Introduction to UML

Yulia Newton, Ph.D.

CS151, Object Oriented Programming

San Jose State University

Spring 2020

UML introduction

- Unified Modeling Language
- Standard graphical modeling language for specifying, visualizing, and documenting software systems
  - Describes the structure and behavior of the system
- Platform independent
- Pictorial language
  - Software blueprint
  - *Picture is worth a thousand words*
- Greatly useful for OOD
- Conceptual mode of UML
  - UML building blocks
  - Rules for how to connect these blocks

Project development process

- Waterfall
  - Analysis, design, implementation and testing
- Iterative
  - Break project into pieces of functionality
  - Most commonly used

Project planning

- Predictive planning
  - Use when you can spec out and plan ahead all the pieces of your project
  - Most efficient and least costly method
  - Not often used as real life tends to be more complicated

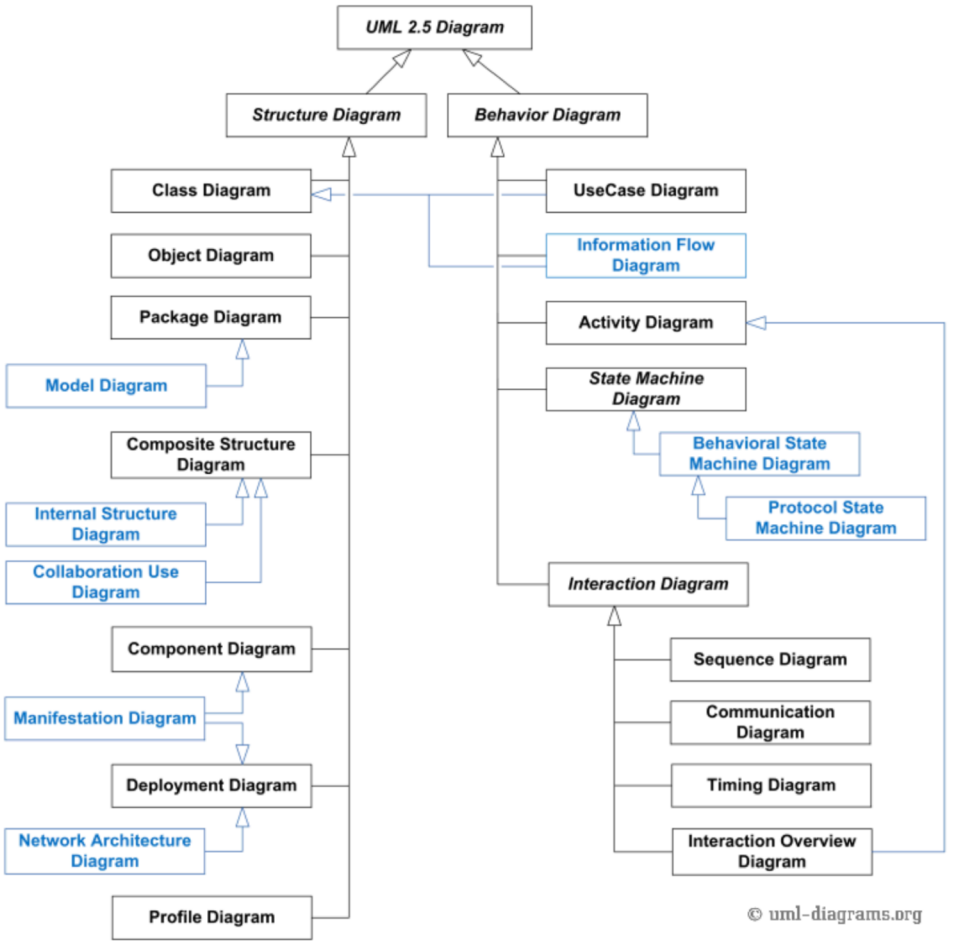
- Adaptive planning
  - Changes in requirements are going to happen and you adapt as you go
  - Agile development

Object oriented concepts used in UML

- **Class –** blueprint of an object (structure, properties, methods)
- **Objects –** fundamental building block of a system, helps us modularize the system
- **Inheritance –** child classes inherit properties and behaviors of parent class
- **Abstraction –** implementation details are hidden from users
- **Encapsulation –** binding related data together
- **Polymorphism –** ability for methods to exist in multiple forms

UML diagrams

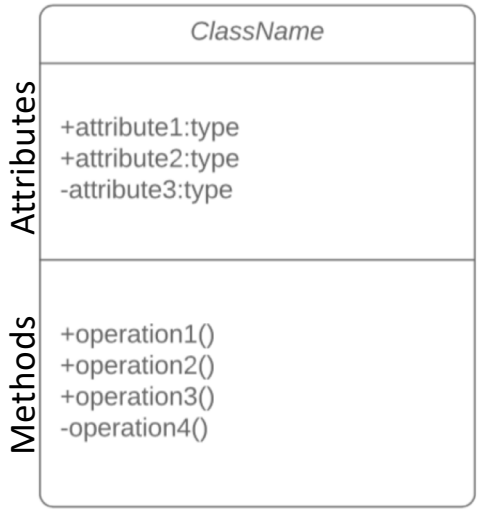
- Structure diagrams
  - Static aspects of the system
- Behavior diagrams
  - Dynamic aspects and behavior of the system



https://www.uml-diagrams.org

Class diagram

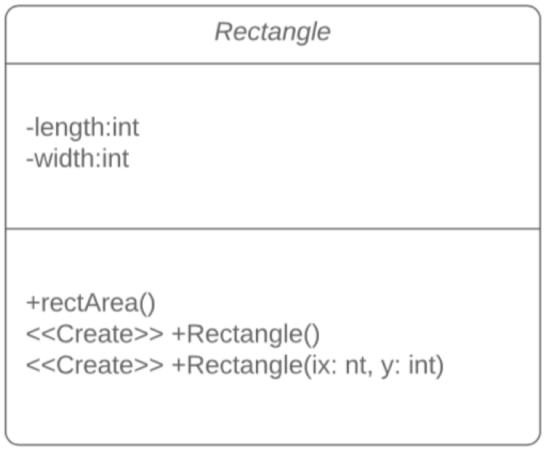
- Describes system structure by describing its classes and class relationships as well as their attributes and methods
- One of the most useful UML diagrams



Visibility:
- private
+ public
# protected
~ package

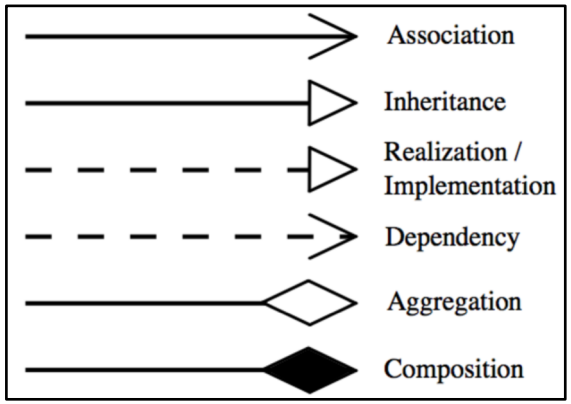
Example: Rectangle class

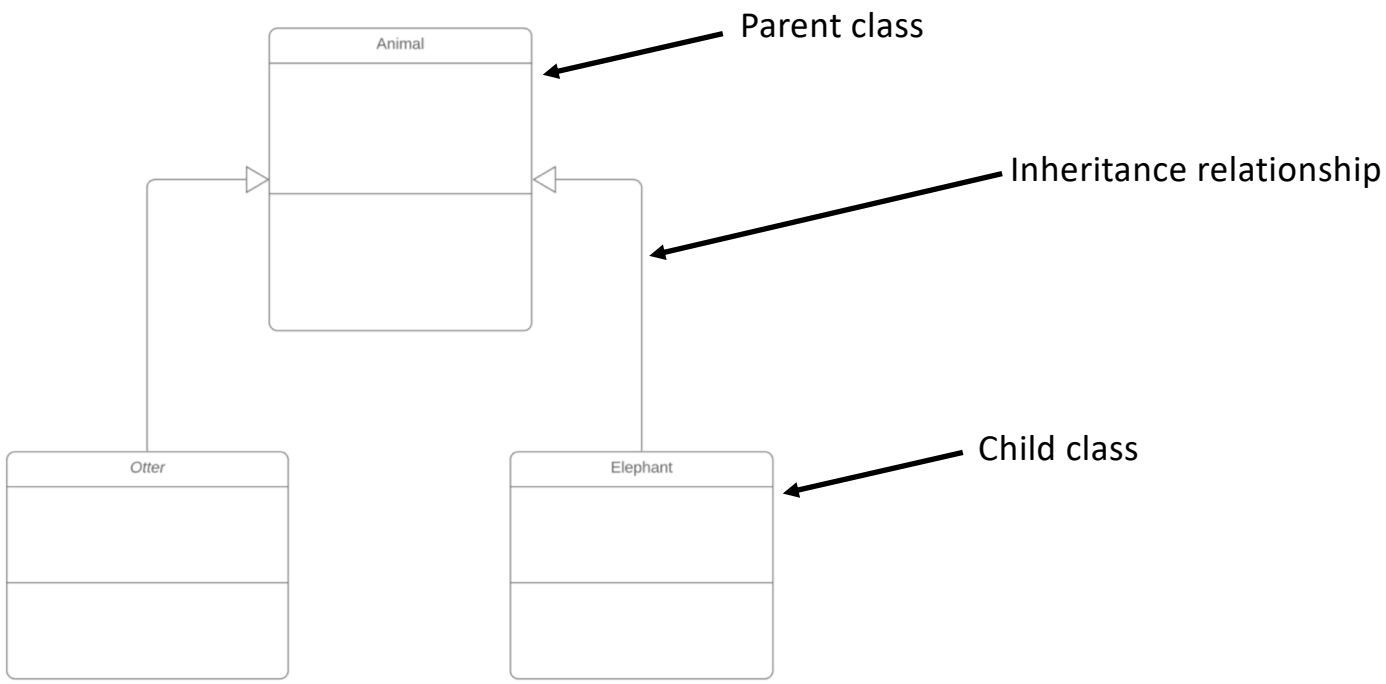
```
class Rectangle{
    int length;
    int width;

    Rectangle(){
        length = 1;
        width = 1;
    }

    Rectangle(int x, int y){
        length = x;
        width = y;
    }

    public void rectArea(){
        int result;
        result = length * width;
        System.out.println("Area of rectangle is: " +result);
    }
}
```



Note: often constructors are not listed in the class diagram

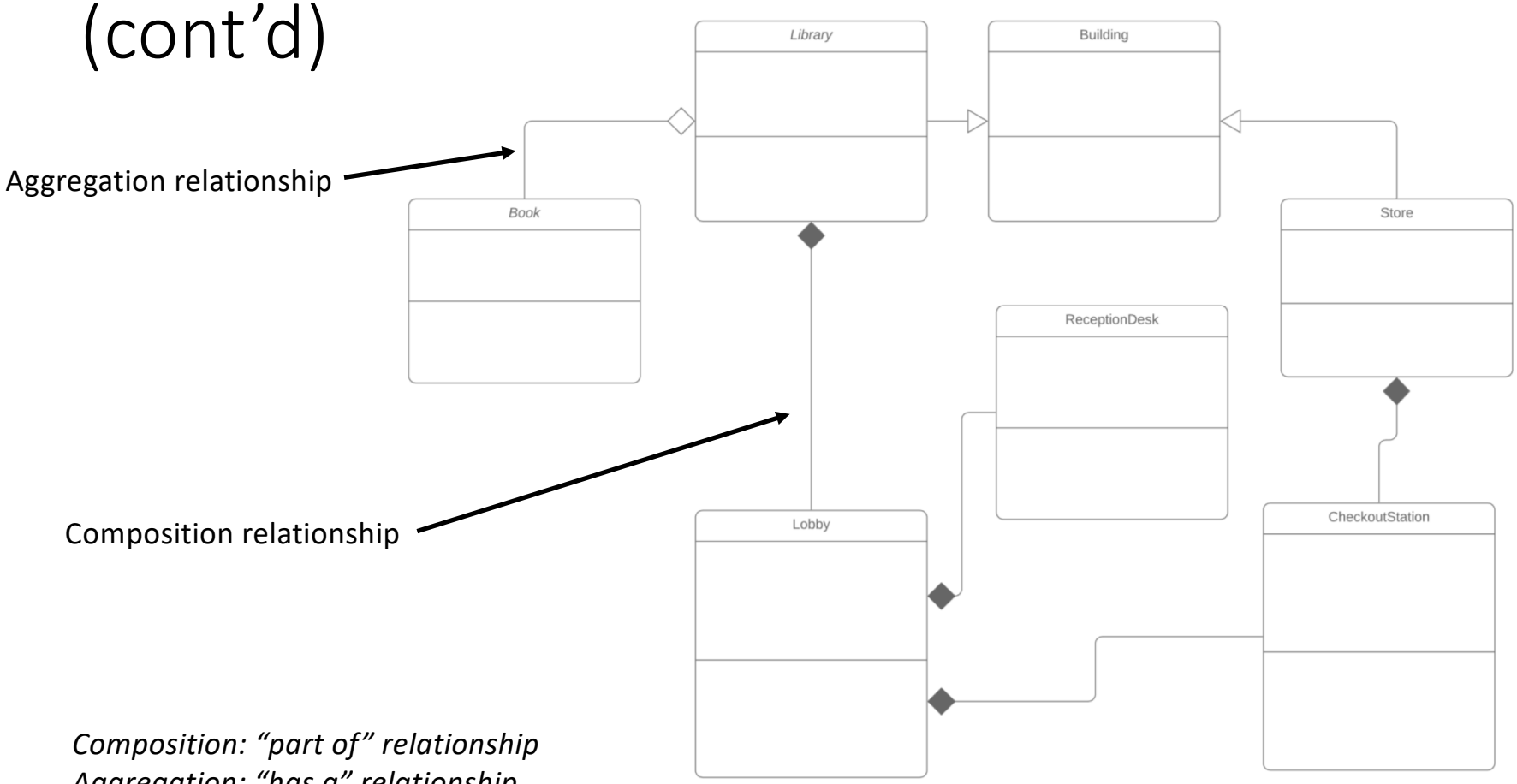
Representing relationships in a class diagram





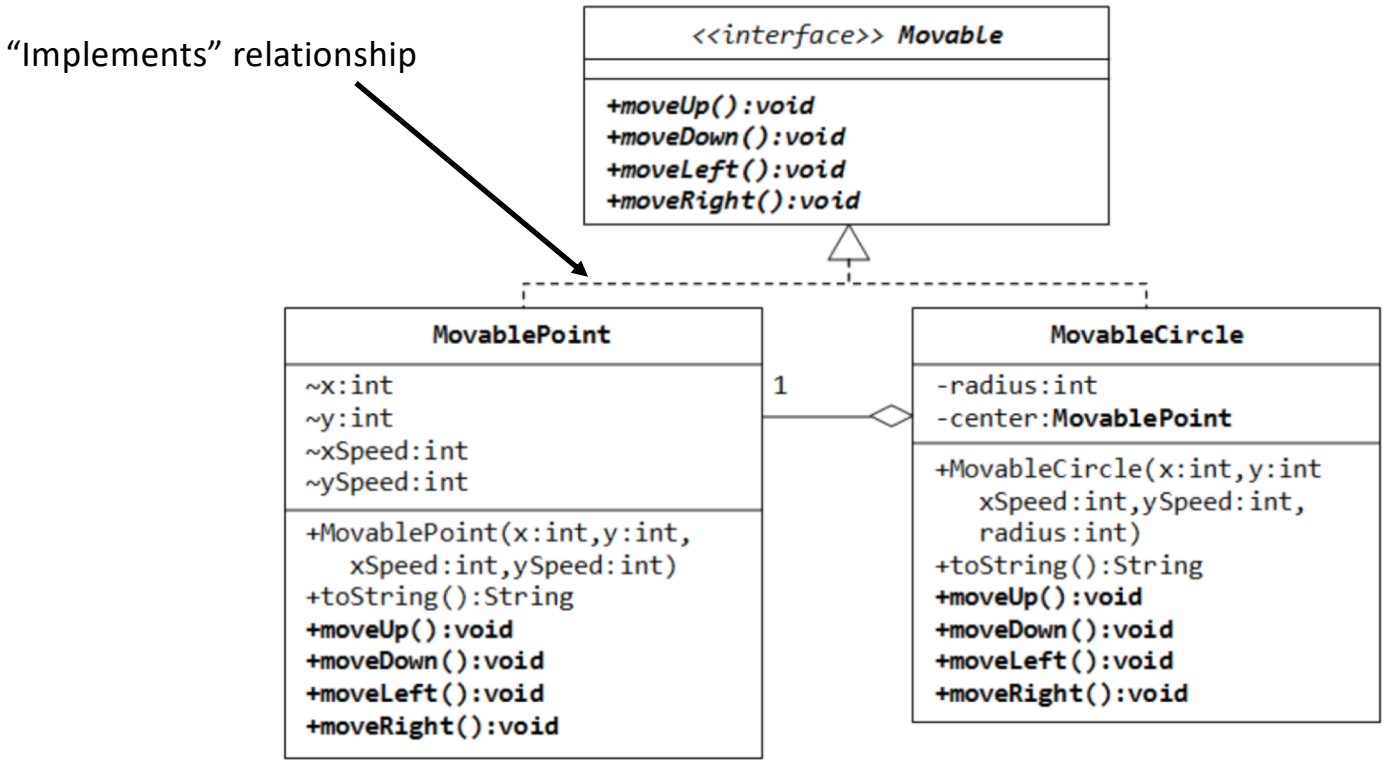
Inheritance: "is a" relationship

Representing relationships in a class diagram (cont'd)



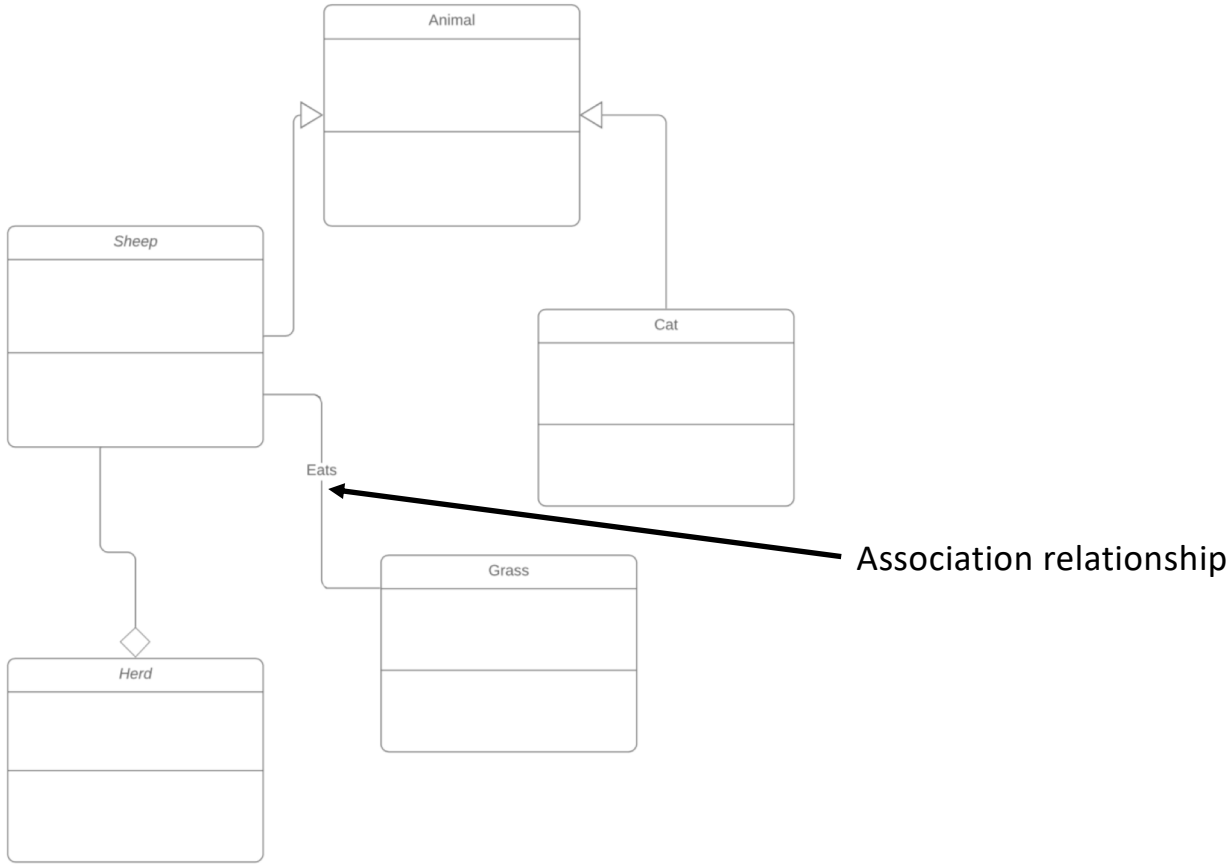
Composition: "part of" relationship
Aggregation: "has a" relationship

Representing relationships in a class diagram (cont'd)



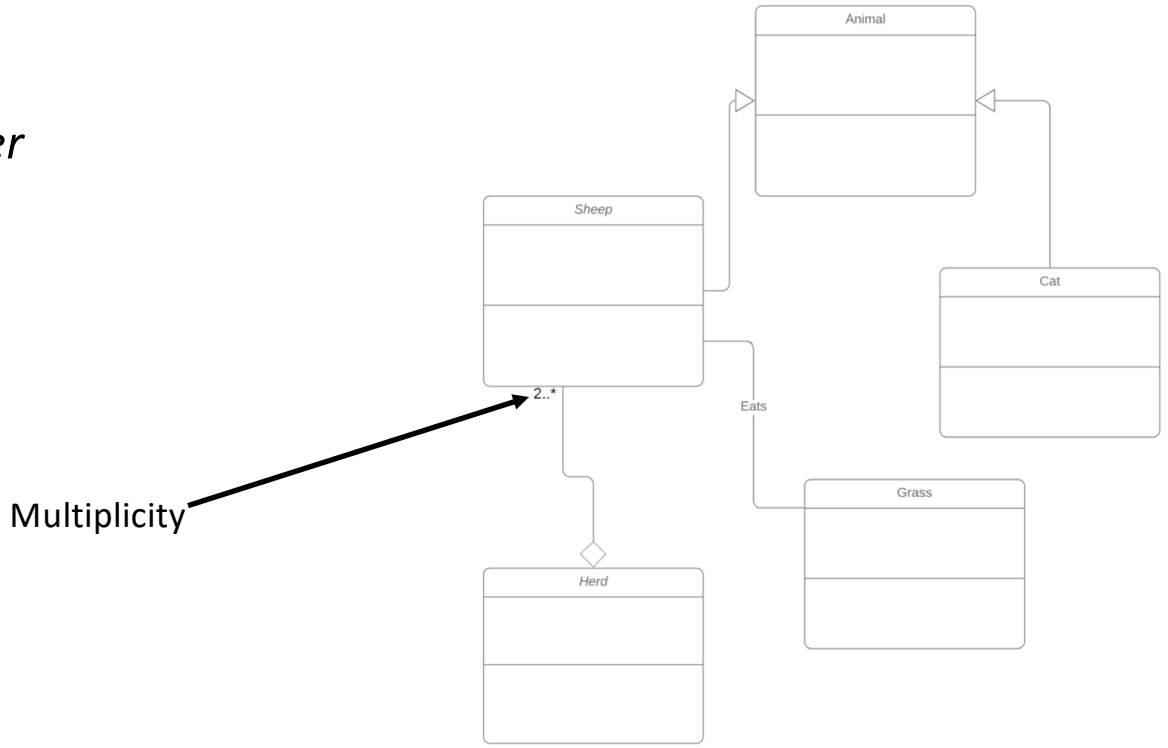
https://medium.com/@smagid_allThings/uml-class-diagrams-tutorial-step-by-step-520fd83b300b

Representing relationships in a class diagram (cont'd)

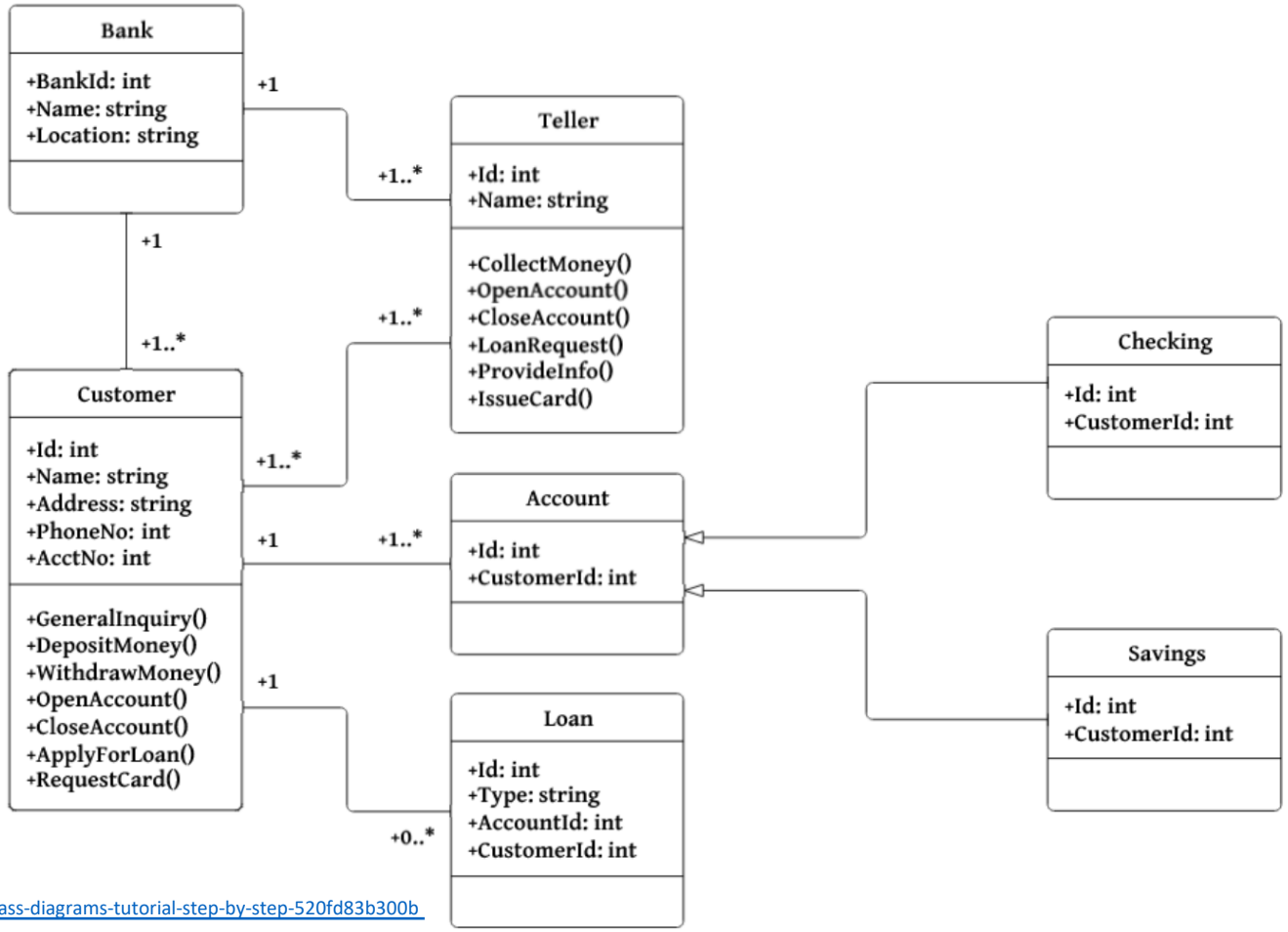


Multiplicity

- Adding a constraint for how many objects have to participate in the relationship
  - 0..1 *zero or one*
  - n *specific number*
  - 0..* *any number*
  - 1..* *at least one*



Example: banking system



https://medium.com/@smagid_allThings/uml-class-diagrams-tutorial-step-by-step-520fd83b300b

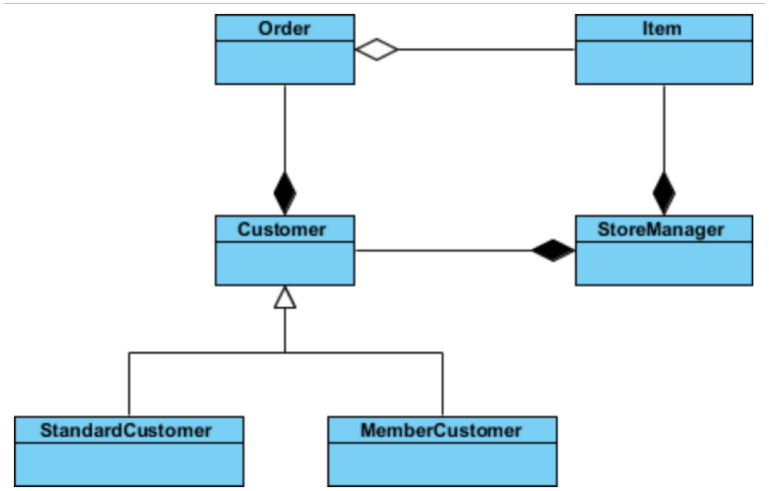
Composite structure diagram

- Models how objects work together
- Similar to class diagram but more elaborate
- Has a container with other objects inside it
  - For each object it shows *role:type[multiplicity]*
    - Multiplicity is optional

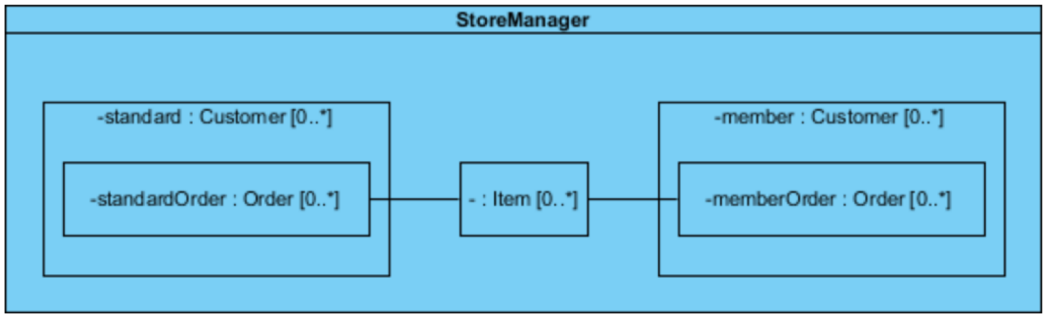
Types of composite structure diagram

- Internal structure diagram
  - Purpose is to allow users to peak inside an object diagram to view how its different parts relate and what it is composed of
- Collaboration diagram
  - Describe cooperation behavior between objects in a system

Converting class diagram to composite diagram

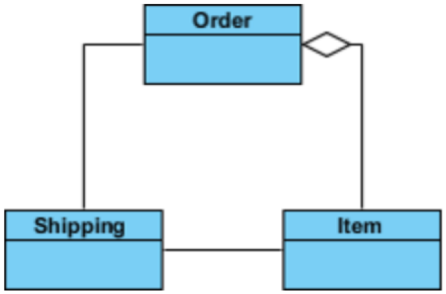


https://www.visual-paradigm.com

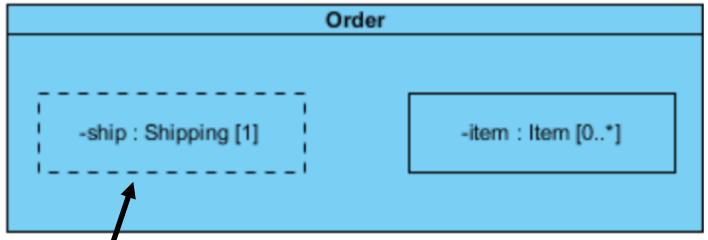


https://www.visual-paradigm.com

Representing references to outside entities



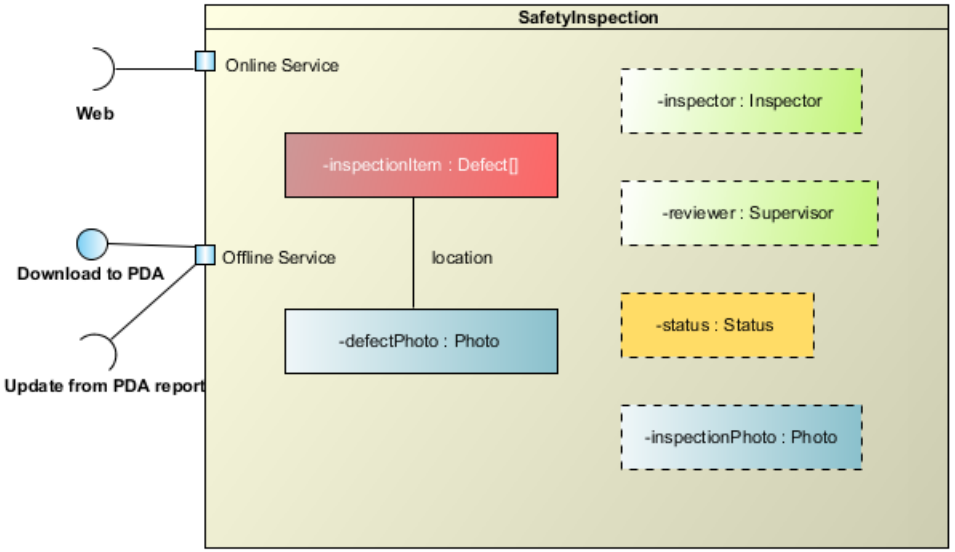
https://www.visual-paradigm.com



https://www.visual-paradigm.com

An outside reference

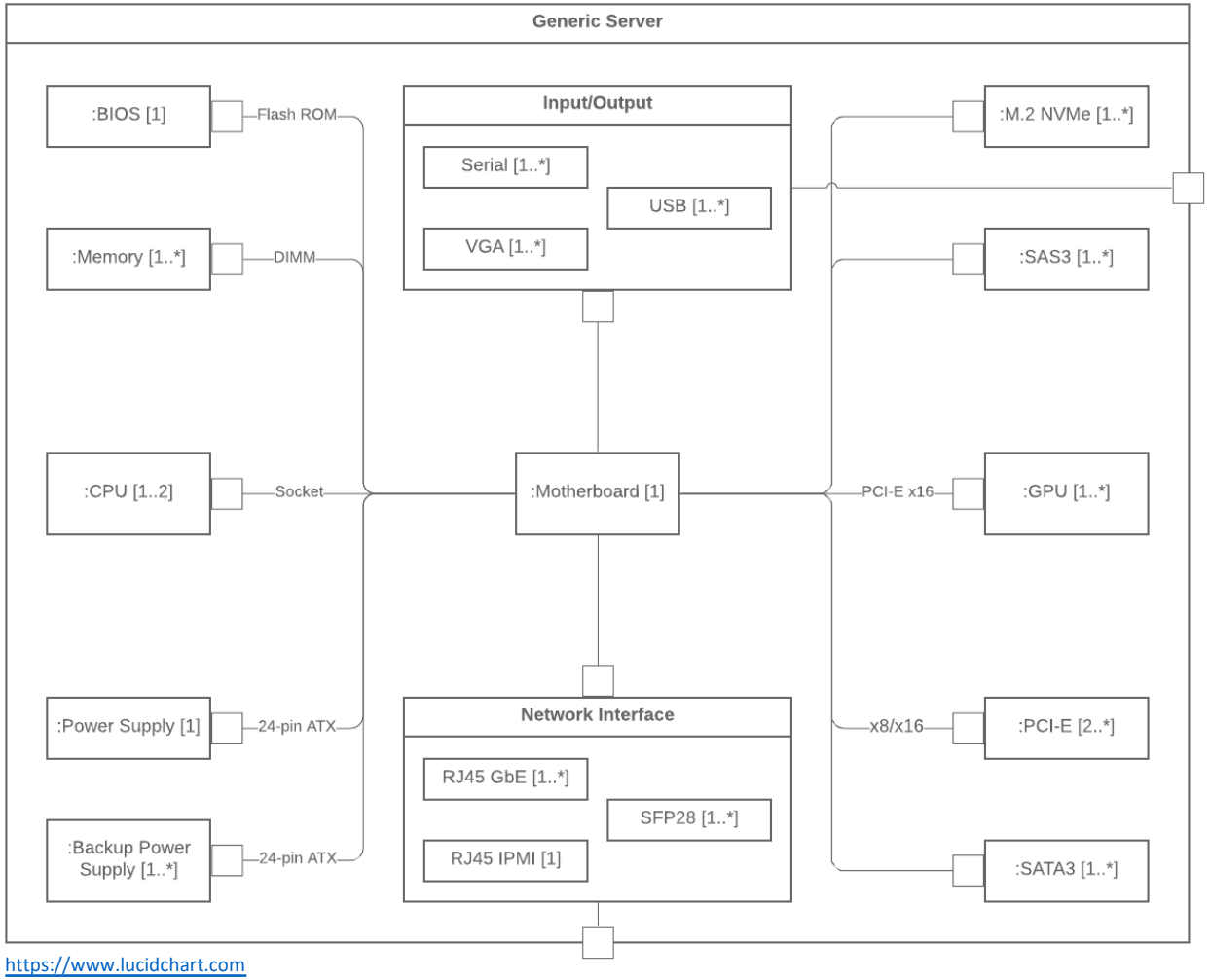
Example: safety inspection system



https://www.visual-paradigm.com

Example: server

Internal structure diagram

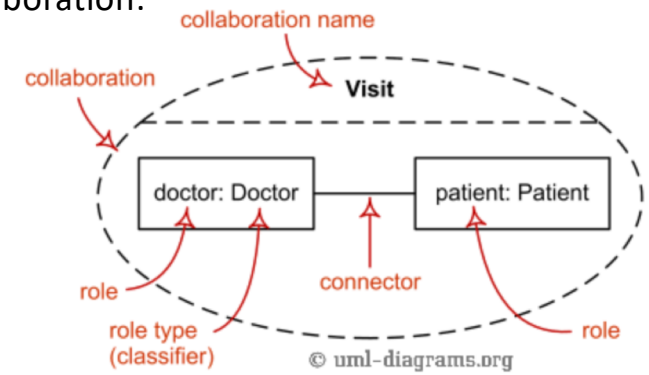


https://www.lucidchart.com

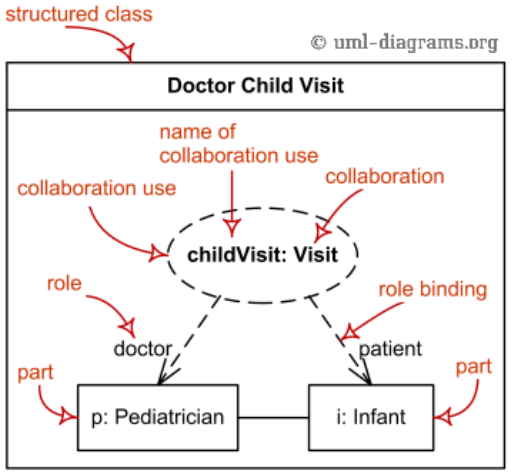
Collaboration use diagram

- Describes cooperating behavior between entities
  - Collaborations describe relevant aspects of cooperation
  - Can be used to describe changing behavior between entities based on interactions

Collaboration:



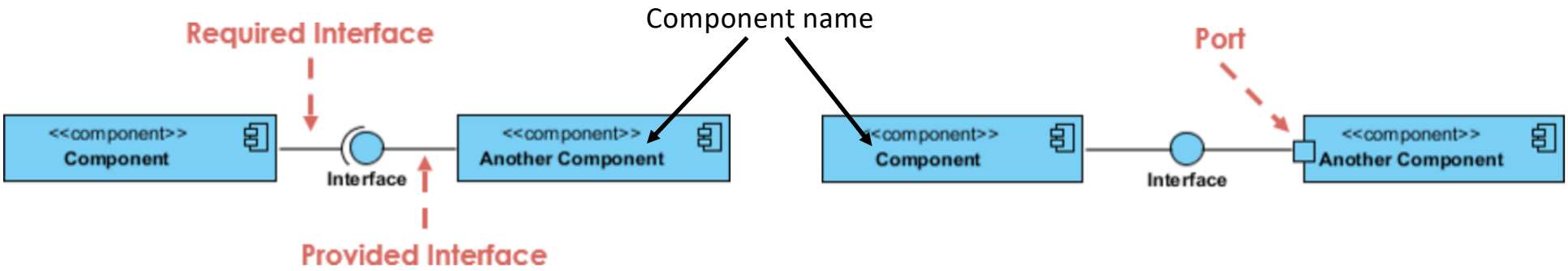
Collaboration use describes an occurrence/instance of a particular collaboration:



https://www.uml-diagrams.org

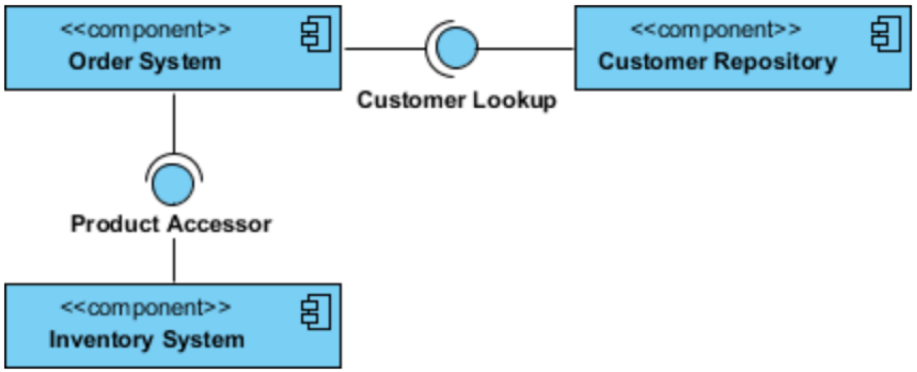
Component diagram

- Represent reusable pieces of code
  - Breaks down the system into high level functionality modules
- Represents one clear objective
- Interacts with other system components through interfaces



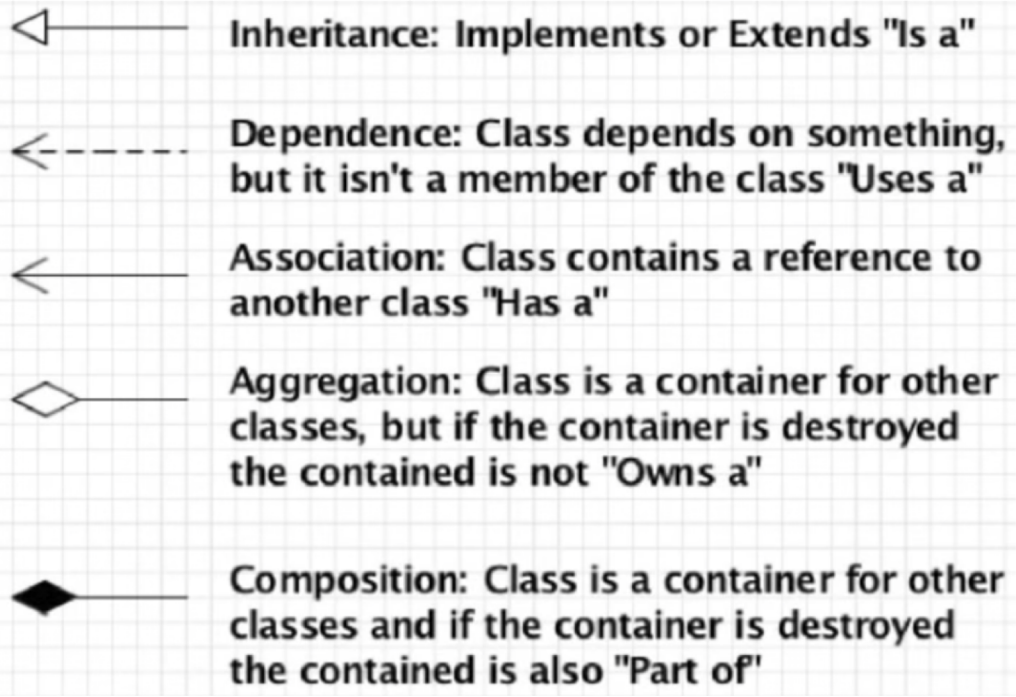
https://www.visual-paradigm.com

Example: component diagram of order system



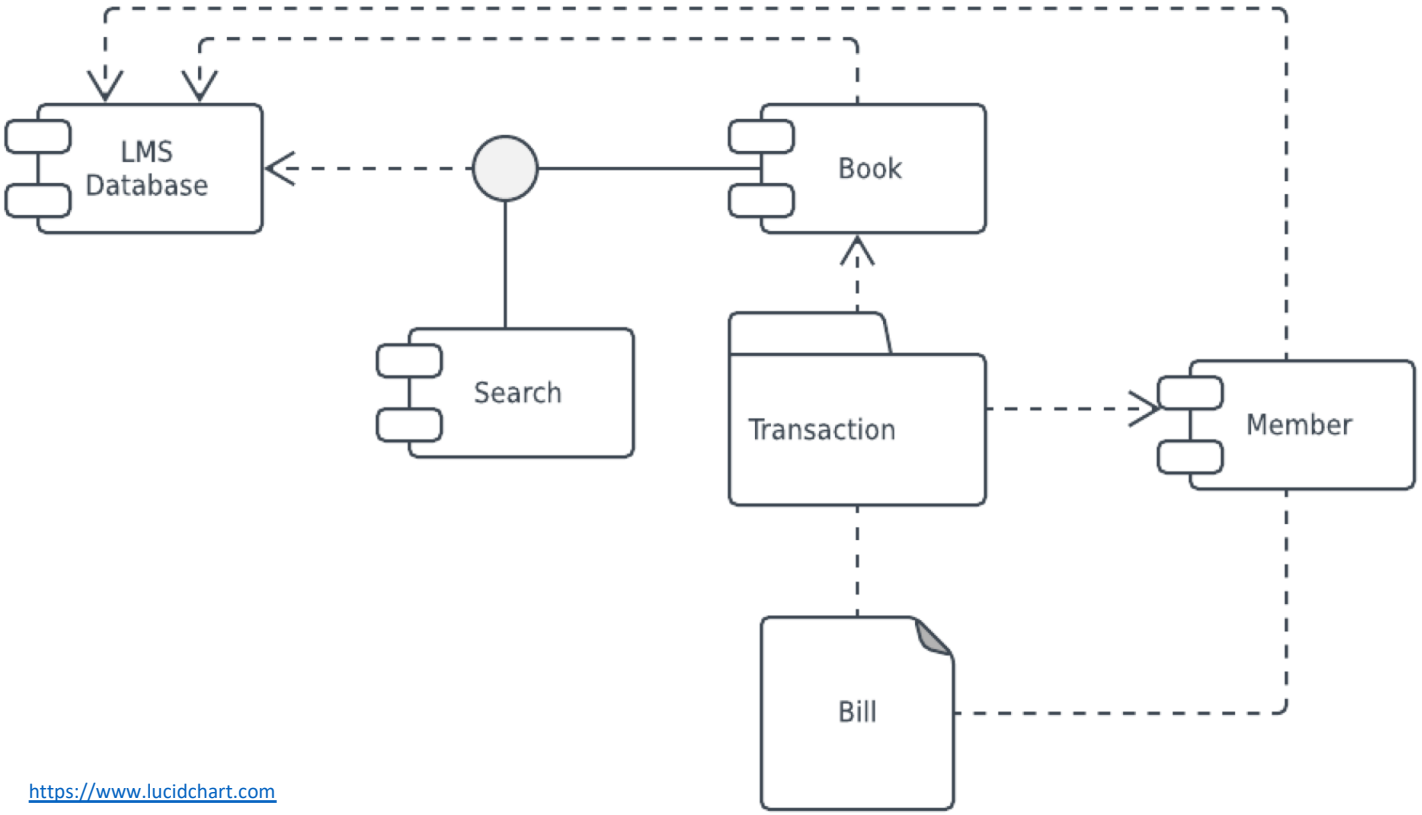
https://www.visual-paradigm.com

Types of relationships in component diagrams



https://www.youtube.com/watch?v=KQUGFFN4M90

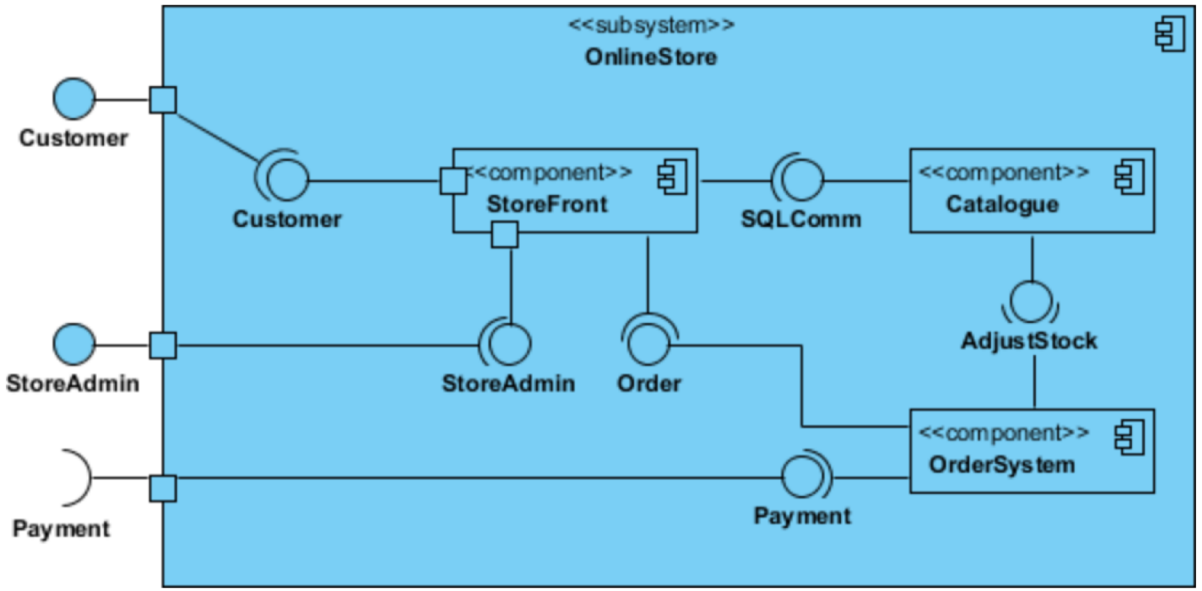
Example: library management system



https://www.lucidchart.com

Subsystems

- Has a keyword *subsystem*



https://www.visual-paradigm.com

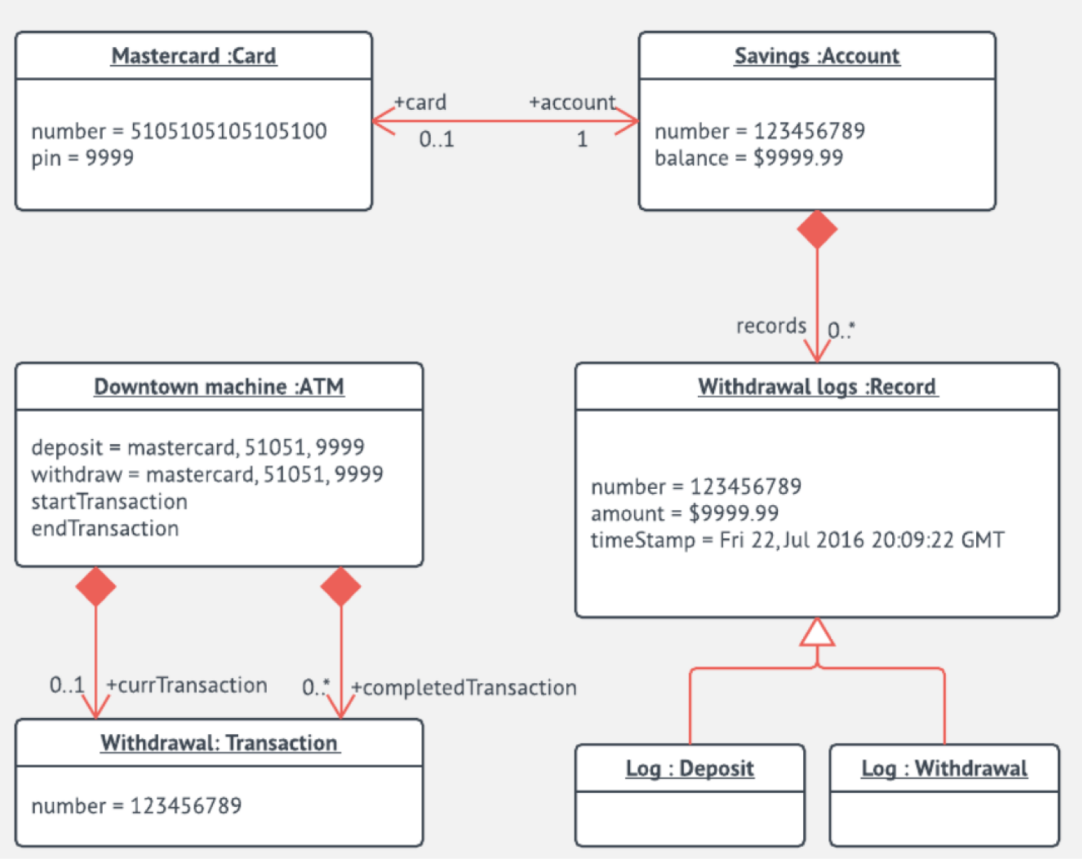
For both component and composite structure diagrams

- https://www.youtube.com/watch?v=KQUGFFN4M90

Object diagram

- Shows data structures at a given point of time or in a given instantiation
  - Examples of data structures
- E.g. class diagram shows class structure but object diagrams show test cases for this class

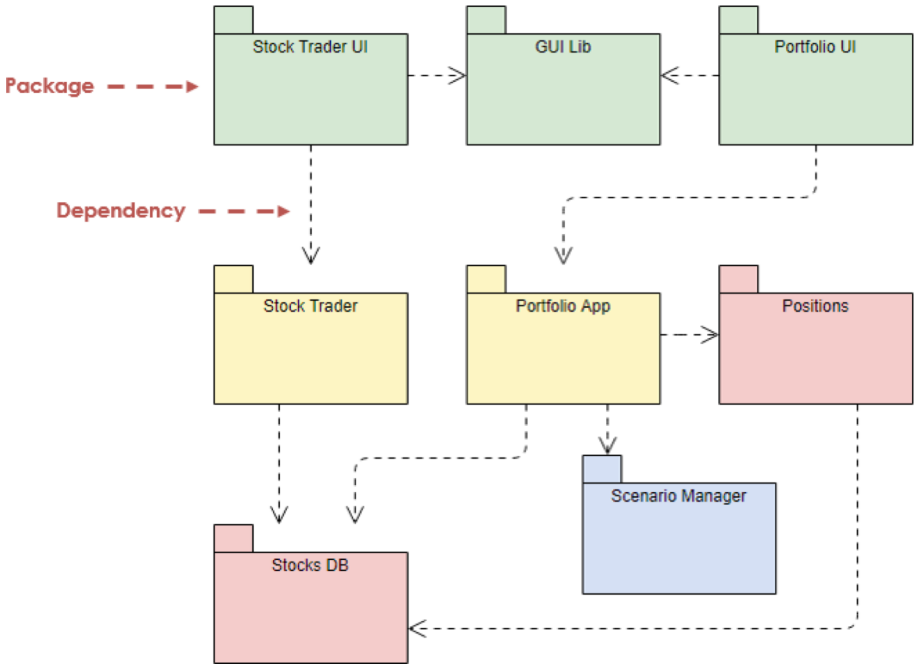
Example: ATM



https://www.lucidchart.com

Package diagram

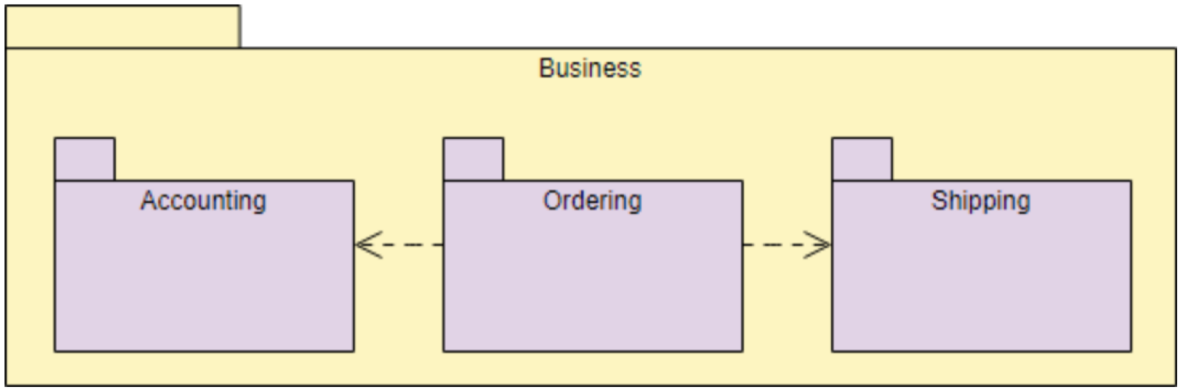
- Shows dependencies between different packages in a given system



https://online.visual-paradigm.com

Using a package diagrams to simplify system description

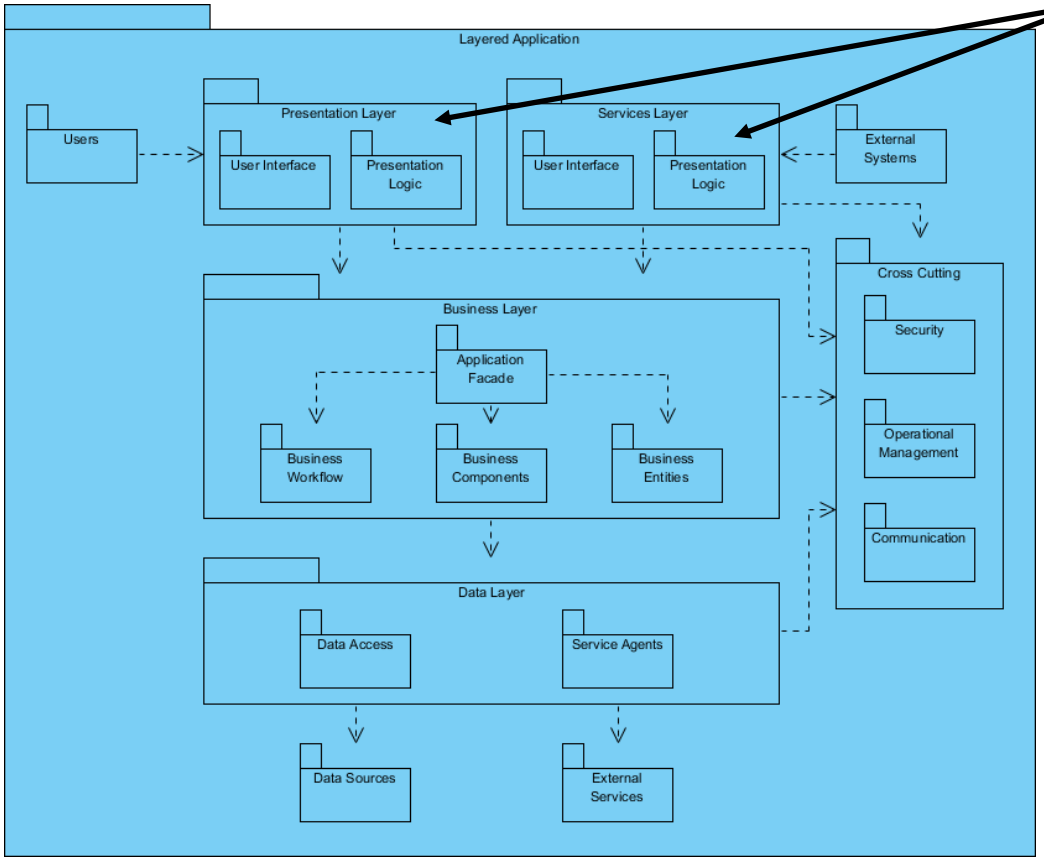
- Can describe composition of a system with high-level entities and dependencies



https://online.visual-paradigm.com

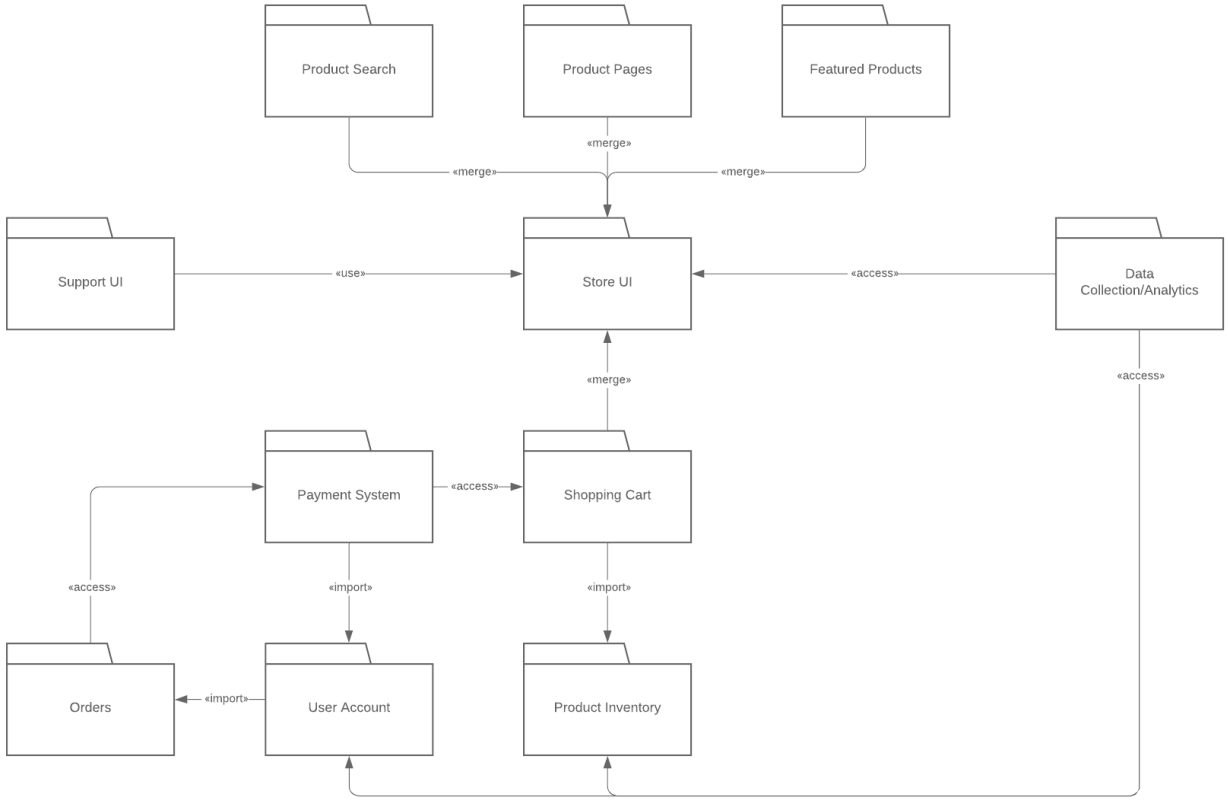
Example: layered application

You can nest entities in a package diagram



https://www.visual-paradigm.com

Annotating relationships in a package diagram

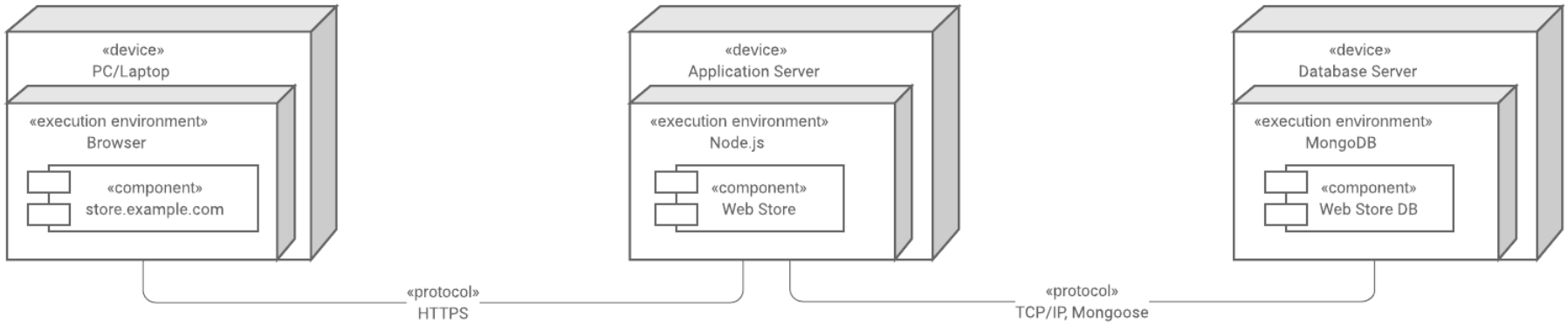


https://www.lucidchart.com

Deployment diagram

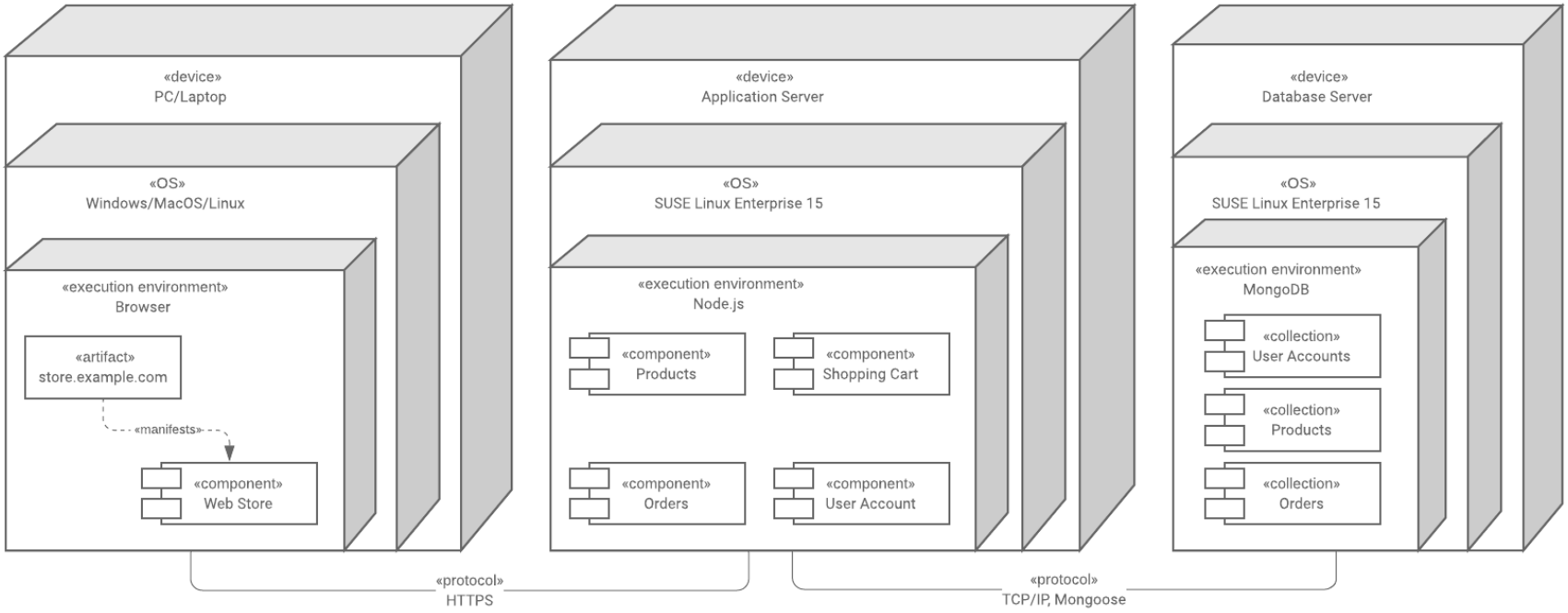
- Shows architecture of the system
- Run-time configuration deployed system and components
  - Most useful to system engineers
  - Can show performance, scalability, maintainability, and portability
- Helps ensure all elements are accounted for during deployment of the system
- Shows how software and hardware work together
- Good video
  - https://www.youtube.com/watch?v=nTtQwGoUUNc

Example: high level system description



https://www.lucidchart.com

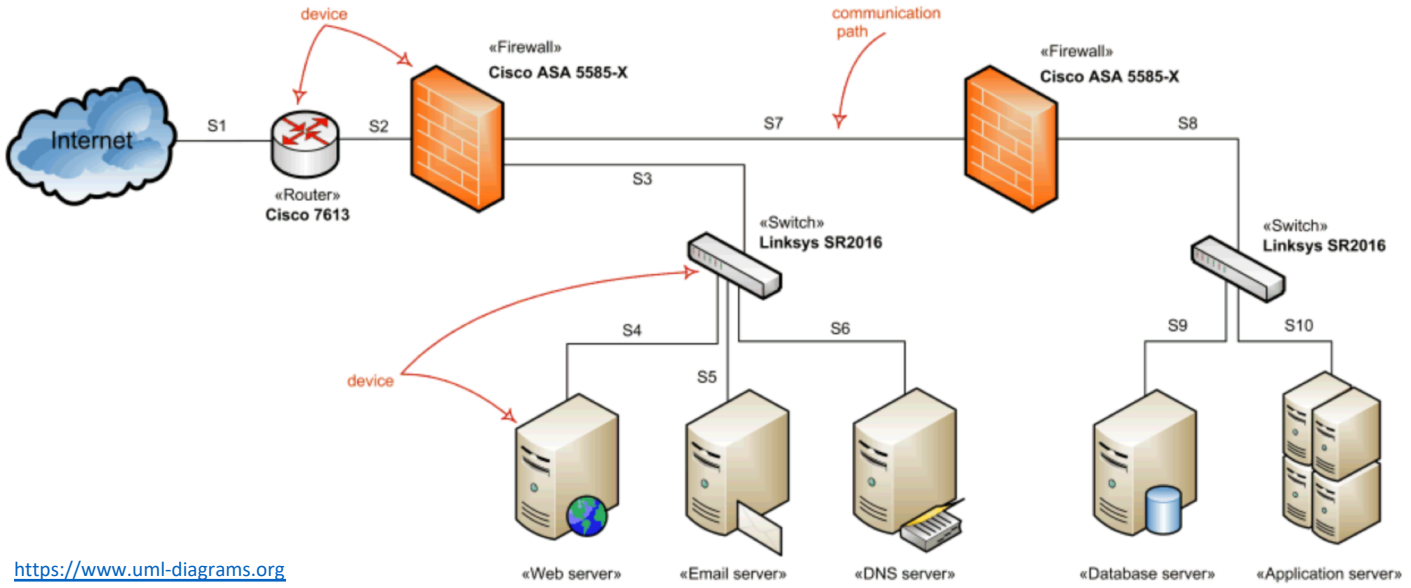
Example: low level system description



https://www.lucidchart.com

Network architecture diagram

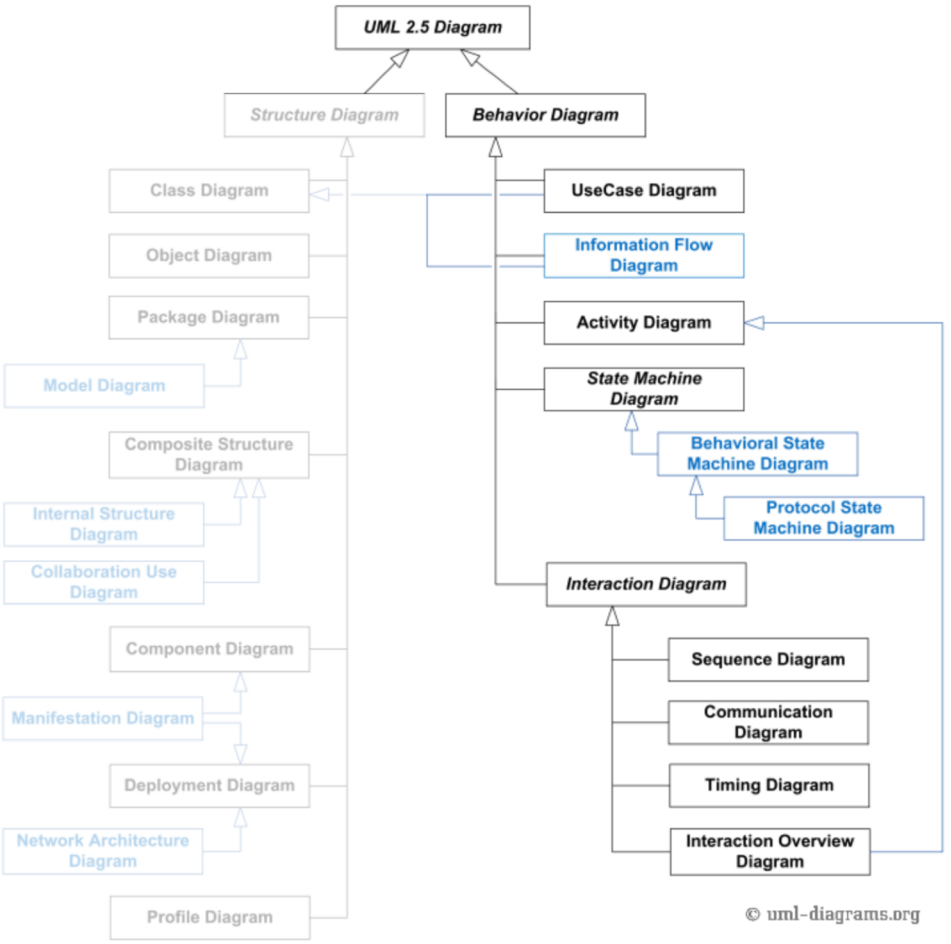
- No UML specification for describing networks
- A modified deployment diagram can be used to describe network architecture



https://www.uml-diagrams.org

UML diagrams

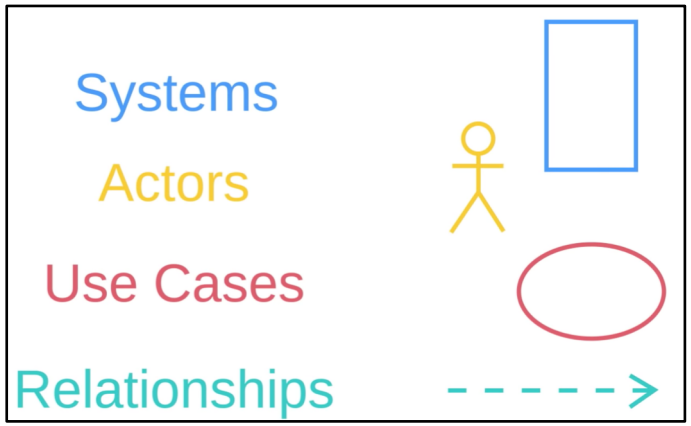
- Structure diagrams
  - Static aspects of the system
- Behavior diagrams
  - Dynamic aspects and behavior of the system



https://www.uml-diagrams.org

Use cases

- Outline different requirements of your program and how your program is going to be used
- Represent functions, processes, or activities performed in the modeled system
- Explains how users interact with the system



https://www.youtube.com/watch?v=zid-MVo7M-E

- Preceded by use case description
  - Trigger

Terminology

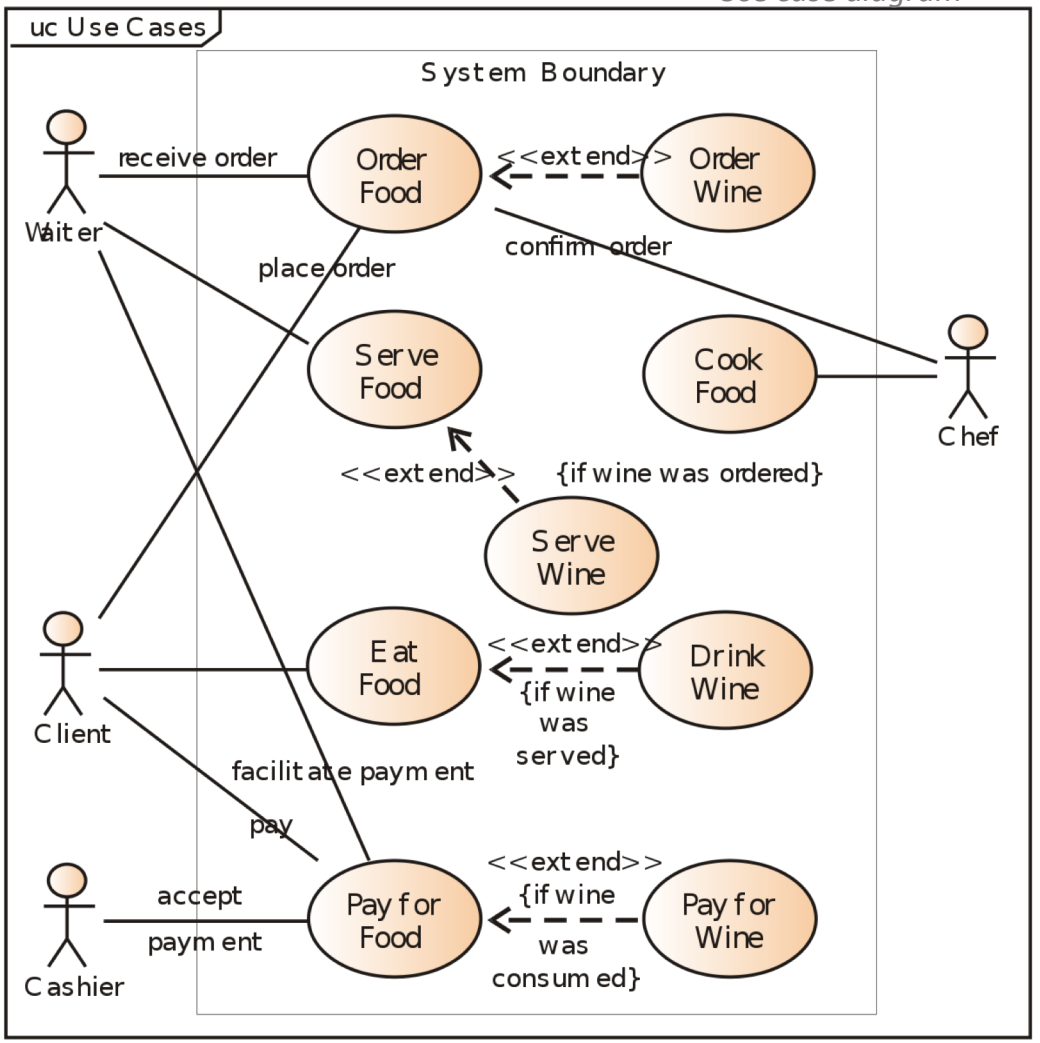
- System is whatever you are describing
  - Represented with a rectangle
  - Everything inside the rectangle happens within the modeled system
- Actor is something or someone who uses the system
  - Primary actor
    - Triggers/initiates the action
  - Secondary actor
    - Reacts to primary actor's or system's actions
  - Most often represented by a person stick figure
- Use case is an action within the system
  - Represented by an oval
- Relationships
  - Represented by lines/arrows
  - Association, include, extend, generalization
  - Multiple use cases might utilize another use case through relationship links

Diving deeper into use case diagram relationships

- Association
  - Between an actor and a use case
  - Represented by a solid line
- Include
  - Included use case is always executed when the base use case is executed
    - E.g. ***password verify*** use case is always performed when the ***login*** use case is performed
  - Can be thought of as “require” relationship (base use case requires included use case)
  - Represented with a dashed arrow pointing from base use case to included use case, annotated with the word <<include>>
- Extend
  - When base use case extends another use case, that use case may not be executed every time
    - E.g. ***baggage check-in*** use case extends ***passenger check-in*** use case
  - Represented with a dashed arrow pointing from extend use case to base use case, annotated with the word <<extend>>
- Generalization
  - Parent use case to child use case relationships
  - Represented by inheritance arrows (solid line with an arrow at the end), pointing from the child to the parent use case
  - Generalizations can apply to both use cases and actors

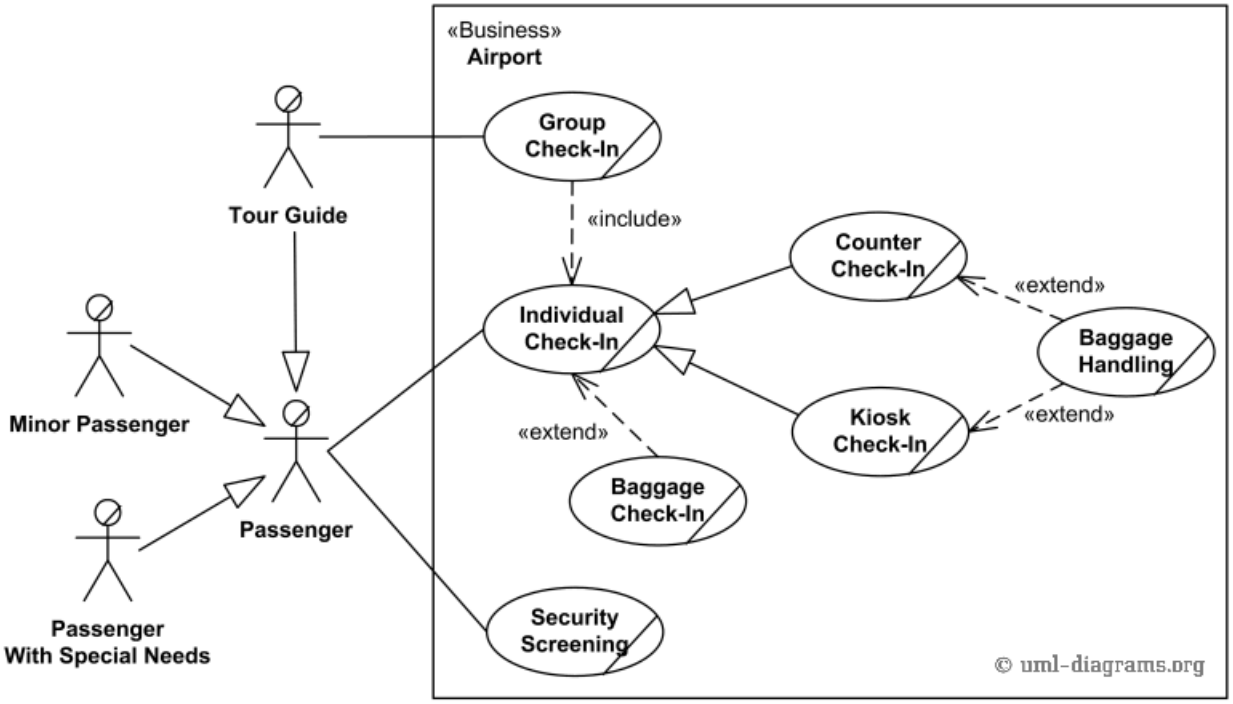
Example: restaurant

Use case diagram



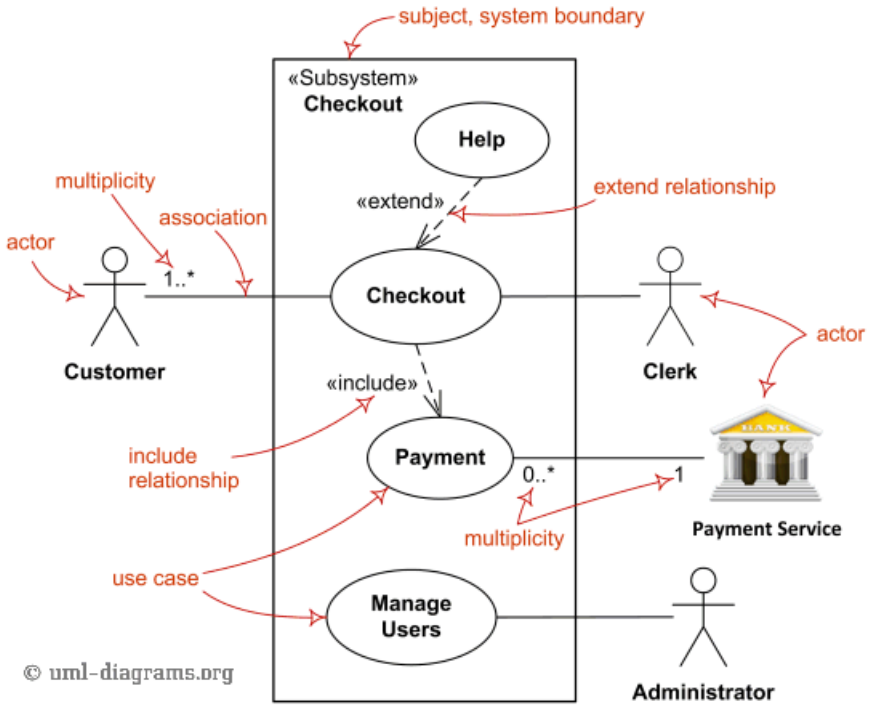
https://en.wikipedia.org/wiki/Use_case_diagram

Example: airport



https://www.uml-diagrams.org

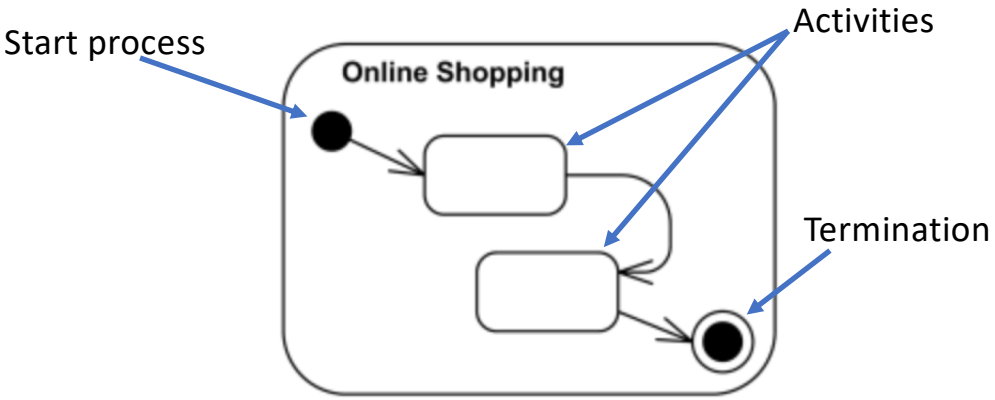
Example: checkout subsystem



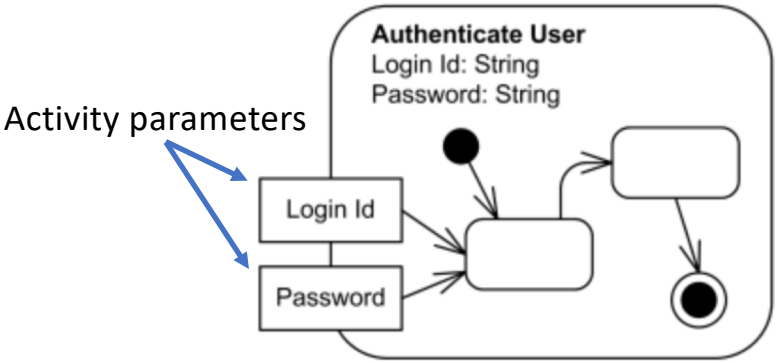
https://www.uml-diagrams.org

Activity diagram

- Visualize steps performed in use case
- Actions invoke activities
- Can have partitioned diagrams
  - Activity groups are partitioned into sections



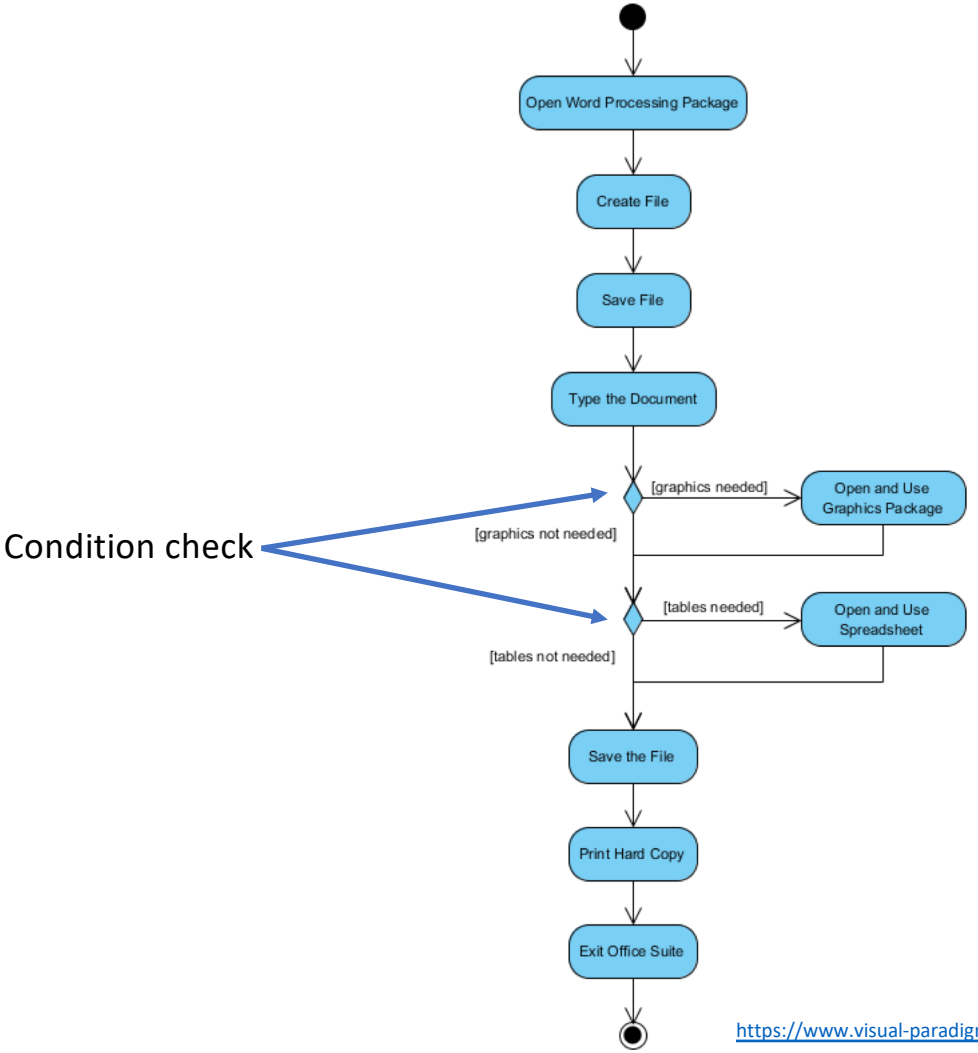
https://www.uml-diagrams.org



https://www.uml-diagrams.org

Example: word processor

Activity diagram



https://www.visual-paradigm.com

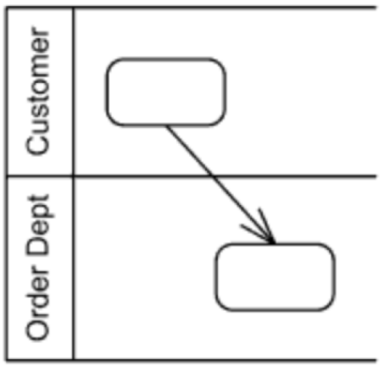
Activity edges

- Can be annotated by the action
- When conditions are involved has to specify what condition branch is satisfied

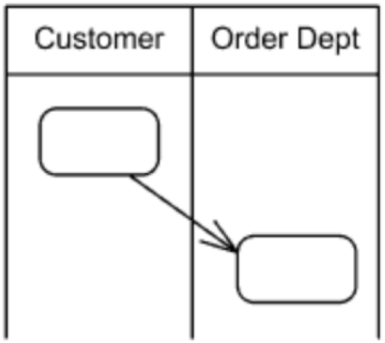
Partitioned activity diagrams

- Partition actions into groups based on a common characteristic
  - E.g. the actor, organizational unit, model, etc.
- We say “this diagram uses swimlanes”
  - Swimlanes can be horizontal or vertical

Horizontal:

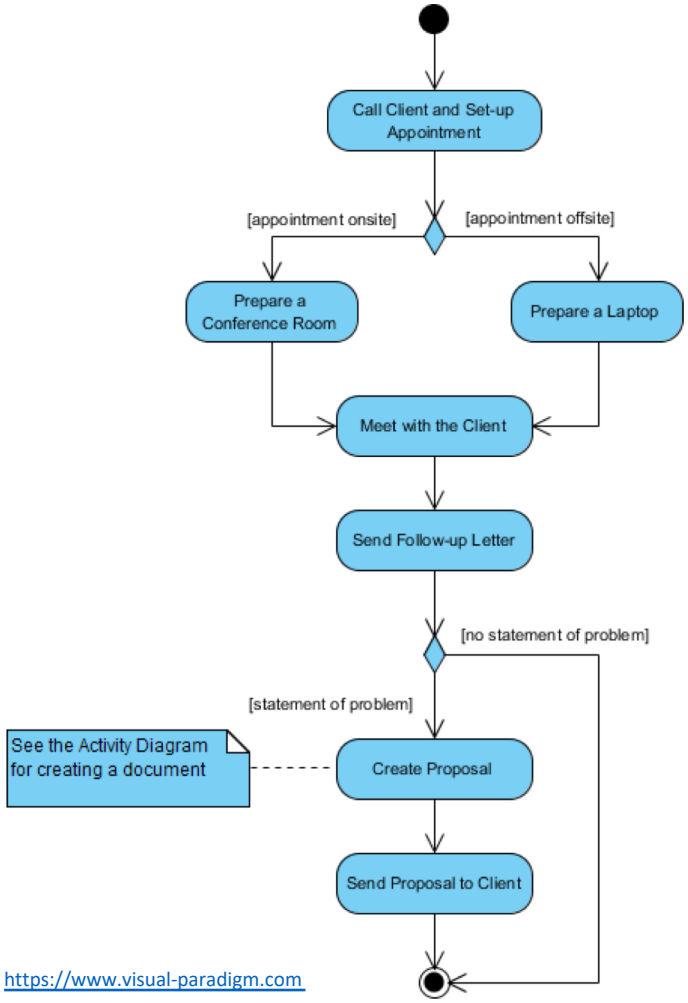


Vertical:



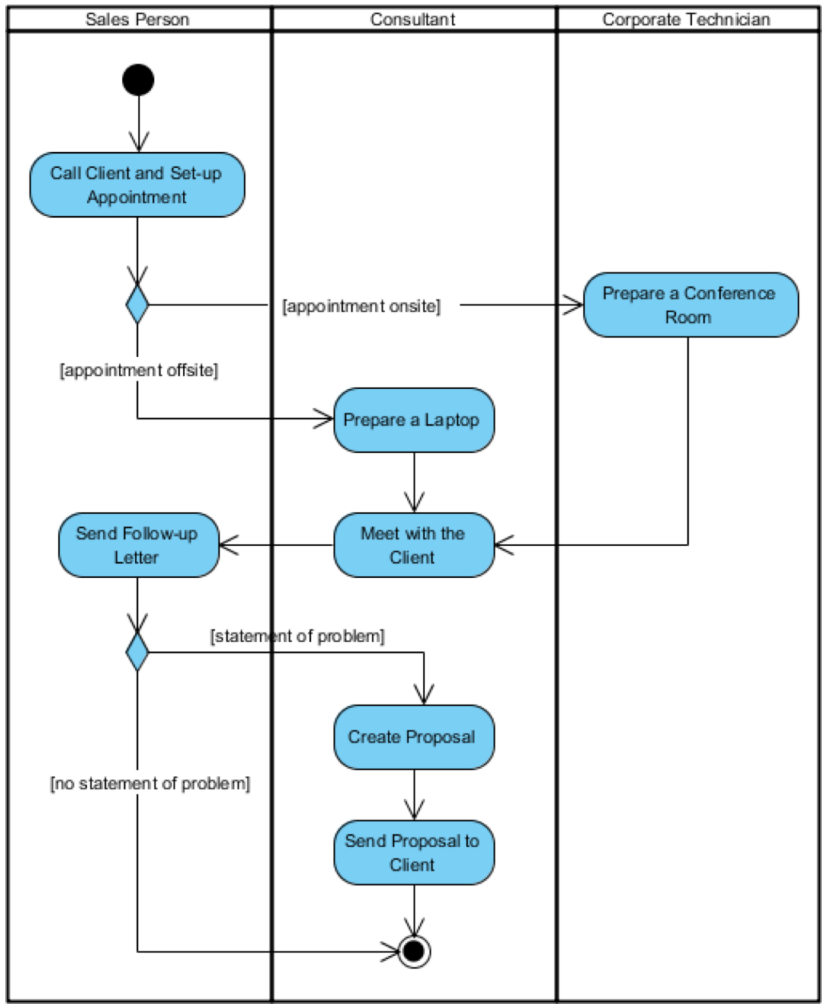
https://www.uml-diagrams.org

Example: new client



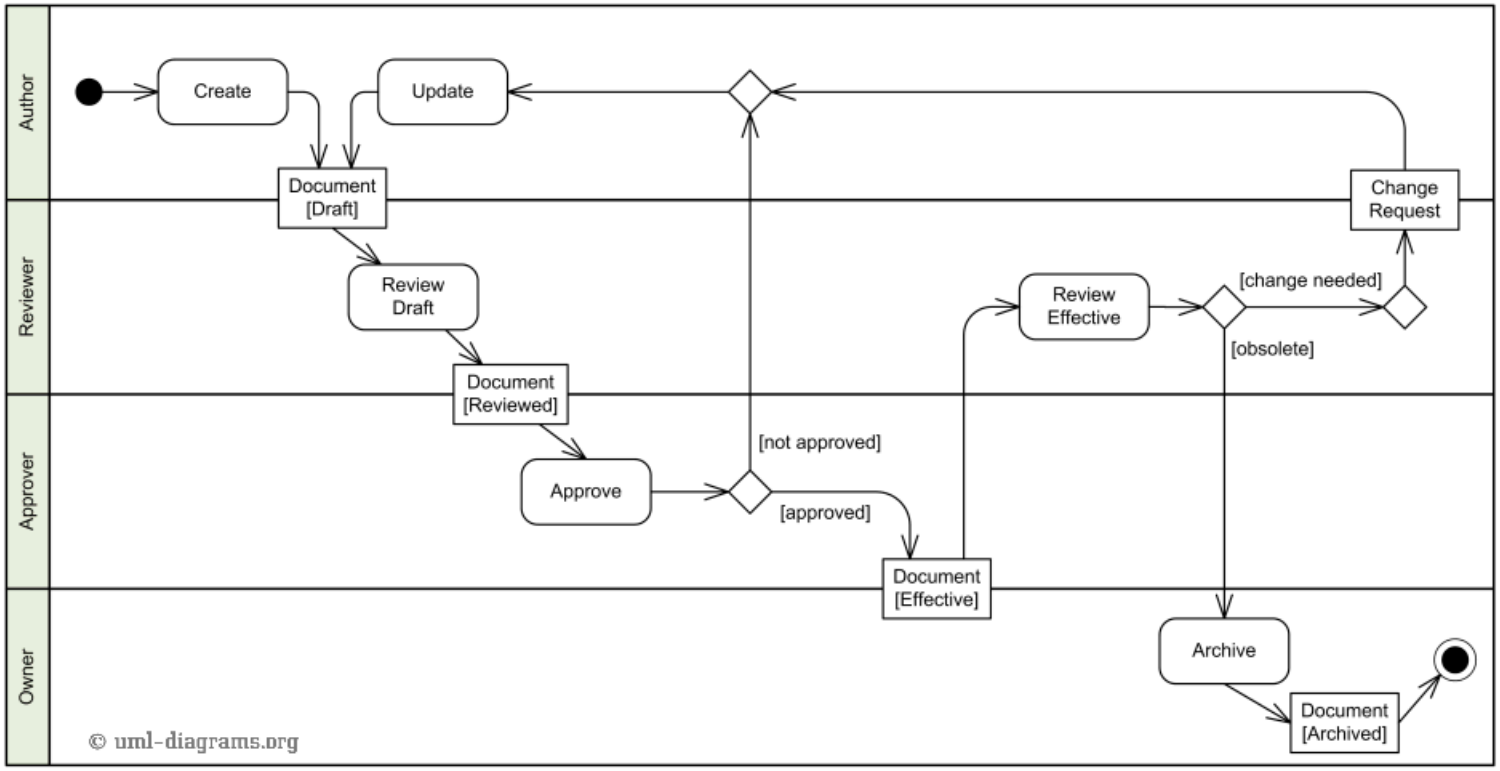
https://www.visual-paradigm.com

Activity diagram



https://www.visual-paradigm.com

Example: document management process



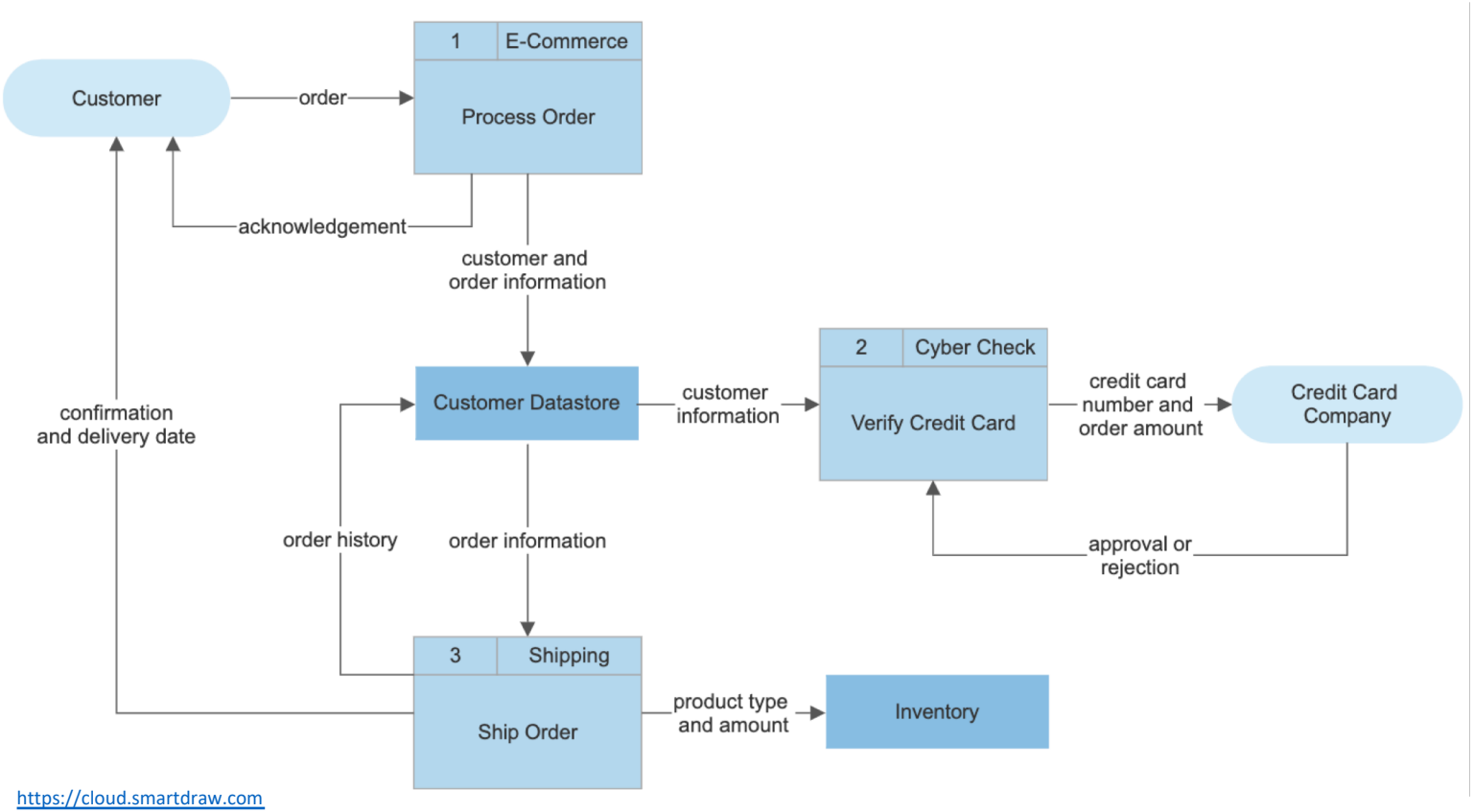
https://www.uml-diagrams.org

Information flow diagram

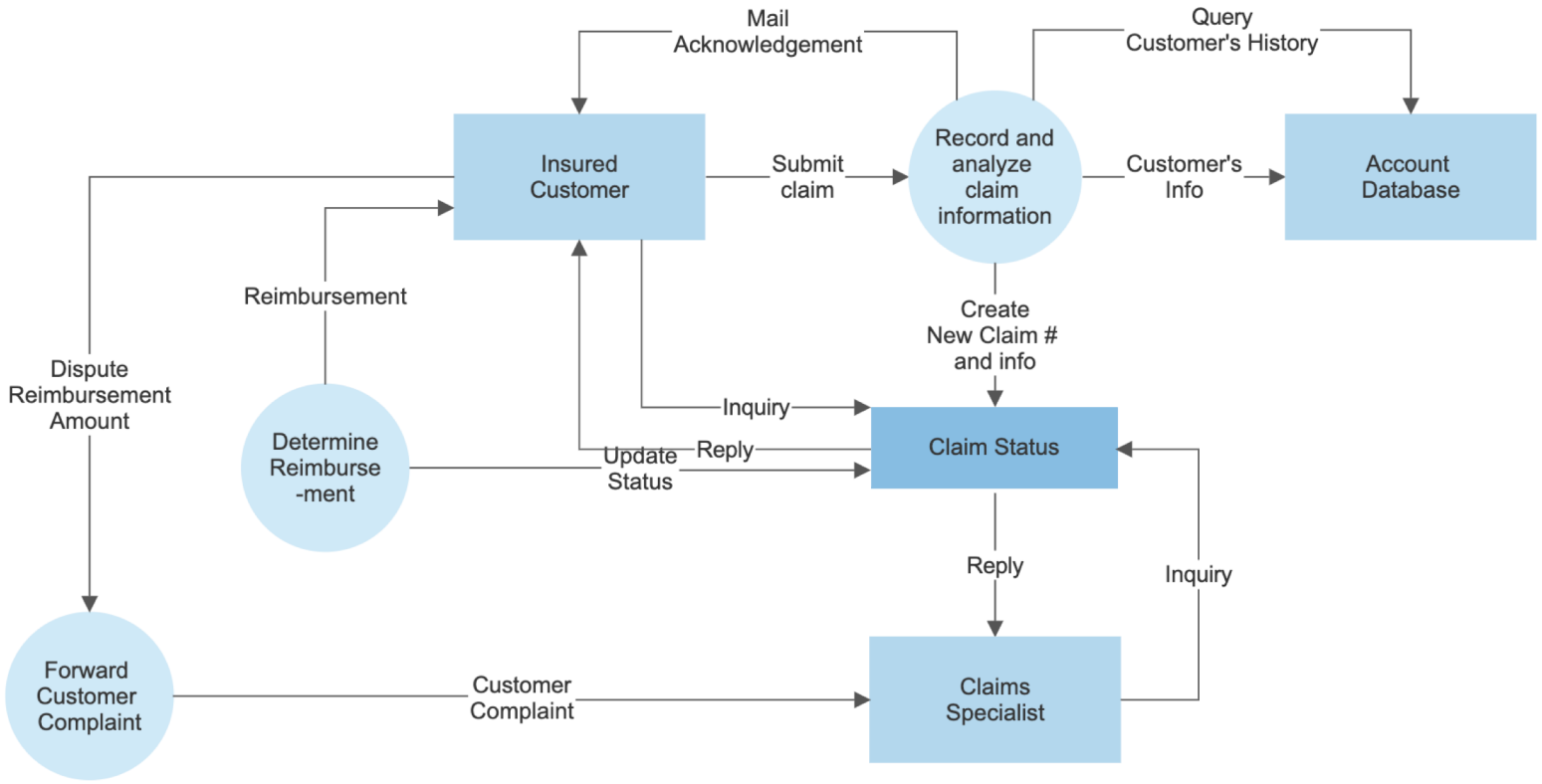
- Describes how data is processed by a system
  - Inputs/outputs
- Can represent the information flow at any abstraction level
  - Context diagram (most generalized)
  - Pseudocode (most detailed)

- External entities
  - Objects outside the system with which the system communicates
  - Can be represented by rectangles or by ovals
- Data flow is represented by arrows directed in the direction of the information flow
- Process transforms incoming data into outgoing data
- Datastores are repositories for data in the system

Example: online order system



Example: insurance claim software



https://cloud.smartdraw.com

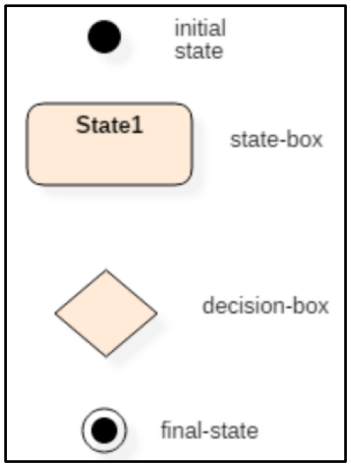
State machine diagram

- Model state changes of objects and events that cause those changes
- Shows states and transitions
  - Transitions represent how states change
  - Transitions are represented by arrows while states are represented by rectangles
  - Similar to activity diagrams, have start and termination nodes

- Two types of diagrams
  - Behavioral state machine
  - Protocol state machine

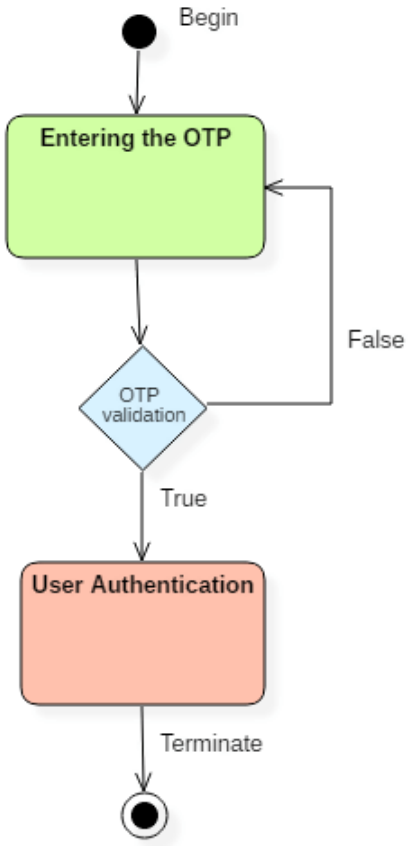
Behavioral state machine

- Models behavior of individual entities in the system
  - E.g. class instances
- Describes behavior in terms of states
- Transition is a change from one state to another state
- Most often are drawn for a single class to show the lifetime behavior of a single object
- Objects may behave differently depending on state
- Events trigger transitions from one state to another



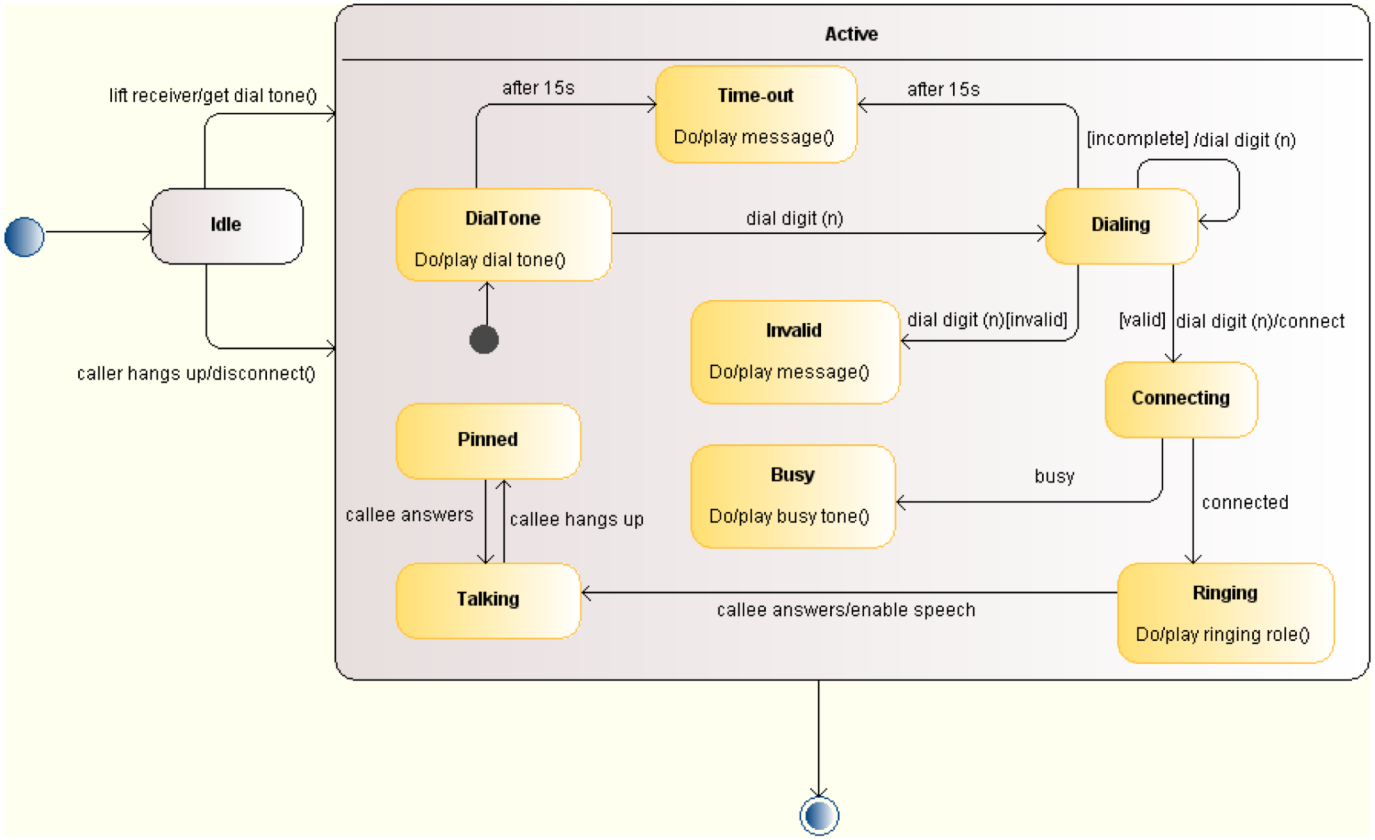
https://www.guru99.com

Example: password authentication



https://www.guru99.com

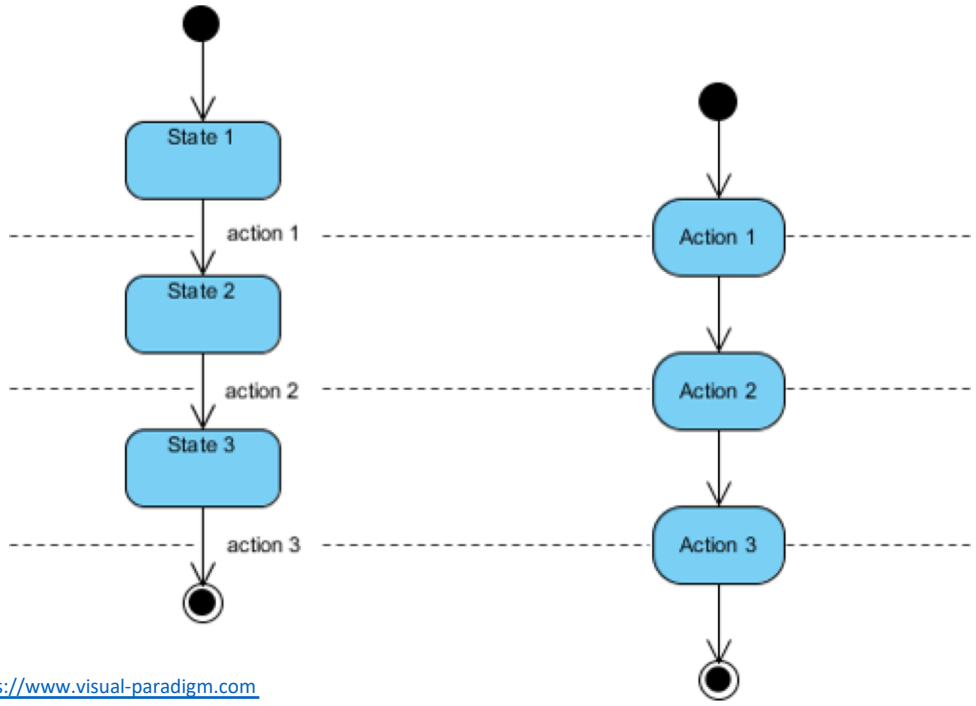
Example: phone call



https://www.modeliosoft.com

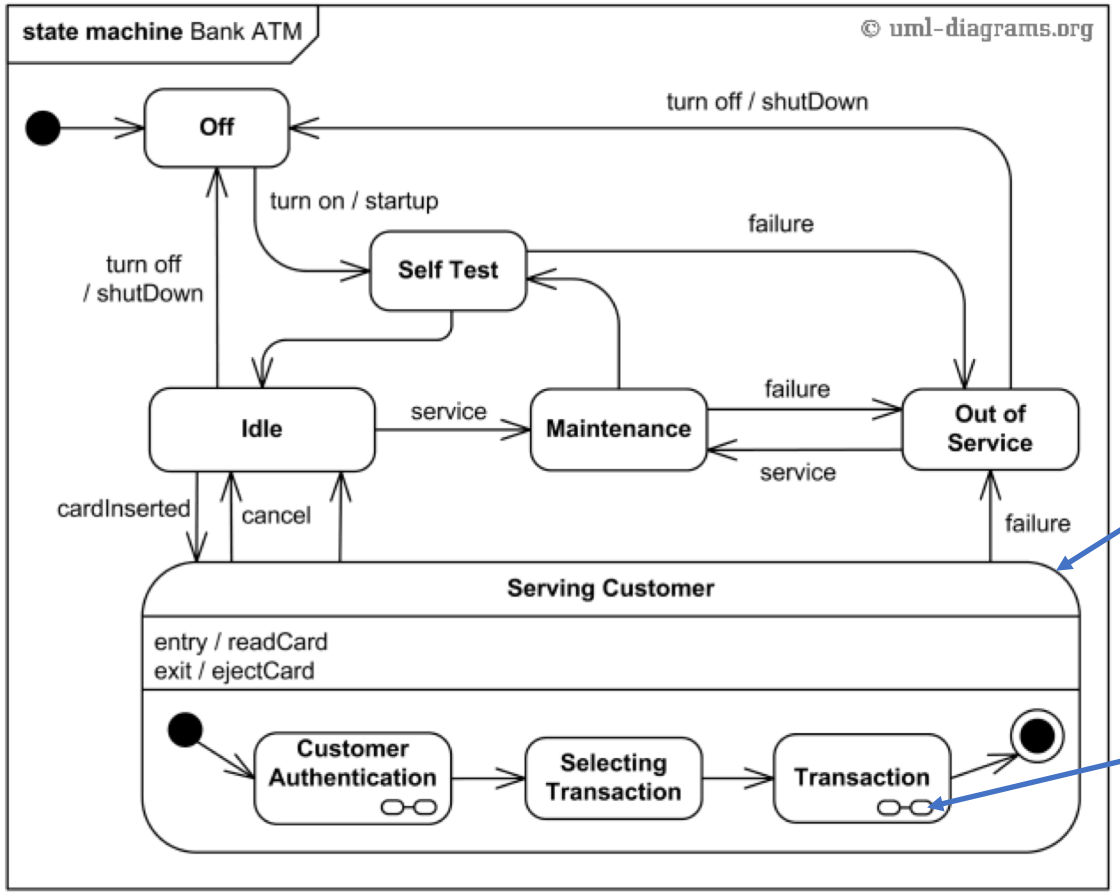
How is behavioral state machine diagram different from activity diagram?

- Activity diagram documents flow of actions/activities while a state machine diagram documents states in the system



https://www.visual-paradigm.com

Example: bank ATM



https://www.uml-diagrams.org

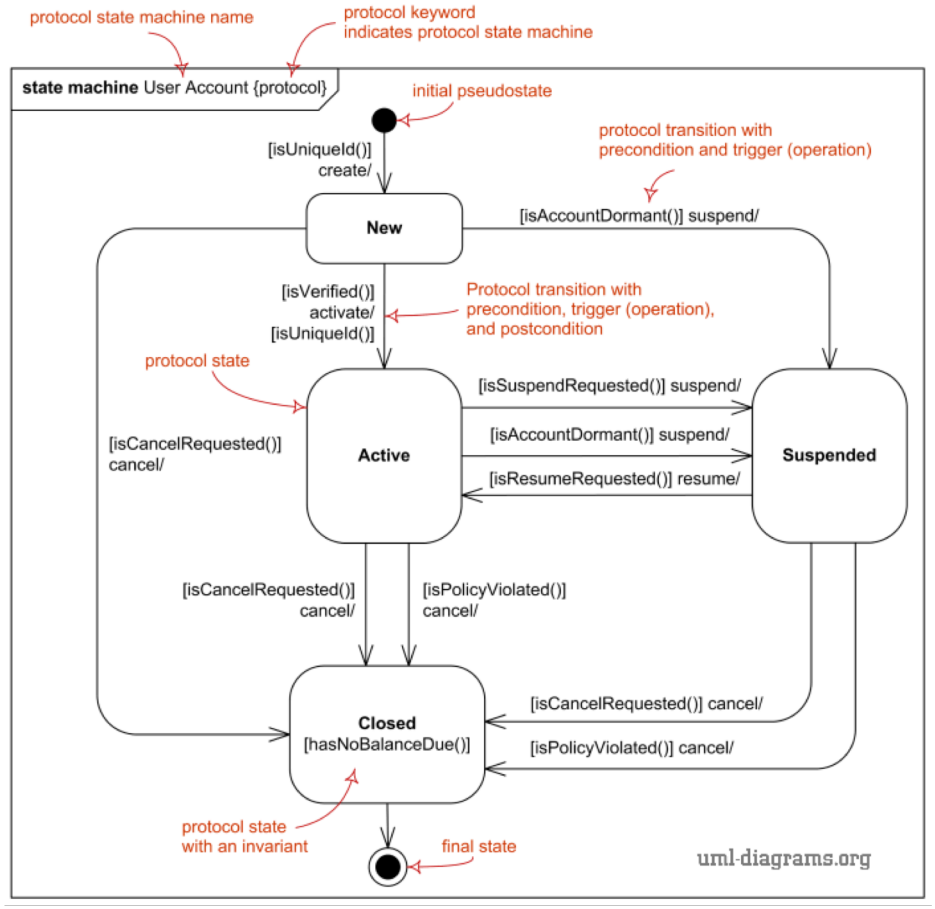
composite state

composite state,
decomposition not shown

Protocol state machine

- Special case of the behavior state machine
- Used to specify usage protocol
  - Valid usage scenarios
  - Protocol is a set of rules
- Shows which operations may be called at which states
- Describes stable states of objects
- Legal sequence of events
- Protocol state machine vs. behavior state machine
  - Protocol state machine is a specification of legal sequences
  - Behavior state machine is a set of behaviors that implement protocol state machine

Example: user account protocol state machine



https://www.uml-diagrams.org

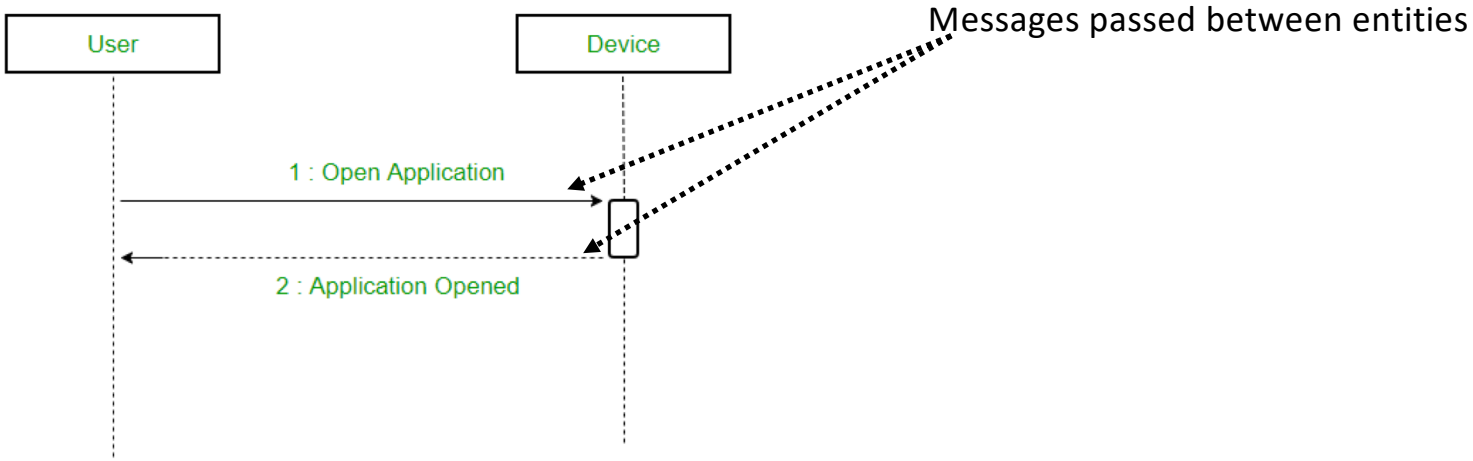
Interaction diagram

- Describe how a group of objects collaborate in a single behavior
- Shows interactive behaviors within the system

- Types of interaction diagrams
  - Sequence
  - Communication
  - Timing
  - Interaction overview

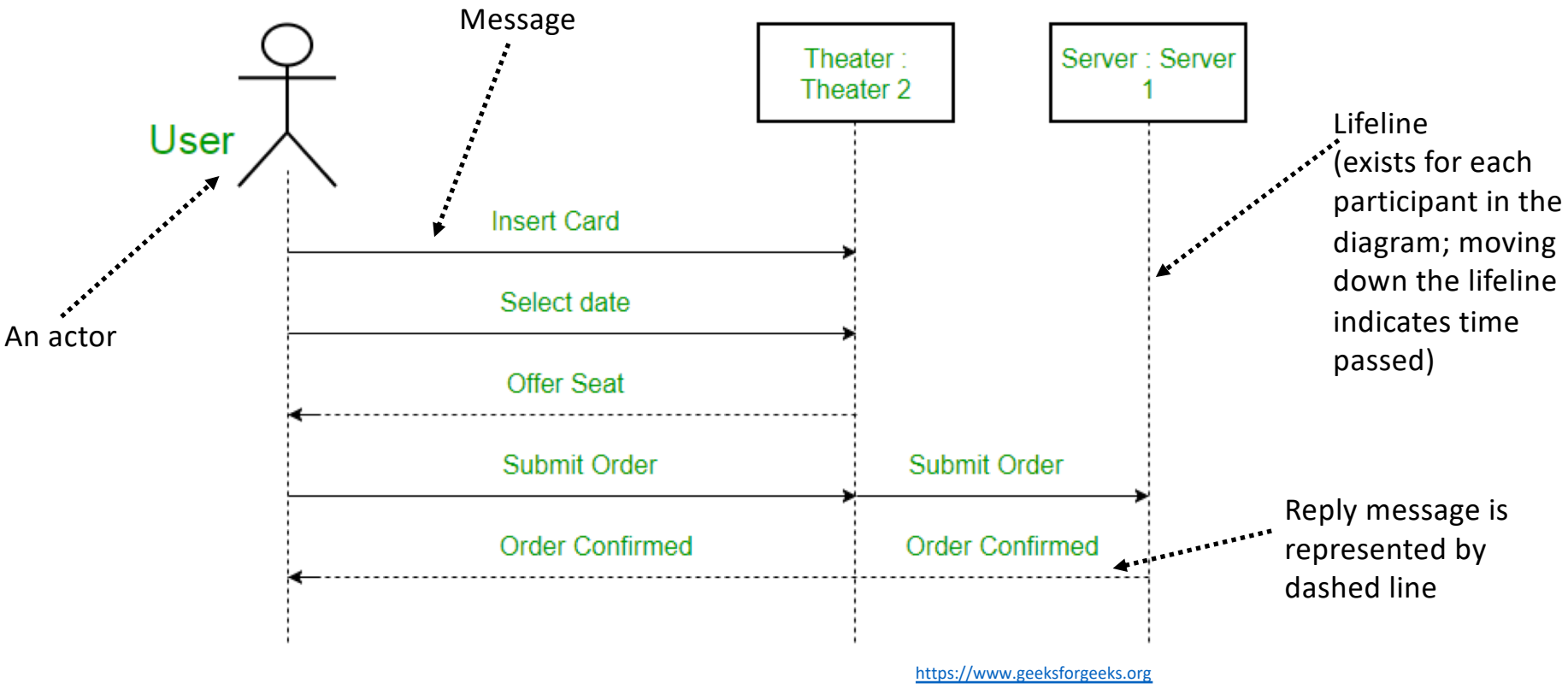
Sequence diagram

- The most commonly used type of interaction diagram
- Describes interactions by showing sequence of messages passed between objects in sequential manner
  - Messages are passed in the order they take place
- Helps document requirements of the system



https://www.geeksforgeeks.org

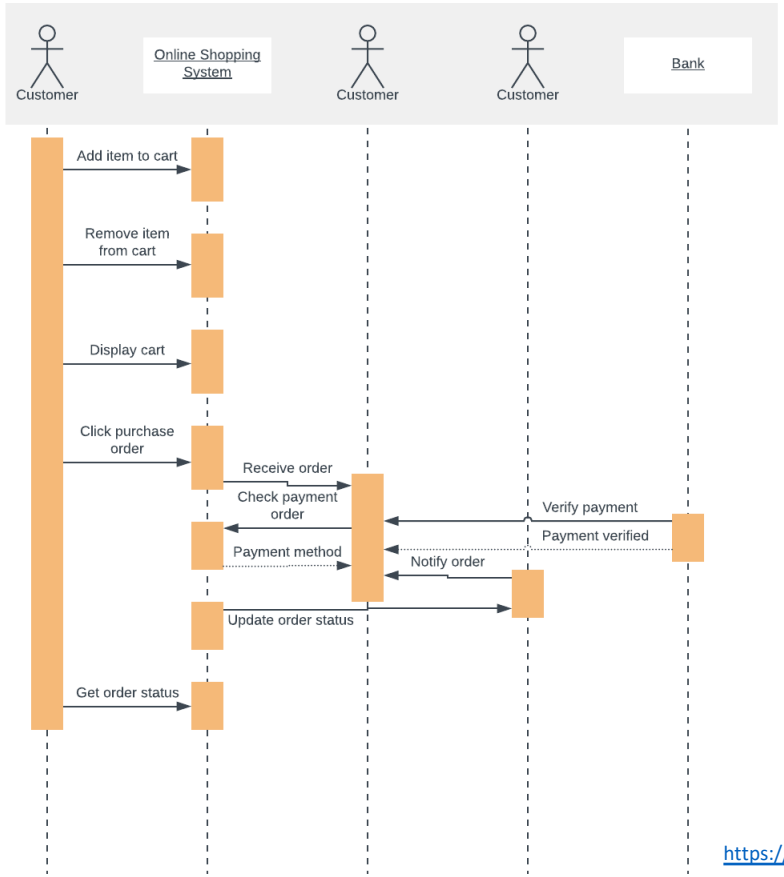
Example: theater seat reservation system



Terminology

- Actor – entity that interacts with the system, external to the system
- Lifeline – each participant in the sequence diagram, usually internal to the system
- Message – communication between objects
  - Synchronous message – waits for a reply from the receiver before interaction can move forward
  - Asynchronous message – does not wait for a reply
  - Create message – message that causes creation of a new object
  - Delete message – message that causes removal of an object
  - Self message – object sends a message to itself
  - Reply message – message sent by the receiver in response
- Gate/guard – models conditions

Example: online shopping



https://www.lucidchart.com

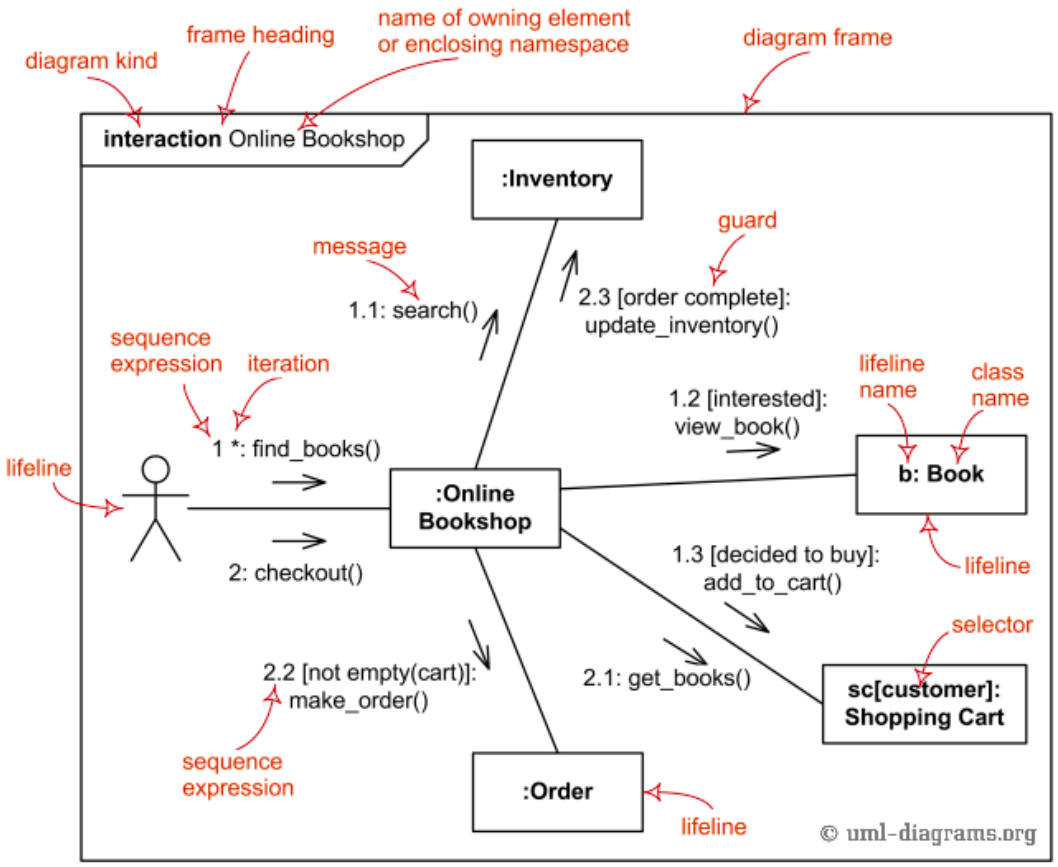
Useful resources

- https://www.youtube.com/watch?v=pCK6prSq8aw

Communication diagram

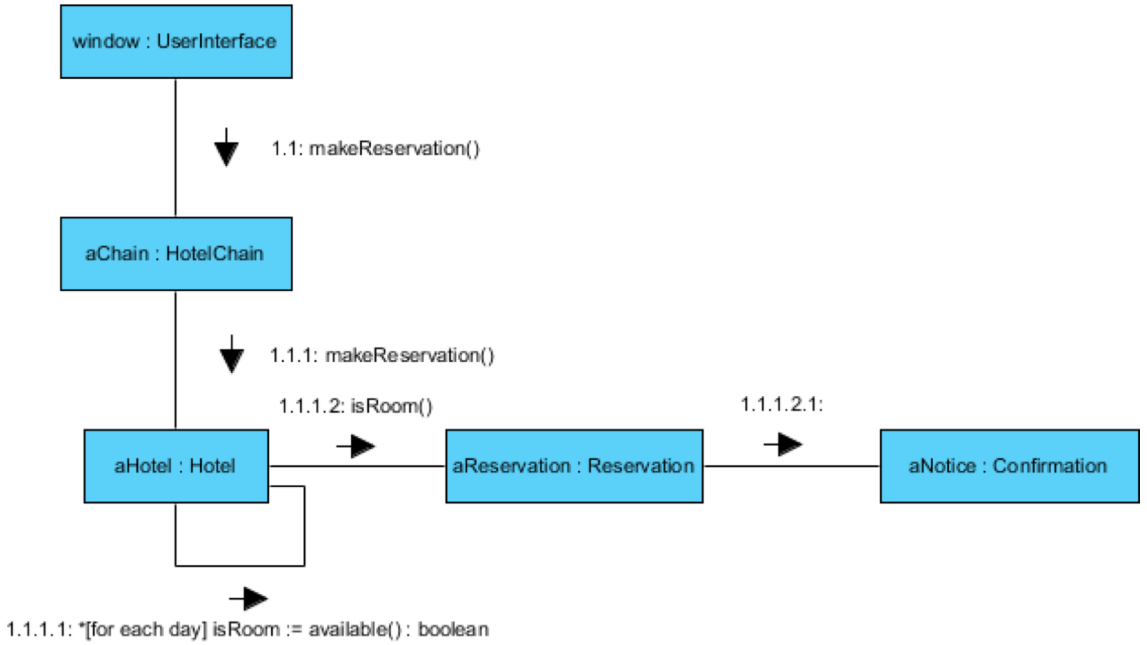
- Describe interactions between objects and part of the system using sequenced messages in free form arrangement
- Show how objects relate to each other
  - Can show collaboration of different objects
- Can show alternative scenarios
- Different from sequence diagrams in that communication diagrams do not show sequential logic
- “Big picture view”; emphasize structural relationships

Example: online bookshop



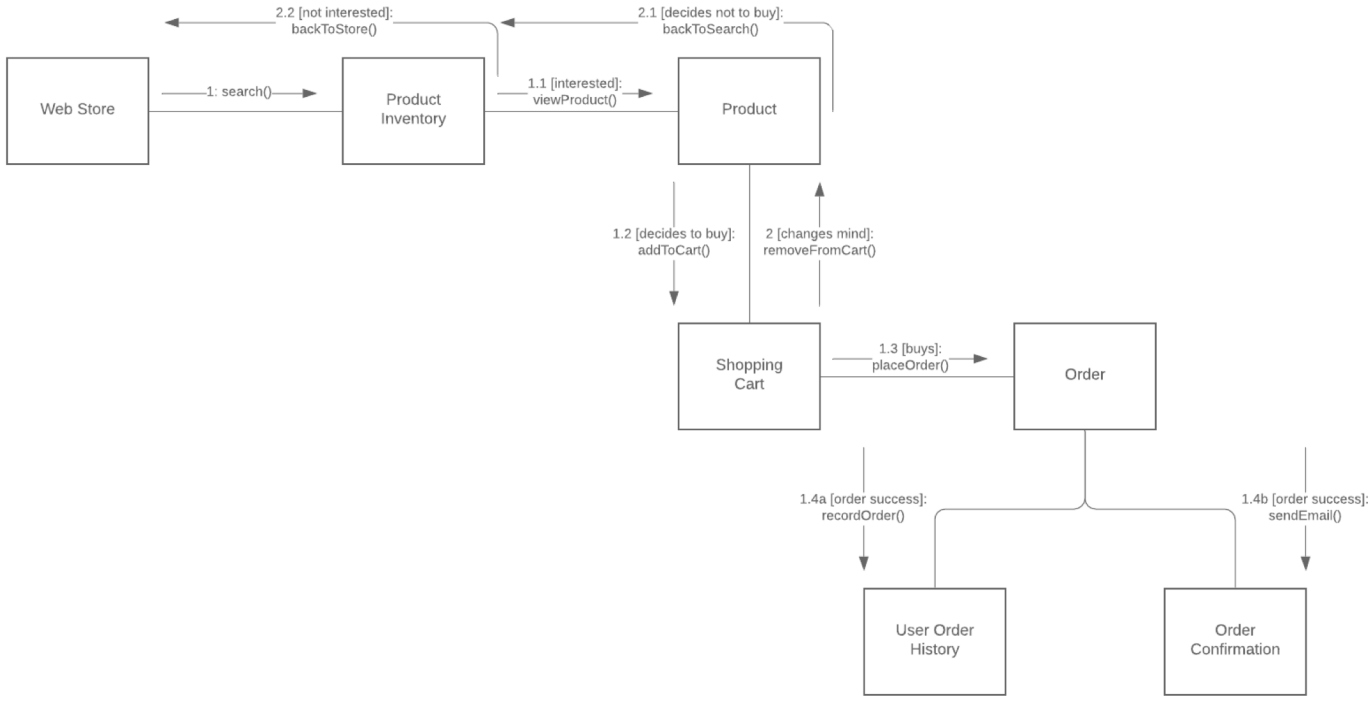
https://www.uml-diagrams.org

Example: hotel reservation



https://www.visual-paradigm.com

Example: web store



https://www.lucidchart.com

Sequence or communication diagram?

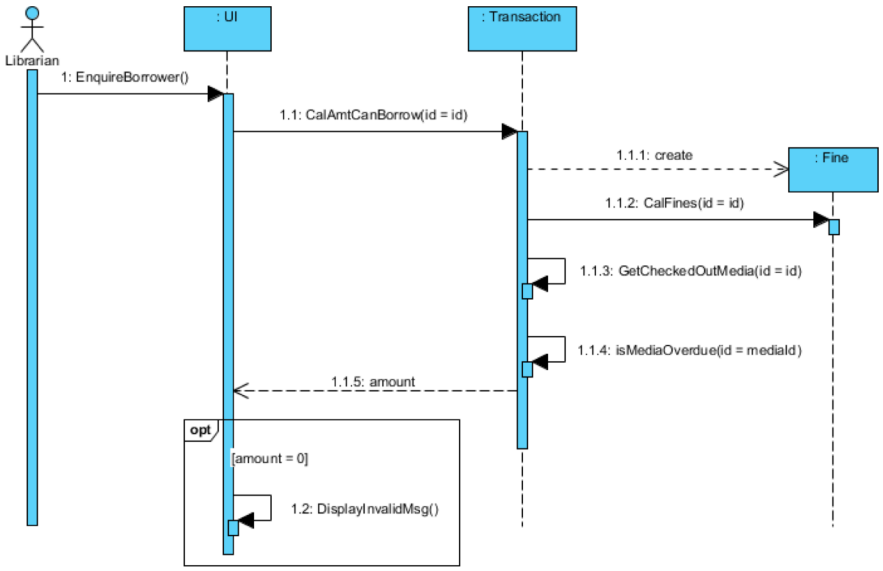
- No right choice; a matter of preference

- Sequence diagram advantages
  - Flow of interactions between objects
  - Read from top to bottom
  - Easy to understand, easy to interpret
  - No need to read sequence numbers to figure out the order

- Communication diagram advantages
  - Unstructured; like a sketch
  - Space efficient
    - No particular vertical or horizontal orientation
  - Easier to modify

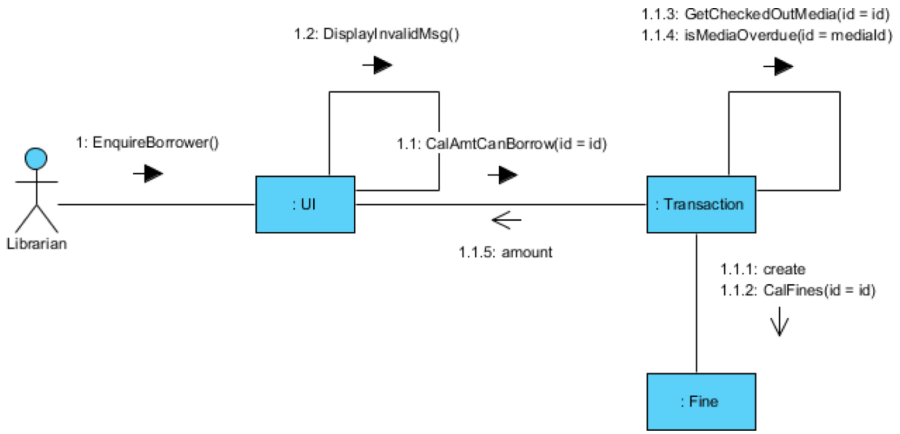
Example: library item overdue

Sequence diagram:



https://www.visual-paradigm.com

Communication diagram:

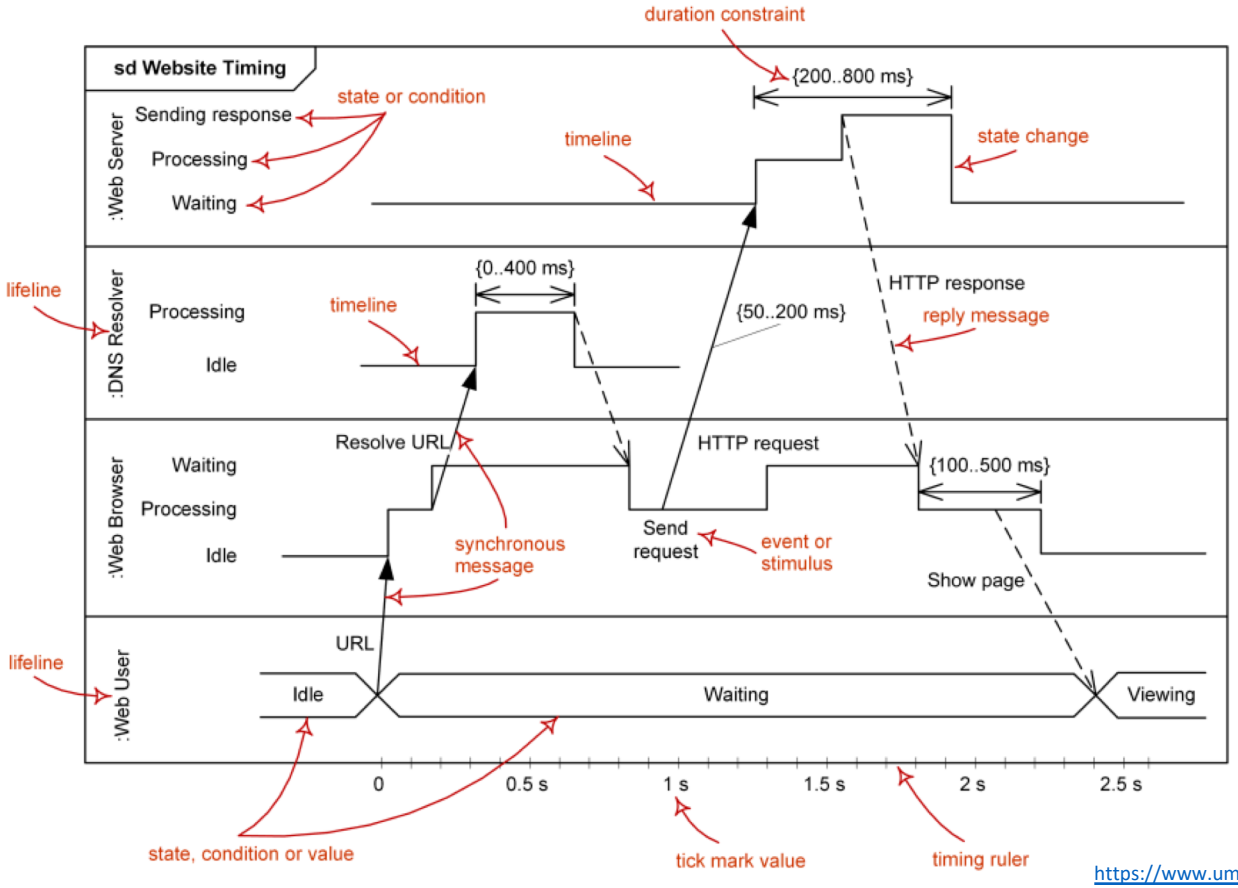


https://www.visual-paradigm.com

Timing diagram

- Purpose is to describe time of events causing changes in conditions of lifelines
- How objects interact with each other within a timeframe

Example: website timing

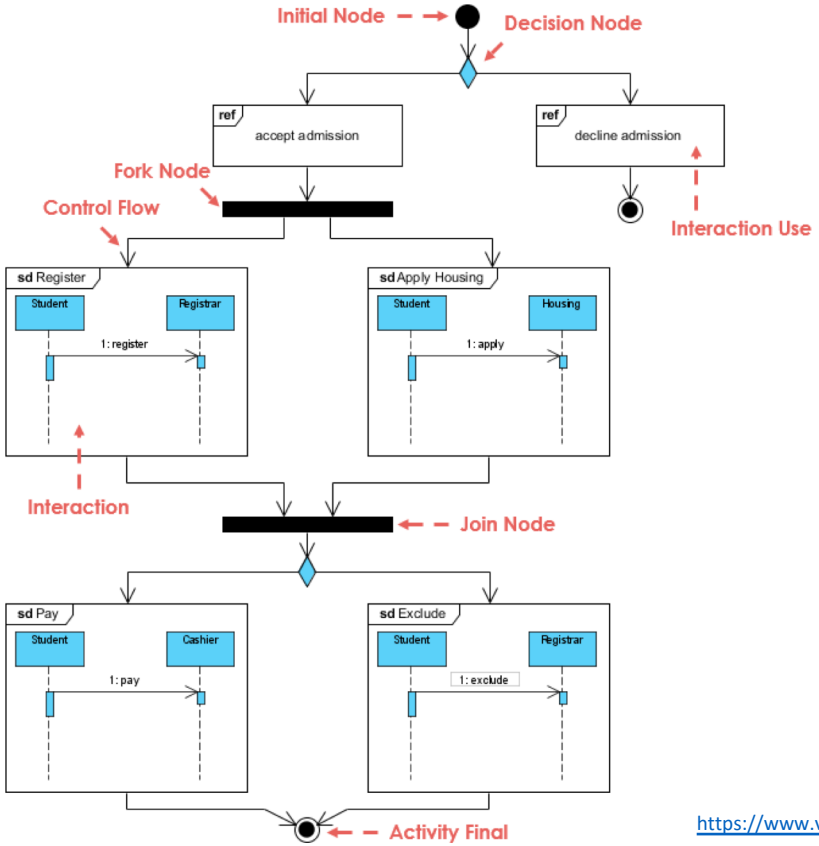


https://www.uml-diagrams.org

Interaction overview diagram

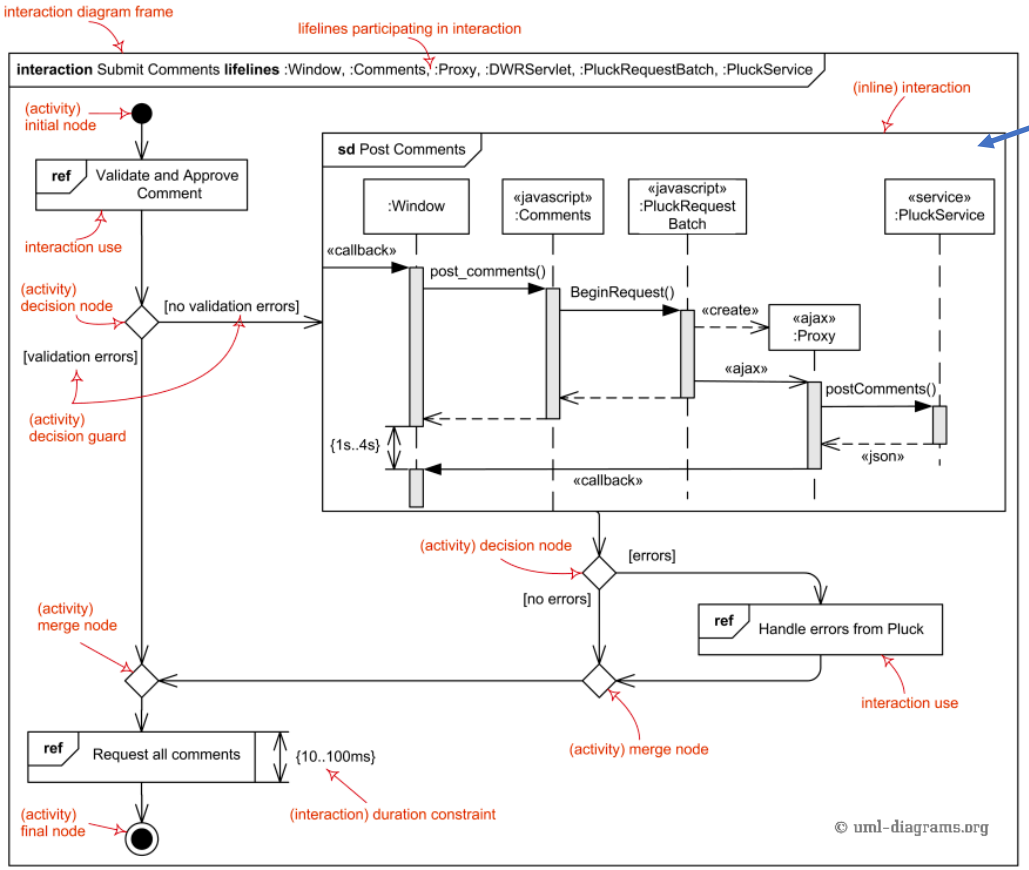
- Describe the flow of control between interacting nodes
  - Nodes represent activity or decision points
- A combination of activity and sequence diagram
- Elements of activity diagram that are used in interaction overview diagram:
  - Initial and final nodes
  - Decision node
  - Merge, fork, and join nodes
- Elements of interaction diagram that are used in interaction overview diagram:
  - Interaction
  - Interaction use
  - Duration and time constraints

Example: student admission



https://www.visual-paradigm.com

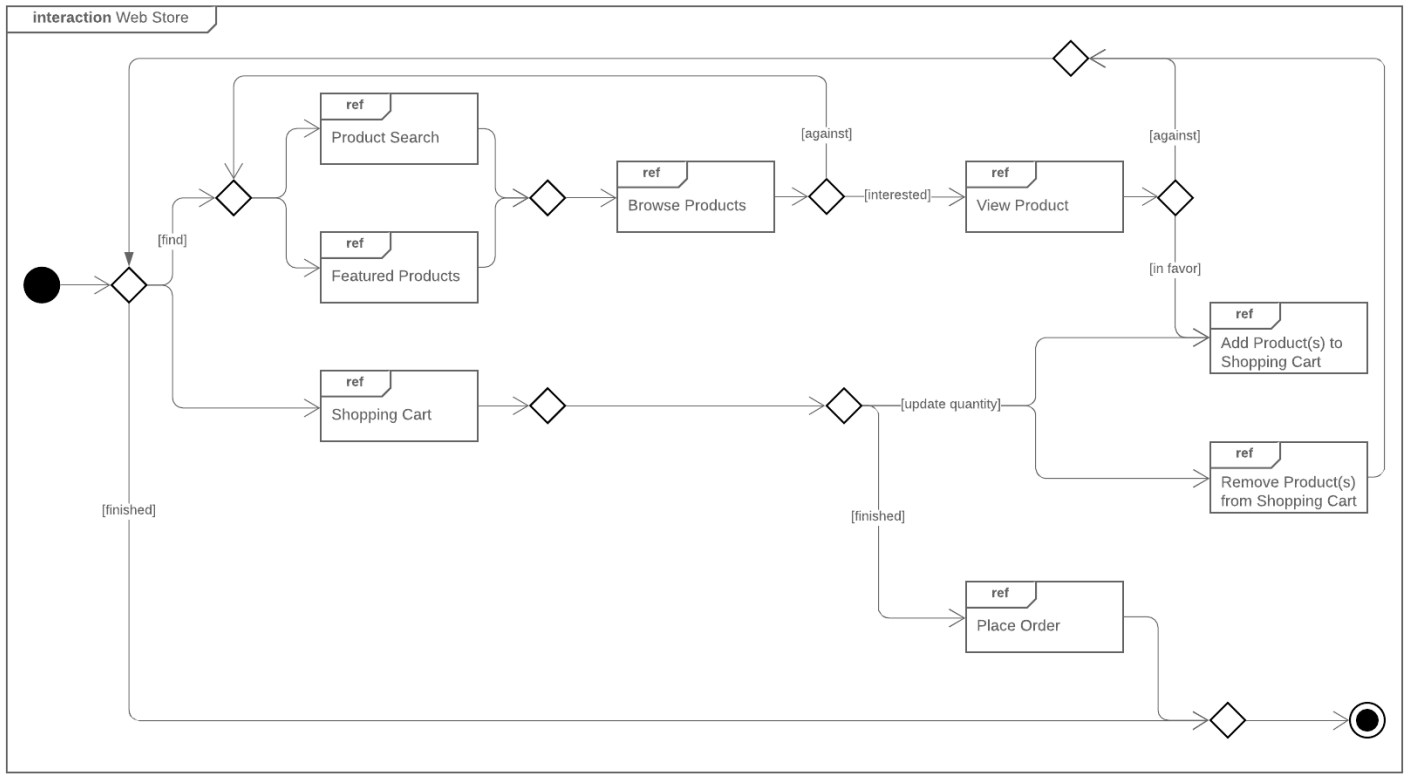
Example: posting online comment



Sequence sub-diagram

https://www.uml-diagrams.org

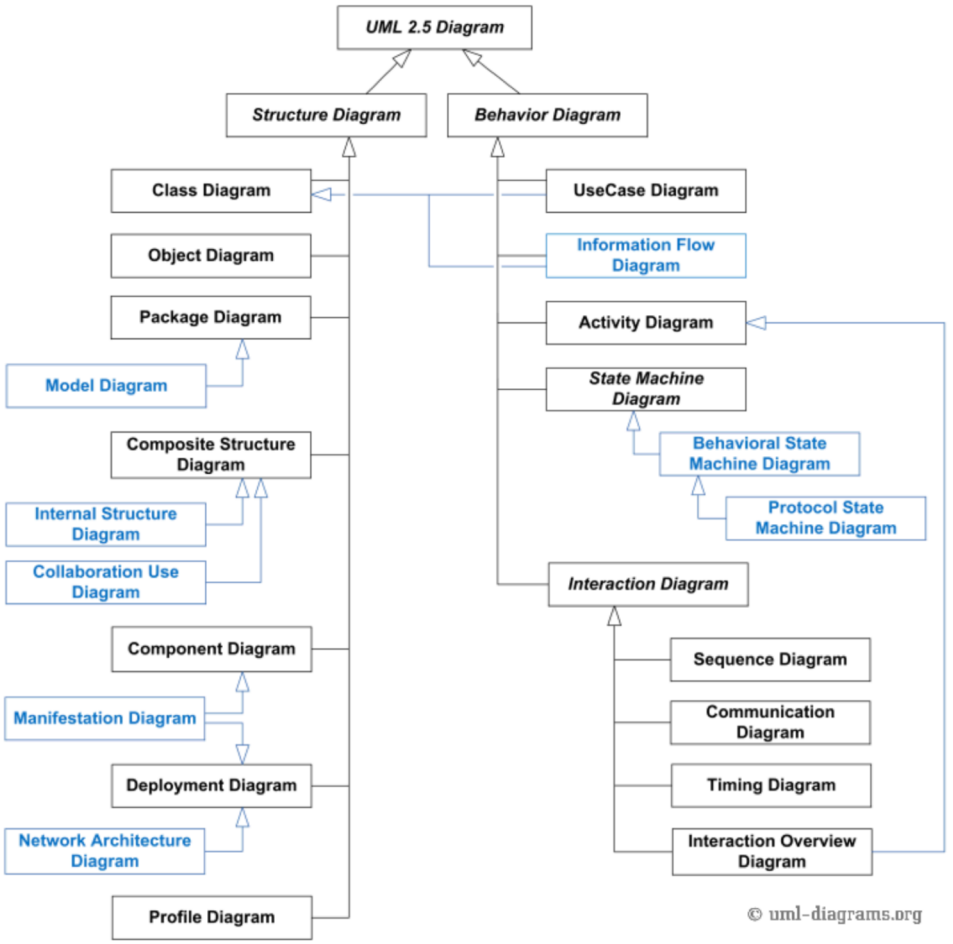
Example: web store



https://www.lucidchart.com

To recap

- Structure diagrams
  - Static aspects of the system
- Behavior diagrams
  - Dynamic aspects and behavior of the system



https://www.uml-diagrams.org

Concluding remarks

- Primary goal of UML is to help design, document and understand the system
- It is a tool to help you deliver a good product