UNIT-8
SOFTWARE ENGINEERING TRENDS AND TECHNOLOGY

- Agile Development
- Extreme Programming
- Cloud Computing and grid computing
- Enterprise Mobility
- Business Intelligent And Approaches
  - ERP, Supply chain Management, Service-oriented architecture and web services
  - Enterprise portals and content management
- Introduction to OOSE

AGILE DEVELOPMENT

- 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, 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.
- These processes share some basic characteristics but do have certain subtle differences among themselves.

Steps in the Agile Model

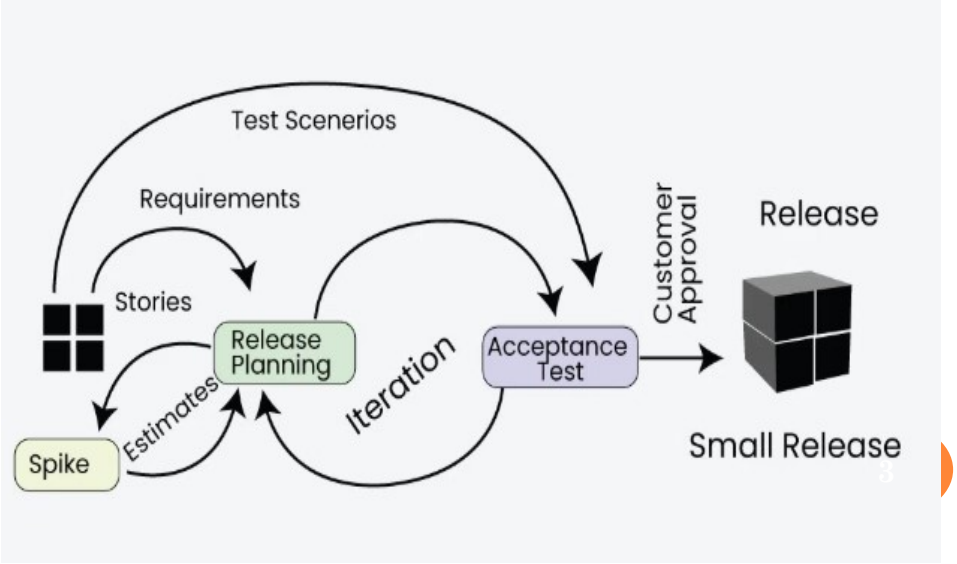
- The agile model is a combination of iterative and incremental process models. The steps involve in agile SDLC models are:
- Requirement gathering
- Design the Requirements
- Construction / Iteration
- Testing / Quality Assurance
- Deployment
- Feedback

Extreme Programming

- Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software and higher quality of life for the development team.
- XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.
- It's a methodology that aims to create high-quality systems based on close interaction with customers, continuous testing, and short development cycles.

The 5 values of XP methodology

- Communication
- Simplicity
- Feedback
- Courage
- Respect



Cloud Computing

- Cloud or cloud computing is the availability of computer resources and systems on-demand, without the user having to manage those systems and the infrastructure required.
- Cloud computing relies on data centers that are located around the world, and larger clouds can utilize dozens or hundreds of these distributed data centers.

DIFFERENT TYPES OF CLOUD COMPUTING

- The four primary types of cloud computing are known as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), and Function as a Service (FaaS).

- **Infrastructure as a Service (IaaS)**
- Infrastructure as a Service (IaaS) is known as the most flexible type of cloud computing.
- IaaS allows businesses to rent IT infrastructures, such as servers, virtual machines (VMs), storage, networks, and operating systems.
- These are rented from a cloud provider on a pay-as-you-go basis.
- This eliminates the capital expense of investing in physical hardware and reduces the need for IT staff to manage it.
- Examples of IaaS providers include the likes of Amazon Web Services (AWS) and Google Cloud Platform (GCP).

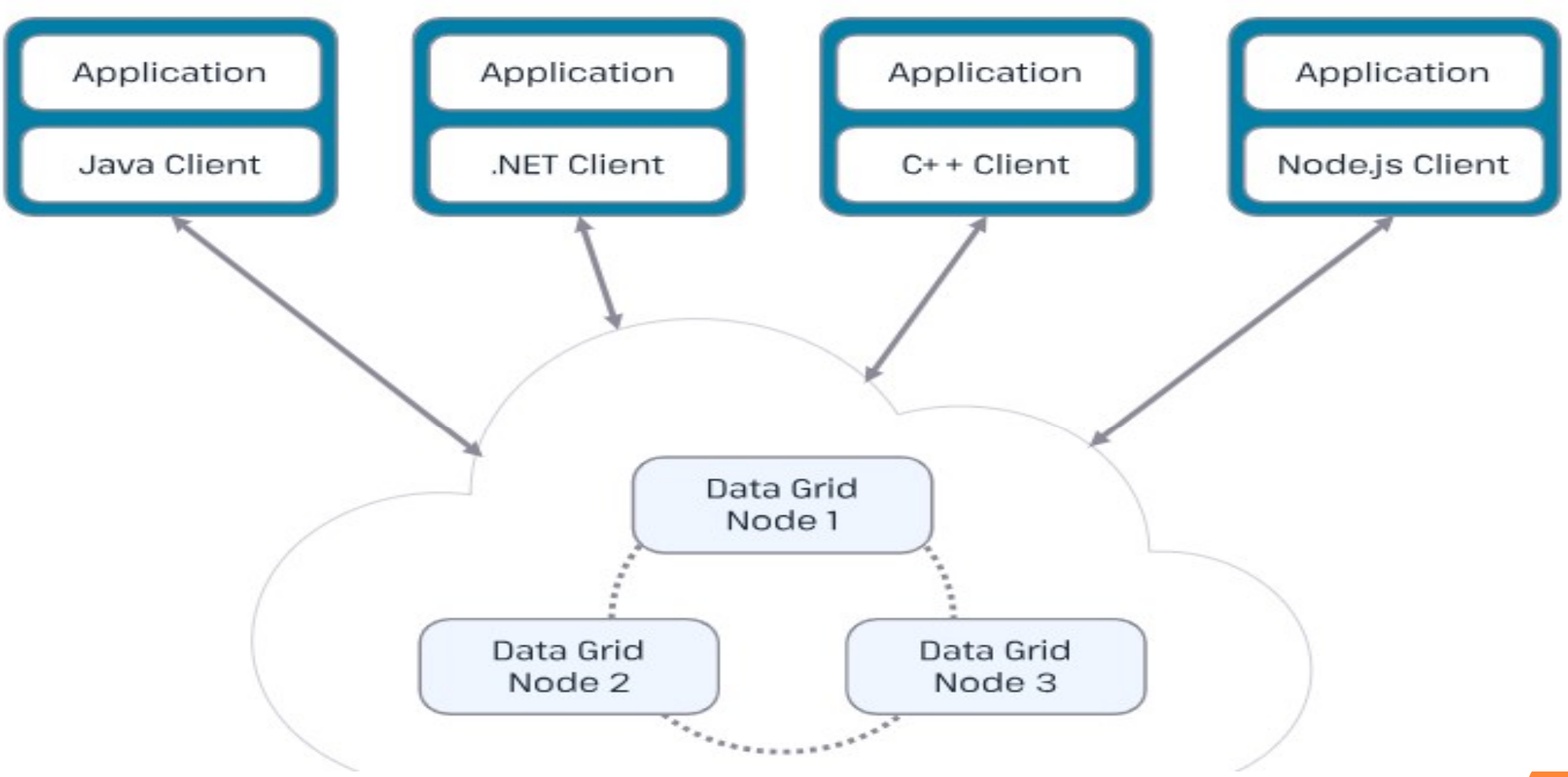
- **Platform as a Service (PaaS)**

- Next up is Platform as a Service (PaaS), which is a cloud computing model in which a service provider can offer a platform for customers to develop, run, and manage applications without having to deal with the complexity of building and maintaining the infrastructure that's often associated with developing and launching an app.
- This includes tools for design, development, testing, and hosting of apps. Examples of PaaS providers are Microsoft Azure and Heroku.

- **Software as a Service (SaaS)**
  - Software as a Service (SaaS) is where applications are hosted by a cloud service provider, and are made available to customers over the Internet.
  - Users don't need to worry about installation, maintenance, or coding, as everything is managed by the service provider.
  - Examples of SaaS applications are Google Workspace (formerly G Suite), Salesforce, and Dropbox.
- **Function as a Service (FaaS)**
  - Function as a Service is a cloud computing category that essentially allows developers to execute a functionality of an application, without having to worry about the infrastructure underpinning that application.
  - Commonly, FaaS is used in serverless computing, and allows developers to write and update code quickly, then executing the code in response to an event.
  - An example of FaaS is AWS Lambda, which allows developers to upload their code, after which it's scaled and run automatically as needed.

GRID COMPUTING

- Grid Computing can be defined as a network of computers working together to perform a task that would rather be difficult for a single machine.
- All machines on that network work under the same protocol to act as a virtual supercomputer.
- The tasks that they work on may include analyzing huge datasets or simulating situations that require high computing power.
- Computers on the network contribute resources like processing power and storage capacity to the network.
- Grid computing is the practice of leveraging multiple computers, often geographically distributed but connected by networks, to work together to accomplish joint tasks.
- It is typically run on a "data grid," a set of computers that directly interact with each other to coordinate jobs.


Application
Java Client
Application
.NET Client
Application
C++ Client
Application
Node.js Client
Data Grid
Node 1
Data Grid
Node 2
Data Grid
Node 3

Why is Grid Computing Important?

- **Scalability**: It allows organizations to scale their computational resources dynamically. As workloads increase, additional machines can be added to the grid, ensuring efficient processing.
- **Resource Utilization**: By pooling resources from multiple computers, grid computing maximizes resource utilization. Idle or underutilized machines contribute to tasks, reducing wastage.
- **Complex Problem Solving**: Grids handle large-scale problems that require significant computational power. Examples include climate modeling, drug discovery, and genome analysis.
- **Collaboration**: Grids facilitate collaboration across geographical boundaries. Researchers, scientists, and engineers can work together on shared projects.
- **Cost Savings**: Organizations can reuse existing hardware, saving costs while accessing excess computational resources. Additionally, cloud resources can be cost-effectively.

Working of Grid Computing

- A Grid computing network mainly consists of these three types of machines
- **Control Node:** A computer, usually a server or a group of servers which administrates the whole network and keeps the account of the resources in the network pool.
- **Provider:** The computer contributes its resources to the network resource pool.
- **User:** The computer that uses the resources on the network.

Advantages of Grid Computing

- Grid Computing provide high resources utilization.
- Grid Computing allow parallel processing of task.
- Grid Computing is designed to be scalable.

Disadvantages of Grid Computing

- The software of the grid is still in the evolution stage.
- Grid computing introduce Complexity.
- Limited Flexibility
- Security Risks

Enterprise Mobility

- When the employees in an organization have the autonomy to work from any location using specific applications and software is termed Enterprise Mobility.
- Not just work, but it also helps store the entire data, information sharing, important announcements, and connecting different departments in an organization.

Enterprise Mobility Management(EMM)

- Enterprise Mobility Management (EMM) is a set of services and technologies designed to secure corporate data on employees' mobile devices while also supporting a wide range of device management policies and business use cases.
- As the workforce becomes increasingly mobile and remote, EMM plays a crucial role in enabling employees to access corporate resources securely from any location, on any device.
- EMM solutions differ a lot between companies.
- Some aim to keep certain apps safe, while others focus on fully securing employee devices.

Business Intelligent and Approaches

- Business intelligence (BI) is a technology-driven process that analyzes business data and transforms it into actionable insights, helping executives and managers make better-informed decisions.
- Business intelligence is a broad term that encompasses data mining, process analysis, performance benchmarking, and descriptive analytics.
- It parses all the data a business generates and presents easy-to-digest reports, performance measures, and trends that inform management decisions.
- Business intelligence should not be confused with business analytics.
- Business intelligence analyzes current operational data to drive decision-making, while business analytics delves deeper into predictive insights for future growth.
- Ultimately, business intelligence drives global economic growth, as it helps organizations identify trends, optimize operations, and seize new opportunities.

BI strategy from the ground up

- Know your business strategy and goals.
- Identify key stakeholders.
- Choose a sponsor from your key stakeholders.
- Choose your BI platform and tools.
- Create a BI team.
- Define your scope.
- Prepare your data infrastructure.
- Define your goals and roadmap.
- Categories of BI analysis

Categories of BI analysis

There are three major types of BI analysis:

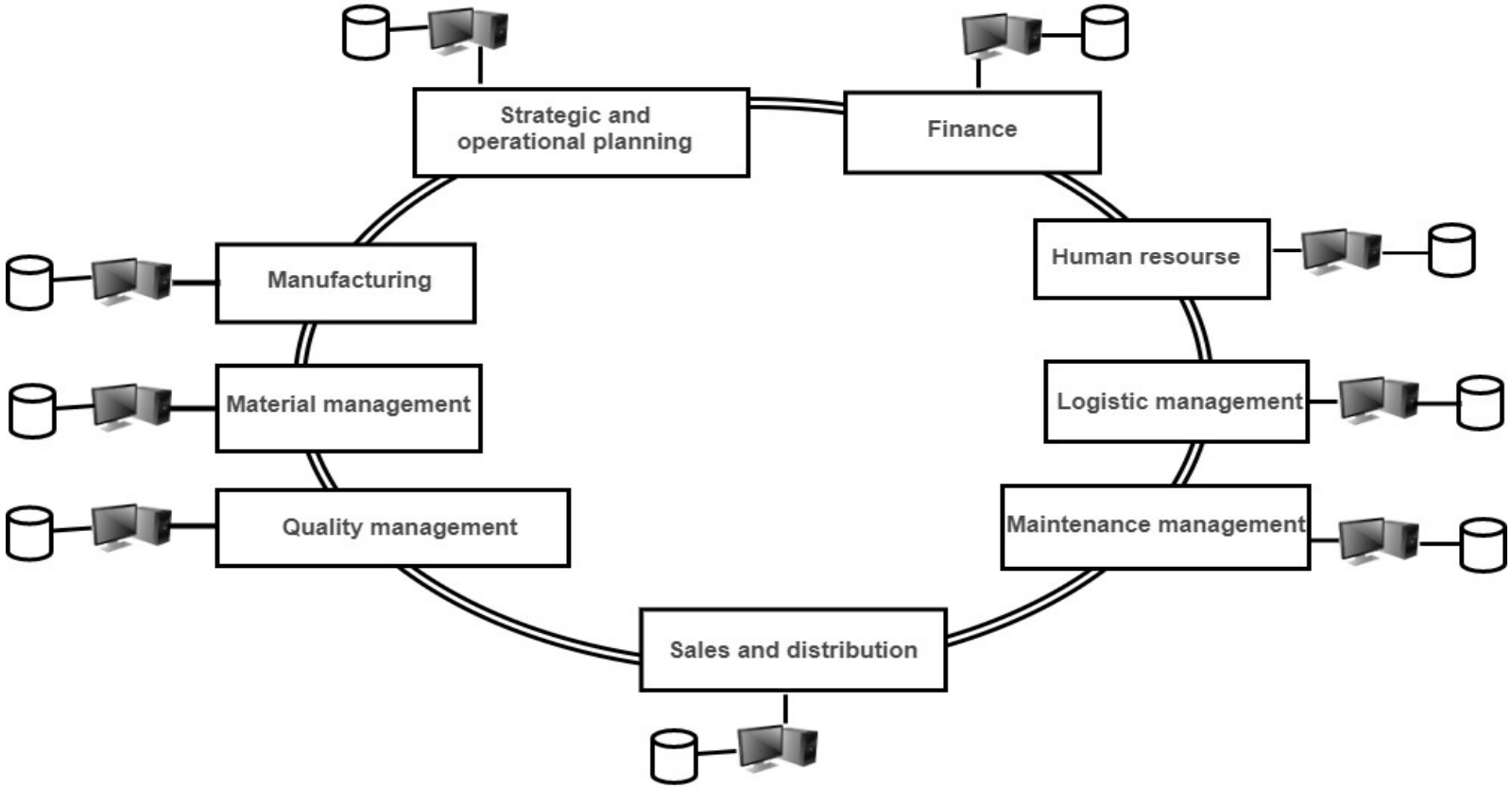
- **Predictive analytics:**
  - Predictive analytics takes historical and real-time data and models future outcomes for planning purposes.
- **Descriptive analytics:**
  - Descriptive analytics is the process of identifying trends and relationships in data using historical and current data.
- **Prescriptive analytics:**
  - Prescriptive analytics takes all the relevant data to answer the question, "what should my business do?"

ERP

- Enterprise Resource Planning (ERP) is a software system that is used by organizations to manage and integrate the important parts of the businesses.
- It is the practice of consolidating an enterprise's planning, manufacturing, sales, and marketing efforts into one management system.
  - It can integrate all the processes that are needed to run a company.
  - It helps to improve efficiency, better reporting, and enhanced data security.
  - These systems can be customized according to the specific needs of the different industries.
  - ERP systems are crucial for large organizations.

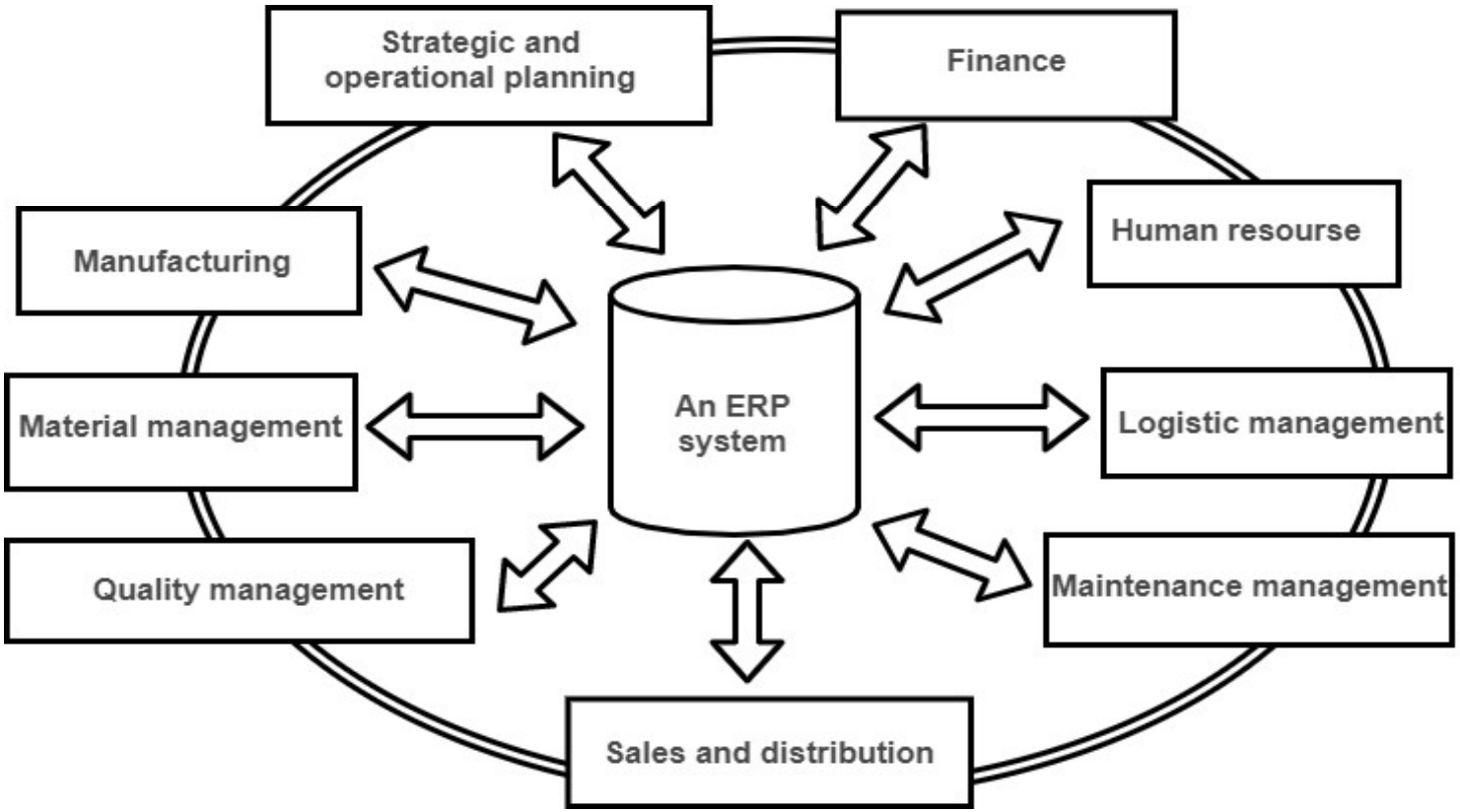
Before ERP

- Before an ERP system, there were different databases of different departments which they managed on their own.
- The employees of one department does not know anything about the other department.



After ERP

- After the ERP system, databases of different departments are managed by one system called the ERP system.
- It keeps track of all the databases within the system.
- In this scenario, employees of one department have information regarding the other departments.



Features of ERP

- **Financial Management:** ERP systems are used to manage financial transactions, produce financial statements such as balance sheets, manage companies' tangible and intangible assets, and track money owned by and to the organization.
- **Supply Chain Management:** ERP systems help to monitor the stock levels, track inventory movement, manage the movement of goods, and automate the purchasing process.
- **Human Resources:** In an organization, ERP systems help to manage employee records, automate payroll processing, and manage recruitment and onboarding.
- **Customer Relationship Management (CRM):** ERP systems help in sales automation, automate marketing campaigns, track customer interactions, and improve customer satisfaction.
- **Project Management:** ERP systems support project planning and scheduling, and monitor project time and expenses for accurate cost estimation.
- **Manufacturing:** ERP systems help in production planning, list all the raw materials required for product manufacture, and monitor the production process.

Benefits of ERPs

- Improved Efficiency
- Integrated Business Processes
- Reduced Redundancy
- Timely Data Access
- Reduced IT Costs
- Maintain Compliance
- Enhances Customer Service

Weakness of ERPs

- High Implementation Costs
- Complex Customization
- Resistance to Change
- Data Security Risks
- Limited Flexibility
- Dependence on Vendor Support

Supply Chain Management

- Supply Chain Management (SCM) is essential in today's fast-paced environment, overseeing the efficient movement of goods, information, and finances from suppliers of raw materials to the final consumers.
- It encompasses a series of complex, interrelated processes designed to ensure that products are produced and delivered in the right quantities, to the right locations, at the right time, and at optimal costs.
- A supply chain is like a network that provides facilities and options like distribution which performs operations like Procurement of material, the transformation of these materials to specific intermediates, and after that finished product and the distribution procedure starts which starts distributing respective products to customers.

Components of a Supply Chain

- **Suppliers:** Provide raw materials, components, and services required to produce finished products.
- **Manufacturers:** Transform raw materials into finished goods through various production processes.
- **Warehouses:** Store raw materials, intermediates, and finished products until they are needed further down the chain.
- **Distribution Centers:** Facilitate the efficient movement and distribution of goods to various locations.
- **Retailers:** Sell finished products directly to end customers.
- **Customers:** The final recipients of the products, whose demands drive the entire supply chain.

Working of Supply Chain Management Work

- Supply chains operate through a network of linked activities and processes that enable the movement of goods, information, and finances from suppliers to customers.

1. Planning: Developing strategies to meet customer demand while optimizing resources.

Key Activities:

- Demand forecasting
- Inventory management
- Capacity planning
- Production scheduling

2. Sourcing: Sourcing involves identifying and selecting suppliers that provide the necessary raw materials, components, and services.

Key Activities:

- Supplier selection
- Supplier relationship management
- Procurement
- Contract negotiation

3. Manufacturing: Coordinating production processes to create finished goods efficiently.

Key Activities:

- Production process design
- Quality control
- Production scheduling
- Equipment maintenance

4. **Warehousing:** Warehousing involves storing raw materials, intermediates, and finished products until they are needed. Efficient warehousing is crucial for maintaining inventory levels and ensuring timely delivery.

Key Activities:

- Inventory management
- Storage solutions
- Order picking
- Packing

5. **Distribution:** Coordinating the movement of goods from manufacturers to customers.

Key Activities:

- Transportation planning
- Logistics management
- Order fulfillment
- Delivery scheduling

6. **Returns and Reverse Logistics**: Returns and reverse logistics involve managing the return of defective or unwanted products from customers back to the manufacturer or supplier. This process includes handling returns, refurbishing products, and recycling.

Key Activities:

- Return processing
- Refurbishing and recycling
- Disposal management
- Customer support

7. **Technology integration:** Implementing systems like Enterprise Resource Planning (ERP) and Transportation Management Systems (TMS) to facilitate information flow.
8. **Performance measurement:** Continuously monitoring and improving supply chain metrics.

SERVICE - ORIENTED ARCHITECTURE

- Service-Oriented Architecture (SOA) is a stage in the evolution of application development and/or integration.
- It defines a way to make software components reusable using the interfaces.

Characteristics of SOA are as follows :

- Provides interoperability between the services.
- Provides methods for service encapsulation, service discovery, service composition, service reusability and service integration.
- Facilitates QoS (Quality of Services) through service contract based on Service Level Agreement (SLA).
- Provides loosely couples services.
- Provides location transparency with better scalability and availability.
- Ease of maintenance with reduced cost of application development and deployment.

Principles of SOA:

- **Standardized service contract:** Specified through one or more service description documents.
- **Loose coupling:** Services are designed as self-contained components, maintain relationships that minimize dependencies on other services.
- **Abstraction:** A service is completely defined by service contracts and description documents. They hide their logic, which is encapsulated within their implementation.
- **Reusability:** Designed as components, services can be reused more effectively, thus reducing development time and the associated costs.
- **Autonomy:** Services have control over the logic they encapsulate and, from a service consumer point of view, there is no need to know about their implementation.
- **Discoverability:** Services are defined by description documents that constitute supplemental metadata through which they can be effectively discovered. Service discovery provides an effective means for utilizing third-party resources.
- **Composability:** Using services as building blocks, sophisticated and complex operations can be implemented. Service orchestration and choreography provide a solid support for composing services and achieving business goals.

Advantages of SOA:

- Service reusability
- Easy maintenance
- Platform independent
- Availability
- Reliability
- Scalability

Disadvantages of SOA:

- High overhead
- High investment
- Complex service management

WEB SERVICES

- A web service is a software application that allows different devices and applications to communicate over the internet.
- Web services are platform-independent, so they can connect applications written in different programming languages.
- A web service is a set of open protocols and standards that allow data to be exchanged between different applications or systems.
- Web services can be used by software programs written in a variety of programming languages and running on a variety of platforms to exchange data via computer networks such as the Internet in a similar way to inter-process communication on a single computer.

Functions of Web Services

- It's possible to access it via the internet or intranet networks.
- XML messaging protocol that is standardized.
- Operating system or programming language independent.
- Using the XML standard, it is self-describing.
- A simple location approach can be used to locate it.

Enterprise Portal and Content Management (EPCM)

- Enterprise Portal and Content Management (EPCM) is the strategies, methods and software tools used to capture, manage, store, preserve, and deliver information and documents related to organizational processes.
- An "enterprise portal" is a centralized online platform that provides employees, partners, and customers with access to various company information and applications through a single point of entry.
- Content management system (CMS) is a software tool that allows users to create, edit, and publish digital content on a website or portal, often within an enterprise setting, essentially managing the content within the portal itself
- Both work together to provide a unified, accessible interface for managing and accessing critical business information.
- Enterprise Portals provide a “gateway” to information and documents, provide security (authentication & authorization), and, have robust content management system, which allows many users to contribute and update the content.
- Portals can improve organizational effectiveness, they can accelerate/automate shared business processes, and they can facilitate information sharing across boundaries for better business insight.

INTRODUCTION TO OOSE

- Object Oriented Software Engineering (OOSE) is a software design technique that is used in software design in object-oriented programming.
- OOSE is developed by Ivar Jacobson in 1992.
- OOSE is the first object-oriented design methodology that employs use cases in software design.
- OOSE is one of the precursors of the Unified Modeling Language (UML) Classes and objects are fundamental building blocks of an Object – Oriented Software Engineering.
- We organize software as a collection of discrete objects that incorporate both data structures and behaviours.

- **Object – Oriented Analysis :**

- Object Oriented Analysis (OOA) is the first technical activity performed as part of object oriented software engineering.
- OOA introduces new concepts to investigate a problem. It is based in a set of basic principles, which are as follows-
  1. The information domain is modeled.
  2. Behavior is represented.
  3. Function is described.
  4. Data, functional, and behavioral models are divided to uncover greater detail.
  5. Early models represent the essence of the problem, while later ones provide implementation details.

- **Object – Oriented Design:**

- In the object-oriented design method, the system is viewed as a collection of objects (i.e., entities).
- The state is distributed among the objects, and each object handles its state data.
- For example, in a Library Automation Software, each library representative may be a separate object with its data and functions to operate on these data.

Benefits of OOSE

- Makes code more modular, flexible, and intuitive .
- Helps manage complexity by allowing designers to focus on interactions at a higher level .
- Provides flexibility in code because each object responds in its own way.

The different terms related to object design are:

1. **Objects:** All entities involved in the solution design are known as objects. For example, person, banks, company, and users are considered as objects. Every entity has some attributes associated with it and has some methods to perform on the attributes.
2. **Classes:** A class is a generalized description of an object. An object is an instance of a class.
3. **Messages:** Objects communicate by message passing. Messages consist of the integrity of the target object, the name of the requested operation, and any other action needed to perform the function.
4. **Abstraction:** Abstraction is the removal of the irrelevant and the amplification of the essentials.

5. **Encapsulation:** Encapsulation is also called an information hiding concept. The data and operations are linked to a single unit. Encapsulation not only bundles essential information of an object together but also restricts access to the data and methods from the outside world.
6. **Inheritance:** OOD allows similar classes to stack up in a hierarchical manner where the lower or sub-classes can import, implement, and re-use allowed variables and functions from their immediate superclasses.This property of OOD is called an inheritance.
7. **Polymorphism:** OOD languages provide a mechanism where methods performing similar tasks but vary in arguments, can be assigned the same name. This is known as polymorphism, which allows a single interface is performing functions for different types.

UNIFIED MODELING LANGUAGE (UML)

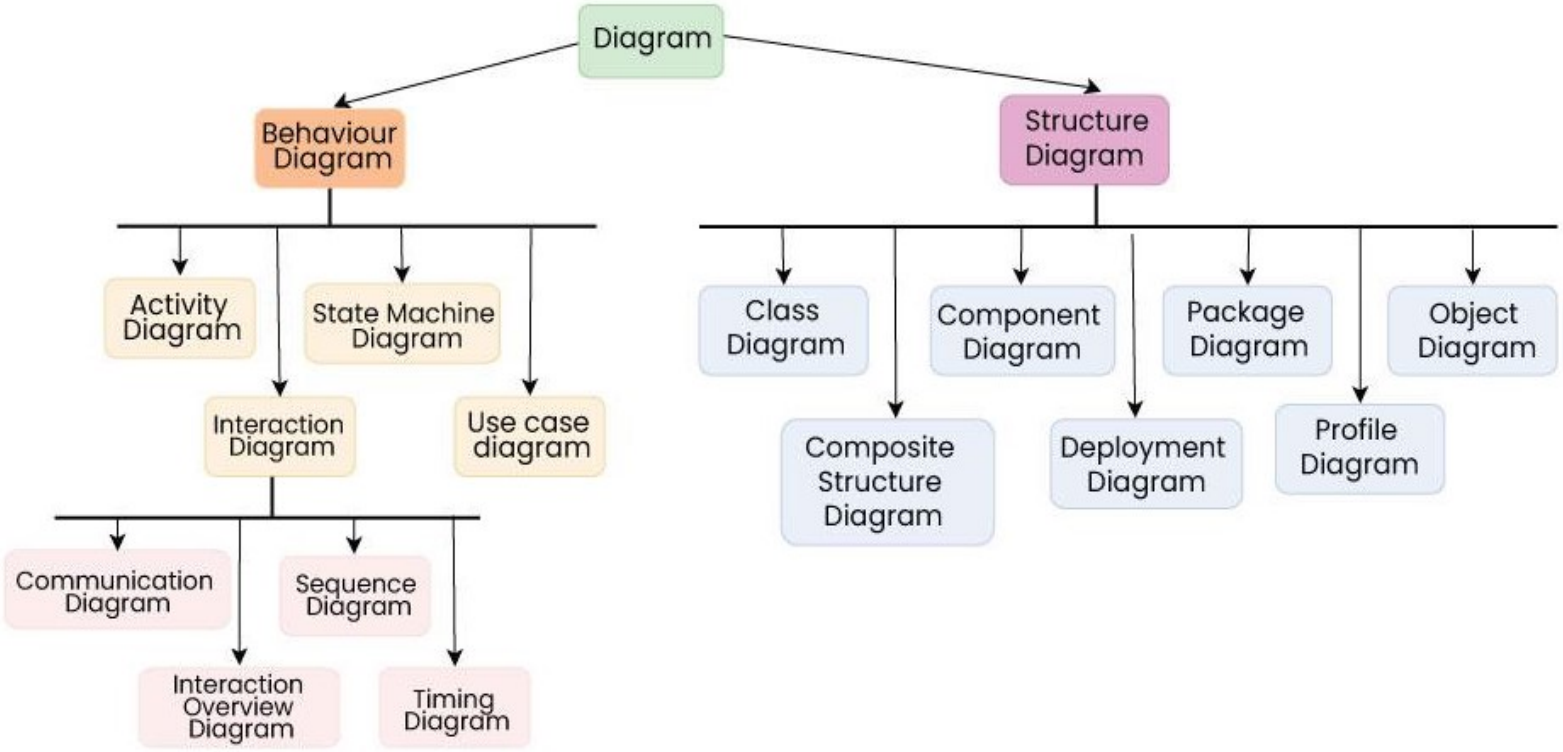
- Unified Modeling Language (UML) is a standardized visual modeling language that is a versatile, flexible, and user-friendly method for visualizing a system's design.
- Software system artifacts can be specified, visualized, built, and documented with the use of UML.
  - We use UML diagrams to show the behavior and structure of a system.
  - UML helps software engineers, businessmen, and system architects with modeling, design, and analysis.

Why do we need UML

- We need UML (Unified Modeling Language) to visually represent and communicate complex system designs, facilitating better understanding and collaboration among stakeholders. Below is why we need UML:
- Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
- Businessmen do not understand code. So UML becomes essential to communicate with non-programmers about essential requirements, functionalities, and processes of the system.
- A lot of time is saved down the line when teams can visualize processes, user interactions, and the static structure of the system.

TYPES OF UML DIAGRAMS

- UML is linked with object-oriented design and analysis.
- UML makes use of elements and forms associations between them to form diagrams.
- Diagrams in UML can be broadly classified as:

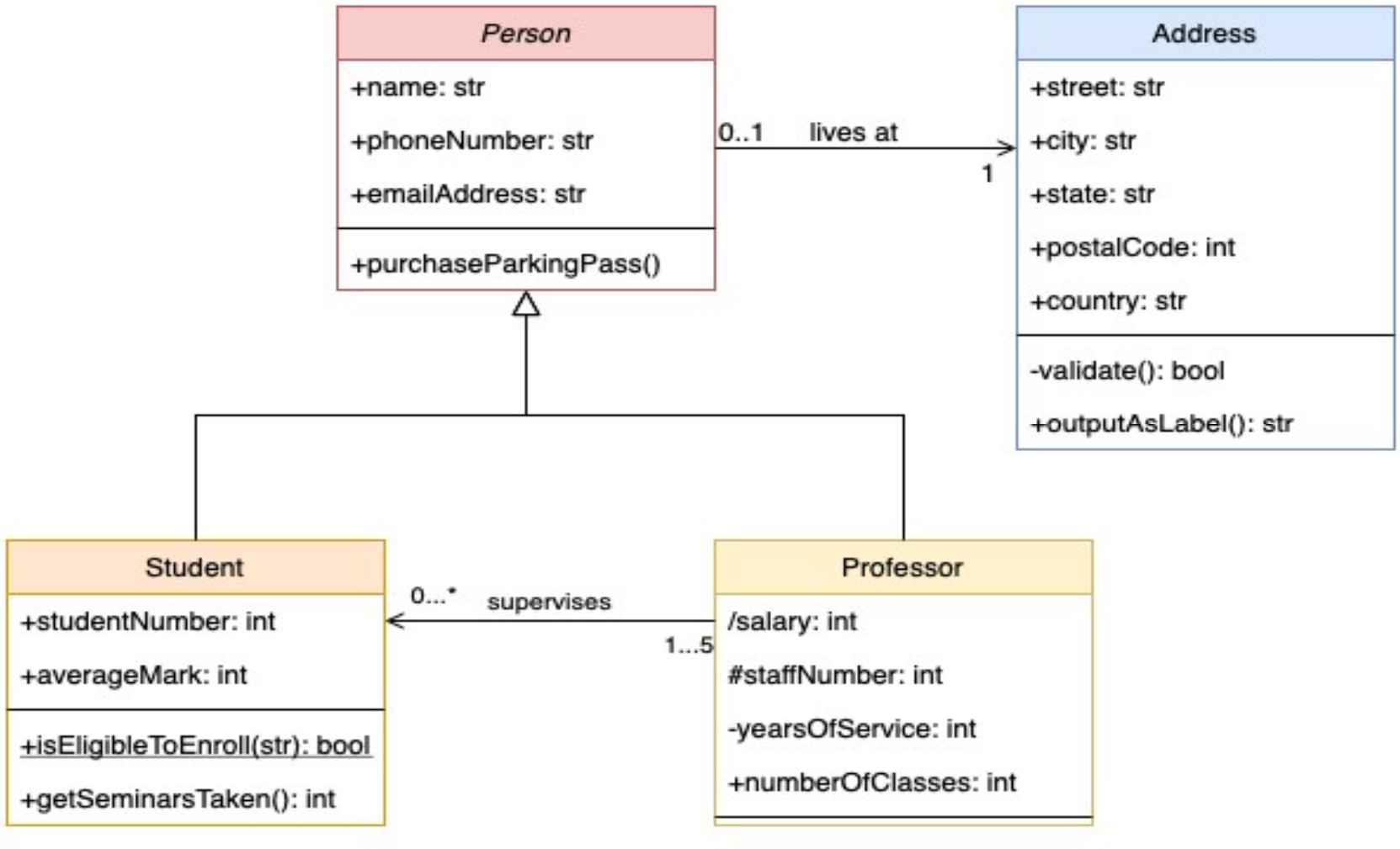


STRUCTURAL UML DIAGRAMS

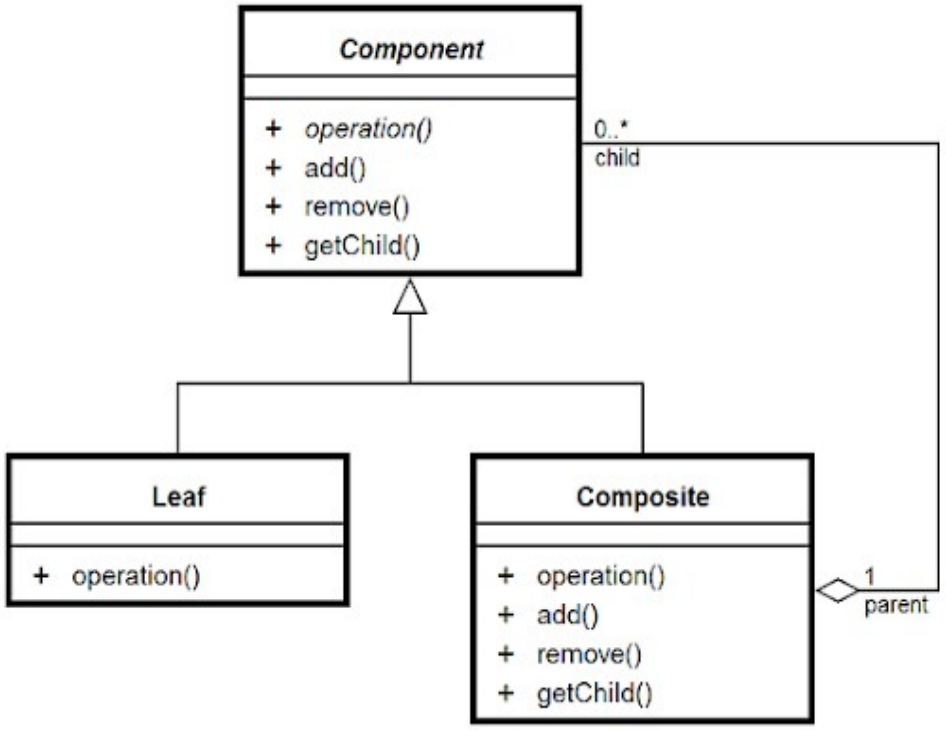
- Structural UML diagrams are visual representations that depict the static aspects of a system, including its classes, objects, components, and their relationships, providing a clear view of the system's architecture.
- Structural UML diagrams include the following types:

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 diagrams also help us identify relationship between different classes or objects.

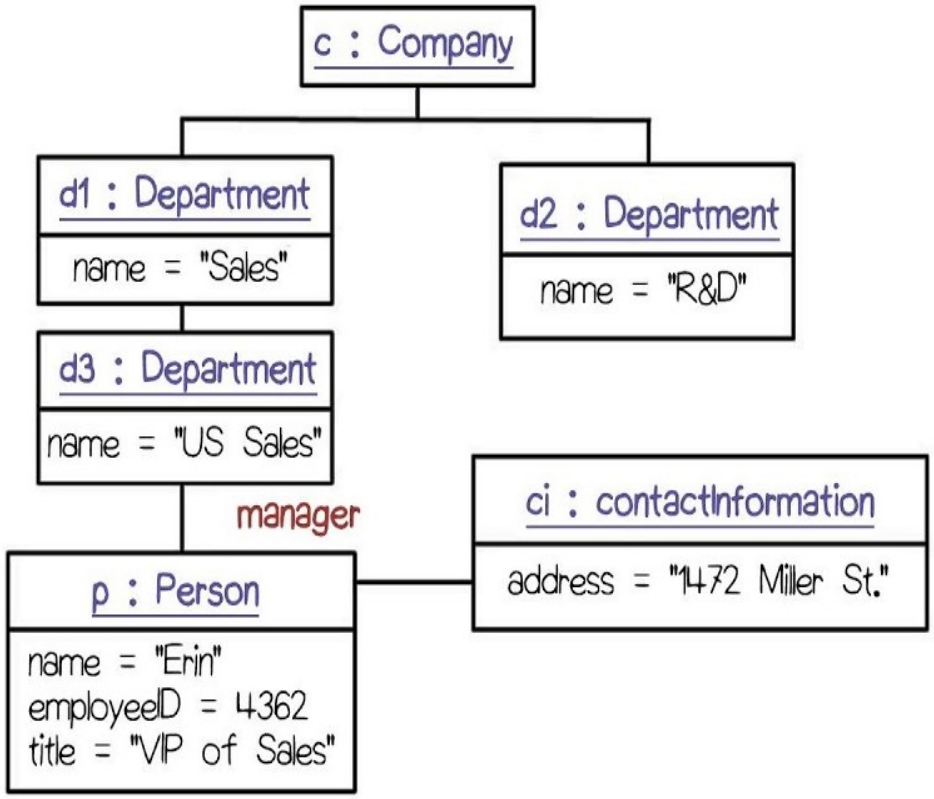

Person
+name: str
+phoneNumber: str
+emailAddress: str
+purchaseParkingPass()
0..1
lives at
1
Address
+street: str
+city: str
+state: str
+postalCode: int
+country: str
-validate(): bool
+outputAsLabel(): str
Student
+studentNumber: int
+averageMark: int
+isEligibleToEnroll(str): bool
+getSeminarsTaken(): int
0...*
supervises
1...5
Professor
/salary: int
#staffNumber: int
-yearsOfService: int
+numberOfClasses: int

- **Composite Structure Diagram**
  - We use composite structure diagrams to represent the internal structure of a class and its interaction points with other parts of the system.
  - A composite structure diagram represents relationship between parts and their configuration which determine how the classifier (class, a component, or a deployment node) behaves.
  - They represent internal structure of a structured classifier making the use of parts, ports, and connectors.
  - We can also model collaborations using composite structure diagrams.
  - They are similar to class diagrams except they represent individual parts in detail as compared to the entire class.



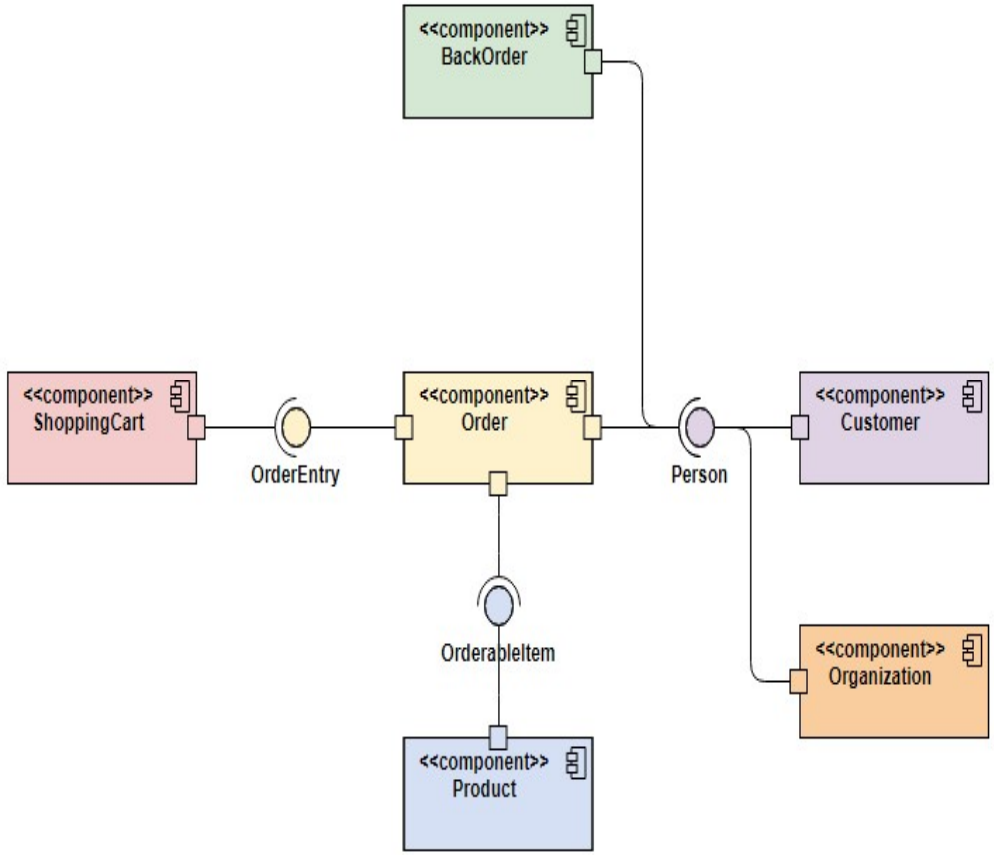
Object Diagram

- An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them.
- An object diagram is similar to a class diagram except it shows the instances of classes in the system.
- We depict actual classifiers and their relationships making the use of class diagrams.
- On the other hand, an Object Diagram represents specific instances of classes and relationships between them at a point of time.



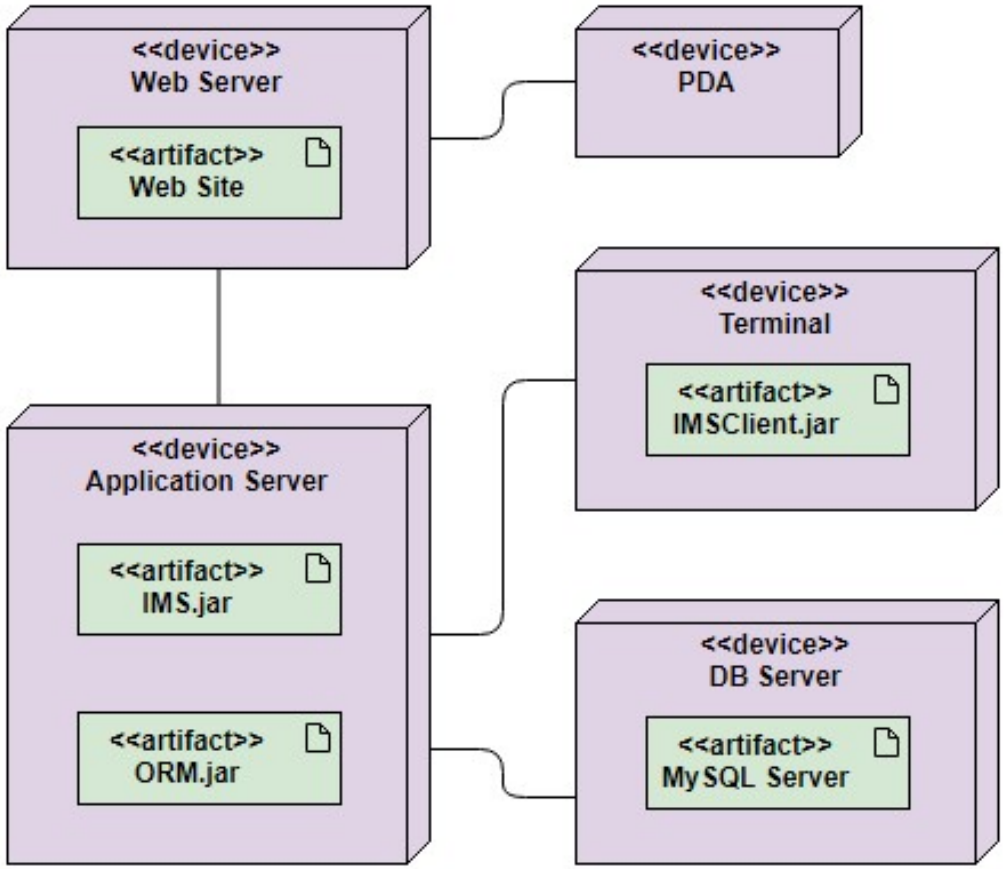
Component Diagram

- Component diagrams are used to represent how the physical components in a system have been organized. We use them for modelling implementation details.
- Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development.
- Component Diagrams become essential to use when we design and build complex systems.
- Interfaces are used by components of the system to communicate with each other.

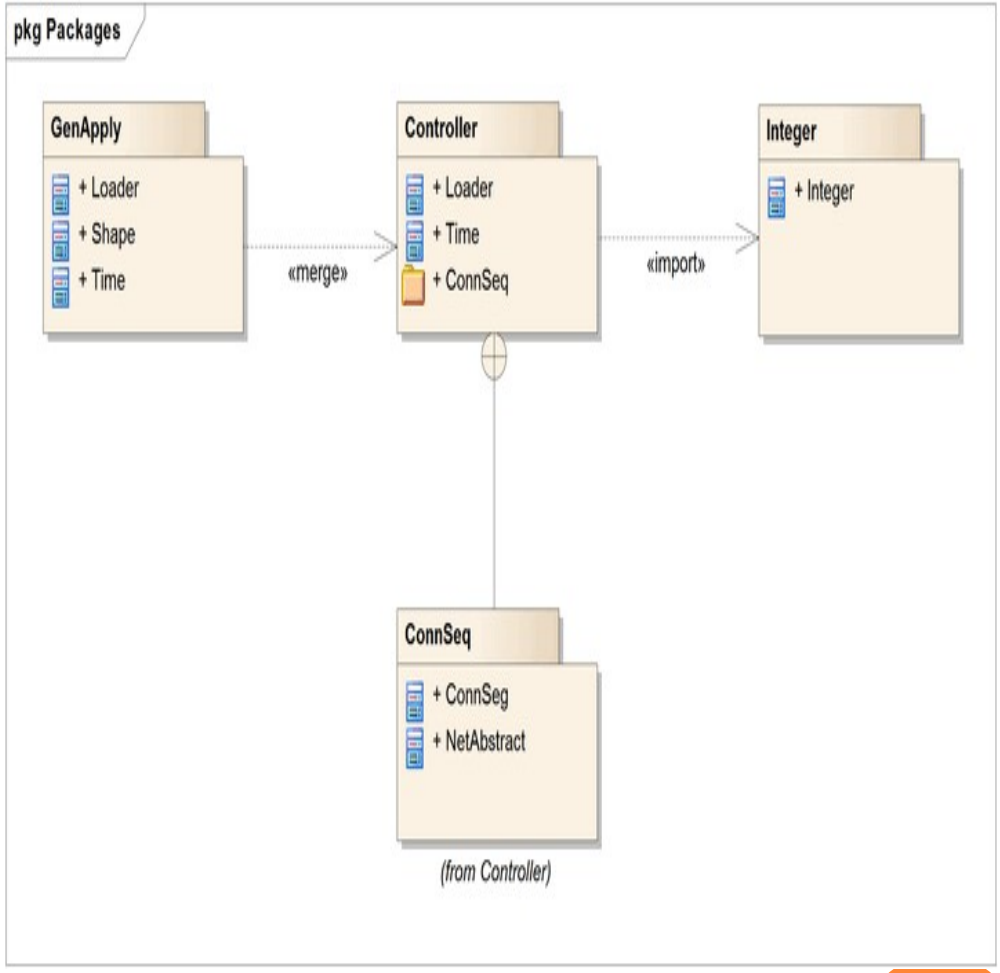


Deployment Diagram

- 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.
- We illustrate system architecture as distribution of software artifacts over distributed targets.
- An artifact is the information that is generated by system software.
- They are primarily used when a software is being used, distributed or deployed over multiple machines with different configurations.

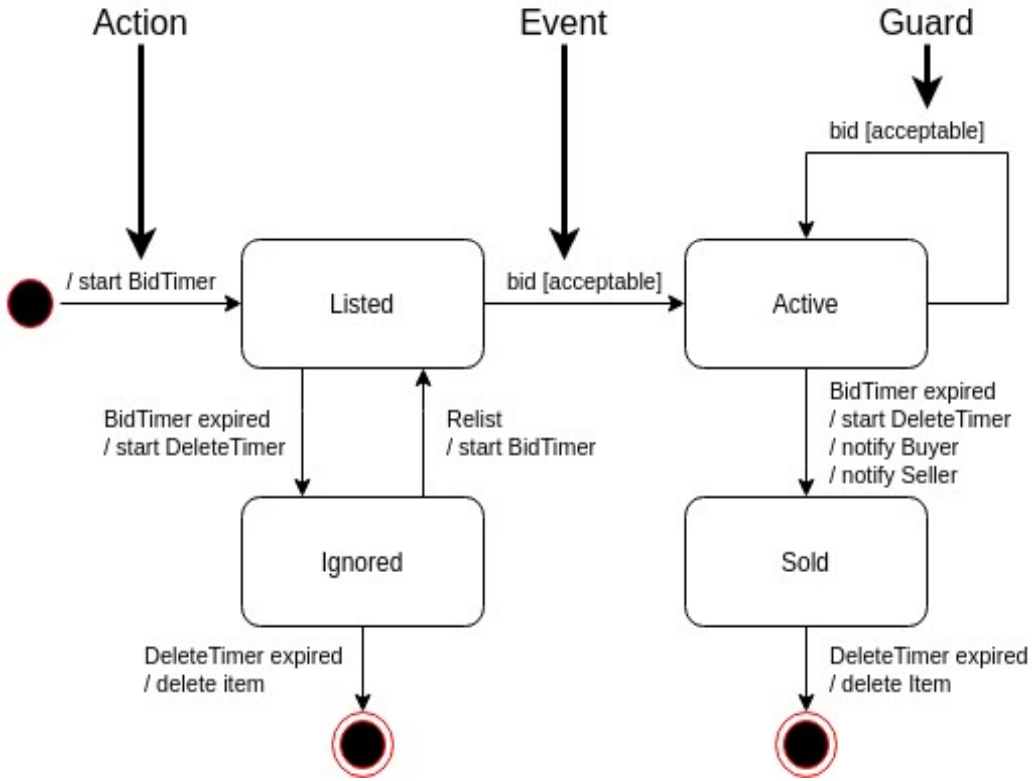


- **Package Diagram**
  - We use Package Diagrams to depict how packages and their elements have been organized.
  - A package diagram simply shows us the dependencies between different packages and internal composition of packages.
  - Packages help us to organise UML diagrams into meaningful groups and make the diagram easy to understand.
  - They are primarily used to organise class and use case diagrams.



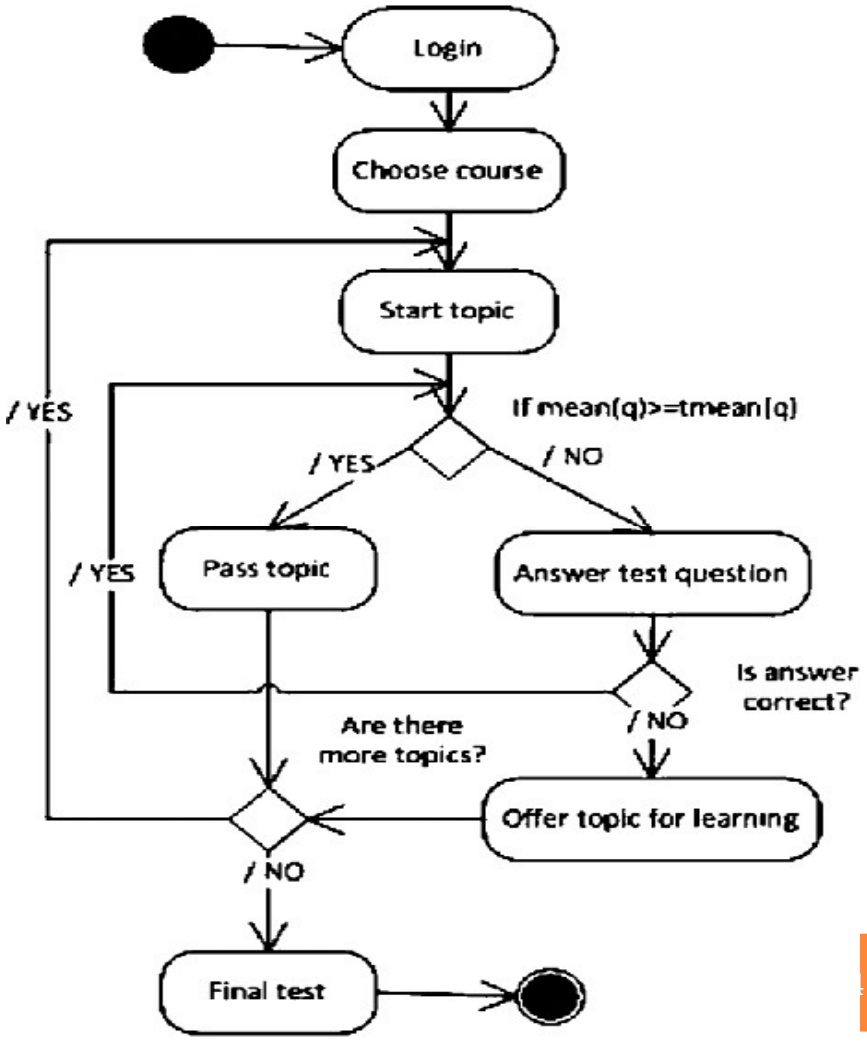
Behavioral UML Diagrams

- 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.
- It's a behavioral diagram and it represents the behavior using finite state transitions.
- State diagrams are also referred to as State machines and State-chart Diagrams
- These terms are often used interchangeably.
- So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli.



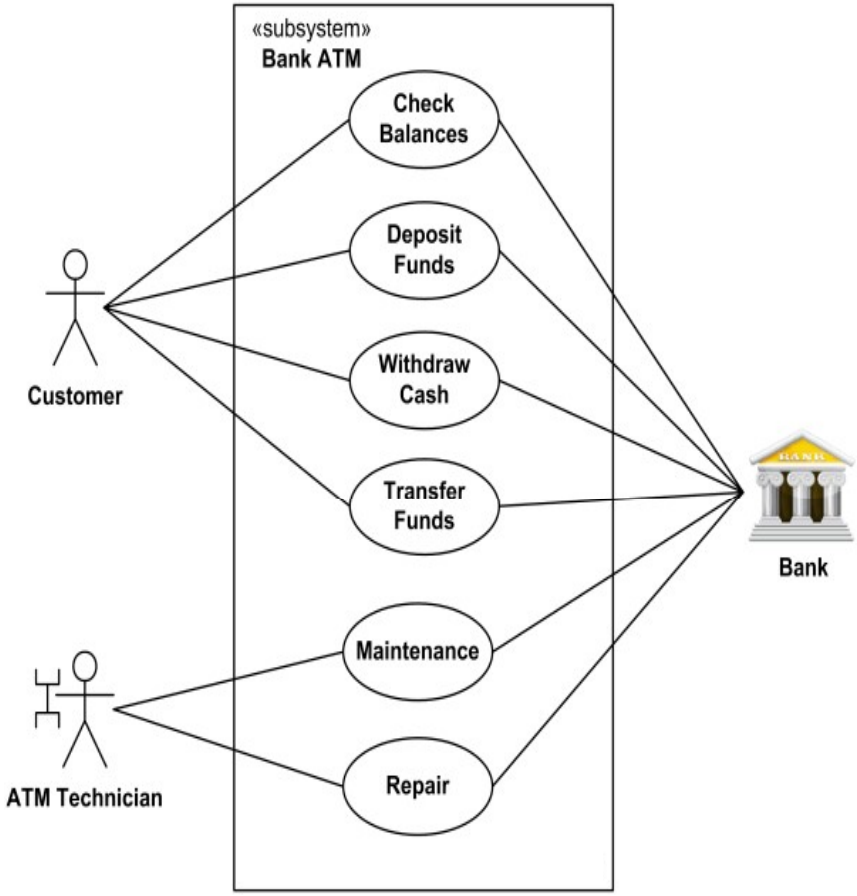
Activity Diagrams

- We use Activity Diagrams to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case.
- We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram.
- An activity diagram focuses on condition of flow and the sequence in which it happens.
- We describe or depict what causes a particular event using an activity diagram.



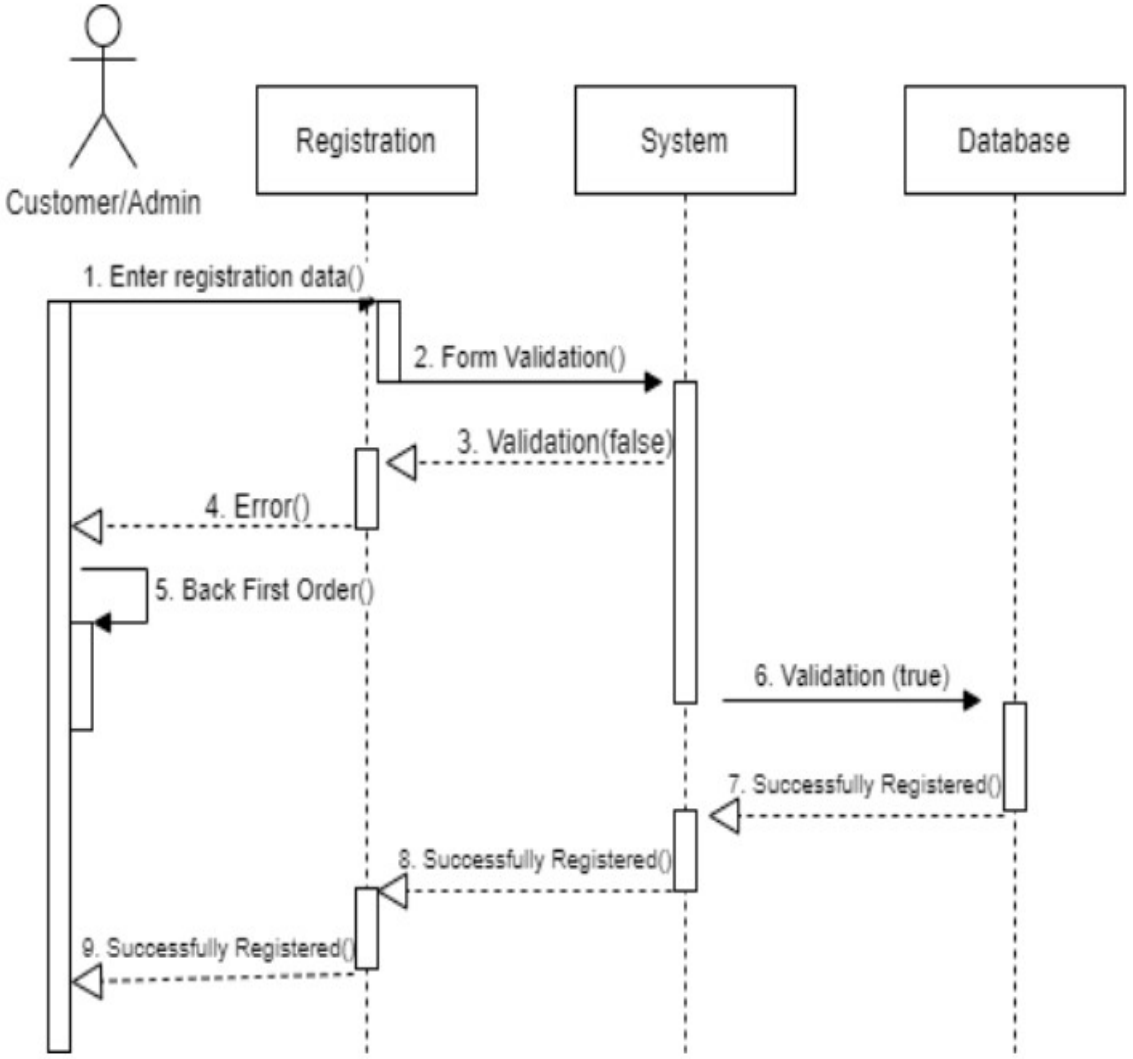
Use Case Diagrams

- 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).
- A use case is basically a diagram representing different scenarios where the system can be used.
- A use case diagram gives us a high level view of what the system or a part of the system does without going into implementation details.



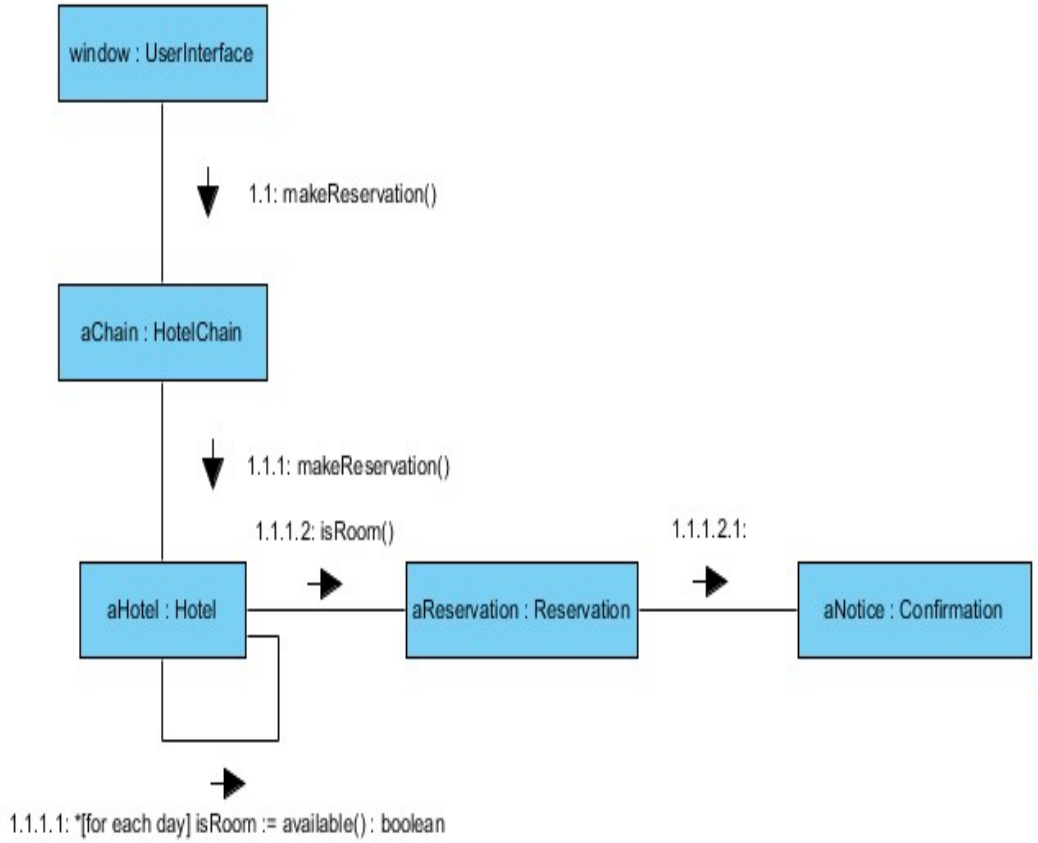
Sequence Diagram

- A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place.
- We can also use the terms event diagrams or event scenarios to refer to a sequence diagram.
- Sequence diagrams describe how and in what order the objects in a system function.
- These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.



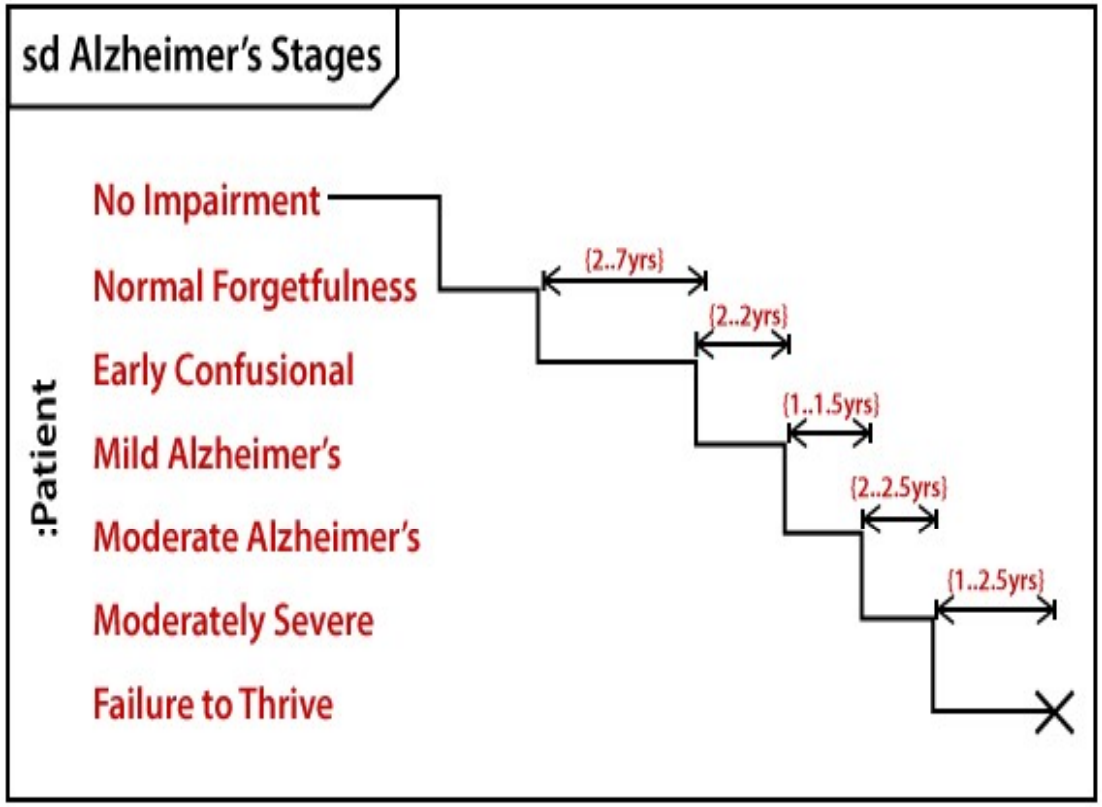
- **Communication Diagram**

- A Communication Diagram (known as Collaboration Diagram in UML 1.x) is used to show sequenced messages exchanged between objects.
- A communication diagram focuses primarily on objects and their relationships.
- We can represent similar information using Sequence diagrams, however communication diagrams represent objects and links in a free form.



Timing Diagram

- Timing Diagram are a special form of Sequence diagrams which are used to depict the behavior of objects over a time frame.
- We use them to show time and duration constraints which govern changes in states and behavior of objects.



Interaction Overview Diagram

- An Interaction Overview Diagram (IOD) is a type of UML (Unified Modeling Language) diagram that illustrates the flow of interactions between various elements in a system or process.
- It provides a high-level overview of how interactions occur, including the sequence of actions, decisions, and interactions between different components or objects.