

Click to prove
you're human



Labeled Modeling Language (UML) is a general-purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering. UML is not a programming language, it is rather a visual language. 1. What is UML? Unified Modeling Language (UML) is a standardized visual modeling language that is versatile, flexible, and user-friendly method for visualizing a system's design. Software system artifacts can be specified, visualized, built, and documented with the use of UML. We use UML diagrams to show the behavior and structure of a system. UML helps software engineers, businessmen, and system architects with modeling, design, and analysis. The International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically. 2. Why do we need UML? We need UML (Unified Modeling Language) to visually represent and communicate complex system designs, facilitating better understanding and collaboration among stakeholders. Below is why we need UML:Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them. Businessmen do not understand code. So UML becomes essential to communicate with non-programmers about essential requirements, functionalities, and processes of the system. A lot of time is saved down the line when teams can visualize processes, user interactions, and the static structure of the system. 3. Types of UML Diagrams UML is linked with object-oriented design and analysis. UML makes use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as follows. Structural UML Diagrams Structural UML diagrams are visual representations that depict the static aspects of a system, including its classes, objects, components, and their relationships, providing a clear view of the system's architecture. Structural UML diagrams include the following types:4.1. Class Diagram The most widely used UML diagram is the class diagram. It is the building block of all object oriented software systems. We use class diagrams to depict the static structure of a system by showing system's classes, their methods and attributes. Class diagrams also help us identify relationship between different classes or objects. Class Diagram4.2. Composite Structure Diagram We use composite structure diagrams to represent the internal structure of a class and its interaction points with other parts of the system. A composite structure diagram represents relationship between parts and their configuration which determine how the classifier (class, a component, or a deployment node) behaves. They represent internal structure of a structured classifier making the use of parts, ports, and connectors. We can also model collaborations using composite structure diagrams. They are similar to class diagrams except they represent individual parts in detail as compared to the entire class.4.3. Object Diagram An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object diagrams depict behaviour when objects have been instantiated, we are able to study the behaviour of the system at a particular instant. An object diagram is similar to a class diagram except it shows the instances of classes in the system. We depict actual classifiers and their relationships making the use of class diagrams. On the other hand, an Object Diagram represents specific instances of classes and relationships between them at a point of time. Object Diagram4.4. Component Diagram Component Diagrams are used to represent how components interact in a system. Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development. Component Diagrams become essential to use when we design and build complex systems. Interfaces are used by components of the system to communicate with each other. Component Diagram4.5. Deployment Diagram Deployment Diagrams are used to represent system hardware and its software. It tells us what hardware components exist and what software components run on them. We illustrate system architecture as distribution of software artifacts over distributed targets. An artifact is the information that is generated by system software. They are primarily used when a software is being used, distributed or deployed over multiple machines with different configurations. Deployment Diagram4.6. Package Diagram We use Package Diagrams to depict how packages and their elements have been organized. A package diagram simply shows us the dependencies between different packages and internal composition of packages. Packages help us to organise UML diagrams into meaningful groups and make the diagram easy to understand. They are primarily used to organise class and use case diagrams. Package Diagram5. Behavioral UML Diagrams Behavioral UML diagrams are visual representations that depict the dynamic aspects of a system, illustrating how objects interact and behave over time in response to events.5.1. State Machine Diagrams A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. State Machine Diagram5.2. Activity Diagrams We use Activity Diagrams to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram. Activity Diagram5.3. Use Case Diagrams Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents(actors). A use case is basically a diagram representing different scenarios where the system can be used. A use case diagram gives us a high level view of what the system or a part of the system does without going into implementation details. '5.4. Sequence Diagram A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems. Sequence Diagram5.5. Communication Diagram A Communication Diagram (known as Collaboration Diagram in UML 1.x) is used to show sequenced messages exchanged between objects. A communication diagram focuses primarily on objects and their relationships. We can represent similar information using Sequence diagrams, however communication diagrams represent objects and links in a free form. Communication Diagram5.6. Timing Diagram Timing Diagrams are a special form of Sequence diagrams which are used to depict the behavior of objects over a time frame. We use them to show time and duration constraints in a diagram. They are similar to sequence diagrams except they represent individual parts in detail as compared to the entire class.4.3. Object Diagram An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object diagrams depict behaviour when objects have been instantiated, we are able to study the behaviour of the system at a particular instant. An object diagram is similar to a class diagram except it shows the instances of classes in the system. We depict actual classifiers and their relationships making the use of class diagrams. On the other hand, an Object Diagram represents specific instances of classes and relationships between them at a point of time. Object Diagram4.4. Component Diagram Component Diagrams are used to represent how components interact in a system. Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development. Component Diagrams become essential to use when we design and build complex systems. Interfaces are used by components of the system to communicate with each other. Component Diagram4.5. Deployment Diagram Deployment Diagrams are used to represent system hardware and its software. It tells us what hardware components exist and what software components run on them. We illustrate system architecture as distribution of software artifacts over distributed targets. An artifact is the information that is generated by system software. They are primarily used when a software is being used, distributed or deployed over multiple machines with different configurations. Deployment Diagram4.6. Package Diagram We use Package Diagrams to depict how packages and their elements have been organized. A package diagram simply shows us the dependencies between different packages and internal composition of packages. Packages help us to organise UML diagrams into meaningful groups and make the diagram easy to understand. They are primarily used to organise class and use case diagrams. Package Diagram5. Behavioral UML Diagrams Behavioral UML diagrams are visual representations that depict the dynamic aspects of a system, illustrating how objects interact and behave over time in response to events.5.1. State Machine Diagrams A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. State Machine Diagram5.2. Activity Diagrams We use Activity Diagrams to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram. Activity Diagram5.3. Use Case Diagrams Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents(actors). A use case is basically a diagram representing different scenarios where the system can be used. A use case diagram gives us a high level view of what the system or a part of the system does without going into implementation details. '5.4. Sequence Diagram A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems. Sequence Diagram5.5. Communication Diagram A Communication Diagram (known as Collaboration Diagram in UML 1.x) is used to show sequenced messages exchanged between objects. A communication diagram focuses primarily on objects and their relationships. We can represent similar information using Sequence diagrams, however communication diagrams represent objects and links in a free form. Communication Diagram5.6. Timing Diagram Timing Diagrams are a special form of Sequence diagrams which are used to depict the behavior of objects over a time frame. We use them to show time and duration constraints in a diagram. They are similar to sequence diagrams except they represent individual parts in detail as compared to the entire class.4.3. Object Diagram An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object diagrams depict behaviour when objects have been instantiated, we are able to study the behaviour of the system at a particular instant. An object diagram is similar to a class diagram except it shows the instances of classes in the system. We depict actual classifiers and their relationships making the use of class diagrams. On the other hand, an Object Diagram represents specific instances of classes and relationships between them at a point of time. Object Diagram4.4. Component Diagram Component Diagrams are used to represent how components interact in a system. Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development. Component Diagrams become essential to use when we design and build complex systems. Interfaces are used by components of the system to communicate with each other. Component Diagram4.5. Deployment Diagram Deployment Diagrams are used to represent system hardware and its software. It tells us what hardware components exist and what software components run on them. We illustrate system architecture as distribution of software artifacts over distributed targets. An artifact is the information that is generated by system software. They are primarily used when a software is being used, distributed or deployed over multiple machines with different configurations. Deployment Diagram4.6. Package Diagram We use Package Diagrams to depict how packages and their elements have been organized. A package diagram simply shows us the dependencies between different packages and internal composition of packages. Packages help us to organise UML diagrams into meaningful groups and make the diagram easy to understand. They are primarily used to organise class and use case diagrams. Package Diagram5. Behavioral UML Diagrams Behavioral UML diagrams are visual representations that depict the dynamic aspects of a system, illustrating how objects interact and behave over time in response to events.5.1. State Machine Diagrams A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. State Machine Diagram5.2. Activity Diagrams We use Activity Diagrams to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram. Activity Diagram5.3. Use Case Diagrams Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents(actors). A use case is basically a diagram representing different scenarios where the system can be used. A use case diagram gives us a high level view of what the system or a part of the system does without going into implementation details. '5.4. Sequence Diagram A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems. Sequence Diagram5.5. Communication Diagram A Communication Diagram (known as Collaboration Diagram in UML 1.x) is used to show sequenced messages exchanged between objects. A communication diagram focuses primarily on objects and their relationships. We can represent similar information using Sequence diagrams, however communication diagrams represent objects and links in a free form. Communication Diagram5.6. Timing Diagram Timing Diagrams are a special form of Sequence diagrams which are used to depict the behavior of objects over a time frame. We use them to show time and duration constraints in a diagram. They are similar to sequence diagrams except they represent individual parts in detail as compared to the entire class.4.3. Object Diagram An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object diagrams depict behaviour when objects have been instantiated, we are able to study the behaviour of the system at a particular instant. An object diagram is similar to a class diagram except it shows the instances of classes in the system. We depict actual classifiers and their relationships making the use of class diagrams. On the other hand, an Object Diagram represents specific instances of classes and relationships between them at a point of time. Object Diagram4.4. Component Diagram Component Diagrams are used to represent how components interact in a system. Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development. Component Diagrams become essential to use when we design and build complex systems. Interfaces are used by components of the system to communicate with each other. Component Diagram4.5. Deployment Diagram Deployment Diagrams are used to represent system hardware and its software. It tells us what hardware components exist and what software components run on them. We illustrate system architecture as distribution of software artifacts over distributed targets. An artifact is the information that is generated by system software. They are primarily used when a software is being used, distributed or deployed over multiple machines with different configurations. Deployment Diagram4.6. Package Diagram We use Package Diagrams to depict how packages and their elements have been organized. A package diagram simply shows us the dependencies between different packages and internal composition of packages. Packages help us to organise UML diagrams into meaningful groups and make the diagram easy to understand. They are primarily used to organise class and use case diagrams. Package Diagram5. Behavioral UML Diagrams Behavioral UML diagrams are visual representations that depict the dynamic aspects of a system, illustrating how objects interact and behave over time in response to events.5.1. State Machine Diagrams A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. State Machine Diagram5.2. Activity Diagrams We use Activity Diagrams to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram. Activity Diagram5.3. Use Case Diagrams Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents(actors). A use case is basically a diagram representing different scenarios where the system can be used. A use case diagram gives us a high level view of what the system or a part of the system does without going into implementation details. '5.4. Sequence Diagram A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems. Sequence Diagram5.5. Communication Diagram A Communication Diagram (known as Collaboration Diagram in UML 1.x) is used to show sequenced messages exchanged between objects. A communication diagram focuses primarily on objects and their relationships. We can represent similar information using Sequence diagrams, however communication diagrams represent objects and links in a free form. Communication Diagram5.6. Timing Diagram Timing Diagrams are a special form of Sequence diagrams which are used to depict the behavior of objects over a time

reverse engineering yields round-trip engineering, meaning the ability to work with in either a graphical or a textual view, while the tools keep the two views consistent. UML is a Language for Documenting A healthy software organization produces all sorts of artifacts in addition to raw executable code. These artifacts include: Requirements Architecture Design Source Code Project Plans Tests Prototypes Releases The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management. Where can UML be used? (Application domains of UML) UML can be used in domains such as: Enterprise information systems Banking and Financial services Telecommunications Transportation Defense Retail Medical electronics Scientific Distributed web-based services UML is not limited to modeling software. In fact, it is expressive enough to model non-software systems, such as workflow in the legal system, the structure and behavior of a patient healthcare system and the design of hardware. Suryateja Pericherla, at present is a Research Scholar (full-time Ph.D.) in the Dept. of Computer Science & Systems Engineering at Andhra University, Visakhapatnam. Previously worked as an Associate Professor in the Dept. of CSE at Vishnu Institute of Technology, India. He has 11+ years of teaching experience and is an individual researcher whose research interests are Cloud Computing, Internet of Things, Computer Security, Network Security and Blockchain. He is a member of professional societies like IEEE, ACM, CSI and ISCA. He published several research papers which are indexed by SCIE, WoS, Scopus, Springer and others.