**Module-5**

**UML**

* 1. **UML**
* UML means Unified Modeling Language
* UML is a modeling language for visualizing, specifying, constructing and documenting a software system and its components.
* UML is a graphical language with set of rules and semantics
* The goal of UML is to provide a standard notation that can be used by all object oriented methods
* Modeling means constructing an abstraction of a system

**5.1.1 Visualizing**

- UML includes both graphical and textual representation

- Notations of UML are visual and well defined

**5.1.2 Specifying**

- UML helps to build models that are precise, unambiguous and complete

**5.1.3 Constructing**

- UML is not a visual programming language

- Models can be directly connected to a variety of programming language

- It is possible to map from model in UML to

* Object oriented languages such as Java or C++
* Tables in relational database

**5.1.4 Documenting**

- UML addresses the documentation of systems architecture and all of its details like requirements, architecture, design, source code, project plan and tests

**5.2 UML Primary Goals**

- The primary goals in the design of UML are:

* Provide users a ready to use, expressive visual modeling language so that they can develop and exchange meaningful models
* Provide extensibility and specialization mechanisms to extend core concepts
* Provide required formal basis for understanding the modeling language
* Encourage the growth of OO tools
* Supports development concepts
  1. **UML Architecture**
* Architecture is used to manage different viewpoints and control iterative and incremental development of systems throughout the cycle
* The five views are:
* **Usecase view**: Focus on behavior
* **Design view**: Focus on vocabulary and functionality
* **Implementation view**: Focus on system assembly and configuration management
* **Process view**: Focus on performance, scalability and throughput
* **Deployment view**: Focus on system topology, distribution, delivery and installation

Vocabulary,functionality

System assembly, configuration management

Design view

Implementation view

Behavior

Process view

Deployment view

System topology, distribution, delivery and installation

Performance, scalability, throughput

**Fig 5.1 Modeling a System’s Architecture**

**5.4 Primary Model Elements of UML**

**Class:** Template for a set of objects that share a common structure and common behavior

**Usecase:** Named behavior

**State:** The condition of an object

**Interface:** The public part of an object

**Active class:** Capable of concurrent activities with other active classes

**Component:** Reusable element typically having both logical as well as physical aspects

**Node:** Hardware device upon which software may reside and execute

**Package:** Container of elements

**Note:** Comment, explanation or annotation

**5.5 Building blocks of UML**

- The building blocks are:

* Things
* Relationships
* Diagrams

**5.5.1 Things**

- There are four kinds of things in UML

**5.5.1.1 Structural Things**

- They are nouns and static part of model

- Eg: class, interface, collaboration, usecase, active class, component and node

**5.5.1.2 Behavioral Things**

- They are verbs and dynamic parts of UML representing behavior over time and space

- Eg: Interaction, state machine

**5.5.1.3 Grouping Things**

- They are organizational part of UML

- Eg: Package, subsystems

**5.5.1.4 Annotational Things**

- Explanatory part of UML

- These are comments applied to describe and remark about any element in a model

**5.5.2 Relationships**

- Relationship is a connection among elements

**5.5.2.1 Association**

- Association is connection between classes

**5.5.2.1.1 Unidirectional association**

Has\_capital

City

Country

**5.5.2.1.2 Binary Association**

Works\_for

employs

Company

Person

**5.5.2.1.3 Qualified Association**

- Qualifiers are used to reduce multiplicity

filename

File

Directory

**5.5.2.1.4 Multiplicity**

- Multiplicity indicates how many instances of one class may relate to single instance of associated class

- Eg: one to one, many to many and zero or one

Has\_capital

(city)

Delhi

(country)

India

**Fig: One to one multiplicity**

intersects

Point

Line

**Fig: Many to many multiplicity**

console

Window

Workstation

**Fig: Zero or one multiplicity**

**5.5.2.1.5 Ternary Association**

- Association among three classes

Project

Language

Person

**Fig: Ternary association**

**5.5.2.2 Aggregation**

- It is a form of association

- It describes part of or part whole relationship

- It is represented as

- Aggregation has two properties:

* Transitivity: A -> B

B -> C

A -> C

* Antisymmetric- A = B

B = A

Document

Paragraph

Sentence

**Fig: Aggregation**

**5.5.2.3 Generalization**

- Generalization is the relationship between class and one or more refined versions of it

- Class being refined is called superclass and refined versions are called subclass

Lamp

Incandescent lamp

Fluroscent lamp

**Fig: Generalization**

**5.5.2.4 Dependency**

- It represents using relationship

depends on

Server

Client

**5.5.2.5 Realization**

- Realization is used to specify the relationship between an interface and the class or component that provides an operation or service for it

**5.5.3 UML Diagrams**

- In UML notation there are 13 standard diagrams

- Six static/structural diagrams

* Class diagram
* Object diagram
* Component diagram
* Composite structure diagram
* Package diagram
* Deployment diagram

- Seven dynamic/behavioral diagrams

* Use-case diagram
* State machine diagram
* Activity diagram
* Interaction diagram
* Sequence diagram
* Communication/collaboration diagram
* Timing diagram
* Interaction overview diagram

**Static/structural diagrams**

**5.5.3.1 Class diagram**

- Class diagram describe the structure of systems in terms of classes and objects.

- Class is a collection of objects that share a set of attributes and operations

- Attributes are data held by objects

- Operations are transformations applied to objects

class name

attributes

operations

copy of

update

utilizes

Item

**Fig: Class diagram for Library management system**

**5.5.3.2 Object Diagram**

- Object diagram is an instance of class diagram

- Captures objects and links

- Object diagram describe static structure of system at a particular time

object Name:class

:class

object Name:class

**Named object Unnamed object Object with attributes**

Attribute type=

”value”

Mary:Member

:Member

Mary:Member

Name=”age”

**Named object Unnamed object Object with attributes**

**5.5.3.3 Component Diagram**

- Component diagram shows the organization and dependencies among set of components built as a part of architectural specification

- Component diagram represents the static implementation view of system

**Fig: Component Diagram for Library Management System**

**5.5.3.4 Composite Structure Diagram**

- Composite structure diagram shows the internal structure of a classifier

- Composite structure diagram describes the way the classes can be displayed the way the classes can be displayed as composite elements

**5.5.3.5 Package Diagram**

- Package diagram is used at lower level to structure the system according to service it provides

- Package diagram is useful to organize use case diagram and class diagram

**5.4.3.6 Deployment Diagram**

- Deployment Diagram specify the distribution of components

- It is the configuration of run time processing nodes

**Fig: Deployment Diagram for Library Management System**

**Dynamic/ Behavioral Diagrams**

**5.4.3.7 Usecase Diagram**

- Usecase diagram is a graph of actors, a set of usecase enclosed by a system boundary, communication associations between the actors and usecase and generalization among the usecase

- Usecase diagram defines the outside and inside of the system behavior

- Usecase diagram captures system functionality

- Usecase diagram captures the requirement of a system

- Usecase is shown as an ellipse containing the name of usecase

- Actors is represented as

**Fig: Usecase Diagram for Library Management System**

**5.4.3.8 State machine diagram (state chart/ transition diagram)**

- State chart diagram models the behavior of single object

- State chart diagram specifies the sequence of events

- It is developed by the designer

- State is represented as

* Initial state is represented as
* Final state is represented

-Final state is also called bulls eye

**Fig: State Chart Diagram for Reservation of Library Management System**

**5.4.3.9 Activity Diagram**

- Activity diagram illustrates the dynamic nature of system

- Activity diagram is used to model internal operations

**Fig: Activity Diagram for Reservation of Library Management System**

**5.4.3.10 Sequence Diagram**

- Sequence diagram describes the interaction between objects

- It represents the object participating in the interaction horizontally and time vertically

**Fig: Sequence Diagram for Library Management System**

**5.4.3.11 Collaboration diagram**

- It represents the sequence of messages by numbering the interactions

- It helps to analyze the complex procedural logic

**Fig: Collaboration Diagram for Library Management System**

**5.4.3.12 Timing Diagram**

- Timing diagram is used to display the change of state or value of one or more elements over time

**5.4.3.13 Interaction Overview Diagram**

- It is a form of activity diagram in which node represents interaction diagram

- It has initial state and final state

- Initial state is represented as

**-** Final state is represented as