UNIT –II

UML DIAGRAMS

USE CASE DIAGRAM

\*A use case captures some user visible function

\* use case captures sequence of actions.

Actors: An actor is a person, organization or external system that plays a role in one or more interaction

Association: Association between actor and use cases are indicated In use case diagram by solid lines.



For e.g use case diagram for students enrolling in courses



CLASS DIAGRAM:

* It is a static diagram
* It represents static view of an application
* It is not only used for visualizing, describing and documenting different aspects of the system but used for constructing software application

Purpose:

* Analysis and designing the static view of an application
* Describes responsibility of a system
* It is a base for component and deployment diagrams
* Used in forward and reverse engineering

Points to remembered for drawing a class diagram

* The name of the class diagram should be meaningful
* Each elements and relationship should be identified in advance
* Responsibility of each class should be clearly identified (attributes and methods)
* For each class minimum number of properties should be specified, because unnecessary properties will make the diagram complicated
* Describe notes whenever necessary to describe some aspects.

e.g class diagram for ordering system



Classifier:

It is a special properties of attributes and operations and different kinds of classes

What is relationship?

\*classifier is a mechanism that describes structural and behavioural features

\* in general modelling elements that can have instances are called classifiers

\* class, instance, data type, signal, component, nodes, use case, subsystem are classifiers (packages are not classifier)

Special properties of attributes and operations

\*scope: The owner scope of a feature specifies whether the feature appears in each instance of the classifier or whether there is just a single instance of feature for all instances of the classifier.

\* instance- each instances holds its own value

\* classifier –just one value for all instance static.

Multiplicity:

* It is reasonable to assume that there may be any number of instances of classes.
* The number of instances a class may have is called multiplicity.

What is relationship ?

* It is a connection among things
* There are four most important relationships in object-oriented modelling:
* DEPENDENCIES
* GENERALIZATION
* ASSOCIATIONS
* REALIZATIONS

DEPENDENCY:

* Specifies a change in the specification of one thing may affect another thing but not necessarily the reverse
* It is rendered as a dashed line [----->]
* UML defines number of stereotype
* There are 8 stereotype that apply to dependency relationship among classes and objects in class diagram
* Two stereo types that apply to dependency relationships among packages.
* Access: source package is granted the right to reference the elements of the target package
* Import: a kind of access but only public content

Two stereo types that apply to dependency relationship among use case:

Extend: Target use case extends the behaviour of source

Include: Source use case explicitly incorporates the behaviour of another use case at a location specified by source.

8 stereotype of dependency are as follows:

1. Bind: specifies the source instantiates the target template using the given actual parameters. e.g relationship between a template container class and an instantiation of that class
2. Derive: specifies that the source may be computed from the target. E.g birthdate and age relationship between two attributes are two associations one which is concrete and the other is conceptual
3. Friend: specifies the source which is given special visibility into the target.
4. Instance of: specifies that the source object is an instance of the target classifier.
5. Instantiate: specifies that the source creates instances of the target. Instantiate is used when you want to specify which elements creates object of another element
6. Power type: it is a classifier whose objects are all the children of a given parent.
7. Refine: source is at a finer degree of abstraction than the target- it is used when the classes are essentially the same, but at different level of abstraction.
8. Use: specifies that the semantic part of the source element depends on the semantic of the public part of the target.

Associations:

* A structural relationship that indicates objects of one class have knowledge of objects of another class
* There are several advanced several adornments that can be used with association.

Navigation:

* Specifies which direction the association flows
* Unless they are indicated associations are bidirectional
* Graphically navigation is indicated as open arrow on one side of the association

Visibility:

* Restricts indirect accessibility of objects
* Identifies a key attribute of a class that can be used to restrict multiplicity from 0..\* to 0..1.

Interface specifier:

* An extension to a role name that shows the interface the class uses in the context of that role. The UML syntax is

RoleName:InterfaceSpecifier

Generalization (inheritance):

* A generalization is a relationship between general thing and more specific kind of thing .
* The specific thing(child) inherits all of the attribute and behaviours of the general thing (parent).
* The child may redefine or ignore any of the parents attributes and behaviour.
* UML supports both single inheritance and multiple inheritance
* There is one predefined stereotype (implementation) and 4 predefined constraints (complete, incomplete, disjoint and overlapping) that may be used with generalization

e.g of generalization



Class diagram for library management



Object diagram:

\*object diagrams are derived from class diagram

\*object diagrams are dependent on class diagram

Purpose:

The purpose of object diagram is similar to class diagram.

* It is more close to actual system behaviour.
* Its purpose is to capture state view of the system at a particular moment.
* First analyse the system and decide which instance are having important data and association.
* Second consider only those instances which will cover the functionality.
* Third make some optimization as the number of instances are unlimited

Before drawing an object diagram the following things should be remembered:

* The object diagram should have meaningful name to indicate its purpose
* The most important elements are to be identified.
* The association among objects should be clarified
* Values of different elements need to be captured in the object diagram.

For e.g

CUSTOMER

ORDER

SPECIAL ORDER

NORMAL ORDER

Now the customer object ( C ) is associated with the three order objects (O1, O2, O3) these order objects are associated with special order and normal order objects (S1, S2 and N1)

The customer is having the following three orders with different numbers (12,32,40) for the particular time considered. The same is for special order and normal order object which are having number of orders as (20,30,60) if a different time of purchase is considered these value will change accordingly.

**OBJECT DIAGRAM FOR ABOVE REQUIREMENTS**



Where to use object diagram ?

Object diagram can be imagined as a snapshot of a running system at a particular moment e.g. a running train. Now if you take a snap of the running train then you will find a static picture of it having the following:

* A particular state which is running
* A particular number of passengers which will change if the snap is taken at different time.

Interaction Diagram

* It is used to describe some type of interactions among the different elements in the model so this interaction is a part of dynamic behaviour of the system.
* The interactive behaviour is represented in UML by two diagrams known as SEQUENCE diagram and COLLABORATION diagram
* Sequence diagram emphasizes on the time sequence of messages. Collaboration diagram emphasizes on the structural organization of the object that send and receive messages.

Sequence Diagram:

* Sequence diagram is a way of describing the behaviour of a system
* It shows an interaction between the system and environment
* It consists of two dimensions vertical and horizontal
* Vertical dimension represents time and horizontal dimension represents different objects.

Vertical line is considered as an object life line . The life line represents the objects existence during the interation.

* Each message is represented by an arrow between the life lines of two objects.
* The messages appears from top to bottom
* Each message is labelled by message name
* It is an alternative way to understand the overall flow of the control of a program.
* Sequence diagram helps to quickly understand the sequence.

Purpose of sequence diagram:

* It is used to show interaction between the objects in the sequential order.
* It is useful for the organizations business staff to communicate how the business currently works by showing various business objects interaction.
* It also acts as a requirement document to communicate for a future system implementation.

Sequence diagram for order system

Sequence diagram for library system



Two examples:

1. Retrieve Borrower Credit Report(ssn) : borrowerCreditReport

or

2. Process Credit Card(name, number, expirationDate, amount : 100)

In example 1, the syntax calls the sequence diagram called Retrieve Borrower Credit Report and passes it the parameter ssn. The Retreive Borrower Credit Report sequence returns the variable borrowerCreditReport.

In example 2, the syntax calls the sequence diagram called Process Credit Card and passes it the parameters of name, number, expiration date, and amount. However, in example 2 the amount parameter will be a value of 100. And since example 2 does not have a return value labeled, the sequence does not return a value (presumably, the sequence being modeled does not need the return value).

Figure 1: A sequence diagram that references two different sequence diagrams

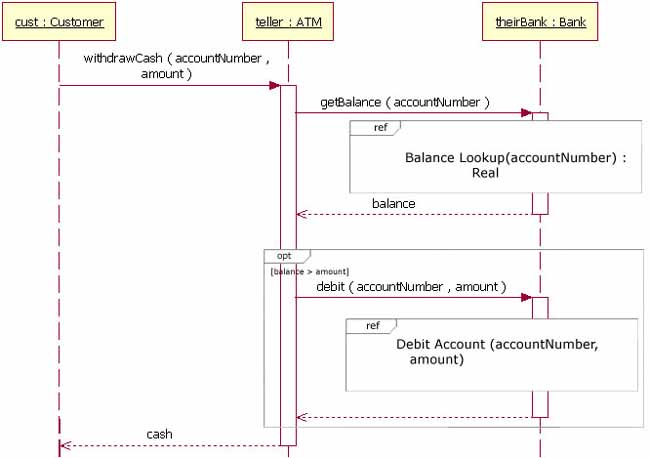


Figure 1 shows a sequence diagram that references the sequence diagrams "Balance Lookup" and "Debit Account." The sequence starts at the top left, with the customer sending a message to the teller object. The teller object sends a message to the their Bank object. At that point, the Balance Lookup sequence diagram is called, with the account Number passed as a parameter. The Balance Lookup sequence diagram returns the balance variable. Then the option combination fragment's guard condition is checked to verify the balance is greater than the amount variable. In cases where the balance is greater than the amount, the Debit Account sequence diagram is called, passing it to the account Number and the amount as parameters. After that sequence is complete, the withdrawCash message returns cash to the customer.

It is important to notice in Figure 11 that the lifeline of their Bank is hidden by the interaction occurrence Balance Lookup. Because the interaction occurrence hides the life line, that means that the their Bank lifeline is referenced in the "Balance Lookup" sequence diagram. In addition to hiding the lifeline in the interaction occurrence, UML 2 also specifies that the lifeline must have the same their Bank in its own "Balance Lookup" sequence.

There will be times when you model sequence diagrams that an interaction occurrence will overlap lifelines that are *not* referenced in the interaction occurrence. In such cases the lifeline is shown as a normal lifeline and is not hidden by the overlapping interaction occurrence.

In Figure 1, the sequence references the "Balance Lookup" sequence diagram. The "Balance Lookup" sequence diagram is shown in Figure 2. Because the example sequence has parameters and a return value, its label —located in the diagram's name box—follows a specific pattern:

Diagram Type Diagram Name [(Parameter Type : Parameter Name)] :

[: Return Value Type]

Collaboration diagram

\*it is an interaction diagram

\*it is a set of object s related in a particular context and interaction.

\*in a collaboration diagram objects are shown as figures

\* as in sequence diagram arrows indicate the message

\* in collaboration diagram the sequence is indicated by numbering the messages.

\*collaboration involves several numbering scheme, you can also use decimal numbering scheme.



Collaboration diagram for order system



Advantage of interaction diagram:--

1.simplicity

2. you can easily see the message by looking at the diagram

Disadvantage

1. They are great only for representing a single sequential process

State chart diagram

\*state chart diagram are used to draw the dynamic nature of the system

\* it is defined as an condition from which the object exists

\* state chart diagram is to model the life time of an object from creation to termination

\*they are used for forward and reverse engineering

Purpose of state chart diagram

* To model dynamic aspect of the system
* To model life time of reactive system
* To describe different states of an object during its life time
* define a state machine to design states of an object



In state chart diagram the initial state is shown as a small dot, final state is shown as a circle surrounding a small dot , a bull’s eye this represents completion of activity.

Activity diagram

\*Activity diagram describe dynamic aspects of the system.

\* Activity diagram is basically a flow chart to represent the flow form 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 deals with all type of flow control by using different elements like fork, join etc.

Purposes of Activity diagram:

* 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.



Activity diagram



Component diagram

\*component diagrams model the physical components such as source code, executeable program, user interface.

\* A component diagram is a graph of the designs components connected by dependency relationship

\* component diagram offer semantically rich grouping mechanism



Deployment diagram

* It contains nodes and connections
* A node represents a piece of hardware in the system
* A connection depicts the communication path used by the hardware to communicate.
* Usually indicates the method such as TCCP/IP.
* Deployment diagram contains artefact
* An artefact is the specification of a physical piece of information e.g source files, binary files, table in database

\*Deployment diagrams are mainly used by system engineers.

\*These diagrams are used to describe the physical components (hardwares), their distribution and association.

\*Software applications are developed to model complex business processes.

So the usage of deployment diagrams can be described as follows:

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

