Excise No.1: Process of sale

AIM: Analyze and Design Process of sale.

<u>Experiment Description:</u>

Main Success Scenario (or Basic Flow):

- Customer arrives at POS checkout with goods and/or services to purchase.
- Cashier starts a new sale.
- Cashier enters item identifier.
- System records sale line item and presents item description, price, and running total.
- Cashier repeats steps 3-4 until indicates done
- System presents total with taxes calculated.
- Cashier tells Customer the total, and asks for payment.
- Customer pays and System handles payment.
- System logs completed sale and sends sale and payment information to the external Accounting system and Inventory system (to update inventory).
- System presents receipt.
- Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows):

- at any time, System fails**:**
- Cashier restarts System, logs in, and requests recovery of prior state.
- Invalid identifier:
- There are multiple of same item category and tracking unique item identity not important (e.g., 5 packages of veggie-burgers)
- Customer asks Cashier to remove an item from the purchase:
- Customer tells Cashier to cancel sale:
- Cashier suspends the sale:
- the system generated item price is not wanted
- System detects failure to communicate with external tax calculation system service:
- Customer says they are eligible for a discount (e.g., employee, preferred customer):
- Customer says they have credit in their account, to apply to the sale:
- Customer says they intended to pay by cash but don't have enough cash:
- Paying by cash:
- Paying by credit:

Excise No. 2: Customer Support system

AIM: Analyze and design Customer Support system.

Experiment Description:

Product Delivery System:

- Choose your shipping and delivery options.
- There, it shows the delivery speed (Between dates to date).
- After we have to press continue button.

Create Order:

- Open the website (Amazon, Flipkart).
- Open the category list and search product.
- Select the item.
- Check the details of item. If it is ok add to the cart.
- If we want to buy multiple items repeat 3 & 4 steps.
- Proceed to check out the item.
- After preceding it shows the login address there we have to enter emailid &password.
- After login we have to select a delivery address if address is already
- Exist click on "deliver to this address" or else we can enter new Delivery address.
- Confirm payment details.

Payment Confirmation:

- Select a payment method .There it will appears another payment method.
  - Credit Card
  - Debit Card
  - Net Banking
  - Cash on Delivery (COD)
- We have to enter the payment information.
- After we have to press continue button.
- It shows valid confirmation about the product.
- Click on confirm order.
- Here it shows the details of our place order of the given product with price.
- Last it will show order successfully placed".

Cancel order:

- For cancelling the order, we should login with our id & password.
- Click on select "your order".
- It displays "cancel request" button and press it.
- They send cancel order confirmation to customer mail & phone number.
- Order will be cancelled automatically within 24 hours.

Excise No.3: Library Management System.

AIM: Analyze and Design Library management System.

Experiment Description:

- The Library System is a web-based application used to automate a library. It allows the librarian to maintain the information about books, magazines and CDs.
- It also allows the librarian to maintain the information about its users. It provides the facilities such as search for items, browse, checkout items, and return items, make reservation, remove reservation etc. to its users.
- To borrow the items from the library, the users must register in the system. The search option allows the users to search for any item in the library.
- If the user finds that the required item is available in the library, he/she can check out the item from the library. If the item is currently not available in the library, the user can make reservation for the item.
- When the item becomes available the respective user who made the reservation for that item first is notified. The reservation is cancelled when the user checks out the item from the library or through an explicit cancellation procedure.
- The system allows the librarian to easily create, update, and delete information about titles, borrowers, items and reservations in the system.
- The librarian is an employee of the library who interacts with the borrowers whose work is supported by the system. The Library System can run on popular web-browser platforms like Windows Explorer, Netscape Navigator etc. It can be easily extended with new functionality.

Excise No.4: Airline Reservation System.

AIM: Analyze and Design Airline Reservation System.

EXPERMENT DESCRIPTION:

The Airlines Reservation system facilitates the user to view the flight schedules, inquire about the flight details, availability of seats and many more. The major functionality of system is to allow the user to book and cancels the flights as per user requirements. It also provides the administrator or manager to modify existing flights or to introduce new flights in the schedule.

Major features provided by the system are:

- **Flight Enquiry**
  The system allows the user or member to perform flight inquiry including flight scheduling, seats availability status, fare details, etc.
- **User Registration**
  It allows the user to register in order to be a member of the organization. User is then granted a privilege to book or cancels flights.
- **Flight Reservation**
  The system allows the member to book the flights as per his/her requirements. The member is prompt to enter the passenger details and credit card details. The member then receives the unique PNR No. and E-ticket.
- **Flight Cancellation**
  The functionality is used by the member to cancel an existing reservation made by the member earlier.
- **Administration**
  The administration module of the system allows the admin/manager to manage the flight scheduling. It provides the admin /manger to modify or change the existing flights or to introduce a new flight's. Apart from scheduling it also allow the admin/manager to generate report of daily or weekly transactions based on requirements.

Excise No.5: Hotel Management System.

AIM: Analyze and Design Hotel Management System.

EXPERMENT DESCRIPTION:

This Project is a fine thought to make the complex procedure of the Hotel management system to an easy manner which is systematic, modular designed, selective menu based user display. The modular design and constructed system is very much user oriented in which user can easily understand the tools and can do edit of his own choice. The system is not any tough more and does not possesses many applications but it is made by focusing on the maintaining records employee’s actions in a computerized system rather than time taking and cumbersome manual system.

Two kinds of users can handle the system.

- Online Users
- Administrator or Hotel Management.

The Online users are the customers or the staff who can see the news and updates of the Hotel and the Administrator are responsible for updating the Hotel details on computer. The Administrator is the authorized user who has power to change or edit the updates as well as the Password.

The daily activities includes the Room activities, Entering details of the new customer check in, To allocate a room as per the customer need and interest, Recording the checkout time and details, Releasing or Empty of room and to record the process in a computer system for future.

The application of the Hotel Management System bears the following functions to use by the Administrator.

- Room status
- New Room inauguration
- Allocated Room Modification
- Details for the Customer Check in and Check out
- New Customer Admission
- Allocation of Room as per the Customer Interest
- Statement and Transactions of the Customer
- Total Customers Present In The Hotel
- Separate Customer Report.

Excise No. 6: ATM (Automated Teller Machine)

AIM: Analyze and design ATM.

Experiment Description

Automated Teller Machine enables the clients of a bank to have access to their account without going to the bank. This is achieved only by development the application using online concepts. When the product is implemented, the user who uses this product will be able to see all the information and services provided by the ATM, when he enters the necessary option and arguments. The product also provides services like request for checks, deposit cash and other advanced requirement of the user. The data is stored in the database and is retrieved whenever necessary. The implementation needs ATM machine hardware to operate or similar simulated conditions can also be used to successfully use the developed product.

To develop this ATM system the entire operation has been divided into the following step:

- verification process
- language, service and account selection
- Banking services
- Transactions
- Special services

The program is designed in such a way that the user has to card and pin number. Once verified, he is provided a menu and he/she had to enter the option provided in the menu. . When the option is entered alone with the respective argument, then the payment history is displayed on the screen. The user also must be given option to browse through the pages like previous page, next page, etc. The user may experience a delay in retrieving or viewing the data, when there are many users logged on to the same bank branch system.

In each excise develop following diagrams and based on analysis:

Use Cases & Use Case scenarios:

Identifying actors

Find the external entities with which the system under development must interact. Candidates include groups of users who will require help from the system to perform their tasks and run the system's primary or secondary functions, as well as external hardware, software, and other systems. Define each candidate actor by naming it and writing a brief description.

These questions are useful in identifying actors:

- Who will supply, use, or remove information from the system?
- Who will use the system?
- Who is interested in a certain feature or service provided by the system?
- Who will support and maintain the system?
- What are the system's external resources?
- What other systems will need to interact with the system under development?

Review the list of stakeholders that you captured in the Vision statement. Not all stakeholders will be actors (meaning, they will not all interact directly with the system under development), but this list of stakeholders is useful for identifying candidates for actors.

Identifying use cases

The best way to find use cases is to consider what each actor requires of the system. For each actor, human or not, ask:

- ➢ What are the goals that the actor will attempt to accomplish with the system?
- ➢ What are the primary tasks that the actor wants the system to perform?
- ➢ Will the actor create, store, change, remove, or read data in the system?
- ➢ Will the actor need to inform the system about sudden external changes?
- ➢ Does the actor need to be informed about certain occurrences, such as unavailability of a network resource, in the system?
- ➢ Will the actor perform a system startup or shutdown?

Understanding how the target organization works and how this information system might be incorporated into existing operations gives an idea of system's surroundings. That information can reveal other use case candidates.

Give a unique name and brief description that clearly describes the goals for each use case. If the candidate use case does not have goals, ask yourself why it exists, and then either identify a goal or eliminate the use case.

Outlining Use Cases

Without going into details, write a first draft of the flow of events of the use cases identified as being of high priority. Initially, write a simple step-by-step description of the basic flow of the use case. The step-by-step description is a simple ordered list of interactions between the actor and the system. For example, the description of the basic flow of the Withdraw Cash use case of an automated teller machine (ATM) would be something like this:

1. The customer inserts a bank card.
2. The system validates the card and prompts the person to enter a personal identification number (PIN).
3. The customer enters a PIN.
4. The system validates the PIN and prompts the customer to select an action.
5. The customer selects Withdraw Cash.
6. The system prompts the customer to choose which account.
7. The customer selects the checking account.
8. The system prompts for an amount.
9. The customer enters the amount to withdraw.
10. The system validates the amount (assuming sufficient funds), and then issues cash and receipt.
11. The customer takes the cash and receipt, and then retrieves the bank card.
12. The use case ends.

As you create this step-by-step description of the basic flow of events, you can discover alternative and exceptional flows. For example, what happens if the customer enters an invalid PIN? Record the name and a brief description of each alternate flow that you identified.

Crud Matrix:

SQL is consists of only 4 statements, sometimes referred to as CRUD:-

- Create - INSERT - to store new data
- Read - SELECT - to retrieve data
- Update - UPDATE - to change or modify data.
- Delete - DELETE - delete or remove data

A CRUD matrix is a table showing the Functions in an application containing SQL statement affecting parts of a database. The CRUD Matrix is an excellent technique to identify the Tables in a database which are used in any User interaction with a Web Site. CRUD means Create, Read, Update or Delete' and the CRUD Matrix identifies the Tables involved in any CRUD operation.

Domain models:

A domain model is a representation of real-world conceptual classes, not of software components. It is *not* a set of diagrams describing software classes, or software objects with responsibilities. Using UML notation, a domain model is illustrated with a set of class diagrams in which no operations are defined. It may show:

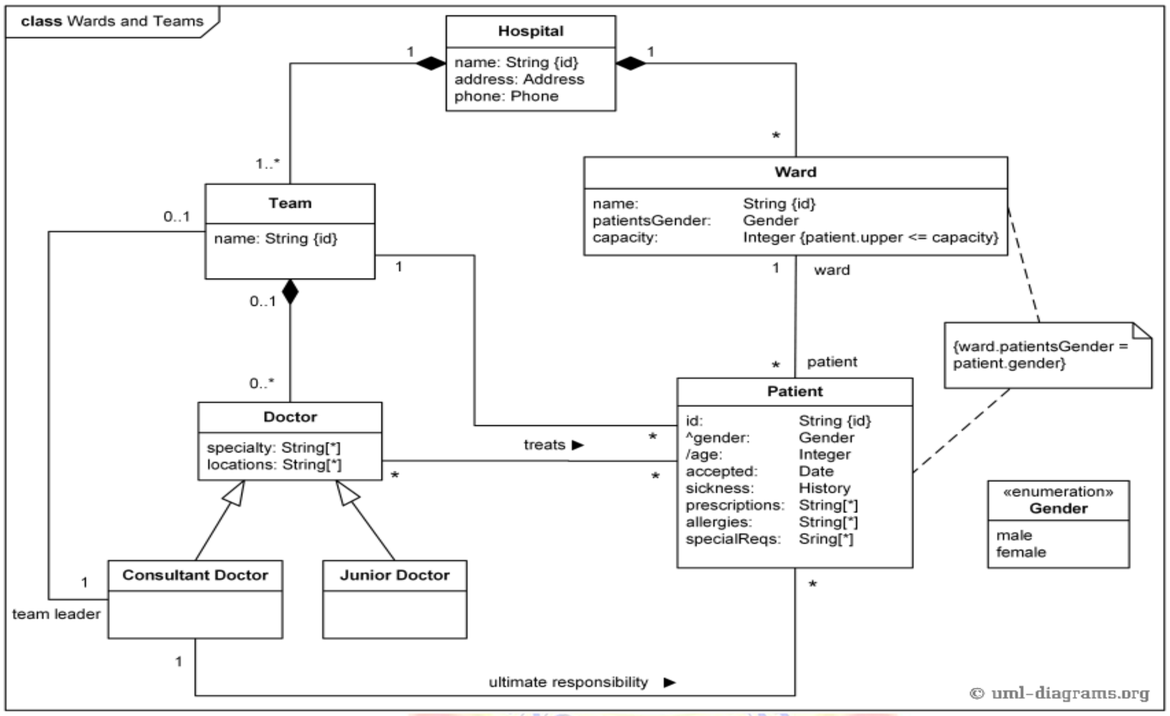
- Domain objects or conceptual classes

- Associations between conceptual classes
- attributes of conceptual classes

Strategies to Identify Conceptual Classes

- Based on category list.
- Identify noun phrases.

Sample domain class diagram:



Use case diagrams:

Introduction:

- A use case diagram describes how a system interacts with outside actors.
- It is a graphical representation of the interaction among the elements and system.
- Each use case representation a piece of functionality that a system provides to its user.
- Use case identifies the functionality of a system.
- Use case diagram allows for the specification of higher level user goals that the system must carry out.
- These goals are not necessarily to tasks or actions, but can be more general required functionality of the system.
- You can apply use case to capture the intended behaviour of the system you are developing, without having to specify how that behaviour is implemented.
- A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case.
- A use case diagram contains four components.
    1. The boundary, which defines the system of interest in relation to the world around it.
    2. The actors, usually individuals involved with the system defined according to their roles.
    3. The use cases, which the specific roles are played by the actors within and around the system.
    4. The relationships between and among the actors and the use cases.

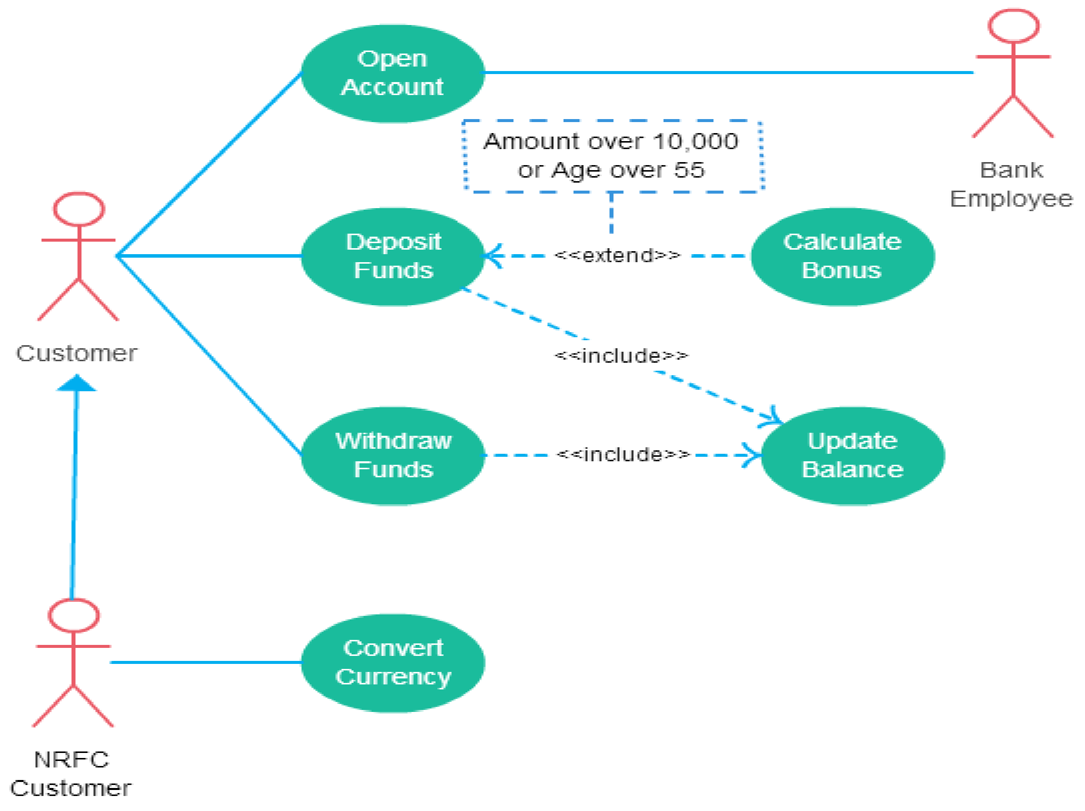
Purpose:

- The main purpose of the use case diagram is to capture the dynamic aspect of a system.
- Use case diagram shows, what software is suppose to do from user point of view.
- It describes the behaviour of system from user’s point.
- It provides functional description of system and its major processes.
- Use case diagram defines the scope of the system you are building.

When to Use: Use Cases Diagrams

- -Use cases are used in almost every project.
- They are helpful in exposing requirements and planning the project.
- During the initial stage of a project most use cases should be defined.

Sample use case Diagram:

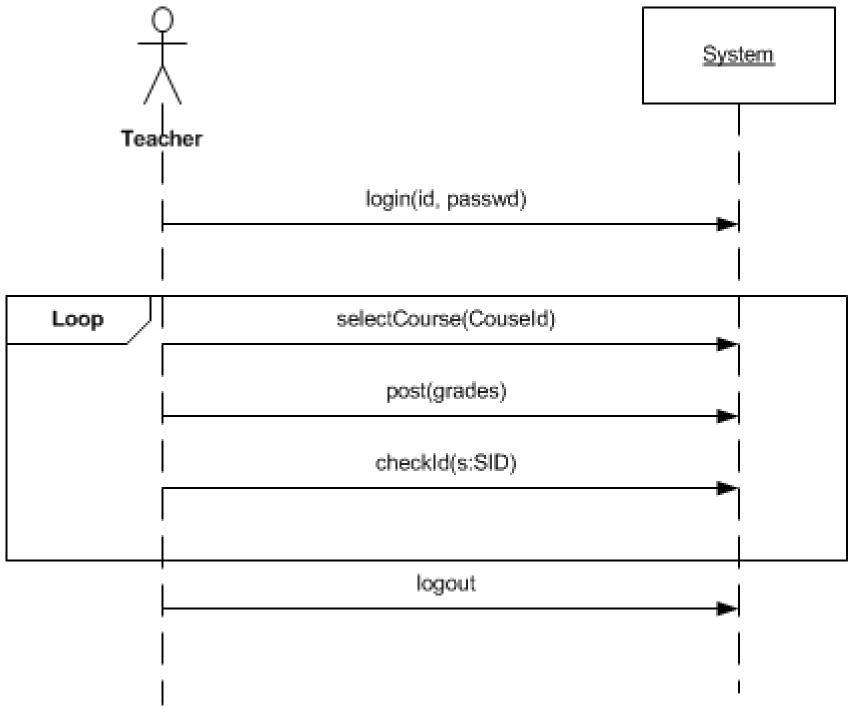


System sequence Diagram:

A system sequence diagram (SSD) is a picture that shows, for a particular scenario of a use case, the events that external actors generate their order, and inter-system events. All systems are treated as a black box; the emphasis of the diagram is events that cross the system boundary from actors to systems.

- It's purely represents interactions with External actors and the System.

Sample SSD:



Sequence diagrams:

A sequence diagram is an interaction diagram that emphasizes the time-ordering of messages.

- Sequence diagrams model the dynamic aspects of a software system.
- The emphasis is on the "sequence" of messages rather than relationship between objects.
- A sequence diagram maps the flow of logic or flow of control within a usage scenario into a visual diagram enabling the software architect to both document and validate the logic during the analysis and design stages.
- Sequence diagrams provide more detail and show the message exchanged among a set of objects over time.
- Sequence diagrams are good for showing the behaviour sequences seen by users of a diagram shows only the sequence of messages not their exact timing.
- Sequence diagrams can show concurrent signals.

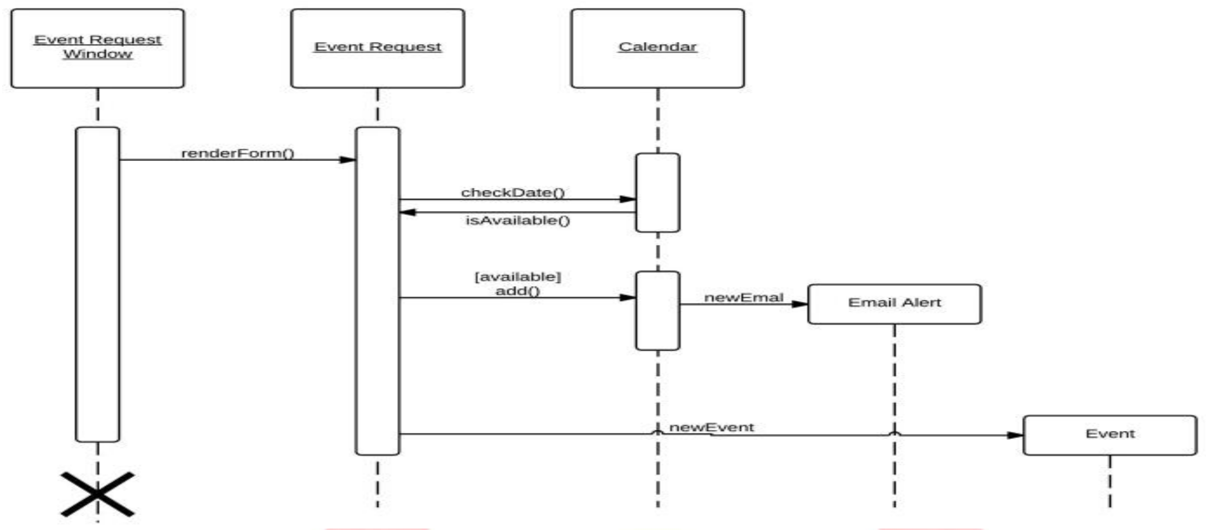
Purpose

- The main purpose of this diagram is to represent how different business objects interact.
- A sequence diagram shows object interactions arranged in time sequence.
- It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

When to use: Sequence Diagram

- Sequence diagram can be a helpful modelling tool when the dynamic behaviour of objects needs to be observed in a particular use case or when there is a need for visualizing the "big picture of message flow".
- A company's technical staff could utilize sequence diagrams in order to document the behaviour of a future system.
- It is during the design period that developers and architects utilize the diagram to showcase the system's object interactions, thereby putting out a more fleshed out overall system design.

Sample sequence diagram:

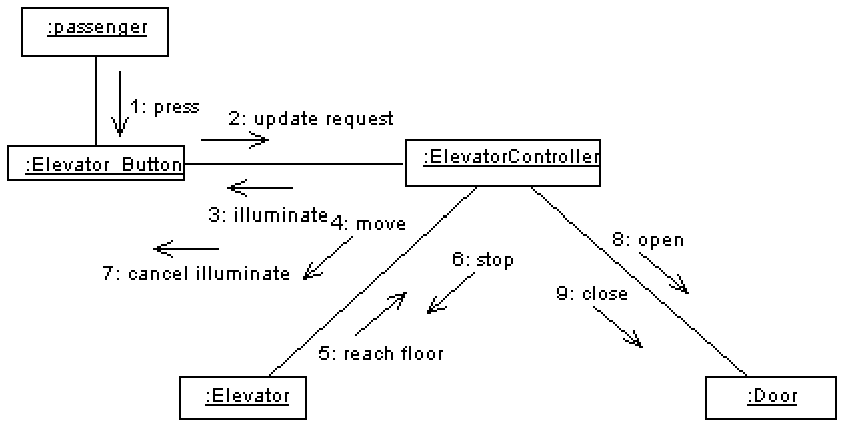


Collaboration Diagram:

Collaboration

- Cross between a symbol diagram and a sequence diagram (interaction).
- Describes a specific scenario.
- Numbered arrows show the movement of messages during the course of a scenario.
- Arrange the object as we like.
- Establish links with messages.
- Every message requires sequential number.

Sample collaboration diagram:



Class diagram:

A class diagram shows a set of classes, interfaces, and collaborations and their relationships.

Introduction

- The class diagram is a static diagram.
- A class model captures the static structure of a system by characterizing the objects in the system, the relationship between the objects, and the attributes and operations for each class of objects.
- The class diagram can be mapped directly with object oriented languages.
- The class model is the most important among the three models.
- Class diagram provide a graphical notation for modelling classes and their relationship.
- They are concise, easy to understand, and work well in practice.
- Class diagrams are the backbone of almost every object-oriented method including UML.
- They describe the static structure of a system.

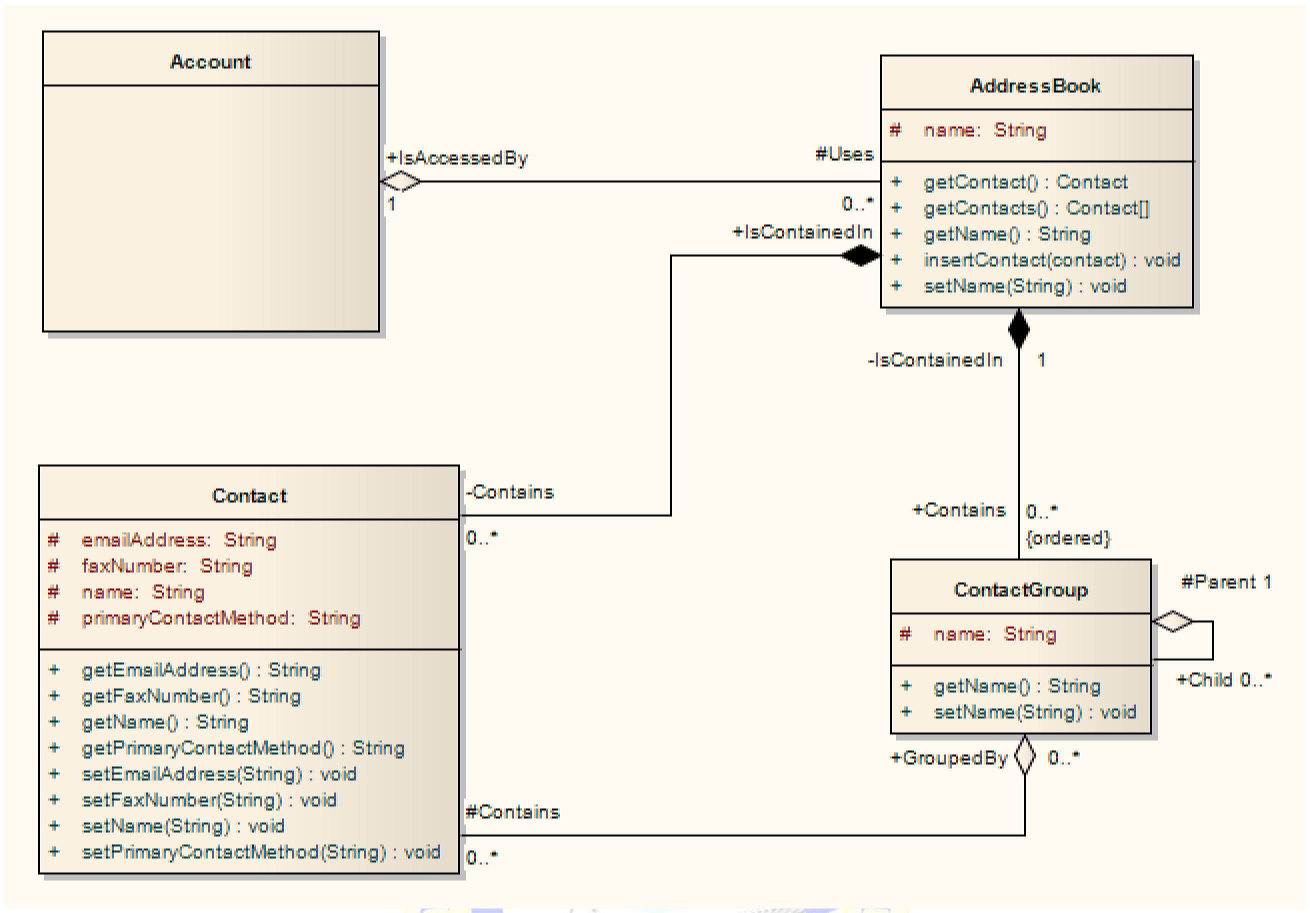
Purpose

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.

When to use: Class Diagram

- Useful for Forward and Reverse engineering.
- Class diagrams are useful both for abstract modelling and for designing actual programs.
- Developer uses class diagram for implementation decision.
- Business analysts can use class diagrams to model systems from the business perspective.

Sample class diagram:



Design to Code:

- Select code generation menu item.
- Select import for reverse engineering and export for forward engineering.
- Select specific language.
- Select required path and click on generate code.

Activity diagrams:

An activity diagram is a special kind of a state chart diagram that shows the flow from activity to activity within a system.

Introduction

- An activity diagram is a type of flow chart with additional support for parallel behaviour.
- This diagram explains overall flow of control.
- Activity diagram is another important diagram in UML to describe dynamic aspects of the system.
- Activity diagram is basically a flow chart to represent the flow from one activity to another activity
- The activity can be described as an operation of the system.
- The control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. This distinction is important for a distributed system.
- Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

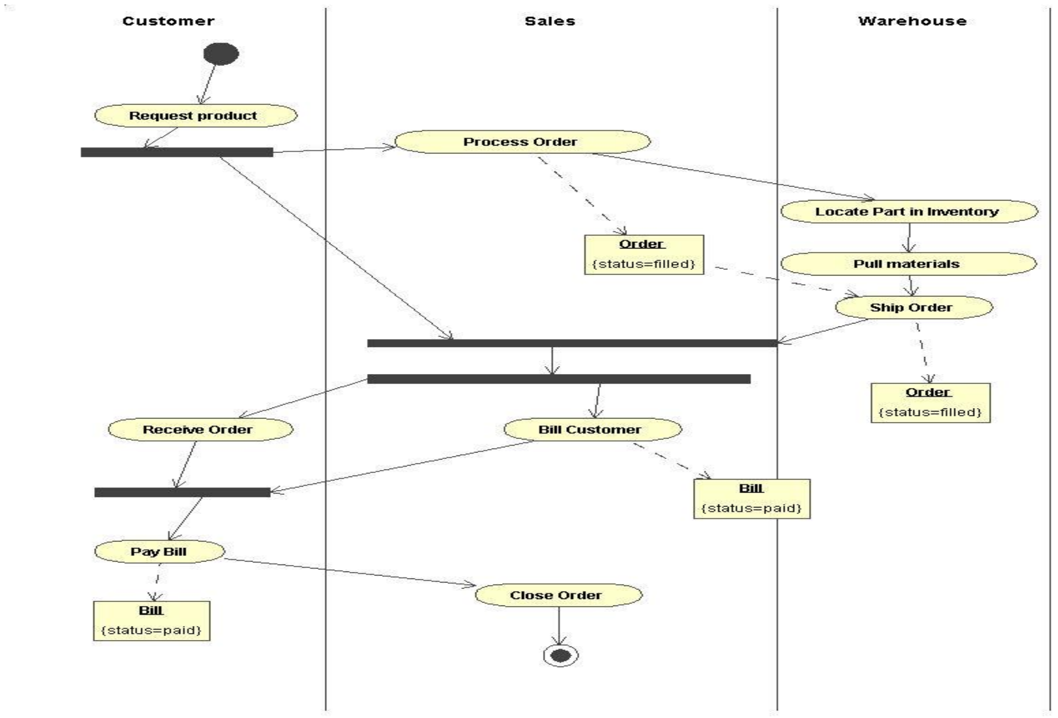
Purpose

- Contrary to use case diagrams, in activity diagrams it is obvious whether actors can perform business use cases together or independently from one another.
- Activity diagrams allow you to think functionally.

When to use: Activity Diagrams

- Activity diagrams are most useful when modelling the parallel behaviour of a multithreaded system or when documenting the logic of a business process.
- Because it is possible to explicitly describe parallel events, the activity diagram is well suited for the illustration of business processes, since business processes rarely occur in a linear manner and often exhibit parallelisms.
- This diagram is useful to investigate business requirements at a later stage.
- An activity diagram is drawn from a very high level. So it gives high level view of a system. This high level view is mainly for business users or any other person who is not a technical person.
- This diagram is used to model the activities which are nothing but business requirements.
- So the diagram has more impact on business understanding rather *implementation details*.

Sample diagram:



Component diagram:

A component diagram shows the dependencies among software components, including source code, binary code and executable components. Some components exist at compile time, some exist at link time, and some exist at run time; some exist at more than one time.

Components relationships are always required and provided interface.

Purpose:

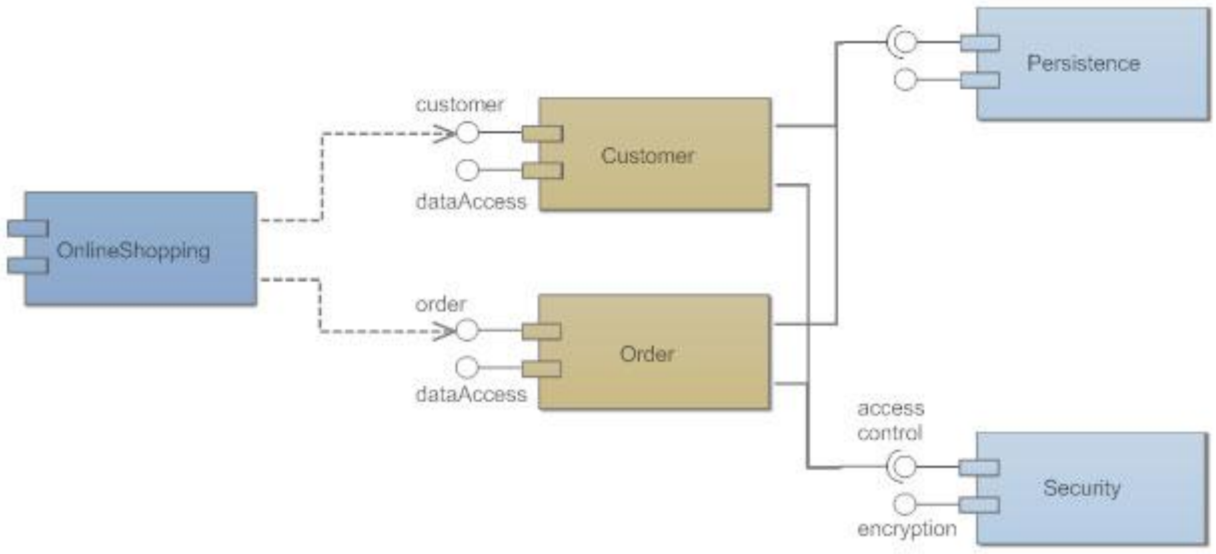
Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities. So from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.

So the purpose of the component diagram can be summarized as:

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

Sample Component Diagram:



State diagram:

State chart diagram

- A state chart diagram shows a state machine, consisting of states, transitions, events, and Activities.
- Provides a very detailed picture of how a specific symbols changes states.
- A state refers to the value associated with a specific attribute of an object and to any actions or side
- Effects that occur when the attribute's value changes.

Introduction

- A **state diagram** is a graph in which nodes correspond to states and directed arcs correspond to transitions labelled with event names.
- A state diagram combines states and events in the form of a network to model all possible object states during its life cycle, helping to visualize how an object responds to different stimuli.
- A state diagram is a graph whose nodes are states and whose directed arcs are transitions between states.
- A state diagram specifies the state sequence caused by event sequence.
- State names must be unique within the scope of a state diagram.
- All objects in a class execute the state diagram for that class, which models their common behaviour.
- We can implement state diagrams by direct interpretation or by converting the semantics into equivalent programming code.

Purpose

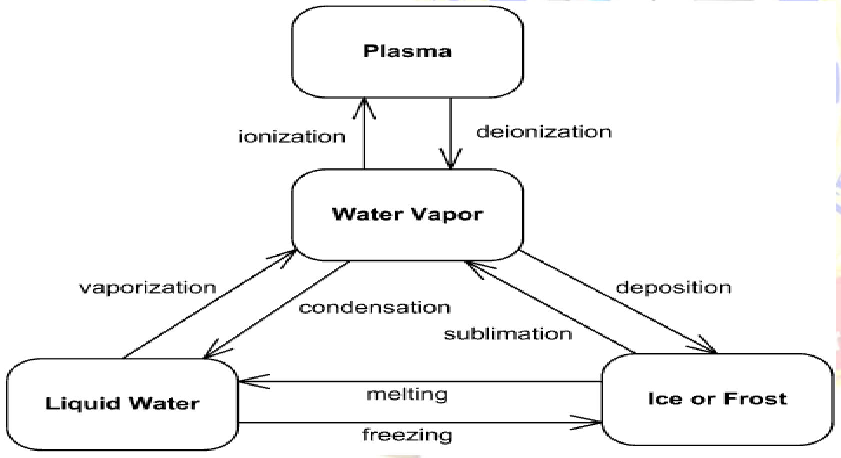
- The state model describes those aspects of objects concerned with time and the sequencing of operations events that mark changes, states that define the context for events, and the organization of events and states.

- They are used to give an abstract description of the behaviour of a system.
- It provides direction and guidance to the individual counties within the states.
- It specifies the possible states, what transitions are allowed between states.
- It describes the common behaviour for the objects in a class and each object changes its behaviour from one state to another.
- It is used to describe the dependence of the functionality on the state of the system that is how the functionality of an object depends on its state and how its state changes as a result of the events that it receives.
- It describes dynamic behaviour of the objects of the system.

When to use: State Diagram

- They are perfectly useful to model behaviour in real time system.
- Each state represents a named condition during the life of an object during which it satisfies some condition or waits for some event.
- It determines how objects of that class react to events.
- For each object state, it determines what actions the object will perform when it receives an event.

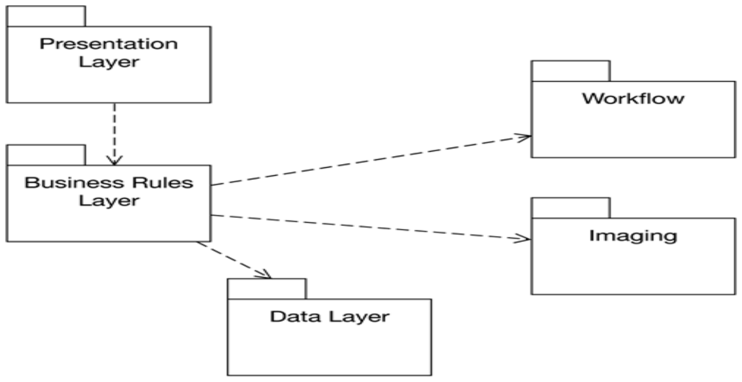
Sample sate diagram:



Layered Package Diagram:

A **package** in the **Unified Modeling Language** is used "to group elements, and to provide a namespace for the grouped elements". A **package** may contain other **packages**, thus providing for a hierarchical organization of **packages**. Pretty much all **UML** elements can be grouped into **packages**.

Sample package diagram:



Deployment diagram:

A Deployment diagram shows the configuration of run-time processing elements and the software components, processes, and objects. Software component instances represent run-time manifestations of code units. Components that do not exist as run-time entities do not appear on these diagrams. These components should be shown on component diagrams.

Purpose:

The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related.

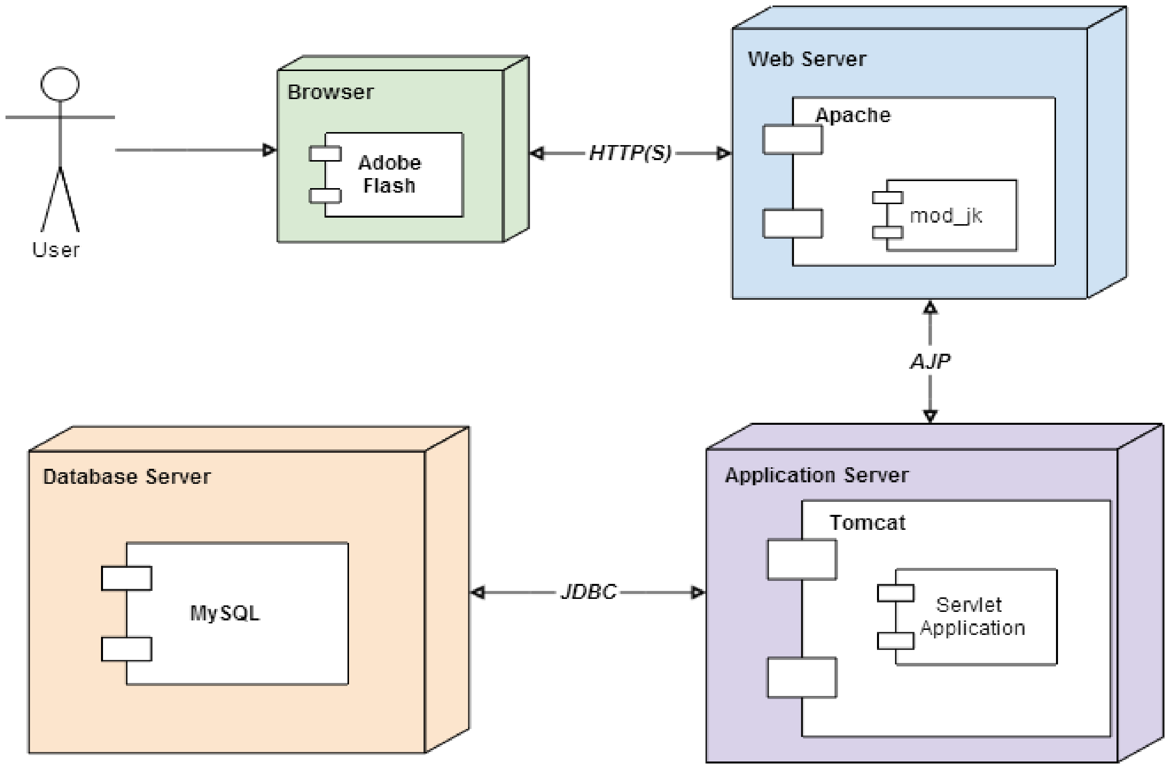
Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.UML is mainly designed to focus on software artefacts of a system. But these two diagrams are special diagrams used to focus on software components and hardware components.

So most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on hardware topology of a system. Deployment diagrams are used by the system engineers.

The purpose of deployment diagrams can be described as:

- Visualize hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe runtime processing nodes.

Sample deployment diagram:



Viva Questions

1. Description of Experiment?
2. Steps for identifying Use cases?
3. How to identify use cases?
4. What is the use of crud Matrix?
5. Who are the actors? What are Events of Actors? Who are the actors in your system?
6. What is Domain class? What are the steps for creating Domain class?
7. Difference b/w Domain class & Design class?
8. What is Use Case?
9. What are Use case scenarios?
10. What are the relationship b/w Use cases?
11. Difference b/w Sequence Diagram & Collaboration Diagram?
12. What is Class? What is Class diagram?
13. What are the steps for Converting Design to Code?
14. What is Package &what is Layered Package diagram?
15. Symbols & Definitions of each diagram
    - Use case diagram
    - Sequence diagram
    - Collaboration diagram
    - Class diagram
    - Component diagram
    - Deployment diagram
    - Object diagram
    - Activity diagram
    - State diagram.
16. Design each diagram.
17. What are the Building blocks of UML?
18. What is UML?
19. Designers of UML?
20. What is Design pattern?
21. What are GRASP Design patterns?
22. What are FURPS and FURPS+?
23. What are the phases of Unified process?
24. What are the phases of Software Development Life Cycle?
25. What is SSD? What is the use of SSD?
26. What are the supplementary specifications?
27. What is difference between Aggregation and composition?