1. Goals of UML in Software Design

- Visualization: UML helps in visualizing software systems, whether they are being newly created or are existing systems that need to be understood or updated. This visualization supports the exploration and explanation of system structure and behavior without delving into the code.
- Specification: It acts as a blueprint or a detailed plan that shows the various elements of a system and their relationships. This can be used as a guide for constructing the actual system.
- Documentation: UML provides a way to document the decisions and structure of systems which is crucial for maintenance and future development. Good documentation helps new team members understand old systems and supports maintenance teams in managing the system.
- Design: By using various types of diagrams, UML aids in the conceptual design of software components and their interfaces, as well as the relationships and interactions between those components.

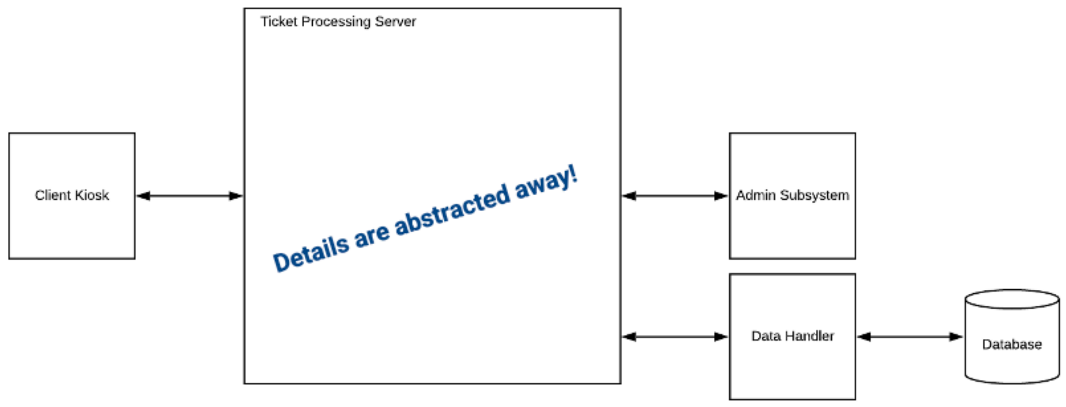
2. Types of UML Diagrams

- UML have wide array of diagrams to cater to different aspects of software development. These diagrams are grouped into two main categories: structure diagrams and behavior diagrams
  - Structure diagrams
    - Component
    - Package
    - Class

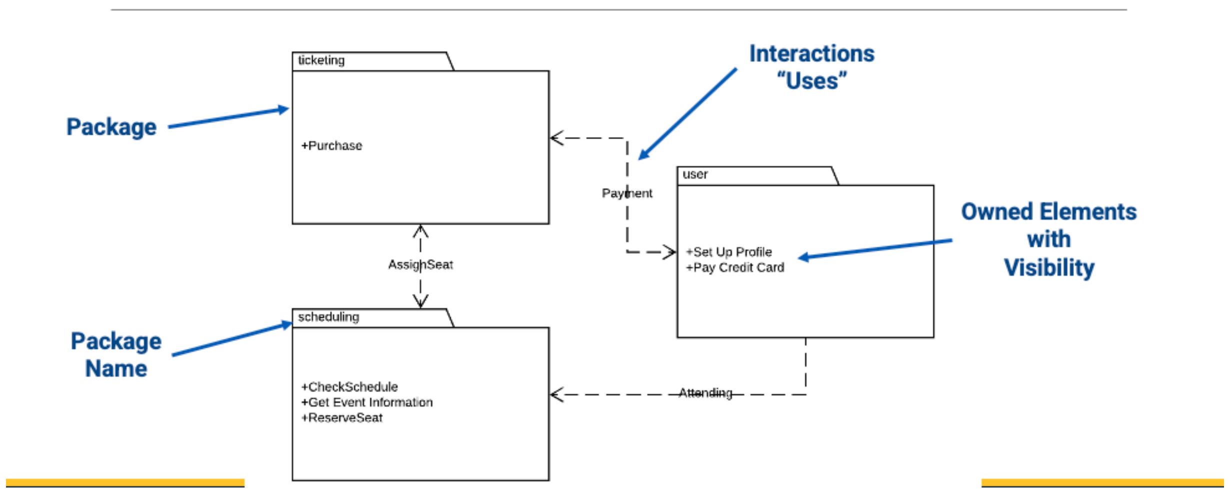
- Behavior Diagrams
    - Use Case
    - Sequence
    - Activity
    - State

STRUCTURE DIAGRAMS:

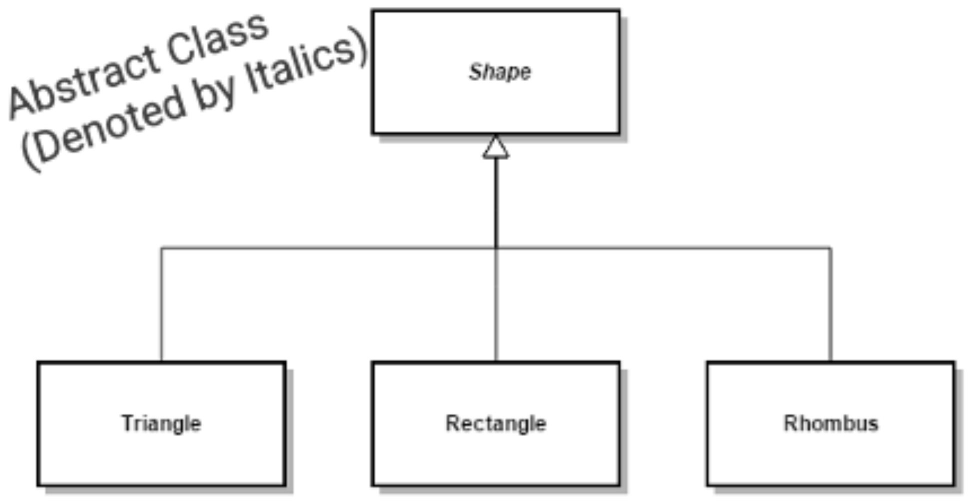
- Component Diagrams: Details the components and interfaces in a system, describing their service provision and requirements.



- Package Diagrams: Explores the use of packages to organize model elements, manage accessibility, and structure the system architecture.



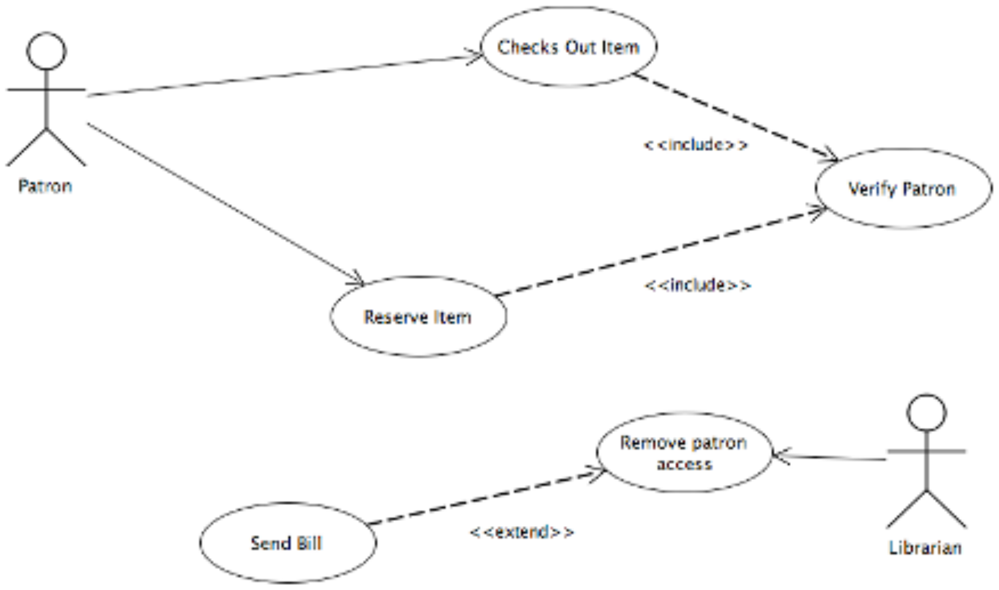
- Class Diagrams: Provides insights into class attributes, methods, visibility, and relationships, emphasizing object-oriented design principles like encapsulation, inheritance, and polymorphism.



BEHAVIOR DIAGRAMS:

- Use Case Diagram: Represents the functionality provided by a system in terms of actors, their goals represented as use cases, and any dependencies between those use cases.

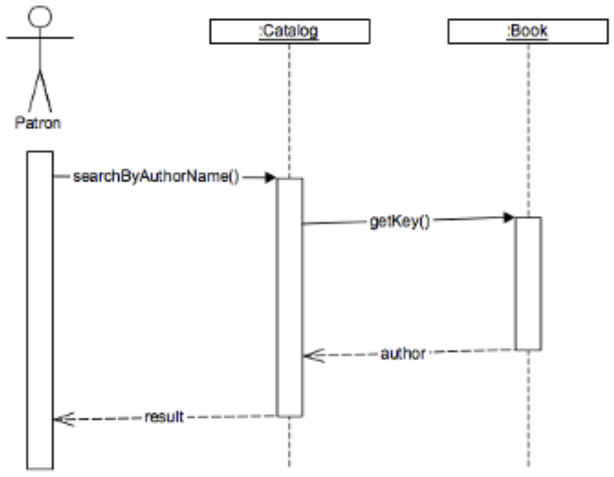
  - Show **actors** (stick person)
  - Show **use case** (oval with a label)
  - Show **relationships** between actors and use cases, between use cases, and between actors using arrows.
  - Do not show details of each use case



Sequence Diagram: Details how operations are carried out what messages are sent and when. Sequence diagrams are organized according to time.

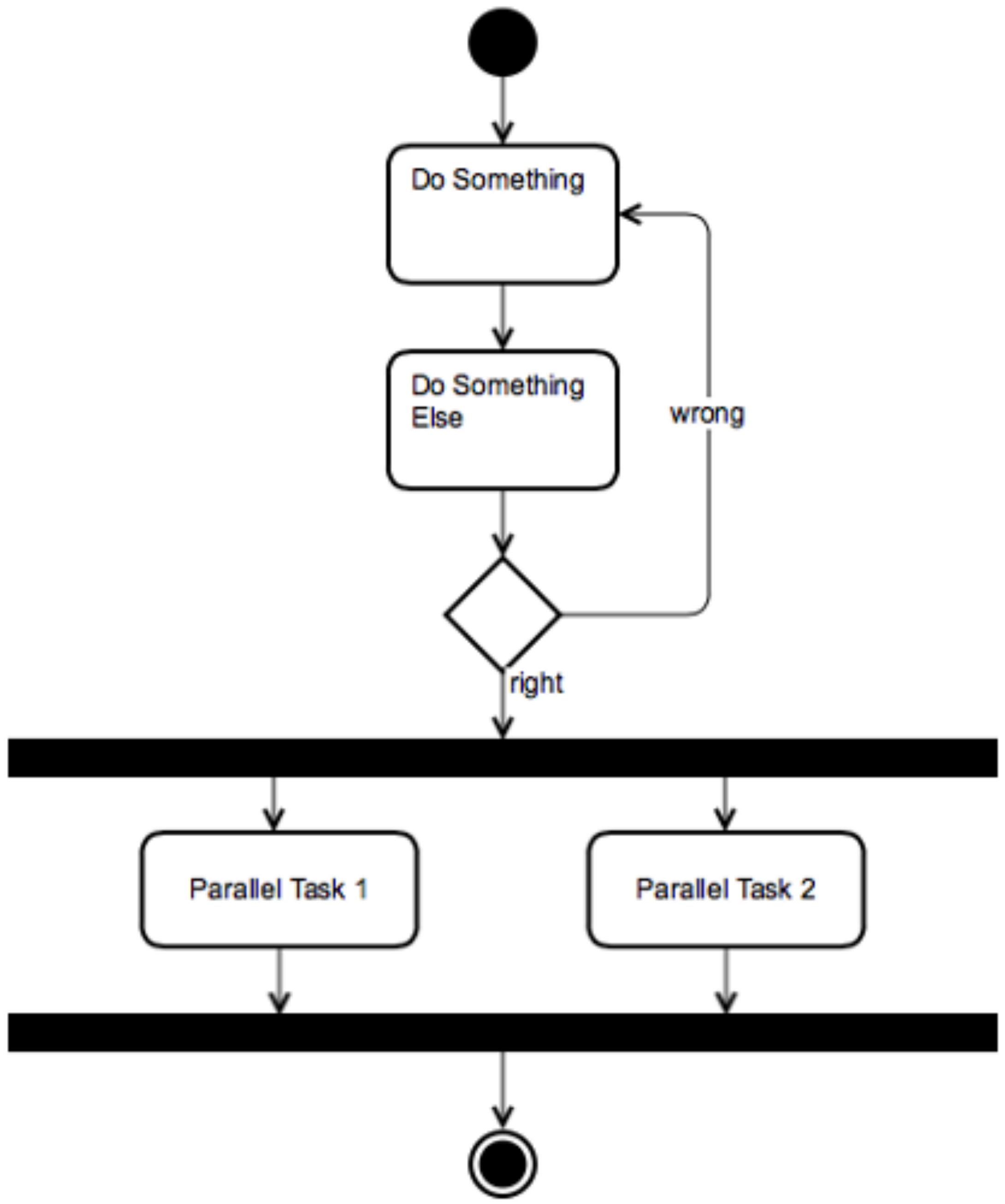
- List objects and actors involved along the top of the diagram.
- Draw a dotted line vertically below objects and actors.
- From top to bottom, draw arrows between interacting objects/actors.
- Annotate arrows with the name of the method called.
- Draw rectangles on the vertical lines to represent method execution.

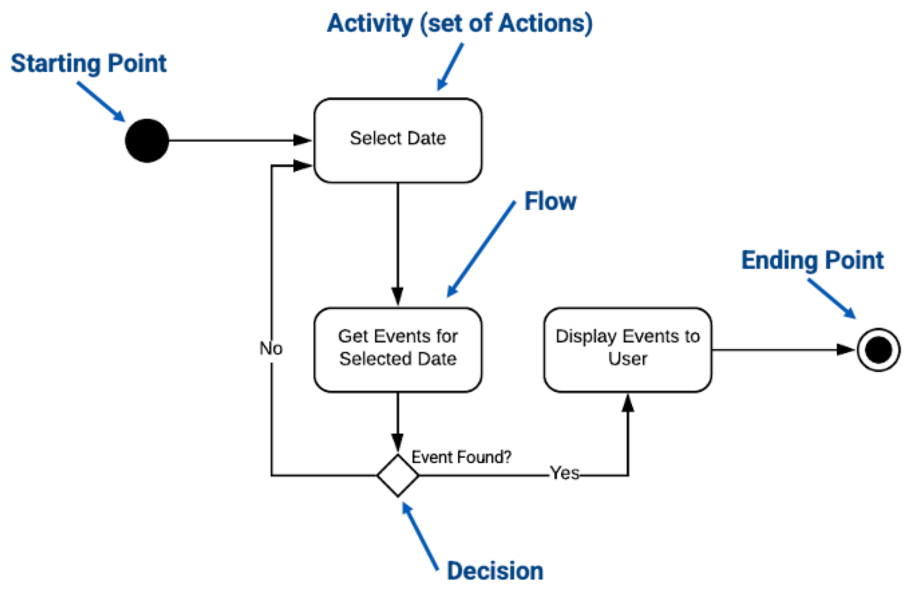
Often, developed directly from Use Case Scenarios!



Activity Diagram: Shows the flow of control or object flow with emphasis on the sequence and conditions of the flow. These diagrams are like advanced flowcharts.

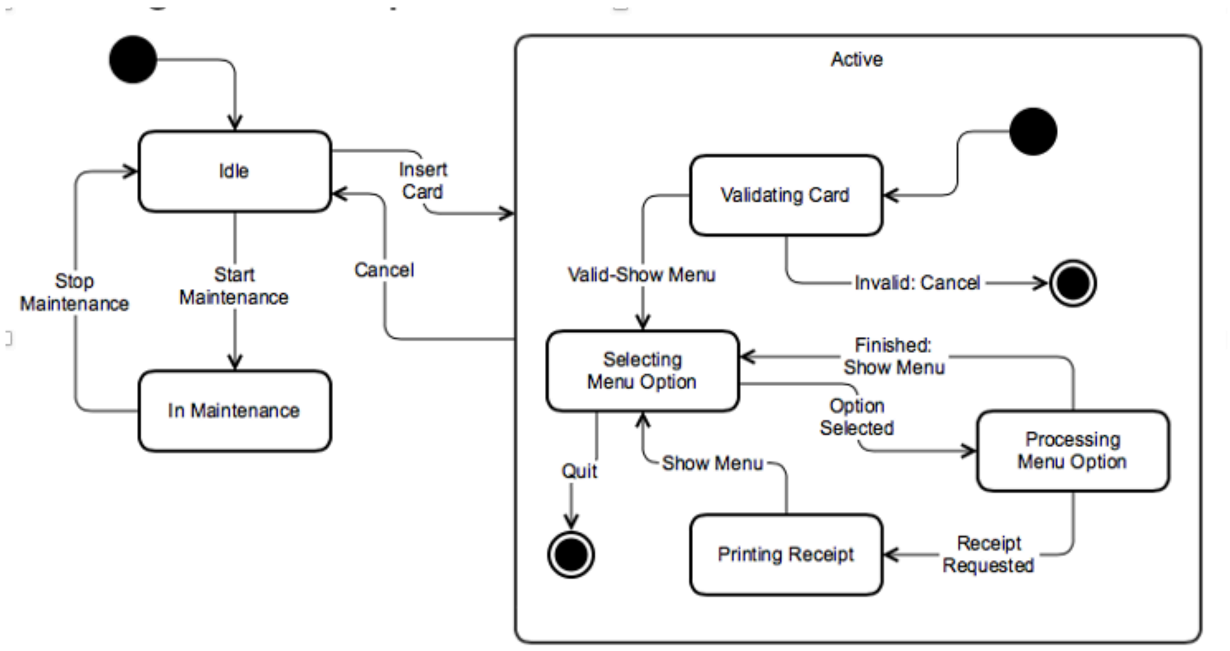
- Shows:
  - Starting Point
  - Activities in Sequence
  - Branching Activities
  - Parallel Activities
  - Ending Point

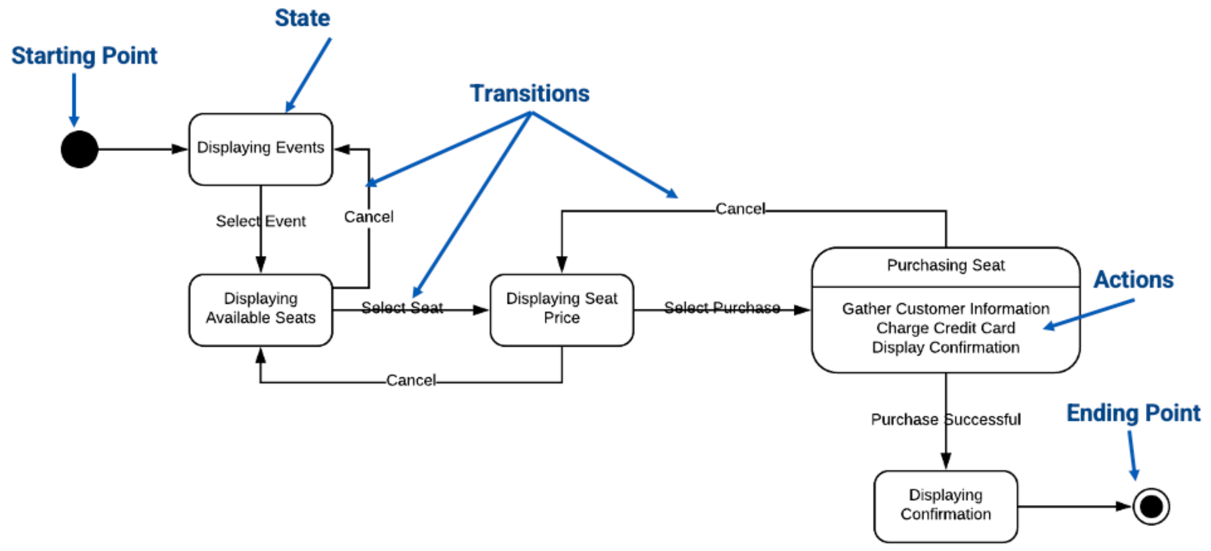

Do Something
Do Something Else
wrong
right
Parallel Task 1
Parallel Task 2



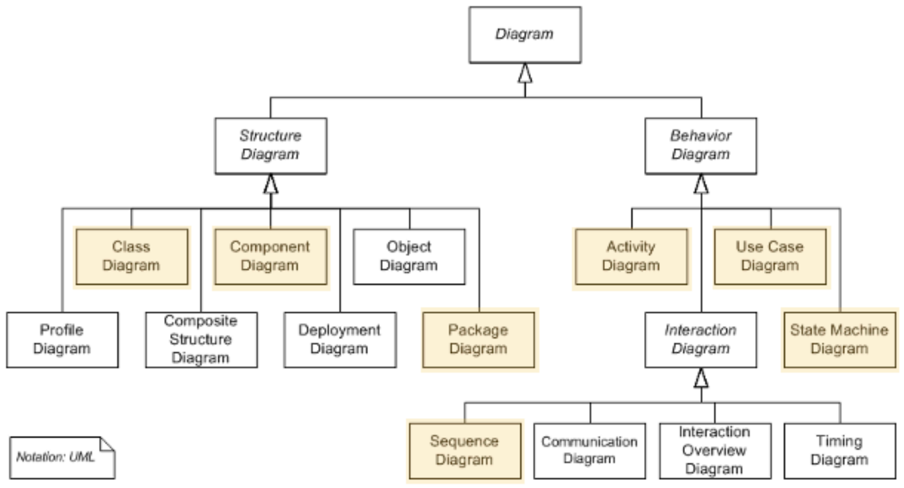
State Diagram: Models the states of an object and the transitions that cause a change from one state to another.

- Similar to Activity Diagrams in appearance, but State Diagram nodes represent the current state of a system.
- <u>States</u> may have <u>substates</u>, which exist as nested states.
- Arcs between nodes represent <u>transitions</u> from between states.
- <u>Events</u> trigger each state transition and are labels on each transition.





3. UML Diagram Tree



Work Cited

https://elearn.etsu.edu/d2l/le/content/9704124/viewContent/94265456/View