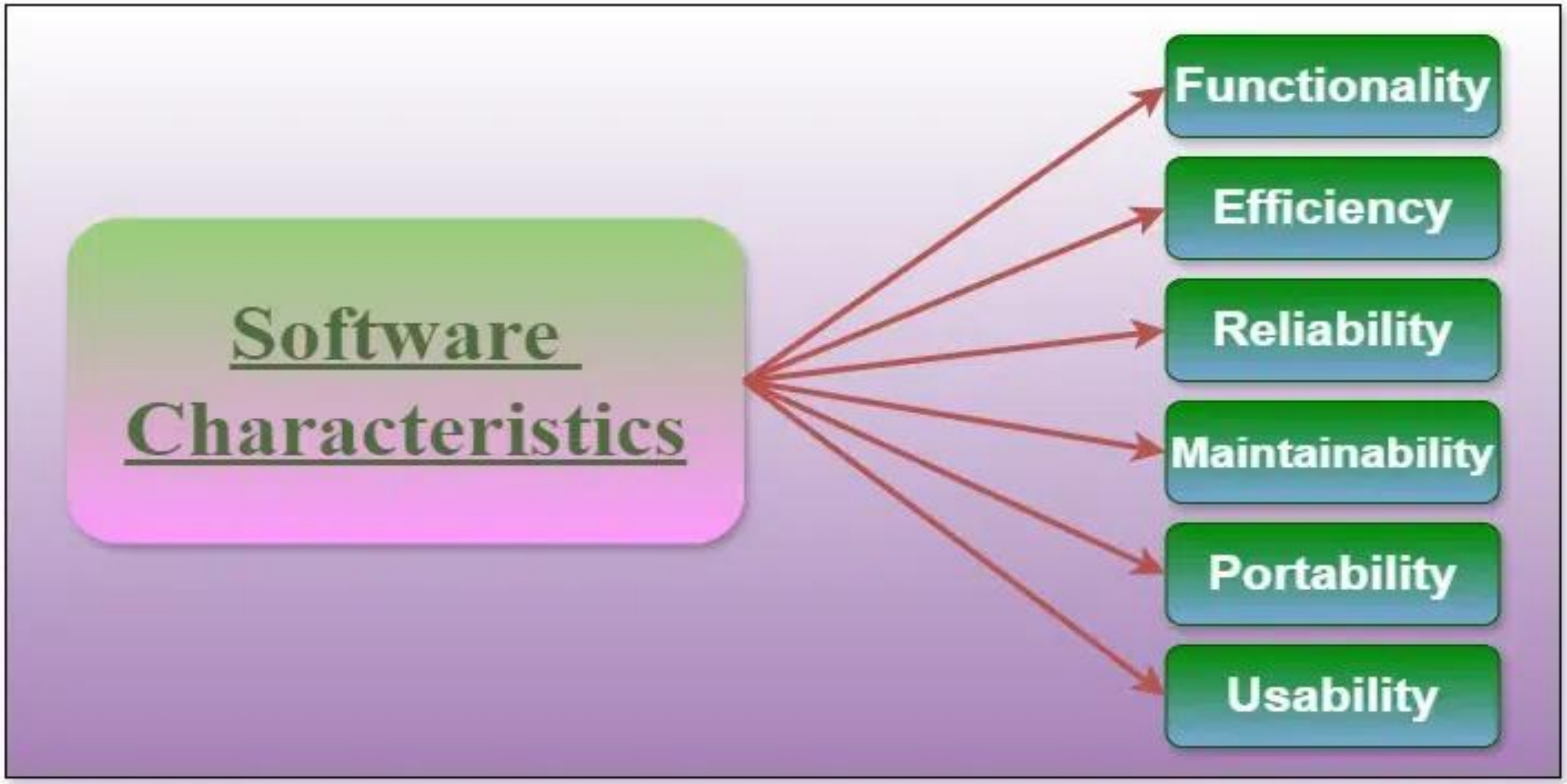
5. Space Exploration and Aerospace

Software engineering plays a vital role in space exploration and aerospace industries, where reliability and precision are paramount. Software engineers develop mission-critical software for spacecraft, satellites, and ground control systems, ensuring the success of space missions and the safety of astronauts. They focus on fault tolerance, real-time processing, and system resilience to withstand the harsh conditions of space.

CHARACTERISTICS OF SOFTWARE

- Software is developed or engineered, but it is not manufactured in the classical sense.
- Software does not wear out, but it deteriorates due to change.
- Software is custom built rather than assembling existing components.

Components of Software Characteristics



Functionality:

- It refers to the degree of performance of the software against its intended purpose.
- Functionality **refers to the set of features and capabilities** that a software program or system provides to its users. Examples of functionality in software include:
    - Data storage and retrieval
    - Data processing and manipulation
    - User interface and navigation
    - Communication and networking
    - Security and access control
    - Reporting and visualization
    - Automation and scripting

Reliability:

A set of attributes that bears on the capability of software to maintain its level of performance under the given condition for a stated period of time. Examples of factors that can affect the reliability of software include:

- ➢ Bugs and errors in the code
- ➢ Lack of testing and validation
- ➢ Poorly designed algorithms and data structures
- ➢ Inadequate error handling and recovery
- ➢ Incompatibilities with other software or hardware

Efficiency:

It refers to the ability of the software to use system resources in the most effective and efficient manner. The software should make effective use of storage space and executive command as per desired timing requirements.

- ➢ Poorly designed algorithms and data structures
- ➢ Inefficient use of memory and processing power
- ➢ High network latency or bandwidth usage
- ➢ Unnecessary processing or computation
- ➢ Unoptimized code

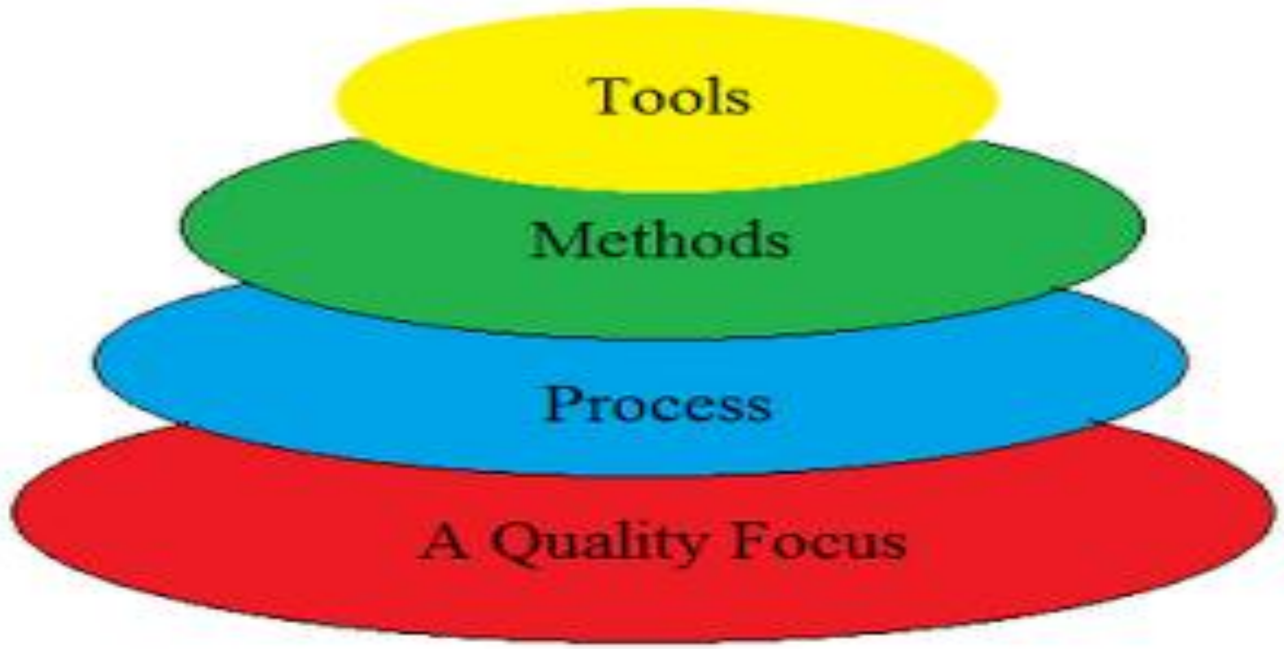
Usability:

It refers to the extent to which the software can be used with ease. the amount of effort or time required to learn how to use the software.

Maintainability:

It refers to the ease with which modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.

LAYERED TECHNOLOGIES



Cont.,

Divided into 4 layers:-

1. A quality Process :-

Any engineering approach must rest on an quality. The "Bed Rock" that supports software Engineering is Quality. Main principle of Software Engineering is Quality Focus.

- An engineering approach must have a focus on quality.
- Total Quality Management (TQM), Six Sigma, ISO 9001, ISO 9000-3, CAPABILITY MATURITY MODEL (CMM), CMMI & similar approaches encourages a continuous process improvement culture

2. Process :-

Foundation for SE is the Process Layer SE process is the glue **that holds all the technology layers together and enables the timely development of computer software.**

It forms the base for management control of software project.

3. Methods :-

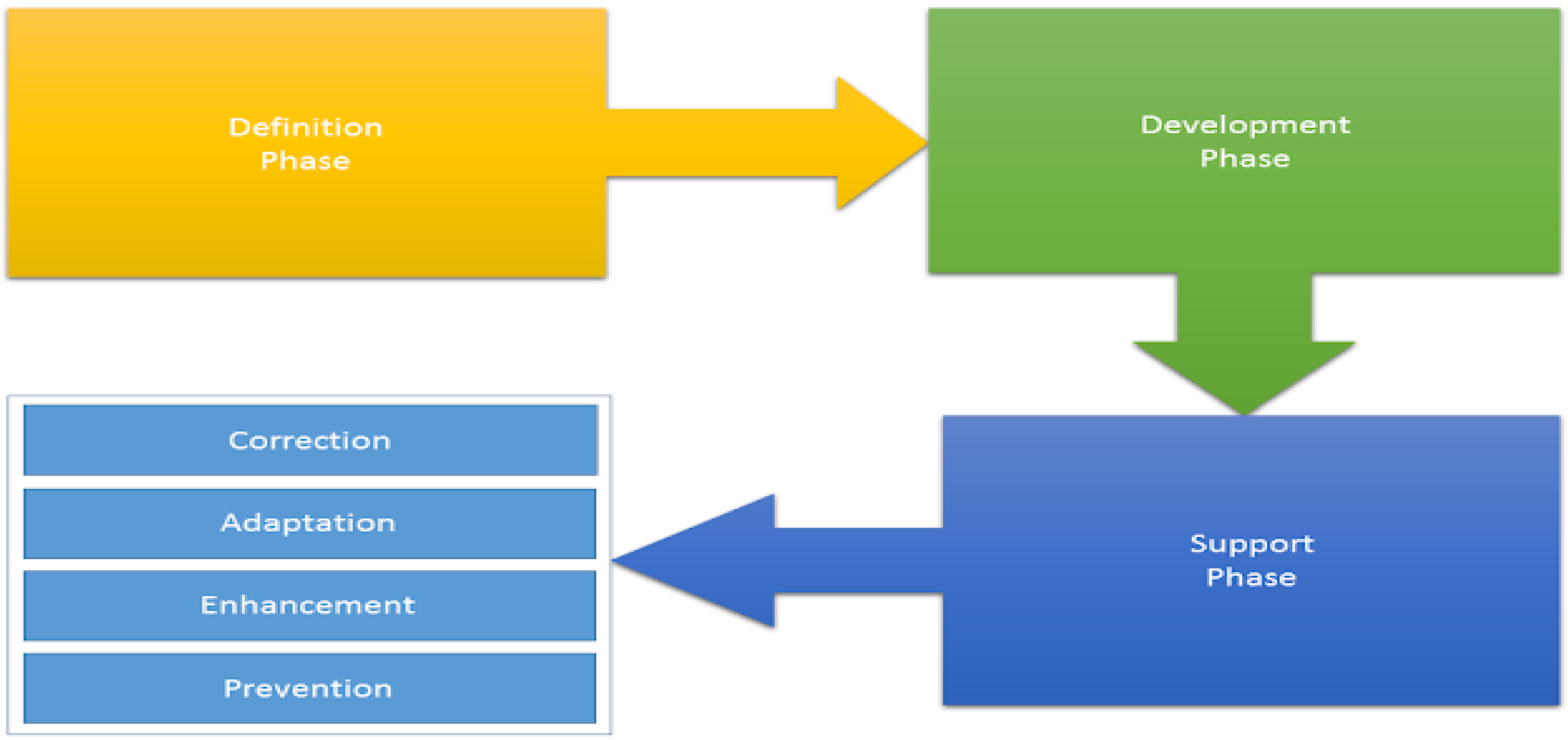
SE methods provide the "Technical Questions" for building Software. **Methods contain a broad array of tasks that include communication requirement analysis, design modeling, program construction testing and support.**

4. Tools :-

SE tools provide automated or semi-automated support for the "Process" and the "Methods".

Tools are combined and interrelated so that information created by one tool can be used by

GENERIC VIEW OF SOFTWARE ENGINEERING



Three Phases of Generic View of Software Engineering

01. Definition Phase:

- The definition phase focuses on “what”.
- what function
- what system behavior can be expected,
- what interfaces are to be established,
- what design constraints exist, and
- what validation criteria are required to define a successful system.

02. Development Phase:

The development phase focuses on “how”.

- how data are to be structured,
- how function is to be implemented within a software architecture,
- how interfaces are to be characterized,
- how the design will be translated into a programming language, and
- how testing will be performed.

03. Support Phase:

The support phase focuses on “change” associated with error correction,
changes due to enhancements brought about by changing customer requirements.

Four types of change are encountered during the support phase:

✓ **Correction:**

Corrective maintenance changes the software to correct defects.

✓ **Adaptation:**

Over time, the original environment, that is, CPU, operating system, business rules etc for which the software was developed is likely to change. flexibility, learning new skills, problem-solving, effective communication, embracing change, and multitasking.

✓ **Enhancement:**

As software is used, the customer/user will recognize additional functions that will provide benefit. Upgrading a phone's camera to improve picture quality.

✓ **Prevention:**

Computer software deteriorates due to change, and because of this, preventive maintenance, often called software re-engineering, must be conducted to enable the software to serve the needs of its end users. nstalling antivirus software to prevent malware attacks.

PROCESS MODELS

A software process model is an abstract representation of a process that presents a description of a process from some particular perspective.

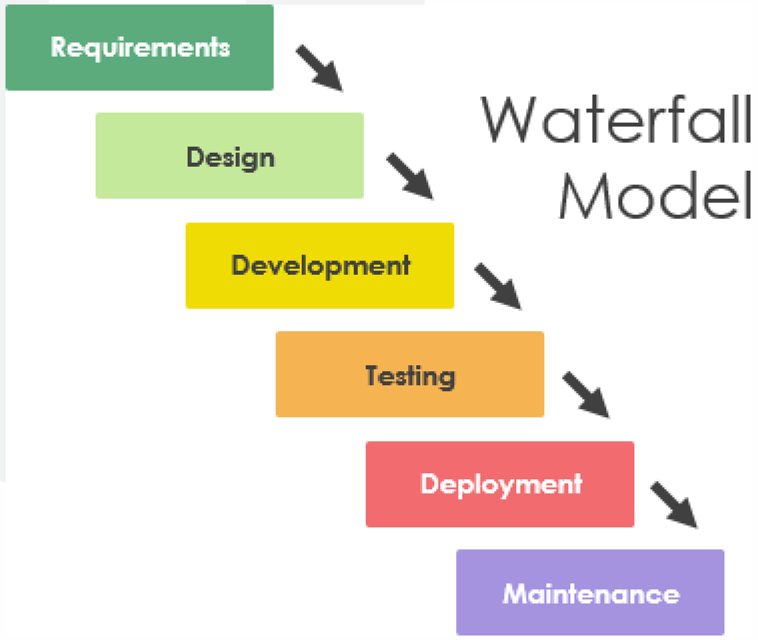
Types of Process Model

- Waterfall model
- V model
- Incremental model
- RAD model
- Agile model
- Iterative model
- Spiral model
- Prototype model

WATERFALL MODEL

The waterfall model is a breakdown of project activities into **linear sequential phases**, where each phase depends on the deliverables of the previous one and corresponds to a specialization of tasks. The approach is typical for certain areas of engineering design.
Eg. Online Banking system

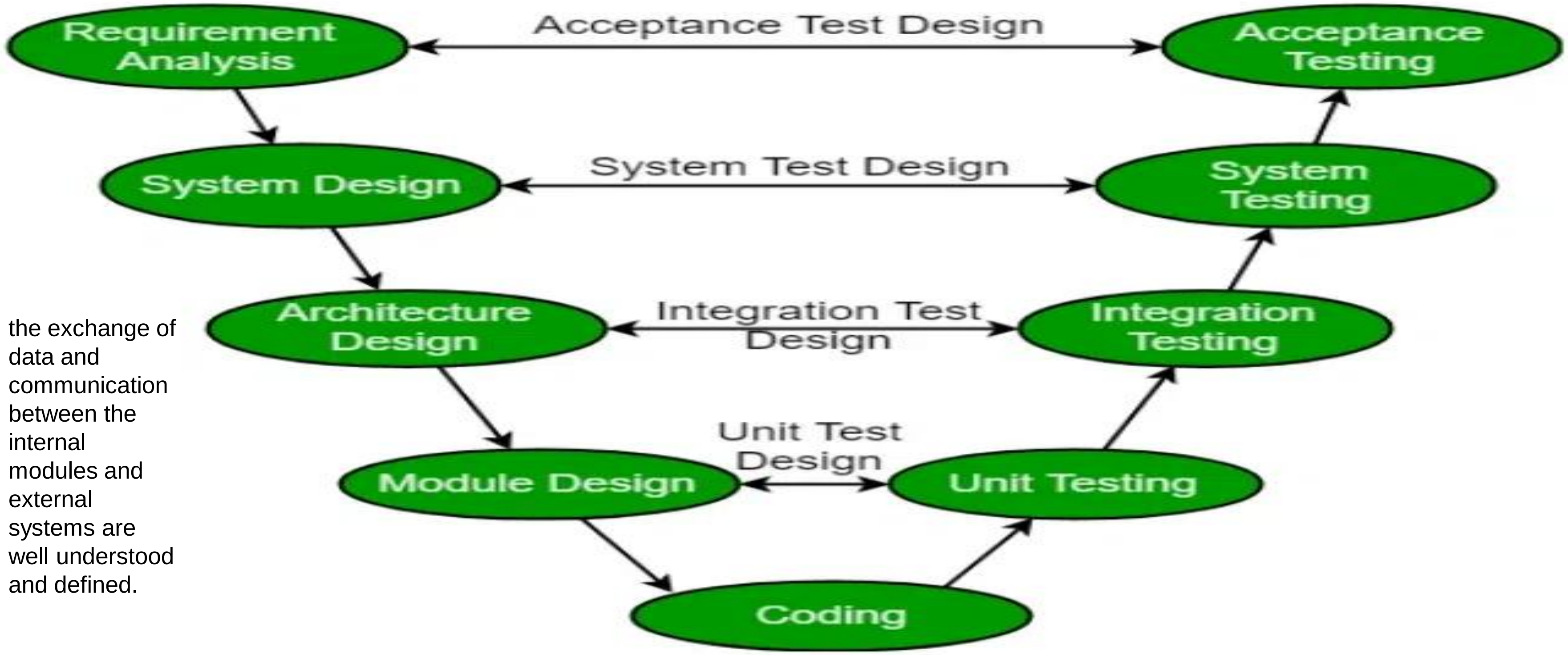
Drawback:
Difficult to accommodate Change Requests



V Model

- It is also known as the Verification and Validation model.
- The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.
- The development of each step is directly associated with the testing phase. **The next phase starts only after completion of the previous phase.**
- Verification helps in examining whether the product is **built right according to requirements**, Verification is the process of checking that software achieves its goal without any bugs. while validation helps in **examining whether the right product is built to meet user needs.**

Requirements Gathering and Analysis: The first phase of the V-Model is the requirements gathering and analysis phase, where the customer's requirements for the software are gathered and analyzed to determine the scope of the project.
Design: In the design phase, the software architecture and design are developed, including the high-level design and detailed design.
Implementation: In the implementation phase, the software is built based on the design.
Testing: In the testing phase, the software is tested to ensure that it meets the customer's requirements and is of high quality.
Deployment: In the deployment phase, the software is deployed and put into use.
Maintenance: In the maintenance phase, the software is maintained to ensure that it continues to meet the customer's needs and expectations.



the exchange of data and communication between the internal modules and external systems are well understood and defined.

Pros and Cons

Pros:

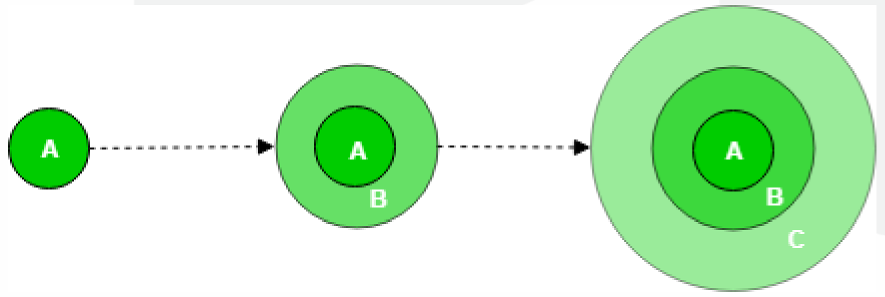
- ✓ This is a highly disciplined model and Phases are completed one at a time.
- ✓ V-Model is used for small projects where project requirements are clear.
- ✓ Simple and easy to understand and use.
- ✓ This model focuses on verification and validation activities early in the life cycle

Cons:

- ✓ It is not good for complex and object-oriented projects.
- ✓ It is not suitable for projects where requirements are not clear and contain a high risk of changing.
- ✓ Time-Consuming:

Incremental Process Model

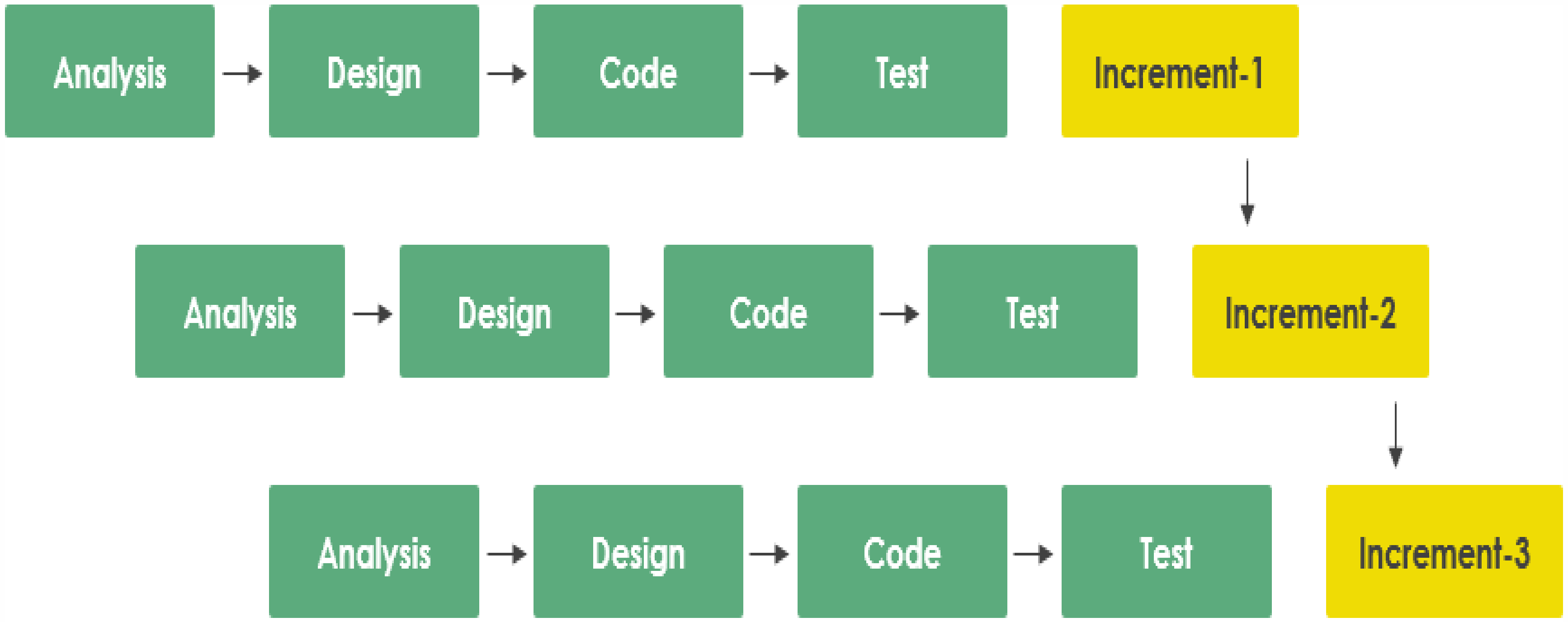
First, a simple working system implementing only a few basic features is built and then that is delivered to the customer. Then thereafter many successive iterations/ versions are implemented and delivered to the customer until the desired system is released.



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.

Cont.,



Incremental Model

Advantages of the Incremental Process Model

Prepares the software fast.

Clients have a clear idea of the project.

Changes are easy to implement.

Provides risk handling support, because of its iterations.

Disadvantages of the Incremental Process Model

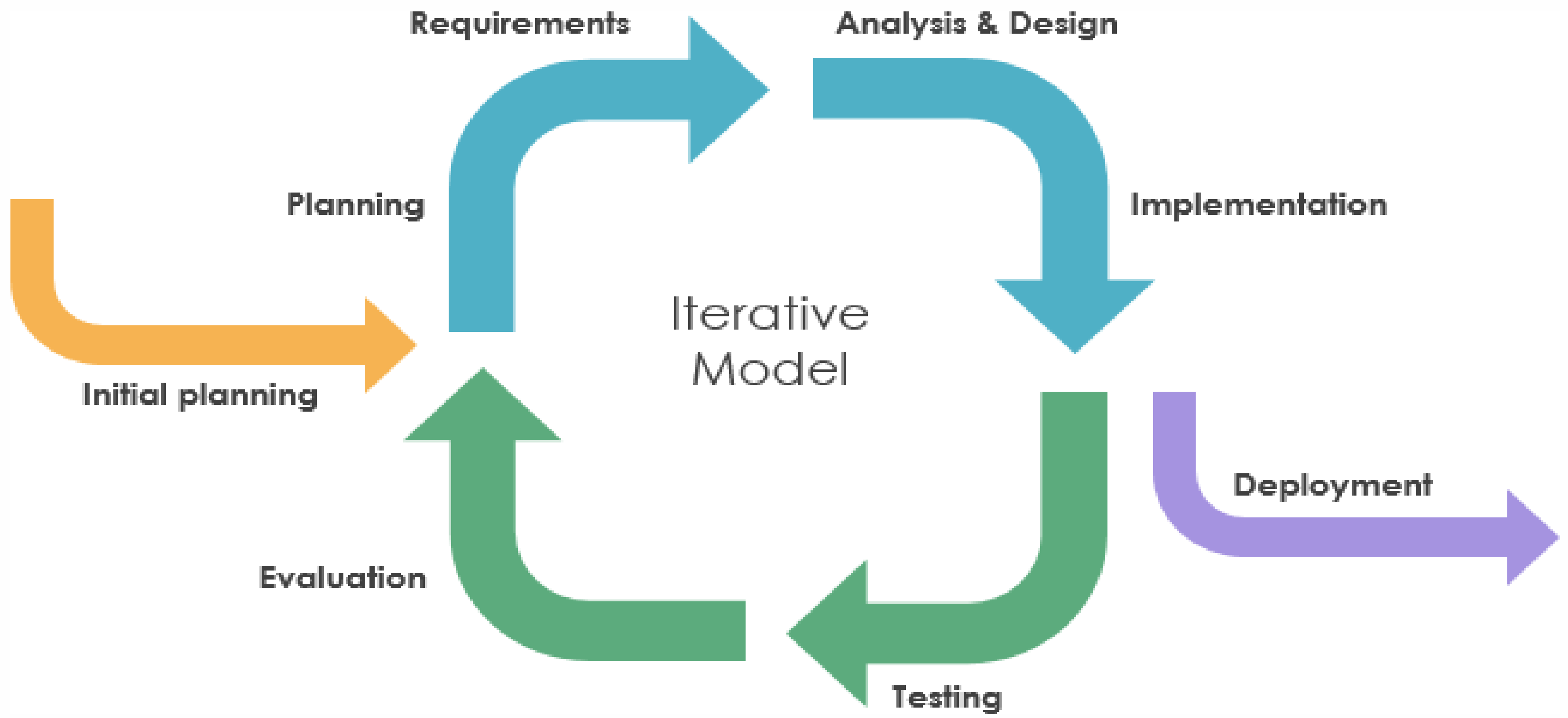
A good team and proper planned execution are required.

Because of its continuous iterations the cost increases.

Iterative Model

- An iterative life cycle model **does not attempt to start with a full specification of requirements.**
- that combines the sequential steps of the traditional Waterfall Model with the flexibility of iterative design.

Cont.,


Requirements
Analysis & Design
Planning
Implementation
Iterative Model
Initial planning
Deployment
Evaluation
Testing

- **Requirements Gathering**: This is the first stage where the business owners and developers meet to discuss the goals and requirements of the website.
- **Design:** In this stage, the developers create a preliminary design of the website based on the requirements gathered in stage 1.
- **Implementation:** In this stage, the developers begin to build the website based on the design created in stage 2.
- **Testing**: Once the website has been built, it is tested to ensure that it meets the requirements and functions properly.
- **Deployment**: The website is then deployed and made live to the public.
- **Review and Improvement:** After the website has been live for a while, the business owners and developers review its performance and make any necessary improvements.
- This process is repeated until the website meets the needs and goals of the business. Each iteration builds upon the previous one, allowing for continuous improvement and iteration until the final product is complete.

Drawbacks of Iterative Waterfall Model

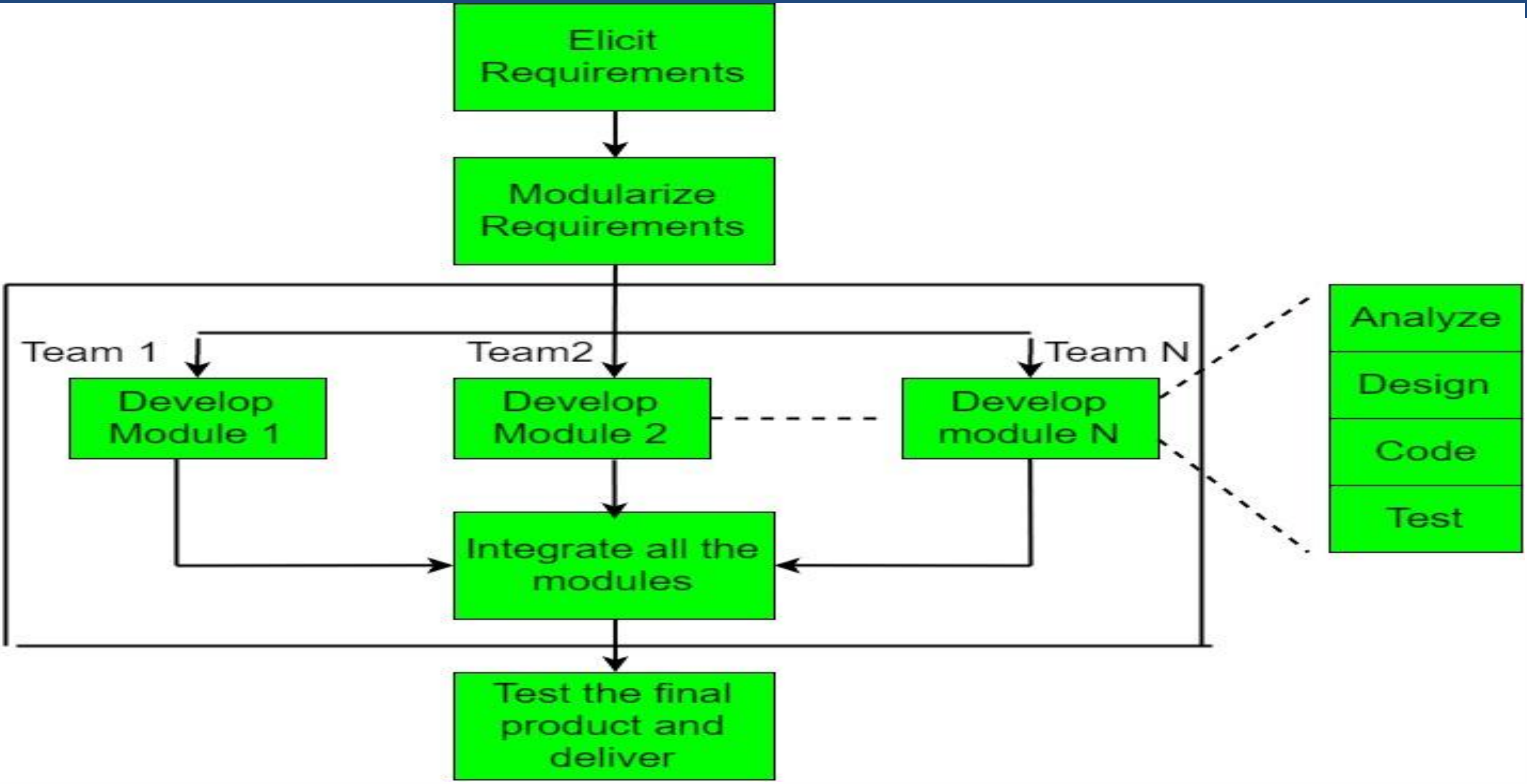
Difficult to incorporate change requests

Risk handling not supported

RAD model

- 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**.
- RAD is designed to be more flexible and responsive to user feedback and changing requirements throughout the development process.

Cont.,


Elicit Requirements
Modularize Requirements
Team 1
Team2
Team N
Develop Module 1
Develop Module 2
Develop module N
Analyze
Design
Code
Test
Integrate all the modules
Test the final product and deliver

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.

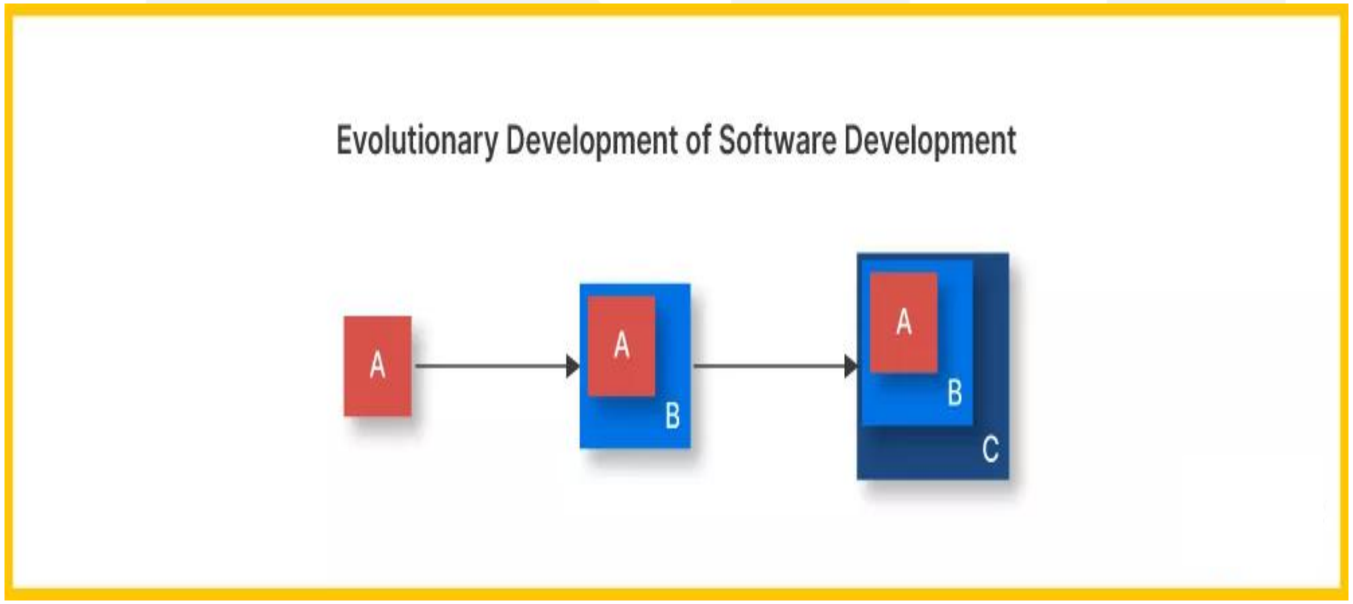
User Description – This phase consists of **taking user feedback and building** the prototype using developer tools. In other words, it includes re-examination and validation of the data collected in the first phase. The dataset attributes are also identified and elucidated in this phase.

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.

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.

Evolutionary process models

Iterative" + "Incremental model" = Evolutionary model.

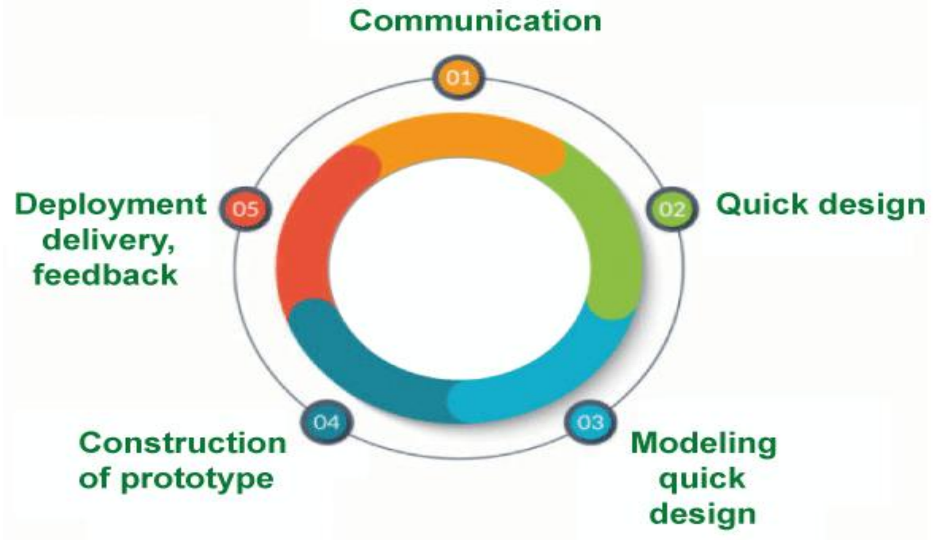


Cont.,

- An evolutionary model involves breaking down the software development process into smaller, manageable increments or iterations.
- Each iteration involves the completion of a subset of the overall software requirements, allowing for continuous testing, feedback, and refinement.

Prototype Model

- We can not identify the input and output of requirement details. We can develop the program quickly with the help of this model.
- We have some different phases of using this prototyping model.



Communication
In this phase, the client and the developer should have a meeting, and in that meeting, they discuss the main objective of creating the project.
Quick design
When we have the requirements known to the developer, then the developer can implement the quick design. The quick design includes implementing important aspects, such as the input and output format of the software. The main focus of this project is on the things visible to the user rather than the detailed plan. With the help of this, we can also create the prototype of the project.

Cont.,

- **Modelling quick design**
  In this phase, we can get a clear idea about the software development because the software is now in the build state. Also, it allows the developer to understand the project's basic requirements.
- **Construction of prototype**
  The customer evaluates the prototype of the project.
- **Deployment, delivery, feedback**
  If the client is unsatisfied with the project, then the developer can recreate the project according to the client's satisfaction. This process repeats until the client is fully satisfied. After the client is happy, the project will be developed based on the final prototype.

Advantages of the Prototyping Model

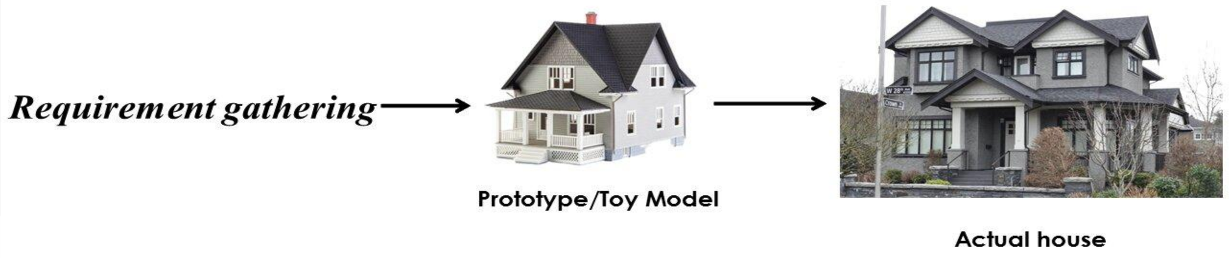
- The prototype model does not need detailed input, output, processes, adaptability of the operating system, or entire machine interaction.
- The **user is fully active** in the development process of this model.
- The development process is the best platform for users to understand the project.
- We can easily detect the error with the help of this prototyping model.
- It also identifies the missing functionality very quickly.
- It also identifies the confusing or difficult functions.

Disadvantages of the Prototyping Model

- In this model, the involvement of the client is greater. So, it is not always considered by the developer.
- The development time is longer, so it is a prolonged process.
- The developer has to perform so many changes according to the client's demand. It disturbs the rhythm of the development process.
- When the user is confused with the prototype, then this prototype will be thrown away.

Example-prototype

- When you go to an architect to build a house, he first asks you about your requirements, like the number of bedrooms, and bathrooms, What type of elevation you need, etc.
- Then, he designs a map or makes a prototype to show you and get your approval. You can suggest some modifications to it. So accordingly, the prototype is refined further. If you approve it, the actual work of building the house starts.



Spiral

- This model is also known as the **risk-driven process model.** With the help of this spiral model, we can generate the software project.
- During the risk analysis, if a risk is found, an alternate solution should be suggested and implemented in the spiral model.
- We can also say that it combines the sequential or waterfall models and the prototype models.

Real-Life Example of Spiral Model: Developing an E-Commerce Website

First Spiral – Planning and Requirements: The initial phase involves gathering basic requirements for the e-commerce website, like product listing, shopping cart, and payment options. The team analyzes any risks, such as security or scalability, and creates a small prototype.

> **Example:** The team builds a simple homepage with a basic product catalog to see how users interact with it and identify any design flaws.

Risk Analysis:

A basic shopping cart and user registration system are added. The payment system is also tested with dummy transactions to ensure security.

Example-Spiral

- The examples of spiral model are that Microsoft used it to develop early versions of Windows.
- The Gantt chart software was also made using spiral model. Game development is another industry who uses spiral model to develop the games.
- As the gaming industry highly depend upon the early versions, at such time spiral model is a solid option.
- With spiral model, the game development industries can get the feedback from their customers at fast and can develop the game as per their convenient.

Advantages of the Spiral Model

- The amount of risk should be reduced in this model.
- This model is best for large and critical projects.
- It provides the developer with solid approval and documentation control.

The software production time is less in this spiral model.

Disadvantages of Spiral Model

- We need a lot of financial support to develop a project.
- We should not use this model for a very small project.

Concurrent Development Model

- We also called this concurrent development model a concurrent model.
- In the first iteration process, all the communication activity has been completed. After completing the first iteration, it exits in the awaiting changes state.
- Also, in the initial phases, The modelling activity was completed. After completion of the initial phases, it goes to the underdevelopment state.
- If the customer wants some changes in the software product, then the modelling activity moves from the under-development state into the awaiting change state.

Stages in the Concurrent Development Model

- System Requirements Gathering
- Design and Prototyping
- Development and Coding
- Integration and Testing
- Deployment and Maintenance

Applications

- **Aerospace and Defense**

For example in aircraft or spacecraft design, concurrent engineering must be employed to guarantee that various systems complement one another and also come up to safety requirements.

- **Automotive Industry:** Concurrent engineering in the automotive industry facilitates the development of multiple components at once of a vehicle for instance engine, safety systems and interfaces. This model assists automotive companies in shortening the overall time it takes to go from a conceptual design to producing the idea.

Parul® University

DIGITAL LEARNING CONTENT

Agile Development :

- **Agile Development:** The meaning of Agile is swift or versatile."**Agile process model**"to facilitate quick project completion.
- Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning.
- Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Agility and Agile Process model

that can be used to generate a large number of ideas related to project requirements.

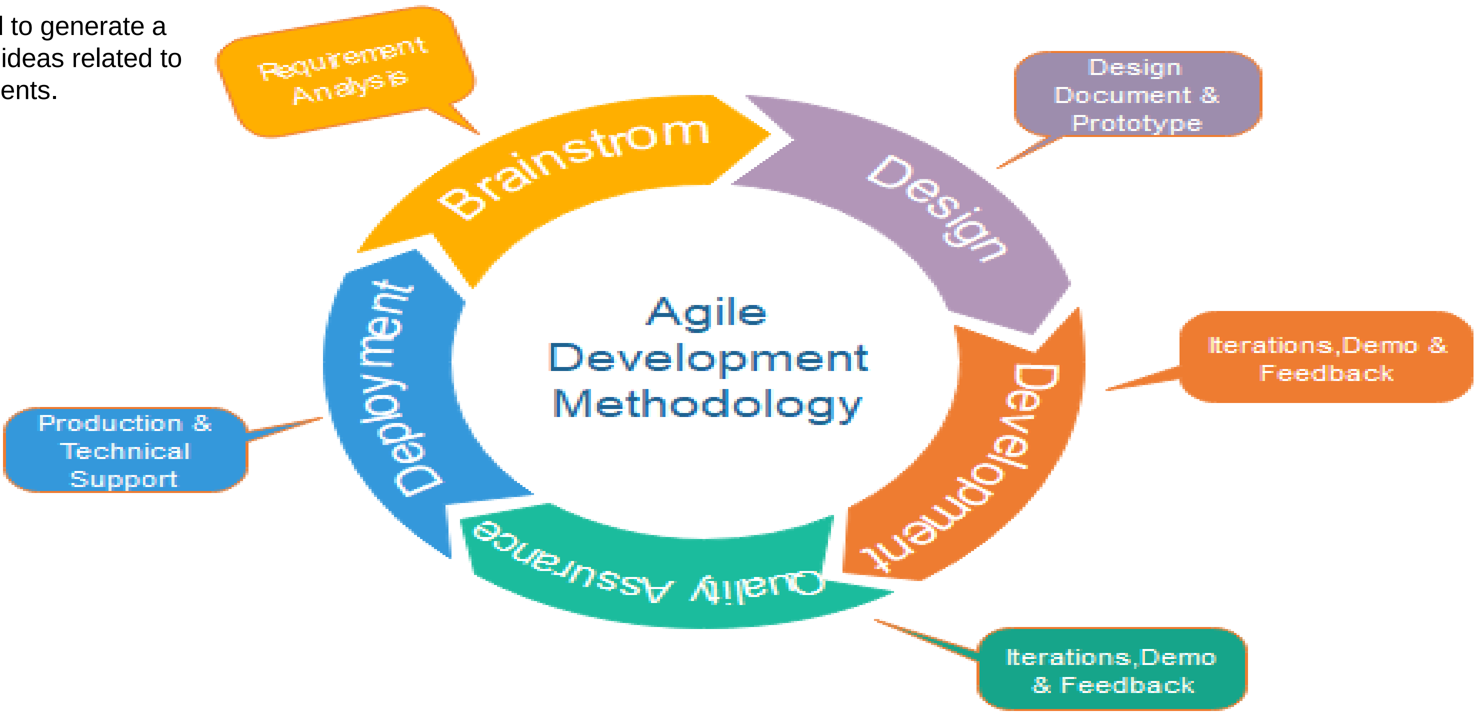

Requirement Analysis
Brainstrom
Design
Design Document & Prototype
Development
Iterations,Demo & Feedback
Quality Assurance
Iterations,Demo & Feedback
Deployment
Production & Technical Support
Agile Development Methodology

Fig. Agile Model

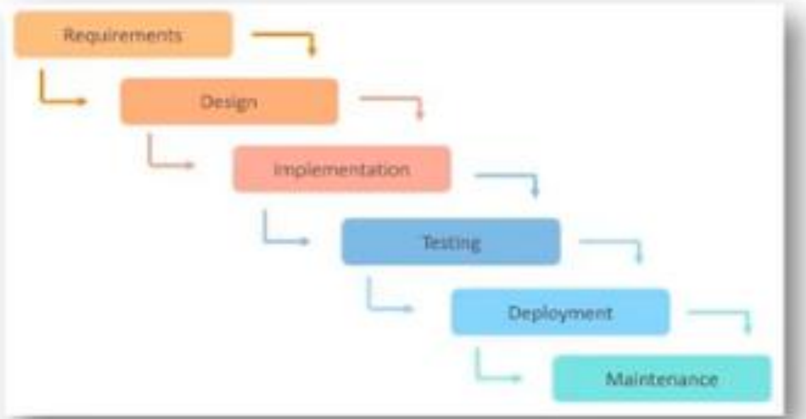
Traditional VS Agile Model Working with Example

For Example:

Instagram Social Application:

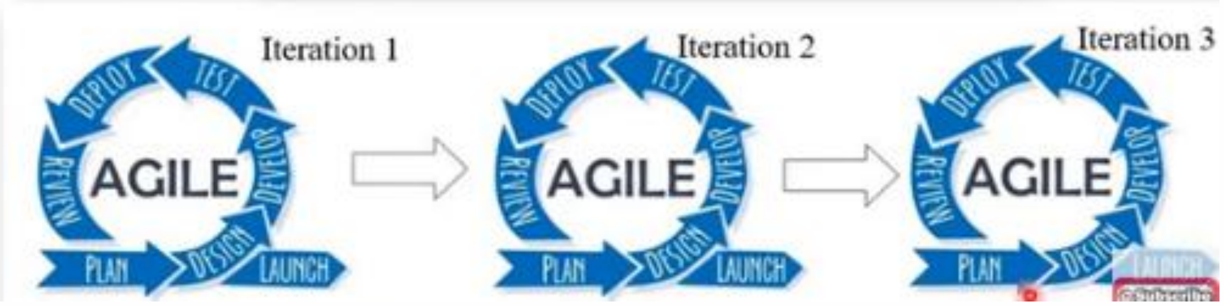
Requirements are:

1. Follow – Unfollow Option
2. Edit profile
3. Search
4. Messaging
5. Post photos
6. Upload story
7. To make reels
8. Go Live

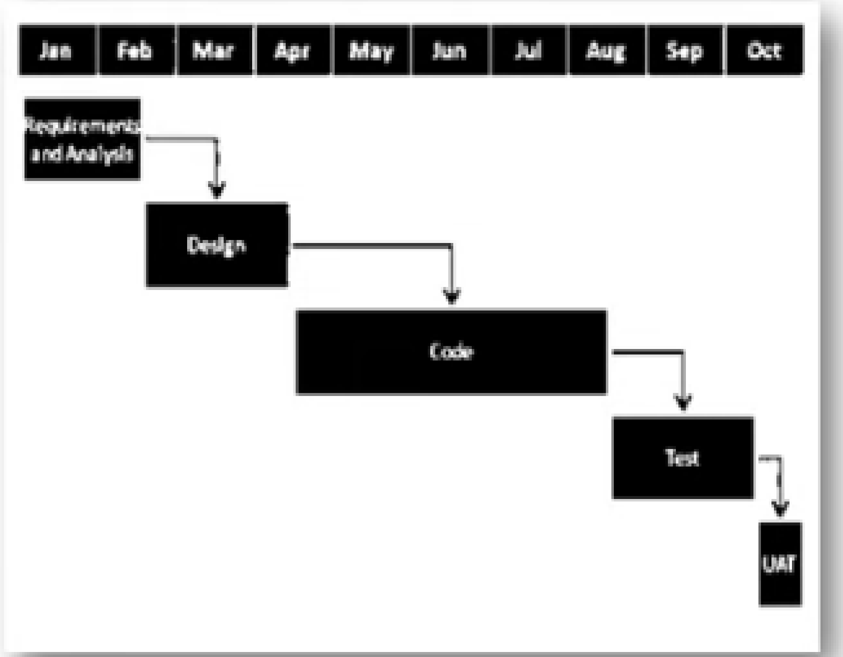


Waterfall Model

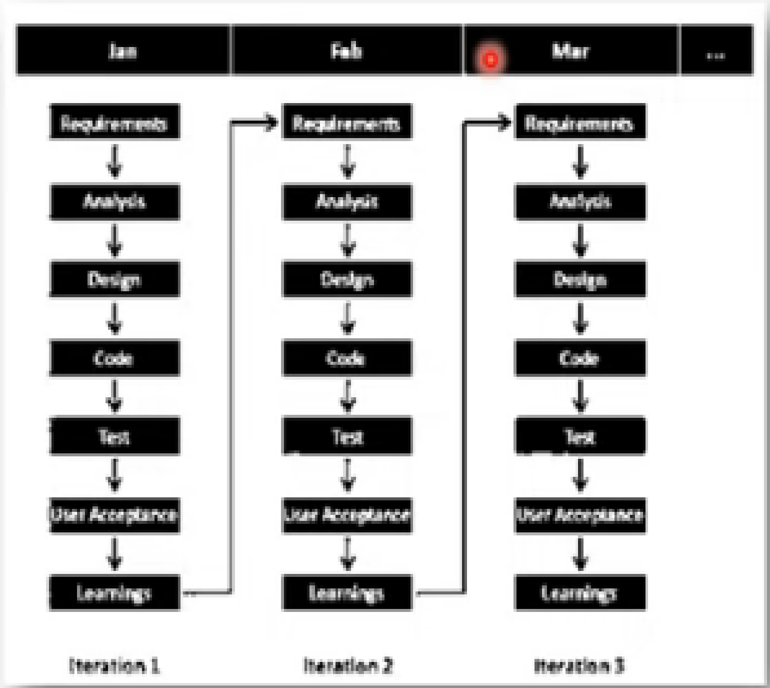
Agile Model



For Example:



Waterfall Model
Time Span



Agile Model
Time Span

Phases of Agile Model:

Following are the phases in the Agile model are as follows:

- Requirements gathering
- Design the requirements
- Construction/ iteration
- Testing/ Quality assurance
- Deployment
- Feedback

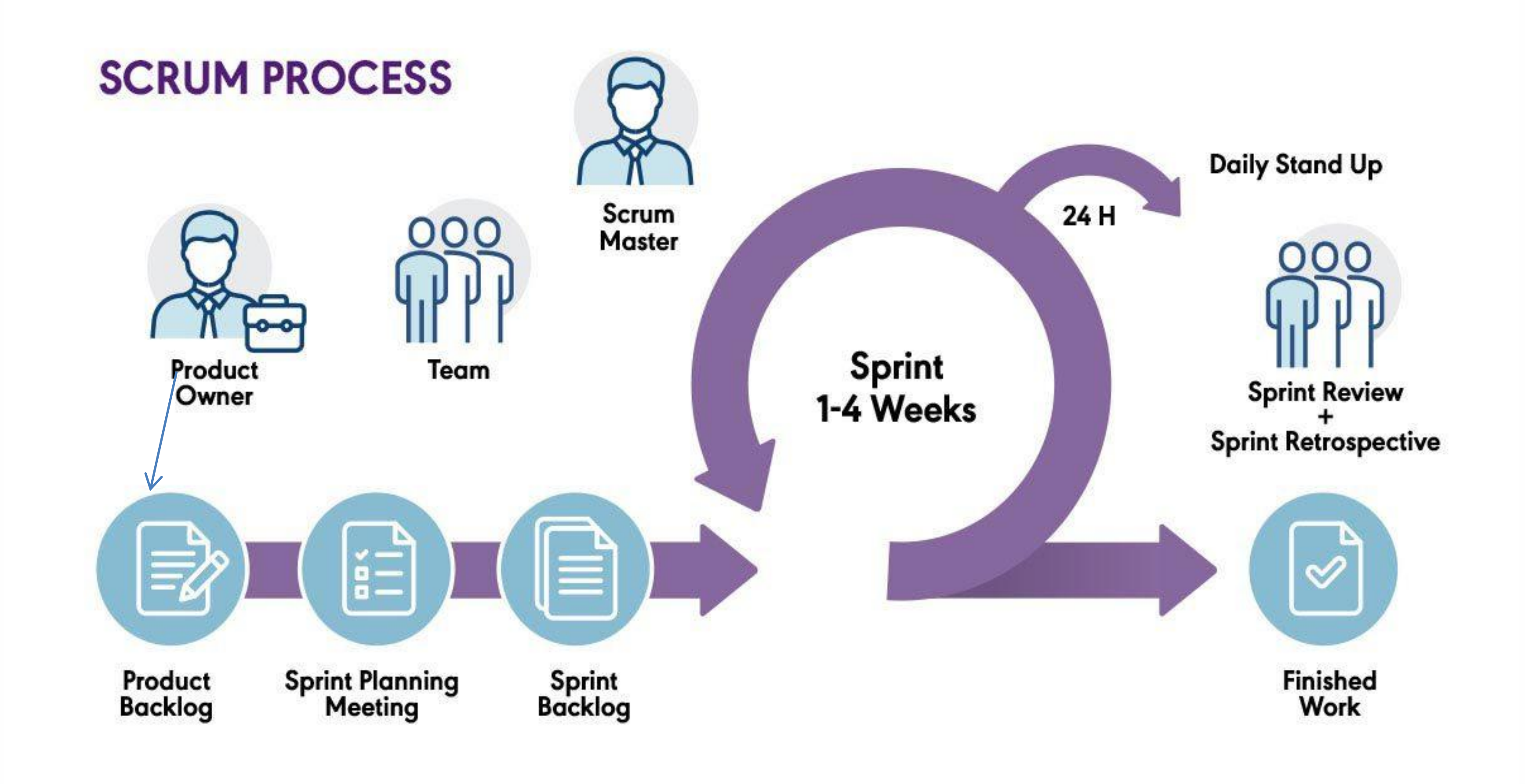
Why Firms Use Agile Methodology

- **Quick response to constant changes** in the market during the software development process
- Focus on **continuous delivery** thereby resulting in quicker software releases
- **Reduction of wasted work / wasted time** resulting in increased productivity and efficiency
- Improved visibility and being able to identify potential show-stoppers before they become an issue

Agile Testing Methods:

- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development
- eXtreme Programming(XP)

▪ Scrum


SCRUM PROCESS
Scrum Master
Product Owner
Team
Daily Stand Up
24 H
Sprint 1-4 Weeks
Sprint Review + Sprint Retrospective
Product Backlog
Sprint Planning Meeting
Sprint Backlog
Finished Work

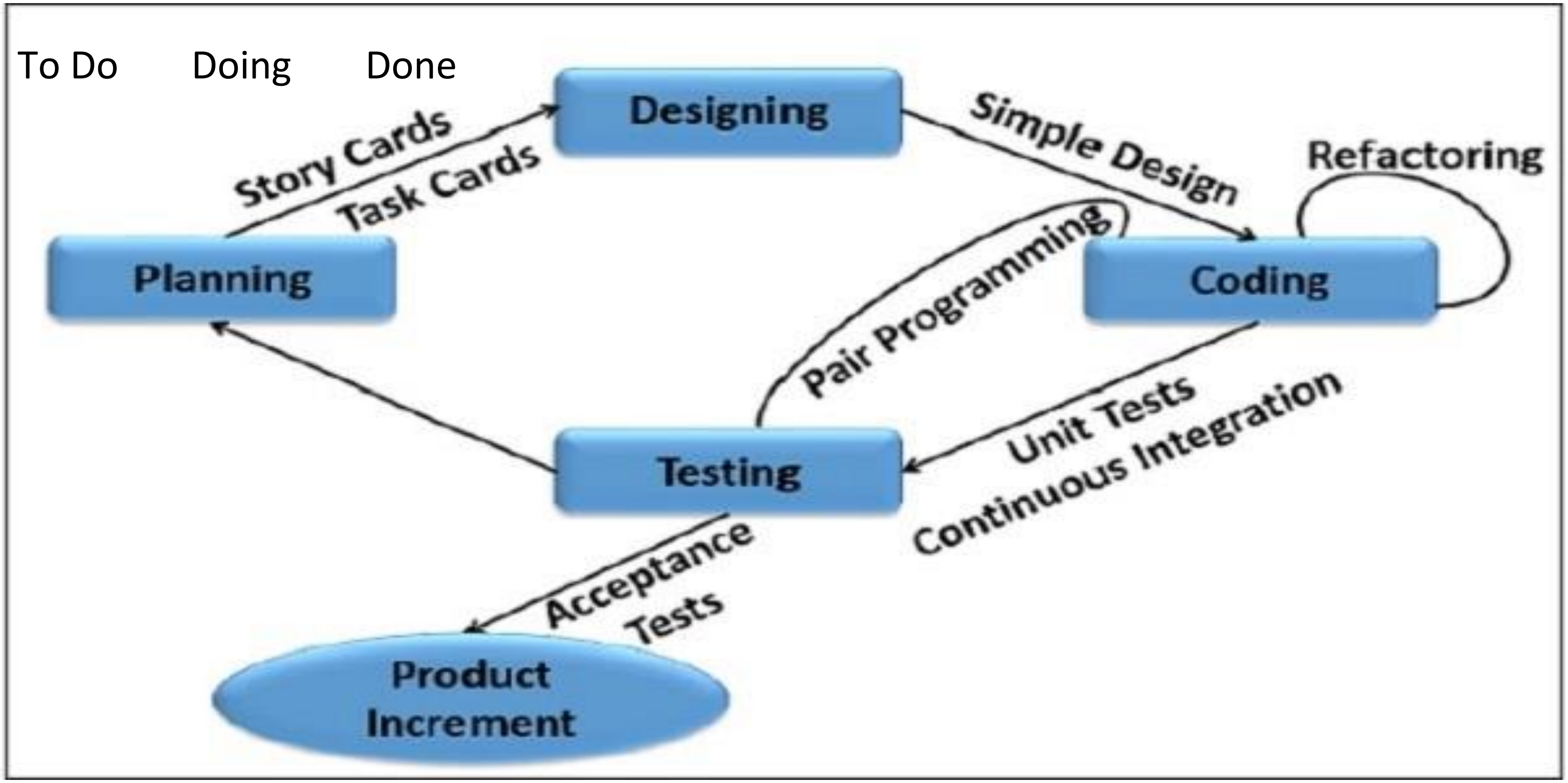
eXtreme Programming(XP)

- ✓ XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop a software.
- ✓ e**X**treme **P**rogramming (XP) was conceived and developed to address the specific needs of software development by small teams in the face of vague and changing requirements.

Cont.,

- Emphasis on continuous feedback from the customer
- Short iterations
- Design and redesign
- Coding and testing frequently
- Eliminating defects early, thus reducing costs
- Keeping the customer involved throughout the development
- Delivering working product to the customer

To Do Doing Done

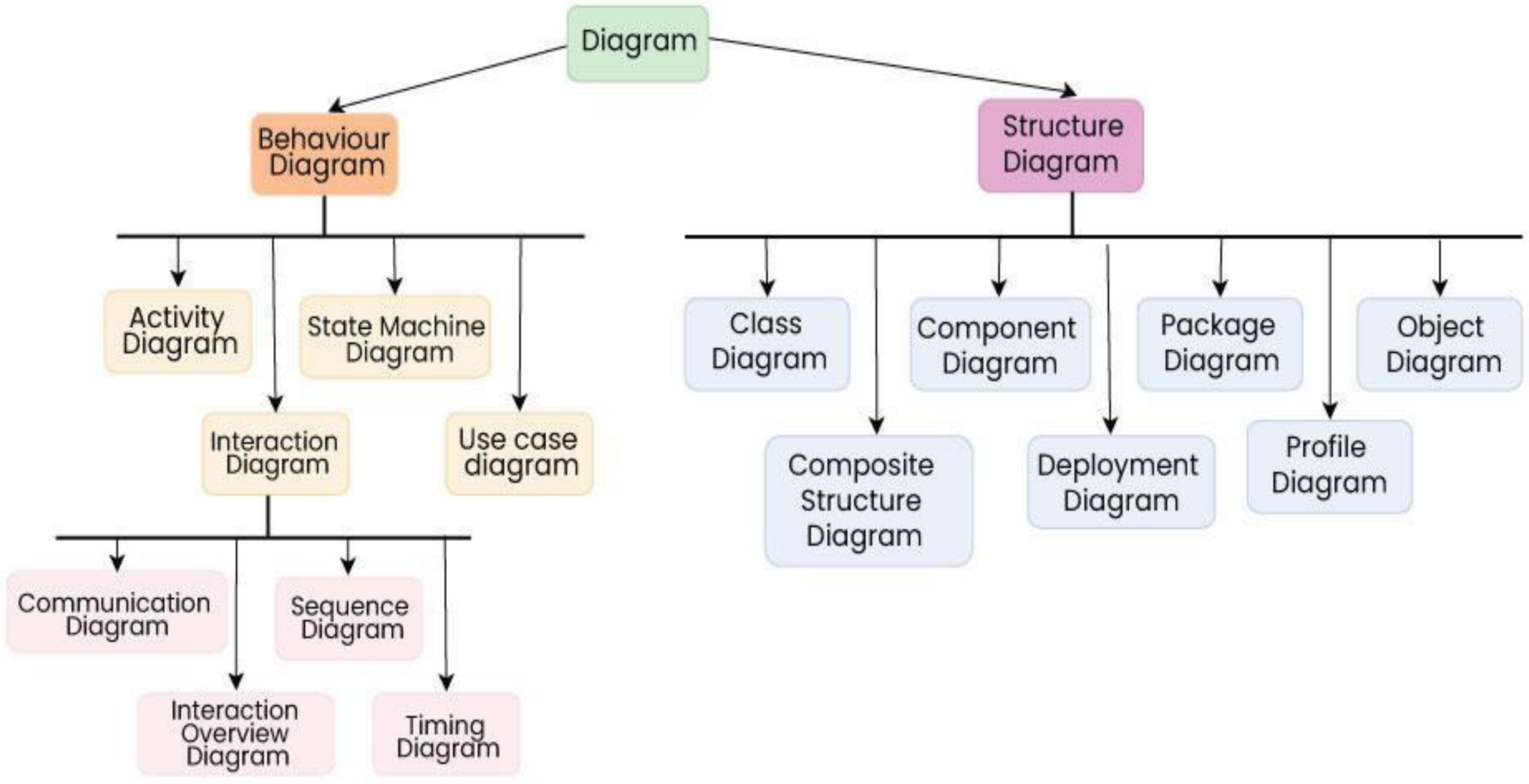


UML

Unified Modeling Language (UML) is a general-purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints

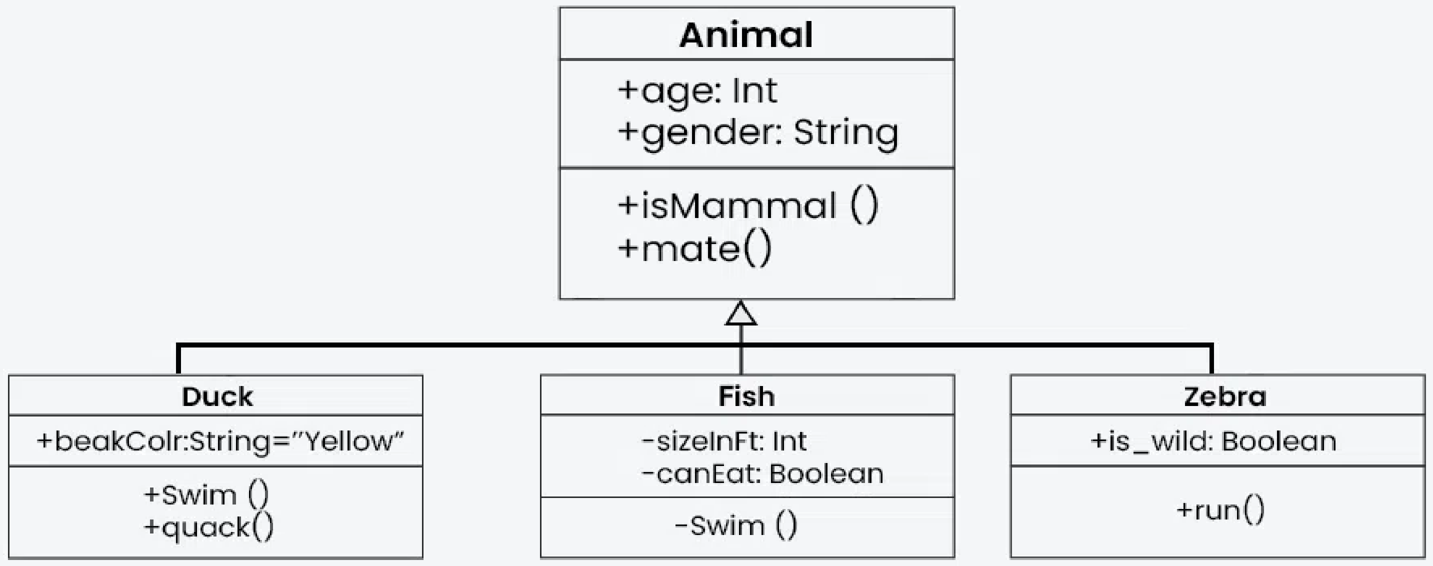
The purpose of a use case diagram in UML is to demonstrate the different ways that a user might interact with a system. Create a professional diagram for nearly any use case using our UML diagram tool.

Types of UML Diagrams



4.1. Class Diagram

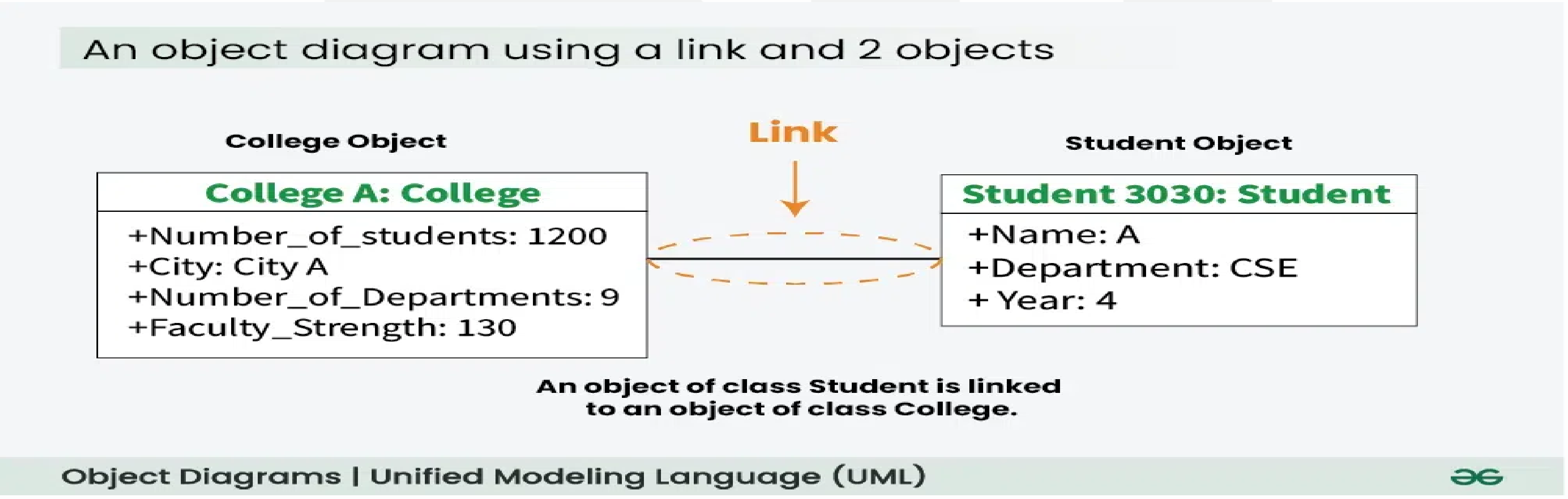
The most widely use UML diagram is the class diagram. It is the building block of all object oriented software systems. We use class diagrams to depict the static structure of a system by showing system’s classes, their methods and attributes



Class Diagram example

object diagram

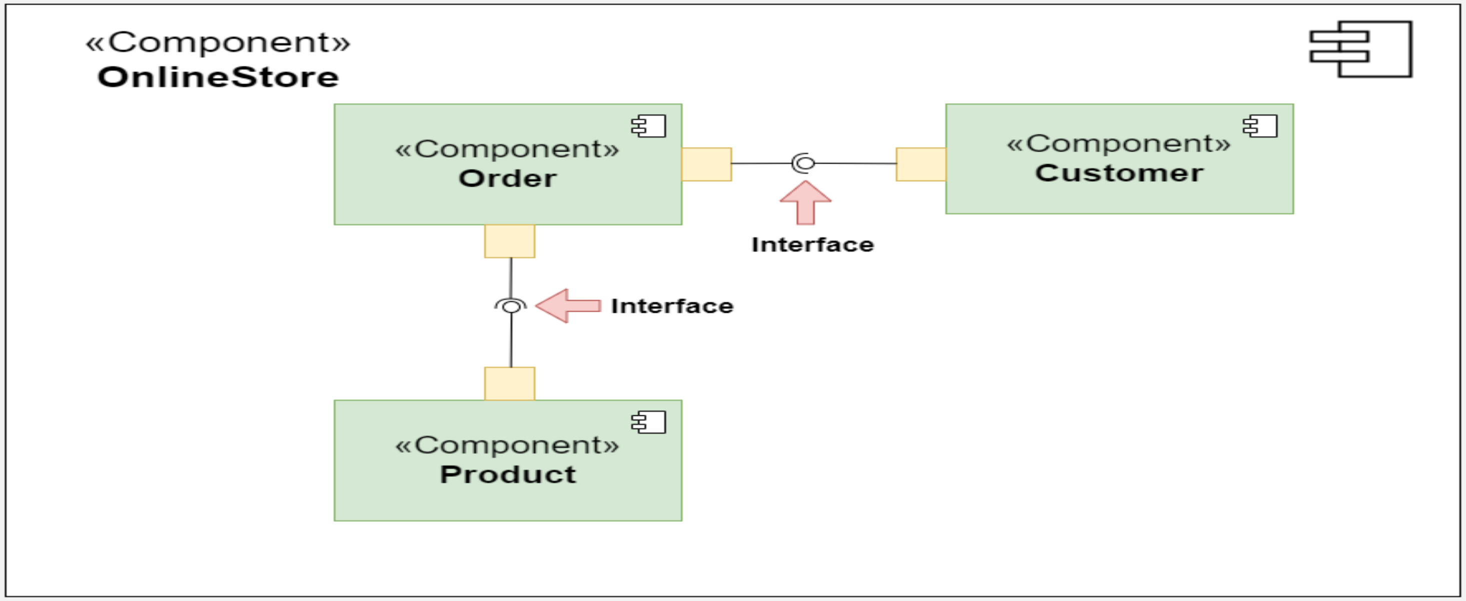
An object diagram is similar to a class diagram except it shows the instances of classes in the system.


An object diagram using a link and 2 objects
College Object
Link
Student Object
College A: College
+Number_of_students: 1200
+City: City A
+Number_of_Departments: 9
+Faculty_Strength: 130
Student 3030: Student
+Name: A
+Department: CSE
+ Year: 4
An object of class Student is linked to an object of class College.
Object Diagrams | Unified Modeling Language (UML)

Component Diagram

Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development.

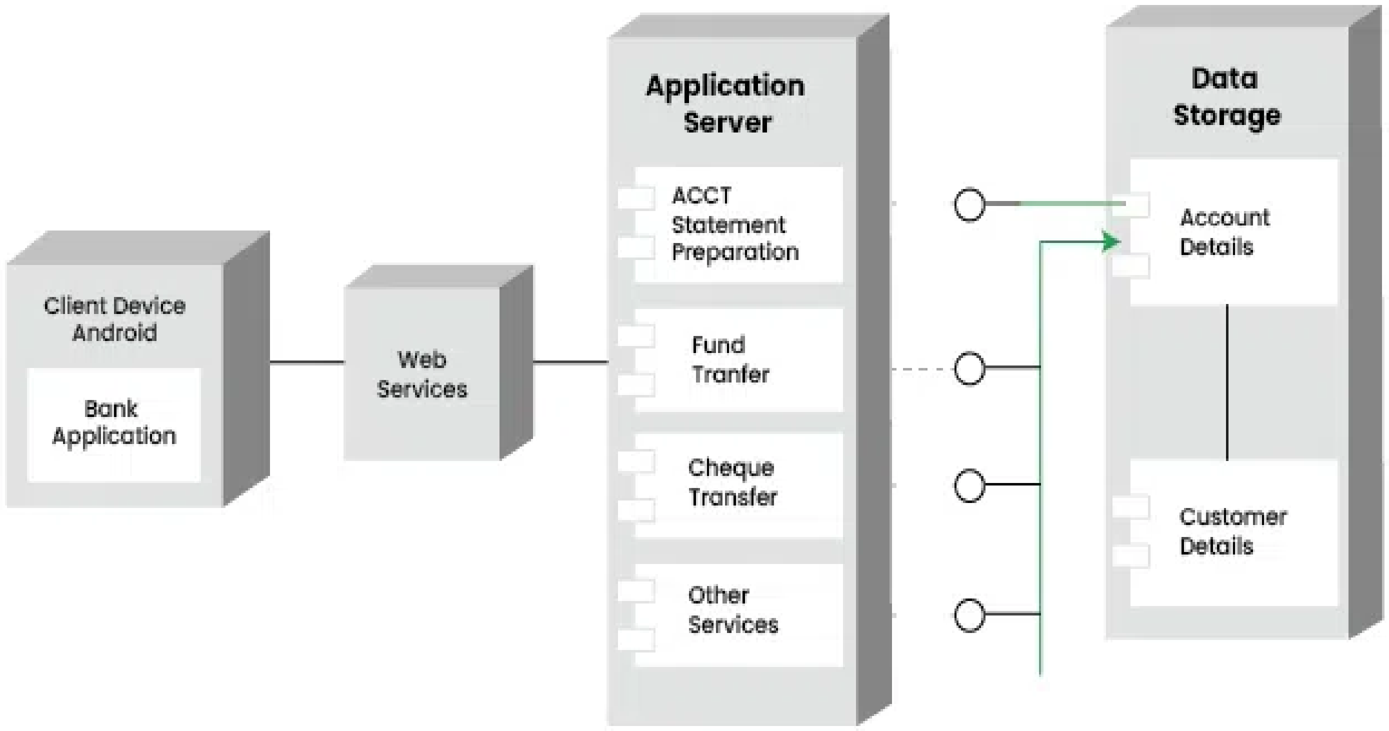
Interfaces in Component-Based Diagram



Deployment Diagram

Deployment Diagrams are used to represent system hardware and its software. It tells us what hardware components exist and what software components run on them.

Deployment Diagram for Mobile Banking Android Services

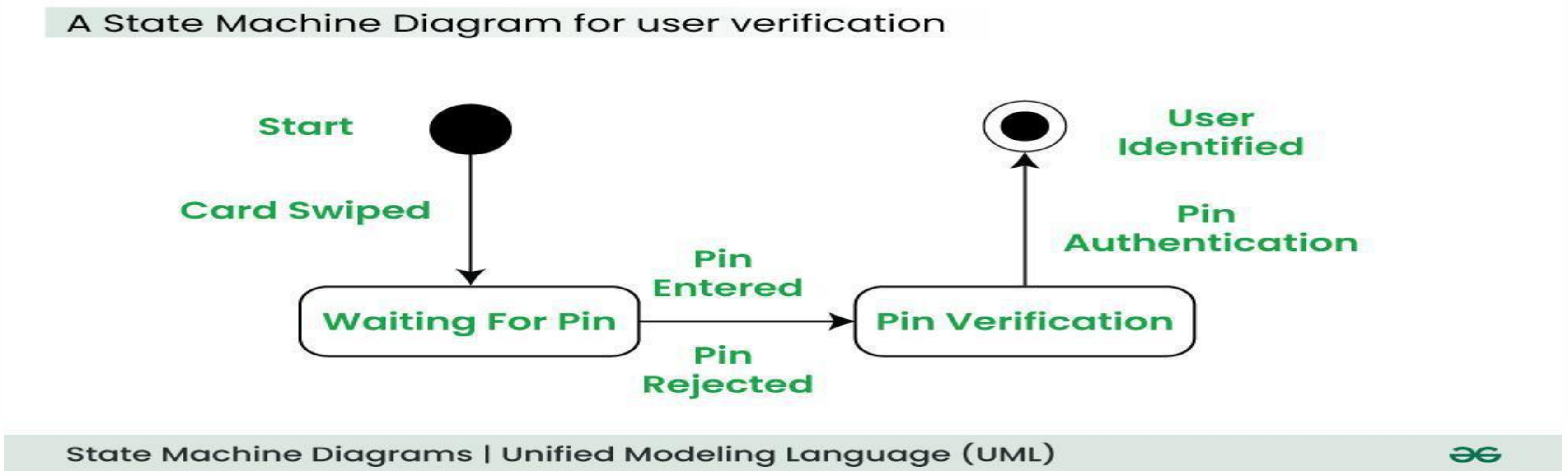


5. Behavioral UML Diagrams

Behavioral UML diagrams are visual representations that depict the dynamic aspects of a system, **illustrating how objects interact and behave over time in response to events.**

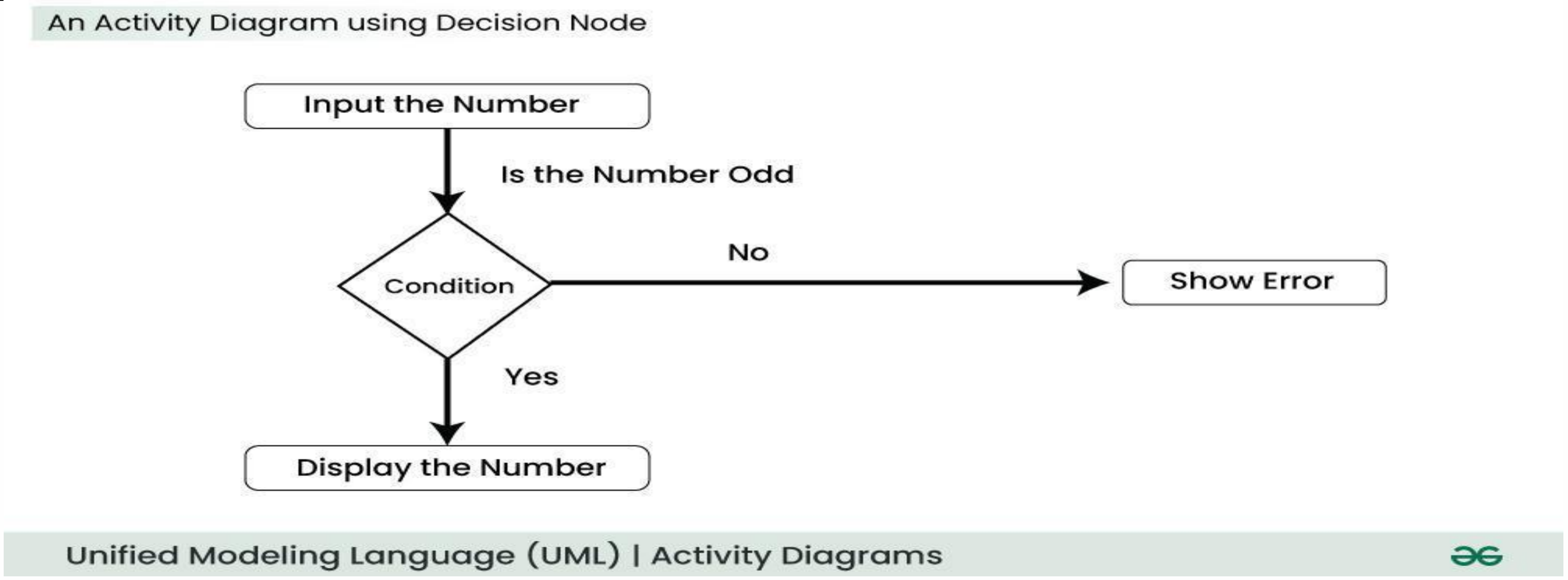
State Machine Diagrams

A state diagram is used to represent the condition of the system or part of the system at finite instances of time.



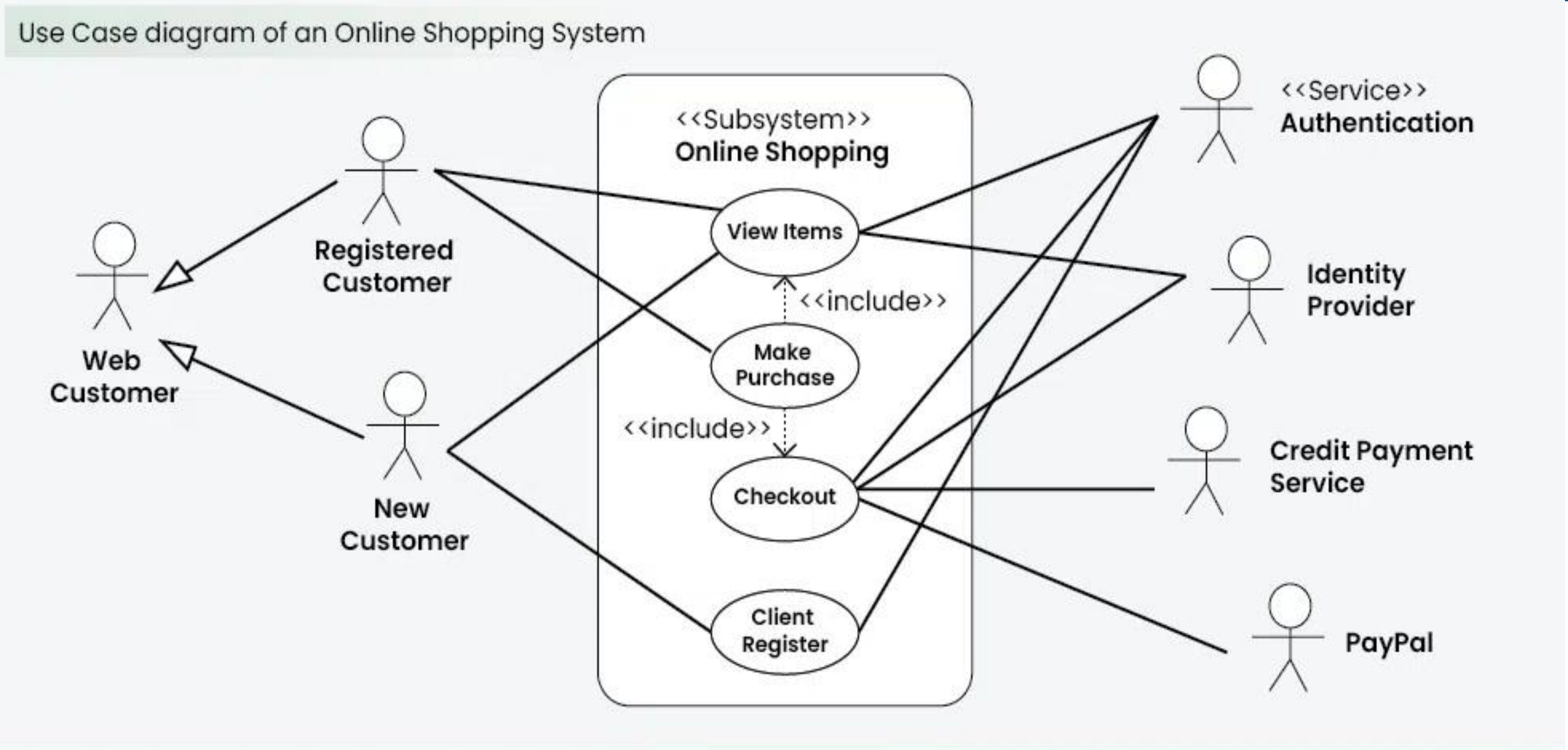
Activity Diagrams

We use Activity Diagrams to illustrate the flow of control in a system


An Activity Diagram using Decision Node
Input the Number
Is the Number Odd
Condition
No
Show Error
Yes
Display the Number
Unified Modeling Language (UML) | Activity Diagrams

Use Case Diagrams

Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents(actors).

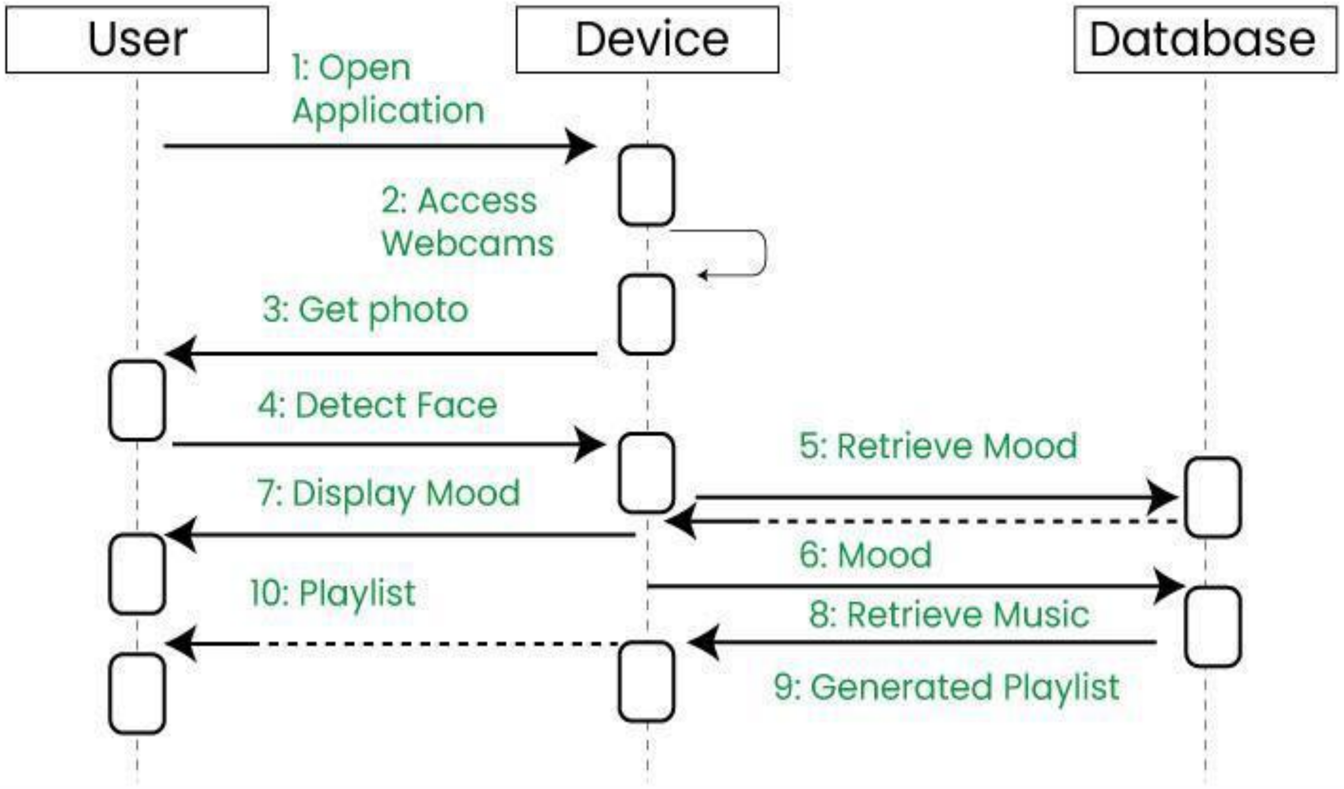


Use Case Diagrams | Unified Modeling Language (UML)

Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place.

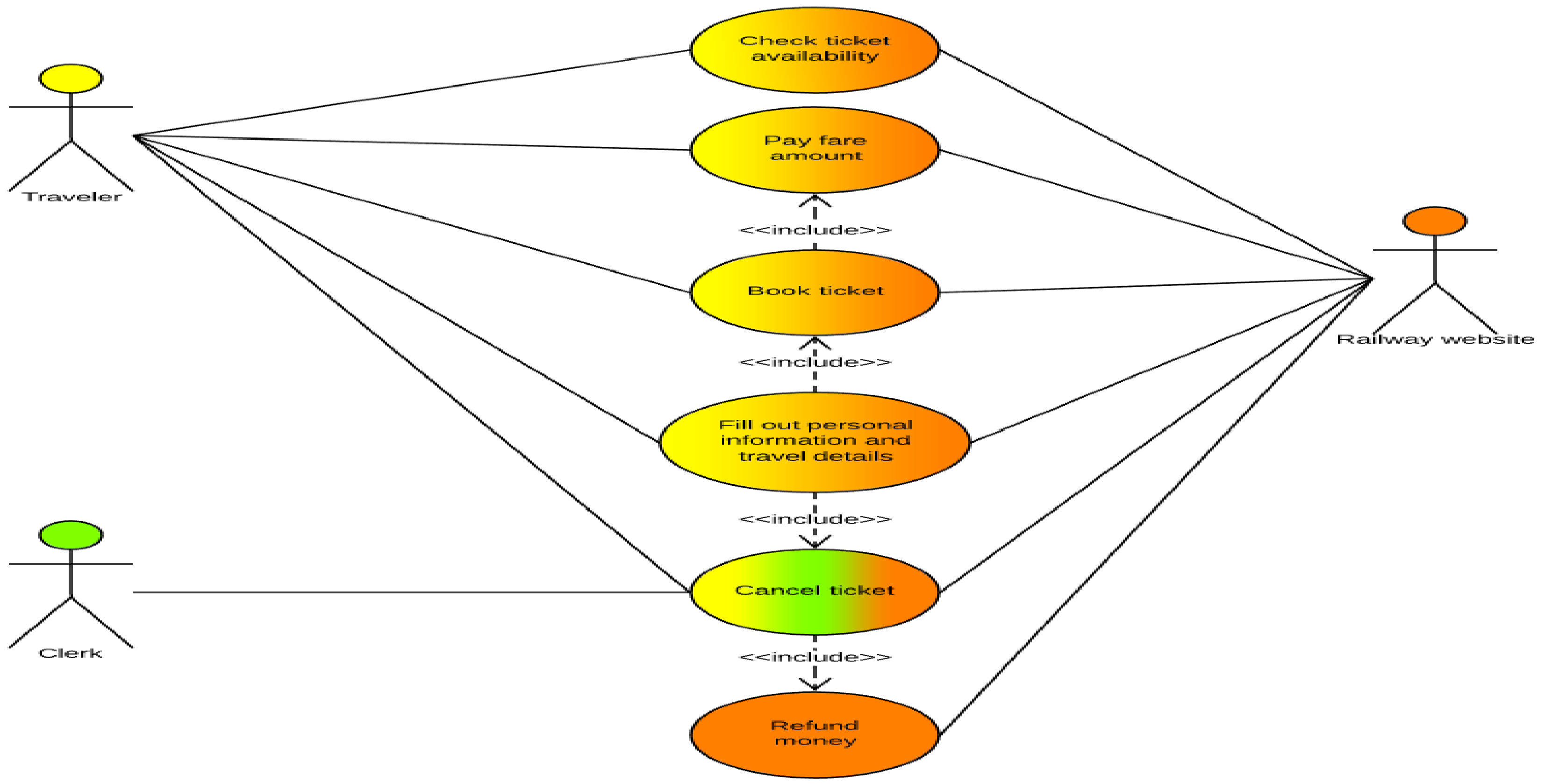
Example sequence diagram



Parul® University

DIGITAL LEARNING CONTENT

Railway reservation use case diagram example



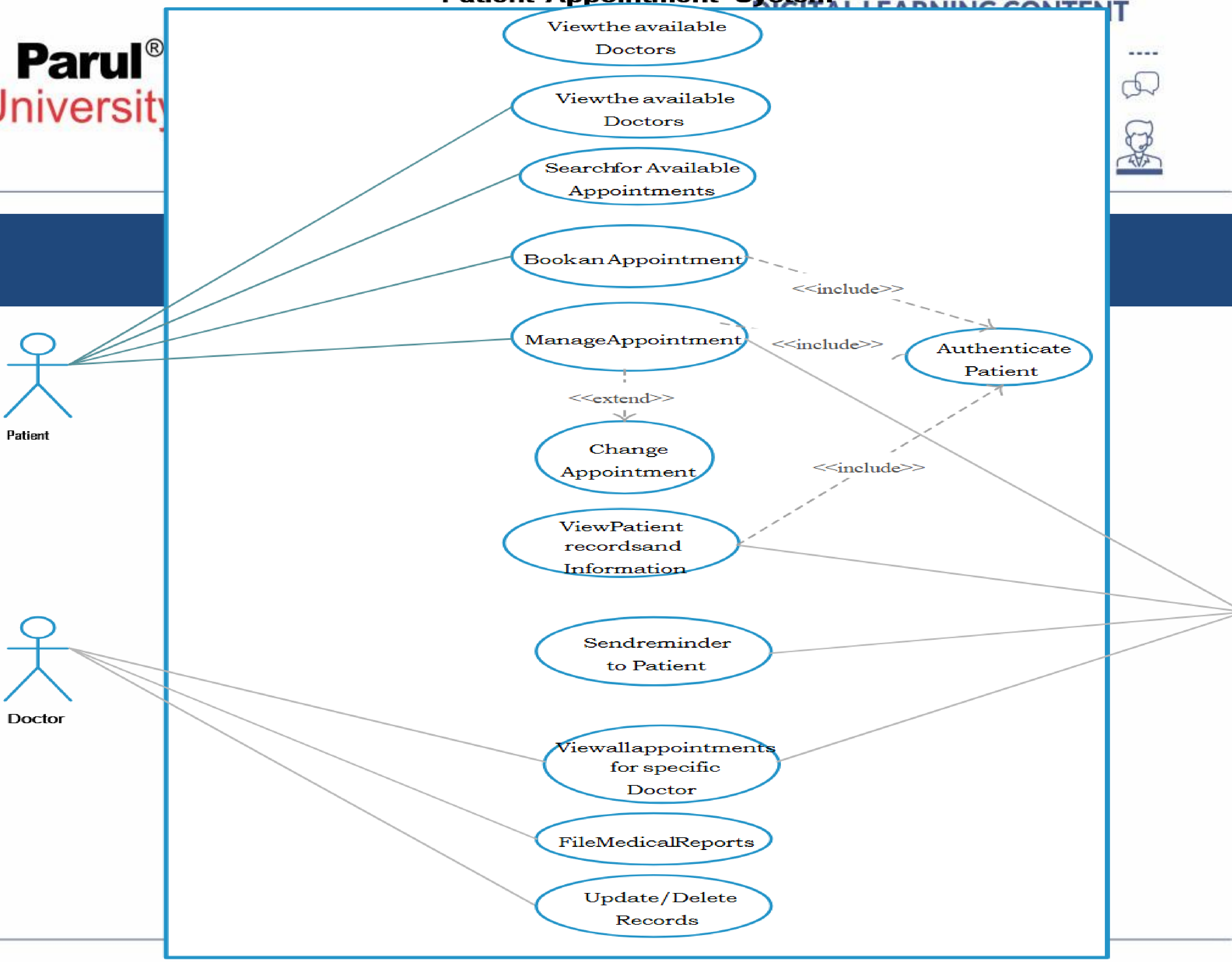
Tools

- **Lucidchart:**
- **Draw.io:**
- **Visual Paradigm:**
- **StarUML:**

Parul® University

DIGITAL LEARNING CONTENT

Patient Appointment System



DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in