### **ON 22/11/2024,**

FRIDAY GROUP B.

GROUP 1 MEMBERS:

• ITEKA Laure Gabriella 22718

• IRADUKUNDA Sylvie 25963

• AKALIZA M. Racia 26628

• UMUGANWA Lina 26644

• KIZUNGU Shyaka Noah Loic 25919

• HAHIRWA Raissa 26586

• ITULA Elie 24857

**SOFTWARE ENGINEERING**

**Assignment 2: UML DAIGRAMS**

### **UML Activity Diagram**

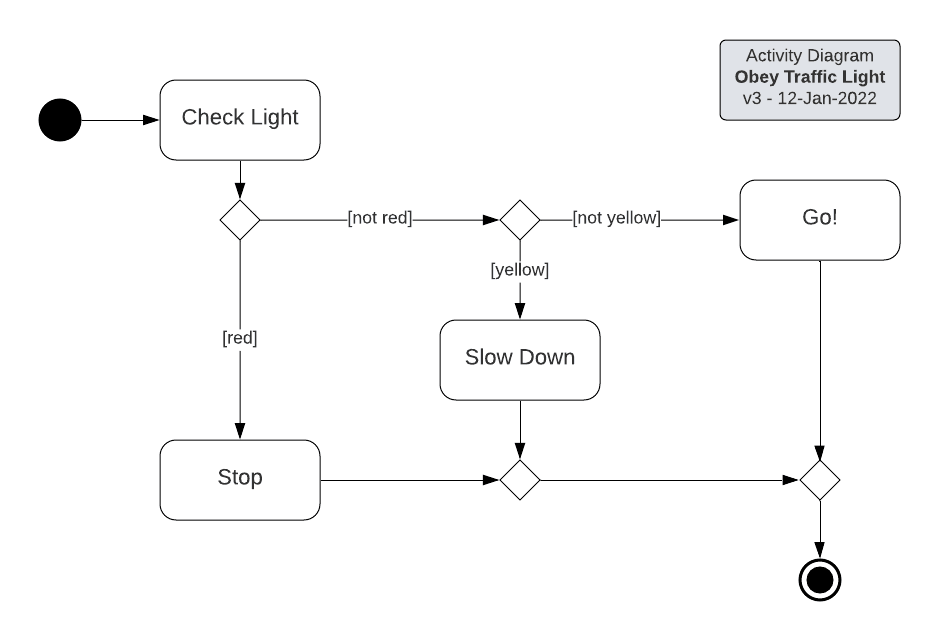
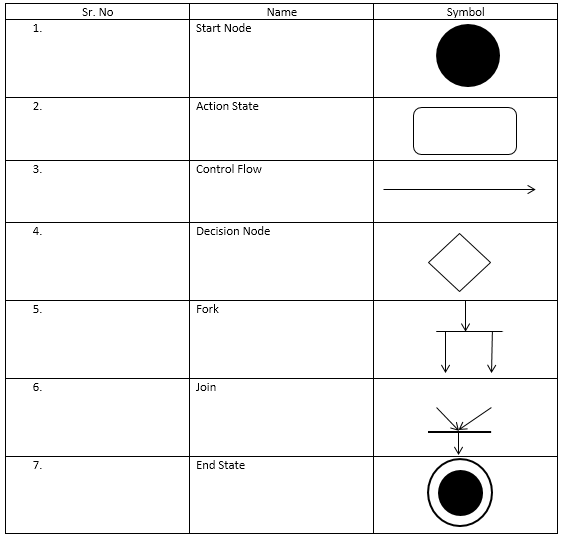
#### **Overview**

A **UML Activity Diagram** is a flowchart-like diagram used to model the dynamic aspects of a system. It represents workflows of stepwise activities and actions, showing the flow of control or data from one activity to another. It is especially useful in describing business processes or software logic.

A type of flowchart that shows the flow of control between actions in a system

#### **Symbols Used**

1. **Rounded Rectangles**: Represent activities or actions.
2. **Arrows**: Show the flow of control or data between activities.
3. **Diamond**: Represents a decision point with alternative flows.
4. **Bars (Horizontal or Vertical)**: Represent forks (splitting into parallel activities) or joins (merging parallel activities).
5. **Black Circle**: Denotes the start of the process (Initial Node).
6. **Black Circle with a Border**: Indicates the end of the process (Final Node).
7. **Swimlanes**: Divide the diagram to show responsibilities of different actors or systems.

****

#### **Advantages and Disadvantages**

**Advantages**:

1. **Clear Representation**: Provides a high-level view of a process or workflow.
2. **Communication Tool**: Helps stakeholders understand system behavior.
3. **Problem Identification**: Reveals bottlenecks or inefficiencies in workflows.
4. **Versatile**: Can model complex workflows and parallel processes.

**Disadvantages**:

1. **Complexity**: For large systems, the diagram can become cluttered and difficult to read.
2. **Overhead**: Creating and maintaining diagrams can be time-consuming.
3. **Misinterpretation**: Stakeholders unfamiliar with UML may misinterpret the symbols.

#### **Example Case Studies**

1. **Order Processing System**:  
   An e-commerce platform models the workflow of an order being placed, verified, shipped, and delivered. Swimlanes can represent different departments (e.g., Sales, Inventory, Shipping).
2. **Library Management System**:  
   Depicts activities like searching for a book, borrowing it, and returning it, with decision points to handle conditions like availability or overdue books.

### **UML Timing Diagram**

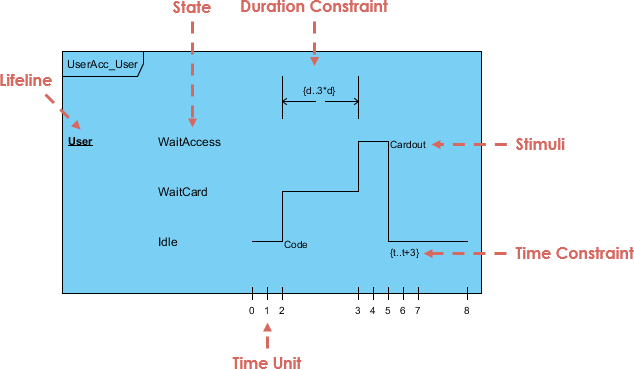
#### **Overview**

A **UML Timing Diagram** is a type of interaction diagram that shows the behavior of objects over time. It is primarily used to understand how objects interact within a specific time constraint and is valuable in real-time systems or embedded systems design.

#### **Symbols Used**

1. **Lifeline**: Represents an object's existence over time.
2. **State Timeline**: Shows the state of an object at different times.
3. **Duration Constraints**: Represent interactions or signals exchanged between lifelines.
4. **Time Ruler**: A horizontal line indicating time progression.

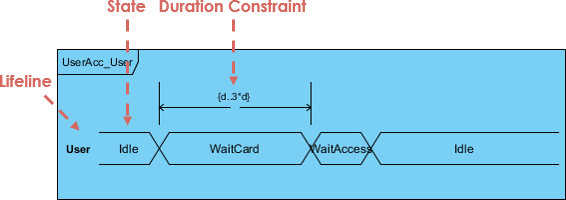
.State Timeline Representation



### **Explanation of Labeled Components**

1. **Lifeline**:
   * The vertical line labeled "User" represents the system element (UserAcc\_User) being modeled over time.
   * It tracks the user's states (e.g., "WaitAccess," "WaitCard," "Idle").
2. **State**:
   * The various labeled regions ("WaitAccess," "WaitCard," "Idle") represent different states of the user as time progresses.
   * Transitions occur between these states based on actions or stimuli.
3. **Duration Constraint**:
   * Shown as a horizontal bracket labeled "{d, 3\*d}."
   * Indicates the permissible time interval for the transition between states (e.g., the time spent in "WaitAccess" before moving to "WaitCard").
4. **Stimuli**:
   * Labels like "Cardout" and "Code" represent events or actions that trigger state transitions in the system.
   * For instance, "Code" might indicate an action such as entering a PIN, leading to a state change.
5. **Time Constraint**:
   * Shown as "{t, t+3}" to indicate the expected range of time for an action or process.
   * Ensures that specific actions (like "Cardout") occur within a defined temporal limit.
6. **Time Unit**:
   * The horizontal timeline at the bottom indicates time units (e.g., seconds or milliseconds) over which the events and state changes occur.
   * The timeline ensures precise tracking of when each state transition happens.

The figure below shows an alternative notation of UML Timing diagram. It shows the state of the object between two horizontal lines that cross with each other each time the state changes



#### **Advantages and Disadvantages**

**Advantages**:

1. **Time Analysis**: Focuses on time-based behavior, useful in real-time systems.
2. **Clear Visualization**: Provides a concise representation of temporal dependencies.
3. **Debugging Aid**: Helps identify timing mismatches or synchronization issues.

**Disadvantages**:

1. **Limited Application**: Not suitable for modeling general workflows or high-level processes.
2. **Complexity**: Requires precise timing information, making it hard to use early in development.
3. **Scalability Issues**: Becomes unwieldy with multiple lifelines and states.

#### **Example Case Studies**

1. **Traffic Light System**:  
   Models the states (e.g., green, yellow, red) of traffic lights over time and how they transition.
2. **Multithreading in Software**:  
   Visualizes threads' states (e.g., running, waiting, terminated) and their interactions during a specific timeframe