Expt. No. 1 (S.S.G.M.C.E.SHEGAON)

Date :

Aim :-

Theory :- Classes are the structural units in object oriented system design approach, so it is essential to know all relationship that exist between the classes in the system.

- Class diagram :- It is a graphical representation for describing a system in context of its static construction.
- Elements in class diagram :- Class diagram contains the system classes with its data member, operations & relationship between the classes.

- Class :- A set of objects containing similar data members and member functions is described by a class. In UML syntax, class is identified by solid outline rectangle with three compartments which contains

1) Class name

A Class is uniquely identified in a system by its name. A textual string is taken as class name. It lies in the first compartment in class rectangle

2) Attributes

Property shared by all instances of a class. It lies in the second compartment in class rectangle.

3) operations

An execution of an action can be performed for any object of a class. It lies in the last compartment in class rectangle.

• Generalization / Specialization :- It describes how one class is derived from another class. Derived class inherits the properties of its parent class.

• Relationships

1) Association :- It is an instance level relationship that allows exchanging message among the objects of both ends of association. A simple straight line connecting two class boxes represent an association. We can give a name to association and also at the both end we may indicate role names & multiplicity of adjacent classes. Association may be uni-directional.

2) Aggregation :- It is special form of association which describes a part whole relationship between a pair of classes. It means, in a relationship, when a class holds some instances of related class, then that relationship can be designed as an aggregation.

3) Composition :- It is strong form of aggregation which describes that whole is completely owns its part. Life cycle of that part depends on the whole.

• Conclusion :-

Expt. No. 2 Date :

Aim :-

Theory :-

- Sequence diagram

It represents the behavioral aspects of a system. Sequence diagram shows the interactions between the objects by means of passing messages from one object to another with respect to time in a system.

- Elements in sequence diagram

Sequence diagram contains the objects of a system and their life-line bar and the messages passing between them.

1) Objects - objects appear at the top portion of sequence diagram. object is shown in rectangle box. Name of object precedes a colon ':' & the class name, from which object is instantiated. The whole string is underlined & appears in a rectangle box. Also, we may use only class name or only instance name. Objects are created at the time of execution of use case & are involved in message passing, are appear in diagram, at the point of their creation.

2) Life-Line bar - A downward vertical line from object box is shown as the life line of object. A rectangle bar on life line indicates that it is active at that point of time.

3) Messages – Messages are shown as an arrow from the life line of sender object to the life line of receiver object & labeled with the message name. Chronological order of the messages passing throughout the objects life line show the sequence in which they occur. There are may exist some different types of messages.

a) Synchronous message : Receiver start Processing the message after receiving it and sender needs to wait until it is made. A straight arrow with close and fill arrow-head from sender life line bar to receiver end, represent a synchronous message.

b) Asynchronous message :- For asynchronous message sender needs not to wait for the receiver to Process the message. A function call that creates thread can be represented as an asynchronous message in sequence diagram. A straight arrow with open arrow head from sender life line bar to receiver end, represent an asynchronous message.

4) Return message :- For a function call when we need to return a value to the object from which it was called then we use return message. But, it is optional, and we are using it when we are going to model our system in much detail. A dashed arrow with open arrow head from sender life line bar to receiver end, represent that message.

d) Response message : one object can send a message to self. we use this message when we need to show the interaction between the same object.

Message type	Notation
Synchronous message	
Asynchronous message	
Response message	

Conclusion :-

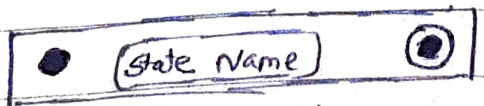
Aim :-

Theory :- In case of object oriented Analysis and design, a system is often abstracted by one or more classes with some well defined behaviour & states. A statechart diagram is a pictorial representation of such system, with all it's states and different events that lead transition from one state to another.

Building Blocks of statechart diagram

1) State :- A state is any 'distinct' stage that an object [system] passes through in it's lifetime. An object remains in a given state for finite time until 'something' happens, which make it to move to another state. All such states can be broadly categorized into following three types

- Initial : The state in which an object remain when created
- Final : The state from which an object do not move to any to other state (Optional)
- Intermediate : Any state, which is neither initial, nor final



An initial state is represented by circle filled with black. An intermediate state is depicted by rectangle with rounded corners. A final state is represented by a unfilled circle

with an inner black filled circle

Intermediate States usually have two compartments, separated by horizontal line. Called name compartment & internal transitions compartment. They are described below.

a) Name Compartment :- Contains the name of the state, which is a short, simple, descriptive string.

b) Internal transitions compartment :- Contains a list of internal activities performed as long as the system is in this state.

The internal activities are indicated using the following syntax action label / action-expression. Action label could be any condition indicator. There are four special action labels.

1) Entry : Indicates activity performed when the system enter this state

2) Exit :- Indicates activity performed when the system exit this state

3) DO :- Indicate any activity that is performed when the system remain in this state or until the action expression result in a completed computation

4) Include :- Indicates invocation of a sub machine

Transition :- Transition is movement from one state to another state in response to an external stimulus. It is represented by a solid arrow from current state to next state.

- Guidelines for drawing Statechart diagrams

1) for the system to developed, identify the distinct state that it passes through.
2) Identify the events that cause the state transitions. Often these would be the methods of a class as identified in a class diagram.
3) Identify what activities are performed while the system remains in a given state.

Expt. No. 5 Date :

Aim :-

Theory :-

Activity Diagram -

Activity diagram fall under the category of behavioural diagram in unified modeling language. It is high level diagram used to visually represent the flow of control in a system. It has similarities with traditional flow charts. However, it is more powerful than a simple flow chart since it can represent various other concepts like concurrent activity their joining and so on.

Activity diagram cannot depict the message passing among related objects. These kind of diagrams are suitable for confirming the logic to be implemented with the business users. These diagram are typically used when the business logic is complex.

Components of Activity Diagram

1) Activity :- An activity denotes a particular action taken in the logical flow of control. This could simply be invocation of a mathematical function, alter an object's properties and so on. There are two special type of activity nodes : initial and final.

Initial node represents the starting point of a flow in an activity diagram. There could be multiple initial nodes

which means that invoking that particular activity diagram would initiate multiple flows.
A final node represents the end point of all activities like an initial node, there could be multiple final nodes. Any transition reaching a final node would stop all activities.

2) Flow :- A flow is represented with a directed arrow. This is used to depict transfer of control from one activity to another, or to other types of components, as we will see below. A flow is often accompained with a label called gaurd condition, indicating the necessary condition for the transition to happen. The syntax to depict it is [guard condition].

3) Decision :- A decision node, represented with a diamond, is a point where a single flow enters and two or more flows leave. The control flow can follow only one of the outgoing paths. The outgoing edges often have guard conditions indicating true-false or if then-else conditions.

4) Merge :- This is represented with diamond shape, with two or more flows entering, and single flow leaving out. A merge node represents the points where at least a single control should reach before further processing could continue.

5) Fork :- Fork is a point where parallel activities begin. For example, a fork is graphically depicted with a black bar, with a single flow entering and multiple flows leaving out.

6) Join :- A Join is depicted with a black bar, with multiple input flows, but a single output flow. Physically it represents the synchronization of all concurrent activities. Unlike a merge, in case of join all of the incoming controls must be completed before any further Progress could be made

- Guidelines for drawing an Activity diagram

1) Identify tiny Pieces of work being Performed by the System
2) Identify the next logical activity that should be Performed
3) Think about all those conditions that should be made, and all those constraints that should be satisfied. before one can move to the next activity.
4) Put non-trivial guard conditions on the edges to avoid confusion.

Conclusion :-

Expt. No. 6 Date :

Aim :-

Theory :-

Use case diagrams

Use case diagrams belong to the category of behavioural diagram of UML diagram. Use case diagram aim to Present a graphical overview of the functionality Provided by the system. It consist of set of actions that the concerned system can perform, one or more actors & dependencies among them.

Actor :- An actor can be defined as an object or set of objects, external to the system, which interacts with the system to get some meaningful work done. Actor could be human, devices or even other systems.

Actor can be classified as :-

1) Primary actor :- They are Principal users of the system, who fulfil their goal by availing some service from the system.

2) Supporting actor :- They render some kind of service to the system.

In use case diagram primary actors are usually drawn on the top left side of the diagram.

Expt. No. (S.S.G.M.C.E.SHEGAON) Page No.

Date :

Use Case Relationship

Three types of relationship exist among use cases :

a) Include relationship b) Extend relationship

c) Use case generalization.

Guidelines for drawing use case diagram

1) determine the system boundary
2) Ensure that individual actors have well defined purpose
3) Use cases identified should let some meaningful work done by the actors.
4) Associate the actors & use cases - there shouldn't be any actor or use case floating without any connection.
5) use include relationship to encapsulate common behaviour among use cases, if any.

Conclusion :-