Ezat Elzalouy

UML

(Unified Model Language)

____

*Is a graphical way of describing Software systems.

*In another way , UML is standard language for specifying , visualizing , constructing and documenting the artifacts of software systems.

*An object-oriented concepts were introduces earlier than UML .At that point of time, there were no standard methodologies to organize and consolidate the object oriented development.

Ezat Elzalouy

A-Coneptual Model

- A conceptual model can be defined as a model which is made of concepts and their relationships.
- A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other.

B-The conceptual model of UML

- can be mastered by learning the following three major elements

- UML building blocks
- Rules to connect the building blocks
- Common mechanisms of UML

Note: UML diagram is a full representation of object oriented concepts . So it becomes important to understand OO concept in detail.

C- Object-Oriented Concepts

- **Objects** – Objects represent an entity and the basic building block.
- **Class** – Class is the blue print of an object.
- **Encapsulation** – Encapsulation is the mechanism of binding the data together and hiding them from the outside world.
- **Inheritance** – Inheritance is the mechanism of making new classes from existing ones.
- **Polymorphism** – It defines the mechanism to exist in different forms.

D-OO Analysis and Design

-The purpose of OO analysis and design can described as

- Identifying the objects of a system.
- Identifying their relationships.
- Making a design, which can be converted to executables using OO languages.

1-UML building blocks

The building blocks of UML can be defined as –

- Things
- Relationships
- Diagrams

1-Things

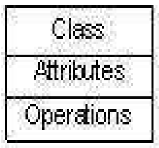
Things are the most important building blocks of UML. Things can be

- Structural
- Behavioral
- Grouping
- Annotational

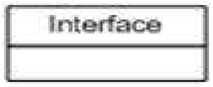
A-Structural things

Structural things define the static part of the model. They represent the physical and conceptual elements.

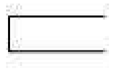
- **Class –** Class represents a set of objects having similar responsibilities.



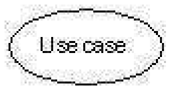
- **Interface –** Interface defines a set of operations, which specify the responsibility of a class.



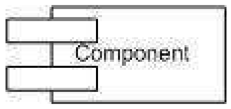
- **Collaboration –**Collaboration defines an interaction between elements.



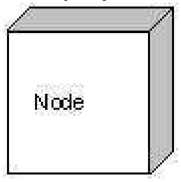
- **Use case –**Use case represents a set of actions performed by a system for a specific goal.



- **Component –**Component describes the physical part of a system.



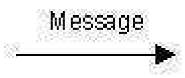
- **Node –** A node can be defined as a physical element that exists at run time.



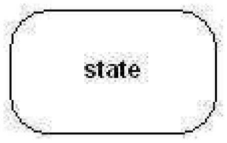
B-Behavioral Things

A behavioral thing consists of the dynamic parts of UML models. Following are the behavioral things –

- **Interaction –** Interaction is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.



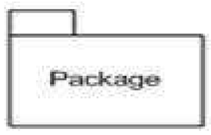
- **State machine –** State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change



C-Grouping Things

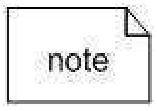
Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available –

- **Package –** Package is the only one grouping thing available for gathering structural and behavioral things.



D-Annotational Things

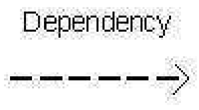
Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. **Note** - It is the only one Annotational thing available. A note is used to render comments, constraints, etc. of an UML element.



2-Relationship

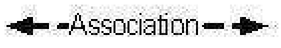
Relationship is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application.

- Dependency is a relationship between two things in which change in one element also affects the other.



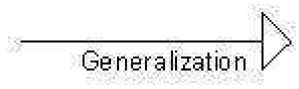
- Association

Association is basically a set of links that connects the elements of a UML model. It also describes how many objects are taking part in that relationship.



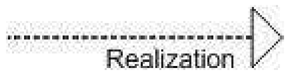
- Generalization

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes the inheritance relationship in the world of objects.



- Realization

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.



UML Diagram

____

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.

UML includes the following nine diagrams, the details of which are described in the subsequent chapters.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- Statechart diagram
- Deployment diagram
- Component diagram

UML - Architecture

UML plays an important role in defining different perspectives of a system. These perspectives are –

- Design
- Implementation
- Process
- Deployment

Design of a system consists of classes, interfaces, and collaboration. UML provides class diagram, object diagram to support this.

Implementation defines the components assembled together to make a complete physical system. UML component diagram is used to support the implementation perspective.

Process defines the flow of the system. Hence, the same elements as used in Design are also used to support this perspective.

Deployment represents the physical nodes of the system that forms the hardware. UML deployment diagram is used to support this perspective

The center is the **Use Case** view which connects all these four. A **Use Case**represents the functionality of the system. Hence, other perspectives are connected with use case.

- ### Structural Things

Graphical notations used in structural things are most widely used in UML. These are considered as the nouns of UML models. Following are the list of structural things.

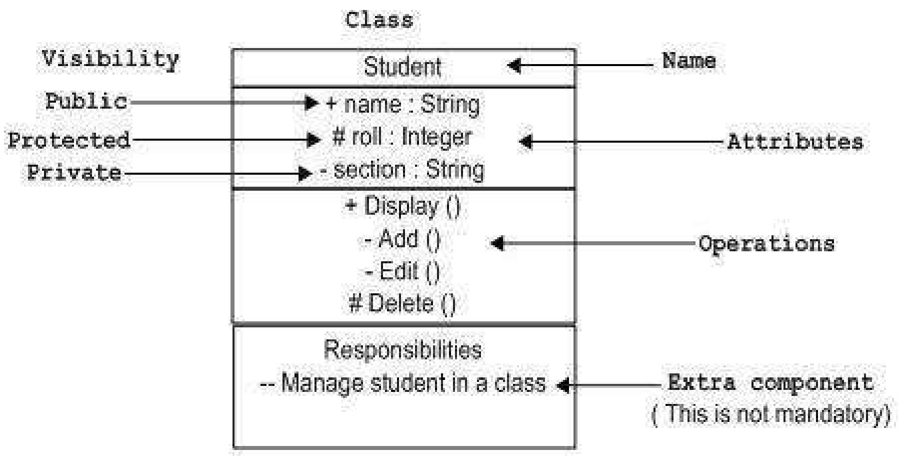
- Classes
- Object
- Interface
- Collaboration
- Use case
- Active classes

- Components
- Nodes

1-Class Notations

UML *class* is represented by the following figure. The diagram is divided into four parts.

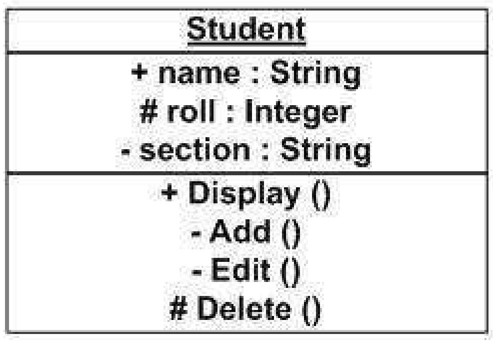
- The top section is used to name the class.
- The second one is used to show the attributes of the class.
- The third section is used to describe the operations performed by the class.
- The fourth section is optional to show any additional components.



Classes are used to represent objects. Objects can be anything having properties and responsibility.

2-Object Notation

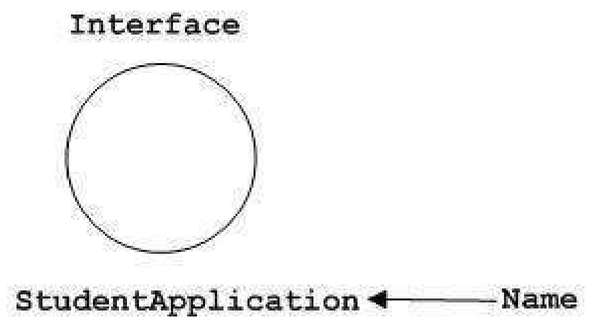
The *object* is represented in the same way as the class. The only difference is the *name* which is underlined as shown in the following figure.



As the object is an actual implementation of a class, which is known as the instance of a class. Hence, it has the same usage as the class.

3-Interface Notation

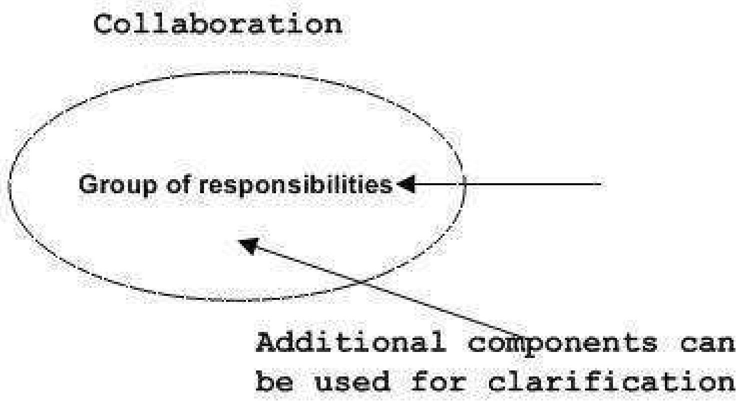
Interface is represented by a circle as shown in the following figure. It has a name which is generally written below the circle.



Interface is used to describe the functionality without implementation. Interface is just like a template where you define different functions, not the implementation. When a class implements the interface, it also implements the functionality as per requirement.

4-Collaboration Notation

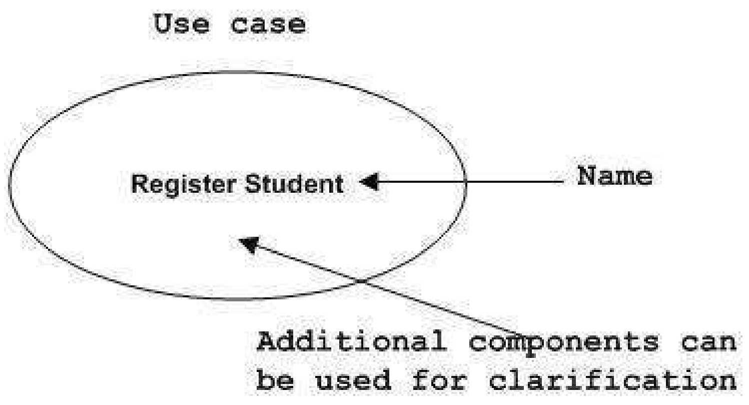
Collaboration is represented by a dotted eclipse as shown in the following figure. It has a name written inside the eclipse.


Collaboration
Group of responsibilities
Additional components can be used for clarification

Collaboration represents responsibilities. Generally, responsibilities are in a group.

5-Use Case Notation

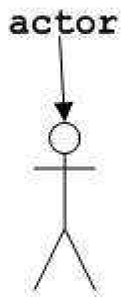
Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.



Use case is used to capture high level functionalities of a system.

6-Actor Notation

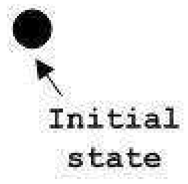
An actor can be defined as some internal or external entity that interacts with the system.



An actor is used in a use case diagram to describe the internal or external entities.

7-Initial State Notation

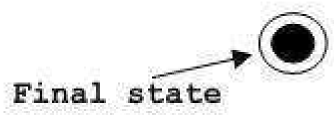
Initial state is defined to show the start of a process. This notation is used in almost all diagrams.



The usage of Initial State Notation is to show the starting point of a process.

8-Final State Notation

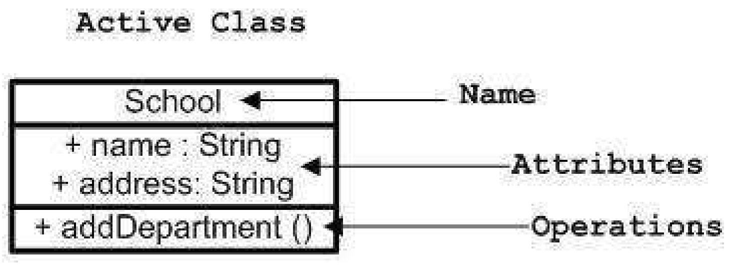
Final state is used to show the end of a process. This notation is also used in almost all diagrams to describe the end.



The usage of Final State Notation is to show the termination point of a process.

9-Active Class Notation

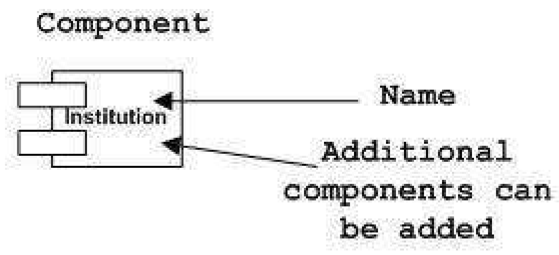
Active class looks similar to a class with a solid border. Active class is generally used to describe the concurrent behavior of a system.



Active class is used to represent the concurrency in a system.

10-Component Notation

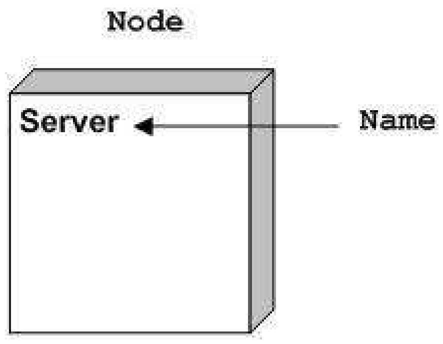
A component in UML is shown in the following figure with a name inside. Additional elements can be added wherever required.



Component is used to represent any part of a system for which UML diagrams are made.

11-Node Notation

A node in UML is represented by a square box as shown in the following figure with a name. A node represents the physical component of the system.



Node is used to represent the physical part of a system such as the server, network, etc.

- ## Behavioral Things

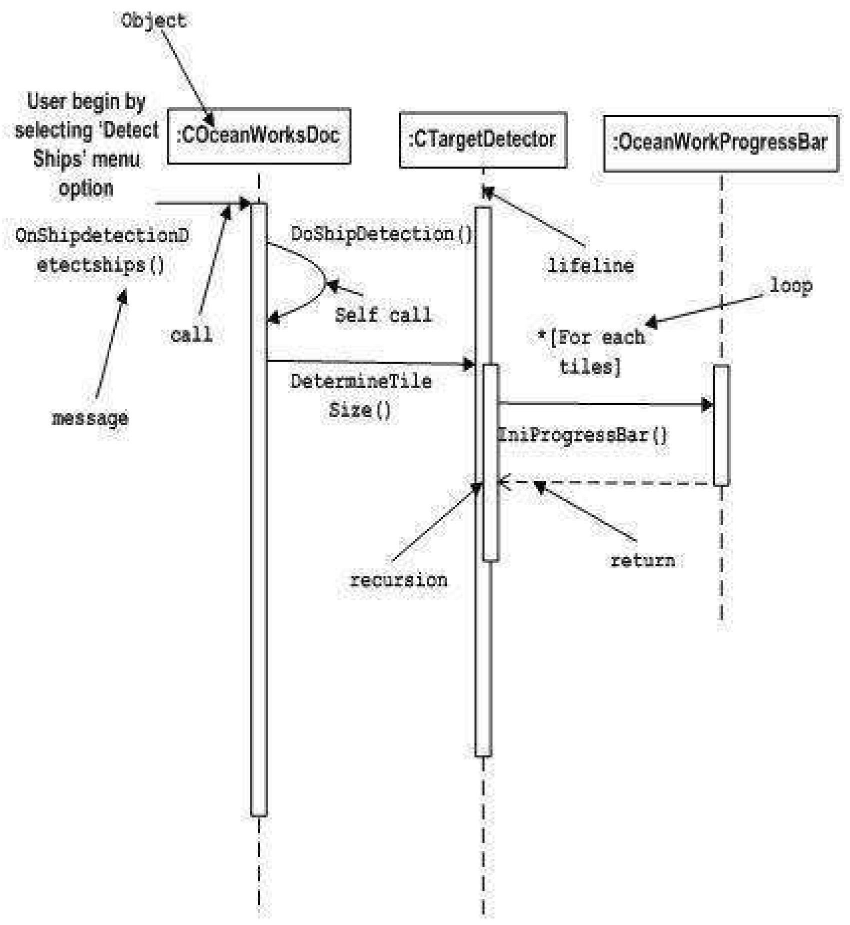
Dynamic parts are one of the most important elements in UML. UML has a set of powerful features to represent the dynamic part of software and non-software systems. These features include *interactions* and *state machines*.

Interactions can be of two types –

- Sequential (Represented by sequence diagram)
- Collaborative (Represented by collaboration diagram)

1-Interaction Notation

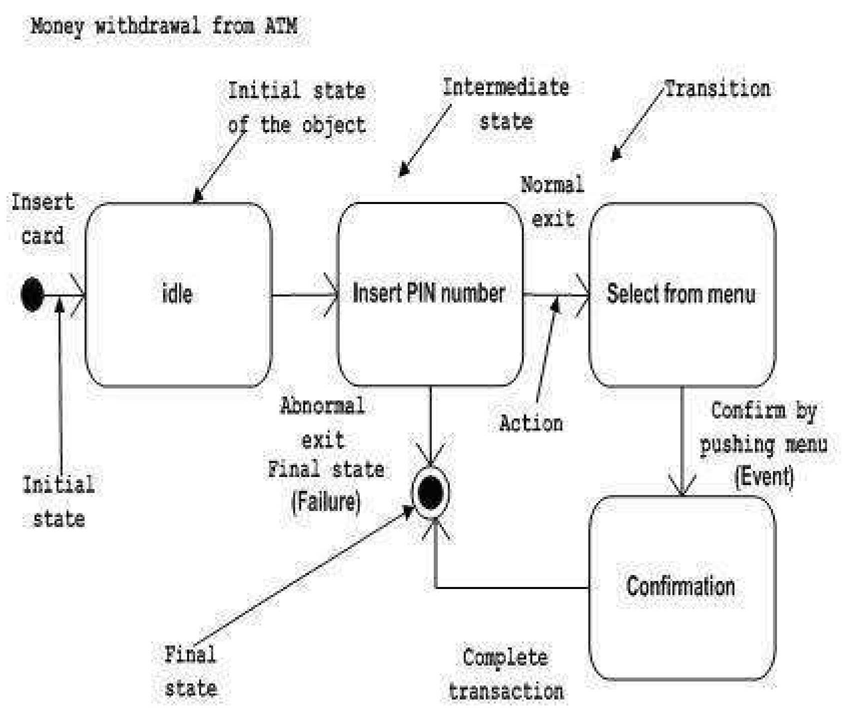
Interaction is basically a message exchange between two UML components. The following diagram represents different notations used in an interaction.



Interaction is used to represent the communication among the components of a system.

2-State Machine Notation

State machine describes the different states of a component in its life cycle. The notations are described in the following diagram.



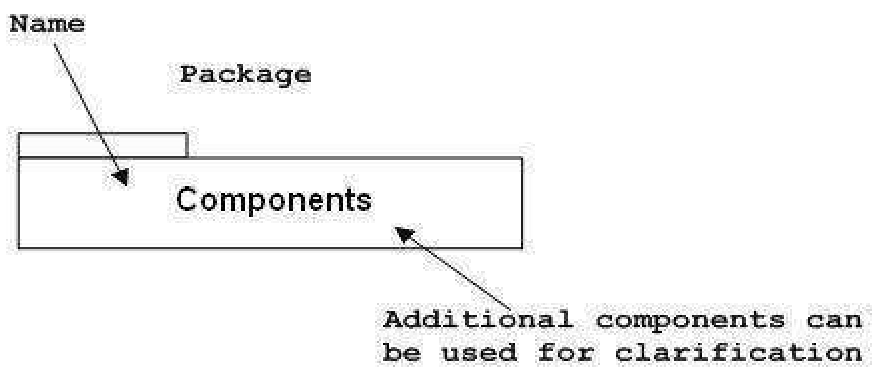
State machine is used to describe different states of a system component. The state can be active, idle, or any other depending upon the situation.

• Grouping Things

Organizing the UML models is one of the most important aspects of the design. In UML, there is only one element available for grouping and that is package.

-Package Notation

Package notation is shown in the following figure and is used to wrap the components of a system.

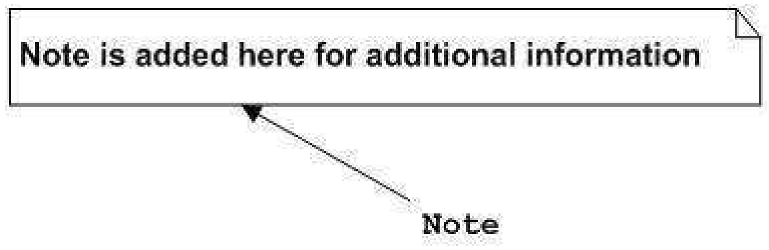


- Annotational Things

In any diagram, explanation of different elements and their functionalities are very important. Hence, UML has *notes* notation to support this requirement.

Note Notation

This notation is shown in the following figure. These notations are used to provide necessary information of a system.



Relationship

model is not complete unless the relationships between elements are described properly. The *Relationship* gives a proper meaning to a UML model. Following are the different types of relationships available in UML.

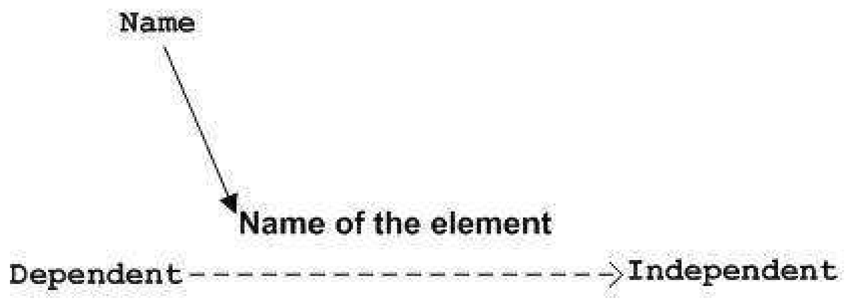
- Dependency
- Association
- Generalization
- Extensibility

1-Dependency Notation

Dependency Notation

Dependency is an important aspect in UML elements. It describes the dependent elements and the direction of dependency.

Dependency is represented by a dotted arrow as shown in the following figure. The arrow head represents the independent element and the other end represents the dependent element.

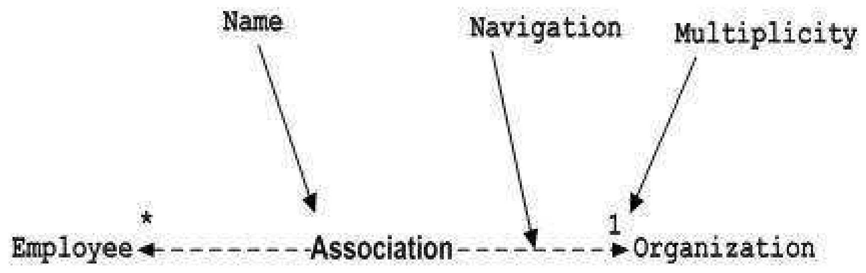


Dependency is used to represent the dependency between two elements of a system

2-Association Notation

Association describes how the elements in a UML diagram are associated. In simple words, it describes how many elements are taking part in an interaction.

Association is represented by a dotted line with (without) arrows on both sides. The two ends represent two associated elements as shown in the following figure. The multiplicity is also mentioned at the ends (1, *, etc.) to show how many objects are associated.

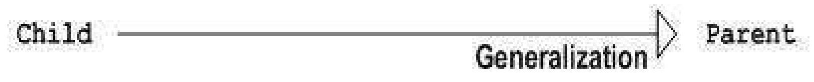


Association is used to represent the relationship between two elements of a system.

3-Generalization Notation

Generalization describes the inheritance relationship of the object-oriented world. It is a parent and child relationship.

Generalization is represented by an arrow with a hollow arrow head as shown in the following figure. One end represents the parent element and the other end represents the child element.

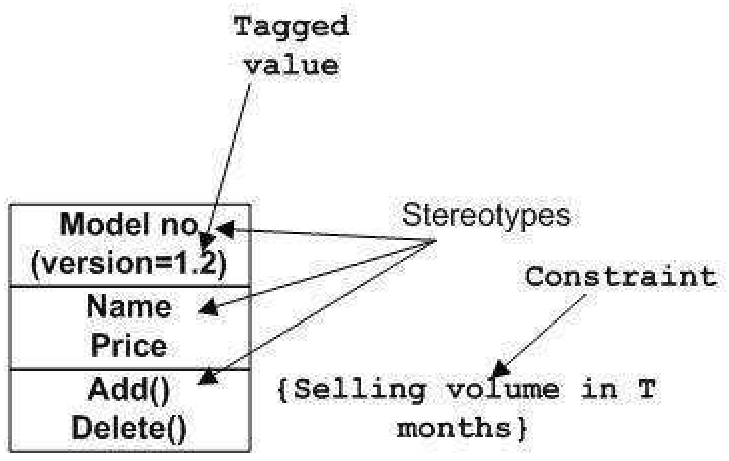


Generalization is used to describe parent-child relationship of two elements of a system.

4-Extensibility Notation

All the languages (programming or modeling) have some mechanism to extend its capabilities such as syntax, semantics, etc. UML also has the following mechanisms to provide extensibility features.

- Stereotypes (Represents new elements)
- Tagged values (Represents new attributes)
- Constraints (Represents the boundaries)



Extensibility notations are used to enhance the power of the language. It is basically additional elements used to represent some extra behavior of the system. These extra behaviors are not covered by the standard available notations.

Ezat Elzalouy

Structural Diagram

____

1-Class Diagram

The purpose of the class diagram can be summarized as –

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

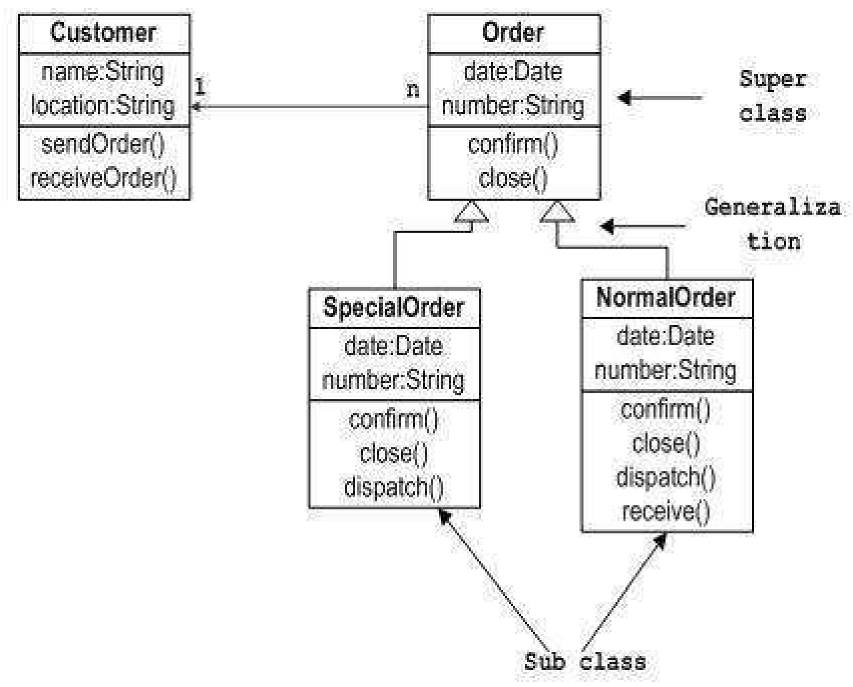
The following diagram is an example of an Order System of an application. It describes a particular aspect of the entire application.

- First of all, Order and Customer are identified as the two elements of the system. They have a one-to-many relationship because a customer can have multiple orders.
- Order class is an abstract class and it has two concrete classes (inheritance relationship) SpecialOrder and NormalOrder.
- The two inherited classes have all the properties as the Order class. In addition, they have additional functions like dispatch () and receive ().

The following class diagram has been drawn considering all the points mentioned above.

Ezat Elzalouy

Sample Class Diagram



2-Object Diagram

- Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.

- The purpose of the object diagram can be summarized as −

1. Forward and reverse engineering.
2. Object relationships of a system
3. Static view of an interaction
4. Understand object behaviour and their relationship from practical perspective

-Example

The following diagram is an example of an object diagram. It represents the Order management system which we have discussed in the chapter Class Diagram. The following diagram is an instance of the system at a particular time of purchase. It has the following objects.

- Customer
- Order
- SpecialOrder
- NormalOrder

Now the customer object (C) is associated with three order objects (O1, O2, and O3). These order objects are associated with special order and normal order objects (S1, S2, and N1). The customer has the following three orders with different numbers (12, 32 and 40) for the particular time considered.

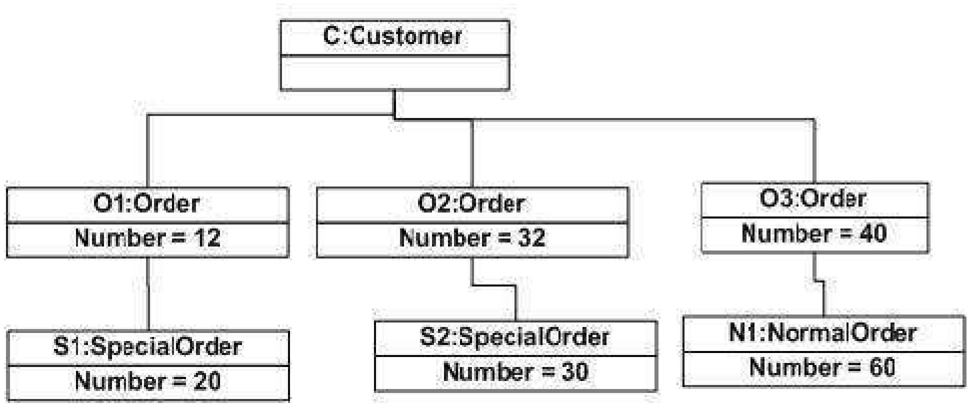
The customer can increase the number of orders in future and in that scenario the object diagram will reflect that. If order, special order, and normal order objects are observed then you will find that they have some values.

For orders, the values are 12, 32, and 40 which implies that the objects have these values for a particular moment (here the particular time when the purchase is made is considered as the moment) when the instance is captured

The same is true for special order and normal order objects which have number of orders as 20, 30, and 60. If a different time of purchase is considered, then these values will change accordingly.

The following object diagram has been drawn considering all the points mentioned above

Object diagram of an order management system



3-Component Diagram

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.

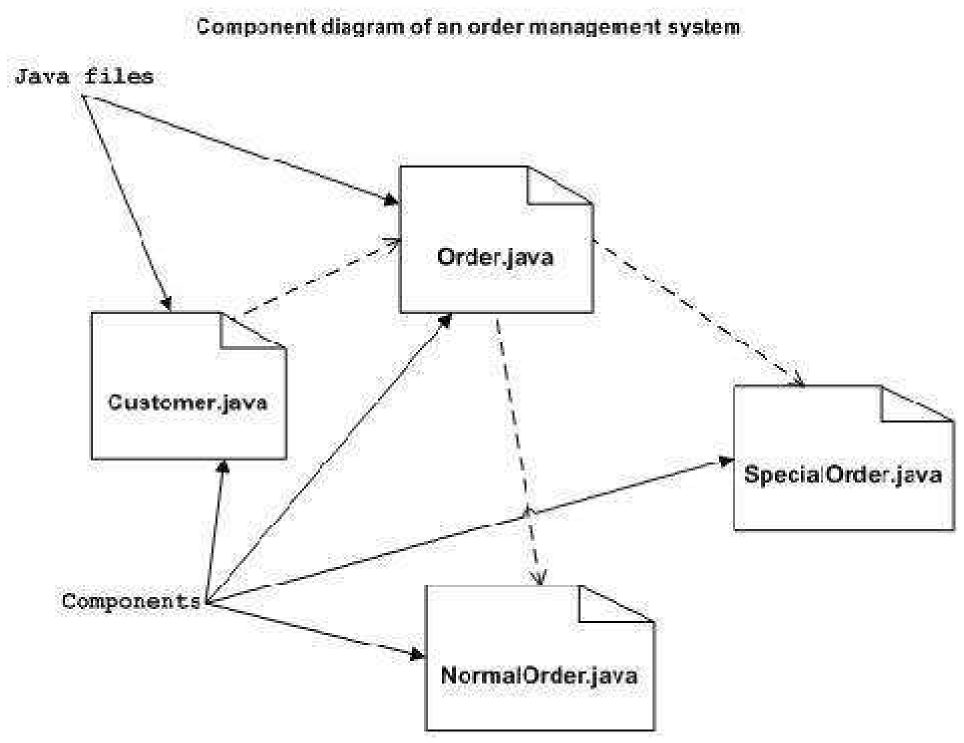
Before drawing a component diagram, the following artifacts are to be identified clearly −

- Files used in the system.
- Libraries and other artifacts relevant to the application.
- Relationships among the artifacts.

Following is a component diagram for order management system. Here, the artifacts are files. The diagram shows the files in the application and their relationships. In actual, the component diagram also contains dlls, libraries, folders, etc.

In the following diagram, four files are identified and their relationships are produced. Component diagram cannot be matched directly with other UML diagrams discussed so far as it is drawn for completely different purpose.

The following component diagram has been drawn considering all the points mentioned above.



4-Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.

The purpose of deployment diagrams can be described as −

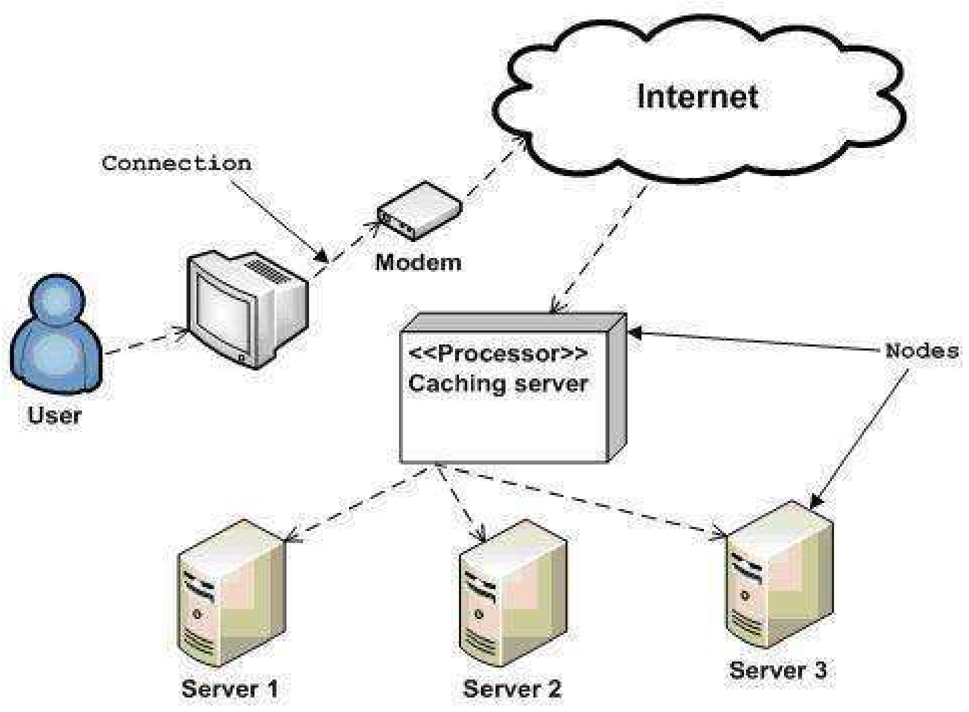
- Visualize the hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe the runtime processing nodes.

Following is a sample deployment diagram to provide an idea of the deployment view of order management system. Here, we have shown nodes as −

- Monitor
- Modem
- Caching server
- Server

The application is assumed to be a web-based application, which is deployed in a clustered environment using server 1, server 2, and server 3. The user connects to the application using the Internet. The control flows from the caching server to the clustered environment.

Deployment diagram of an order management system



Deployment diagrams can be used –

- To model the hardware topology of a system.
- To model the embedded system.
- To model the hardware details for a client/server system.
- To model the hardware details of a distributed application.
- For Forward and Reverse engineering.

5-Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

In brief, the purposes of use case diagrams can be said to be as follows −

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors.

Following is a sample use case diagram representing the order management system. Hence, if we look into the diagram then we will find three use cases **(Order, SpecialOrder, and NormalOrder)** and one actor which is the customer.

The SpecialOrder and NormalOrder use cases are extended from *Order* use case. Hence, they have extended relationship. Another important point is to identify the system boundary, which is shown in the picture. The actor Customer lies outside the system as it is an external user of the system.

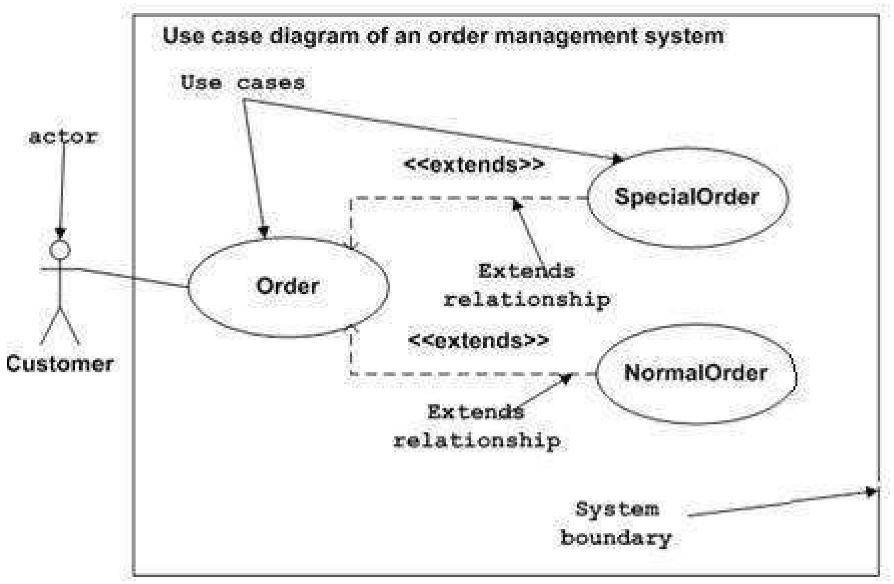


Figure: Sample Use Case diagram

Use case diagrams can be used for −

- Requirement analysis and high level design.
- Model the context of a system.
- Reverse engineering.
- Forward engineering.

6-Interaction Diagram

From the term Interaction, it is clear that the diagram is used to describe some type of interactions among the different elements in the model. This interaction is a part of dynamic behavior of the system.

This interactive behavior is represented in UML by two diagrams known as **Sequence diagram** and **Collaboration diagram**. The basic purpose of both the diagrams are similar.

Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

The purpose of interaction diagram is −

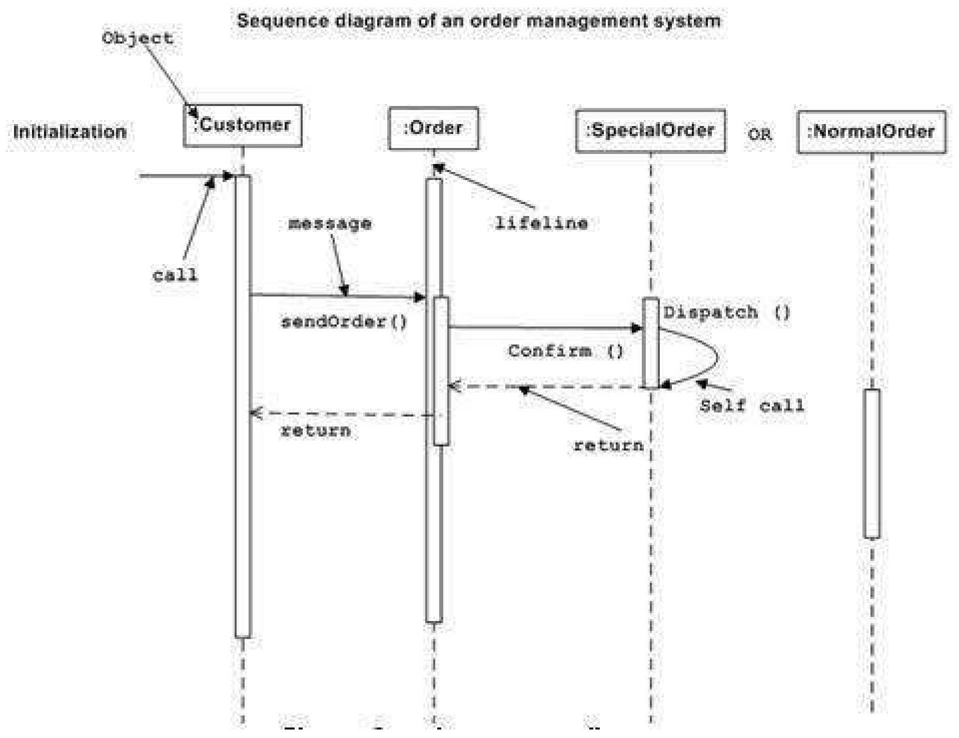
- To capture the dynamic behaviour of a system.
- To describe the message flow in the system.
- To describe the structural organization of the objects.
- To describe the interaction among objects.

A-The Sequence Diagram

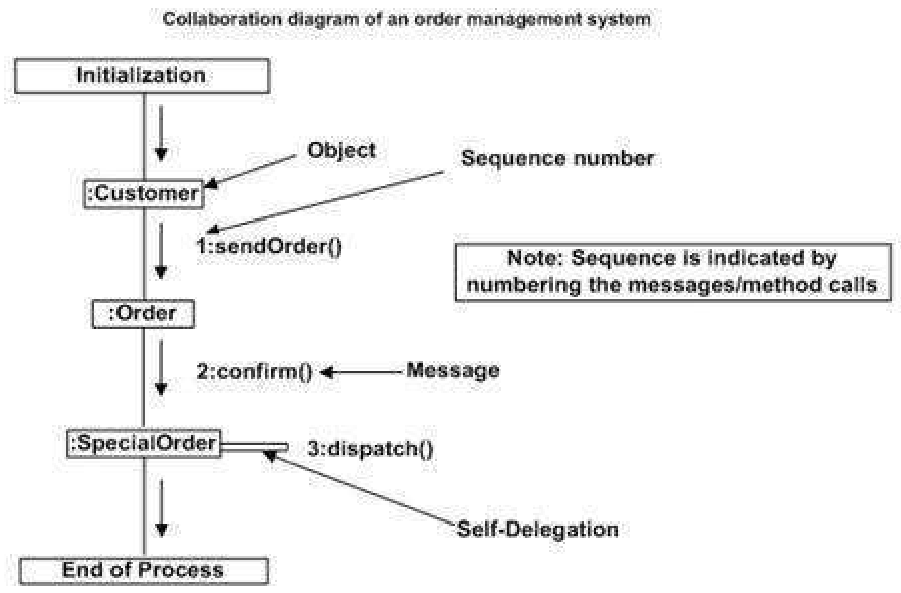
The sequence diagram has four objects (Customer, Order, SpecialOrder and NormalOrder).

The following diagram shows the message sequence for *SpecialOrder* object and the same can be used in case of *NormalOrder* object. It is important to understand the time sequence of message flows. The message flow is nothing but a method call of an object.

The first call is *sendOrder ()* which is a method of *Order object*. The next call is *confirm ()* which is a method of *SpecialOrder* object and the last call is *Dispatch ()* which is a method of *SpecialOrder* object. The following diagram mainly describes the method calls from one object to another, and this is also the actual scenario when the system is running.



B-The Collaboration Diagram



Interaction diagrams can be used −

- To model the flow of control by time sequence.
- To model the flow of control by structural organizations.
- For forward engineering.
- For reverse engineering.

6-Statechart Diagram

The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.

A Statechart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

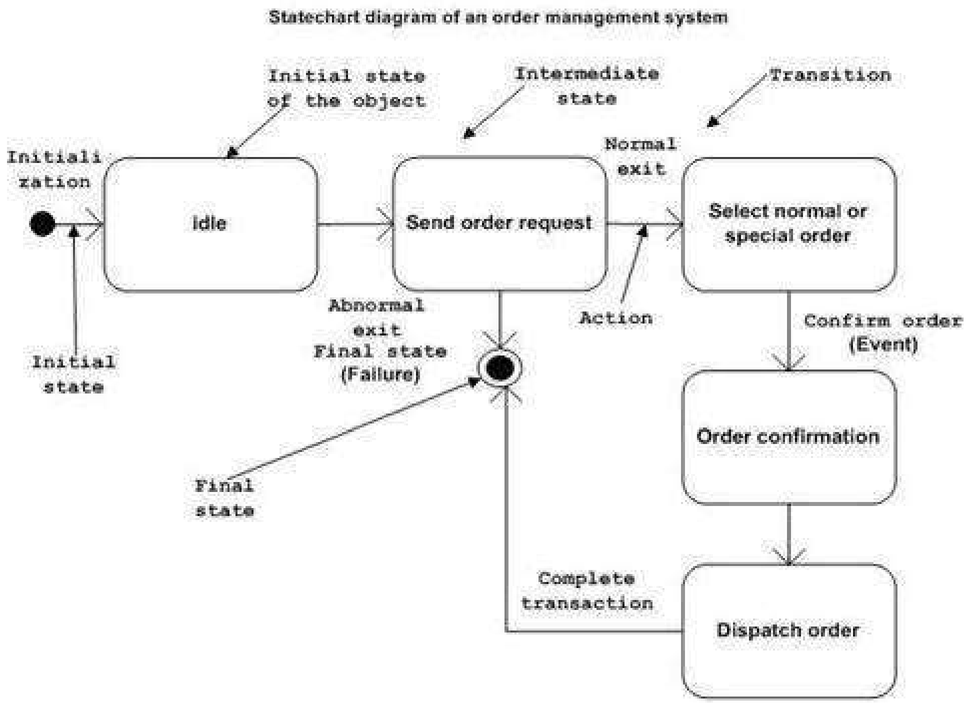
Following are the main purposes of using Statechart diagrams −

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object.

Following is an example of a Statechart diagram where the state of Order object is analyzed

The first state is an idle state from where the process starts. The next states are arrived for events like send request, confirm request, and dispatch order. These events are responsible for the state changes of order object.

During the life cycle of an object (here order object) it goes through the following states and there may be some abnormal exits. This abnormal exit may occur due to some problem in the system. When the entire life cycle is complete, it is considered as a complete transaction as shown in the following figure. The initial and final state of an object is also shown in the following figure.



The main usage can be described as –

- To model the object states of a system.
- To model the reactive system. Reactive system consists of reactive objects.
- To identify the events responsible for state changes.
- Forward and reverse engineering.

7-Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

Activity diagram is basically a flowchart 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. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

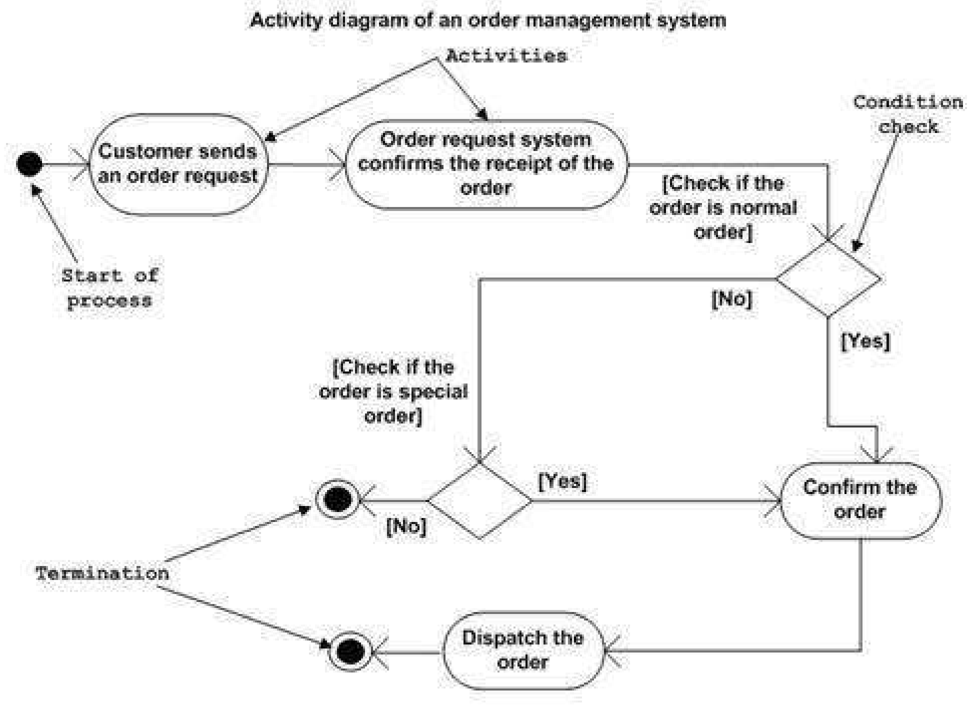
The purpose of an activity diagram can be described as –

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

Following is an example of an activity diagram for order management system. In the diagram, four activities are identified which are associated with conditions. One important point should be clearly understood that an activity diagram cannot be exactly matched with the code. The activity diagram is made to understand the flow of activities and is mainly used by the business users

Following diagram is drawn with the four main activities –

- Send order by the customer
- Receipt of the order
- Confirm the order
- Dispatch the order


Activity diagram of an order management system
Activities
Condition check
Customer sends an order request
Order request system confirms the receipt of the order
[Check if the order is normal order]
Start of process
[No]
[Yes]
[Check if the order is special order]
[Yes]
[No]
Confirm the order
Termination
Dispatch the order

Activity diagram can be used for –

- Modeling work flow by using activities.
- Modeling business requirements.
- High level understanding of the system's functionalities.
- Investigating business requirements at a later stage.