UNIFIED
MODELING
LANGUAGE™

UML

Diego Pacheco

About me...

- □ Cat's Father
- □ Principal Software Architect
- □ Agile Coach
- □ SOA/Microservices Expert
- □ DevOps Practitioner
- □ Speaker
- □ Author

diegopacheco

@diego_pacheco

http://diego-pacheco.blogspot.com.br/

Diego Pacheco

Building Applications with Scala

Write modern, scalable, and reactive applications with the power of Scala

Packt>

Diego Pacheco

Building Effective Microservices

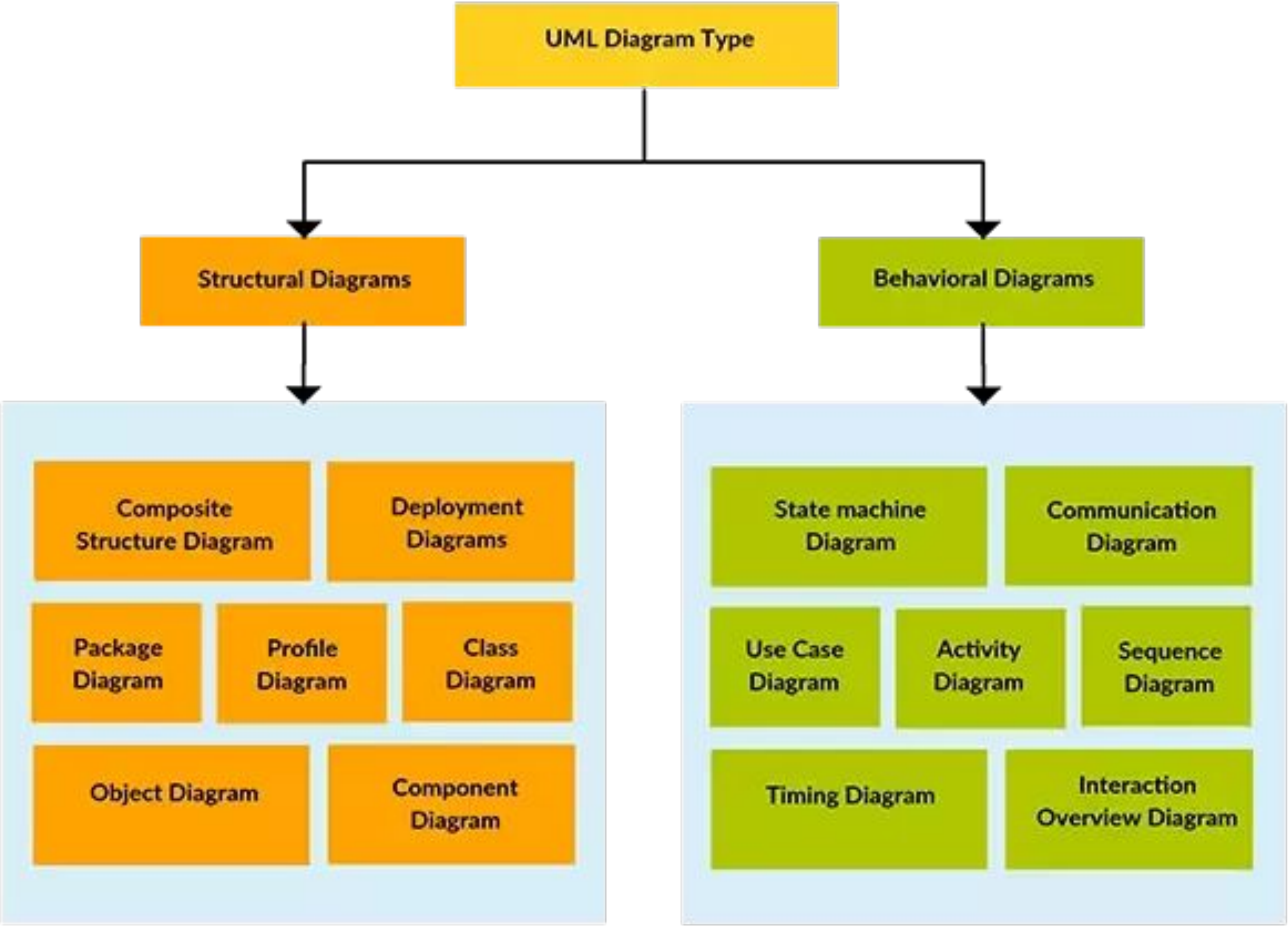
Explore microservices and their implementation hands-on

Packt>

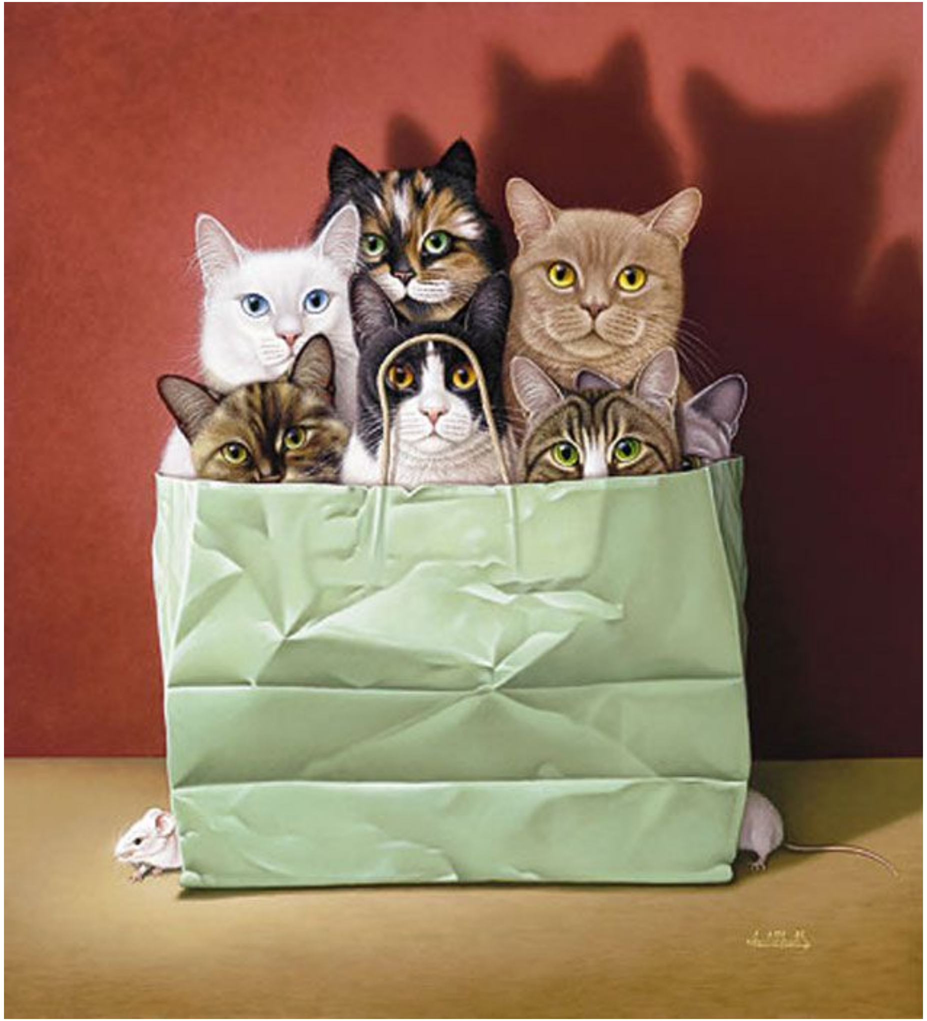
https://diegopacheco.github.io/

UML

- □ 1994
- □ Unified Modeling Language
    - □ Grady Booch, OMT (James Rumbaugh)
    - □ OOSE (Ivar Jacobson)
- □ UML is not a method / methodology.
- □ Good for concurrent / distributed systems modeling.


UML Diagram Type
Structural Diagrams
Behavioral Diagrams
Composite Structure Diagram
Deployment Diagrams
Package Diagram
Profile Diagram
Class Diagram
Object Diagram
Component Diagram
State machine Diagram
Communication Diagram
Use Case Diagram
Activity Diagram
Sequence Diagram
Timing Diagram
Interaction Overview Diagram

MAKE ME THINK

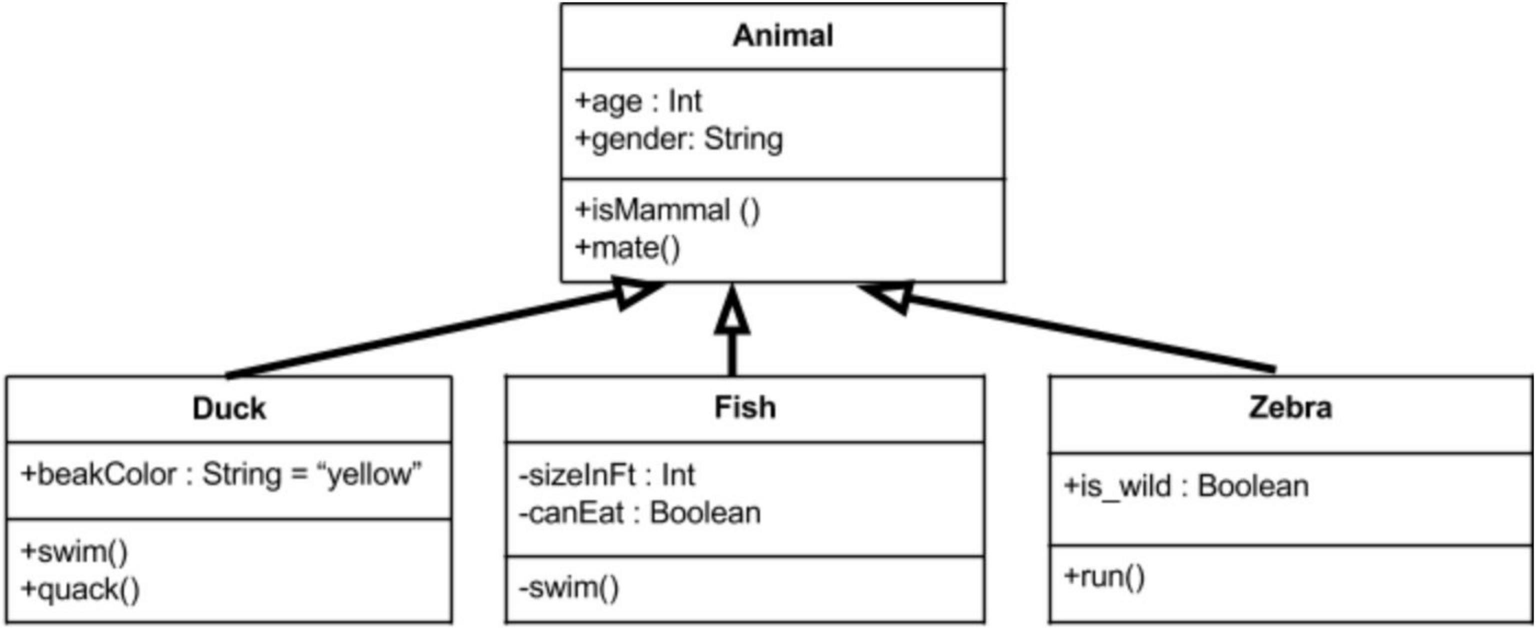


- ❑ I can do anything in code. Should I?
- ❑ UML does not guarantee QUALITY.
- ❑ ASK before you SHOOT.
- ❑ THINK before you DO.
- ❑ UML is just a WAY to structure your thinking you can do without. But you should be THINKING right ???

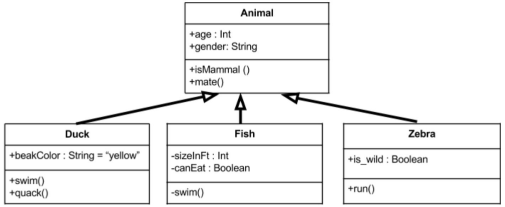
UML is a way to a MEAN...

- ❑ BEFORE we CODE
- ❑ Where is the “NORTH”
- ❑ Where we should PUT things(code)?
- ❑ Why we should PUT or not in ONE place or ANOTHER
- ❑ What it means?
  - ❑ Side effects? Dependencies? Benefits? Issues?
  - ❑ How easy is to extend? Maintain? Grow? Refactor? Test?
  - ❑ Aggregate? Split? Generic? Specific?

Class Diagram



Class Diagram



Animal

+age : Int
+gender: String

+isMammal ()
+mate()

Duck

+beakColor : String = “yellow”

+swim()
+quack()

Fish

-sizeInFt : Int
-canEat : Boolean

-swim()

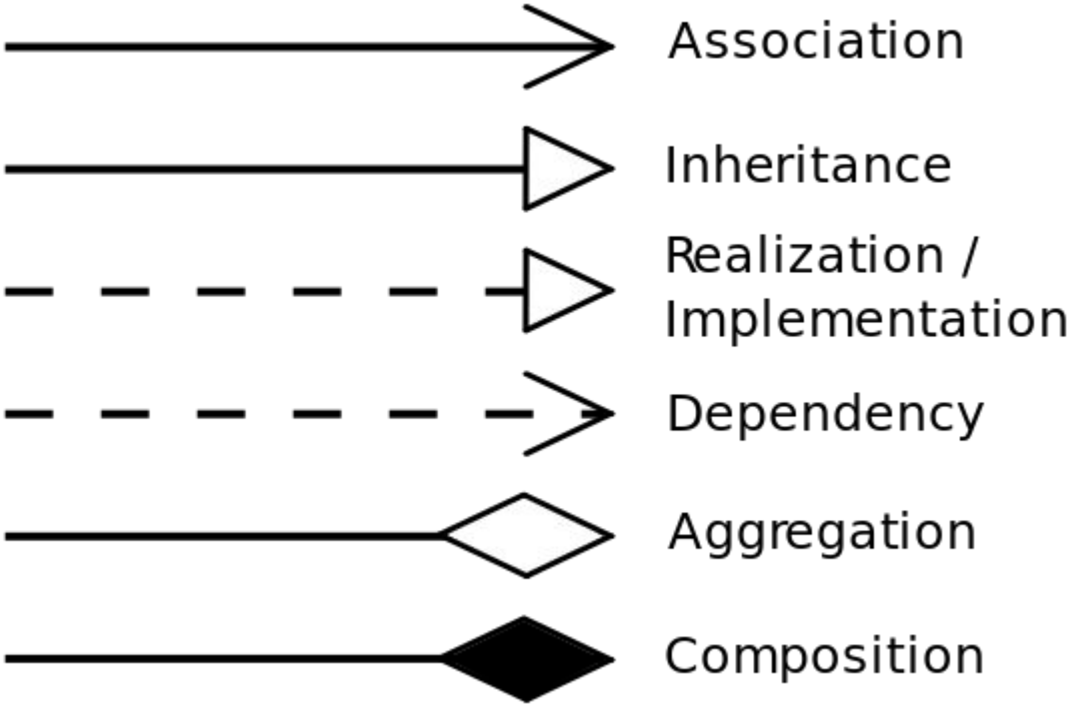
Zebra

+is_wild : Boolean

+run()

- □ Describe static STRUCTURE
- □ General Conceptual modeling
- □ Main Elements & Interactions
- □ Relationship:
  - □ Dependency: uni
  - □ Association: uni, multi, several,
  - □ Aggregation / Composition: specific, 2.
  - □ Inheritance

Class Diagram



Association

Inheritance

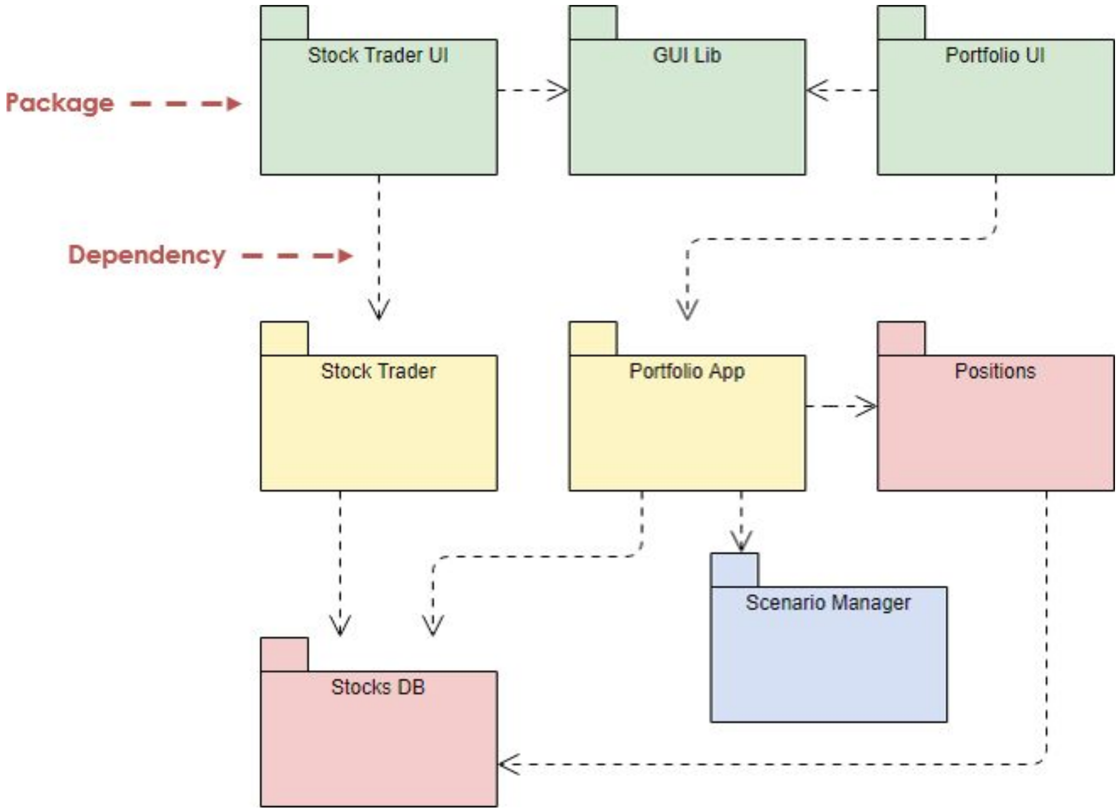
Realization /
Implementation

Dependency

Aggregation

Composition

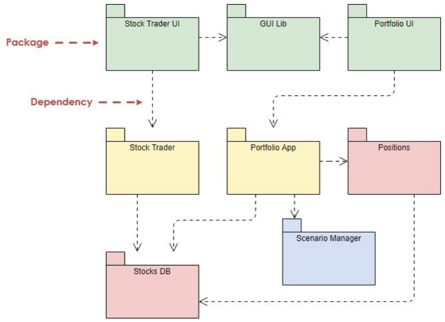
Package Diagram



Package

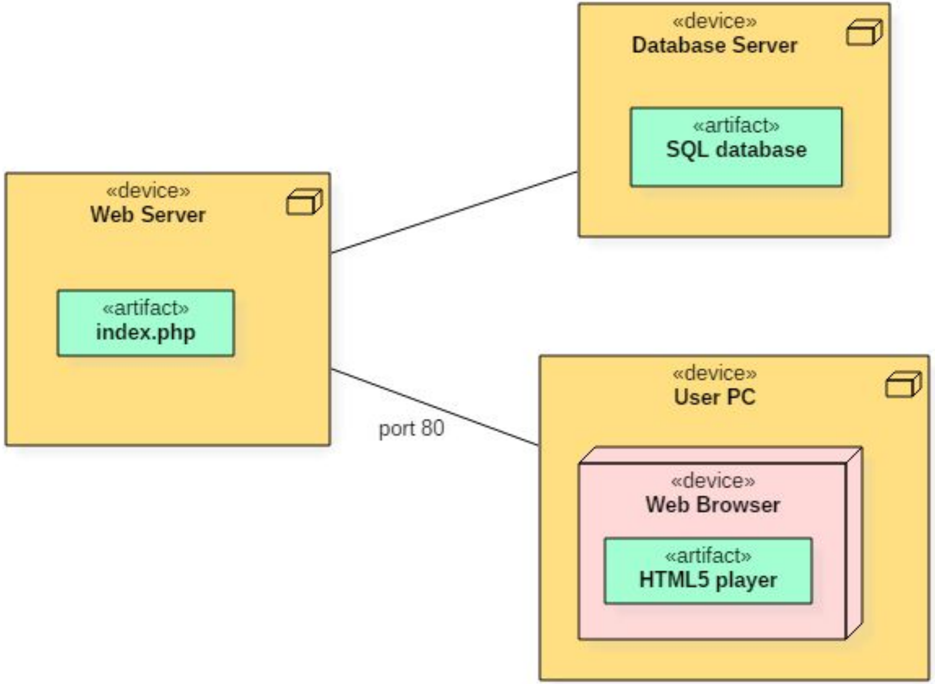
Dependency

Package Diagram

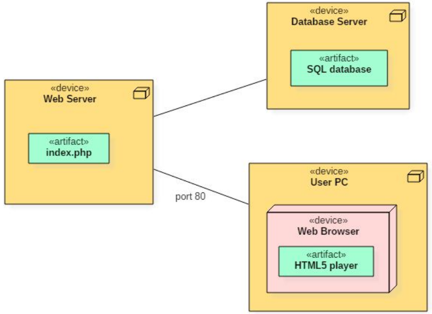


- ❑ Packages == modules
- ❑ Dependencies between packages
- ❑ Optionaly: Important classes in a module.
- ❑ Great for: Layers & Responsibilities.
- ❑ Constructs:
  - ❑ Import (dependency)
  - ❑ Merge Constructs (combine)

Deploy Diagram

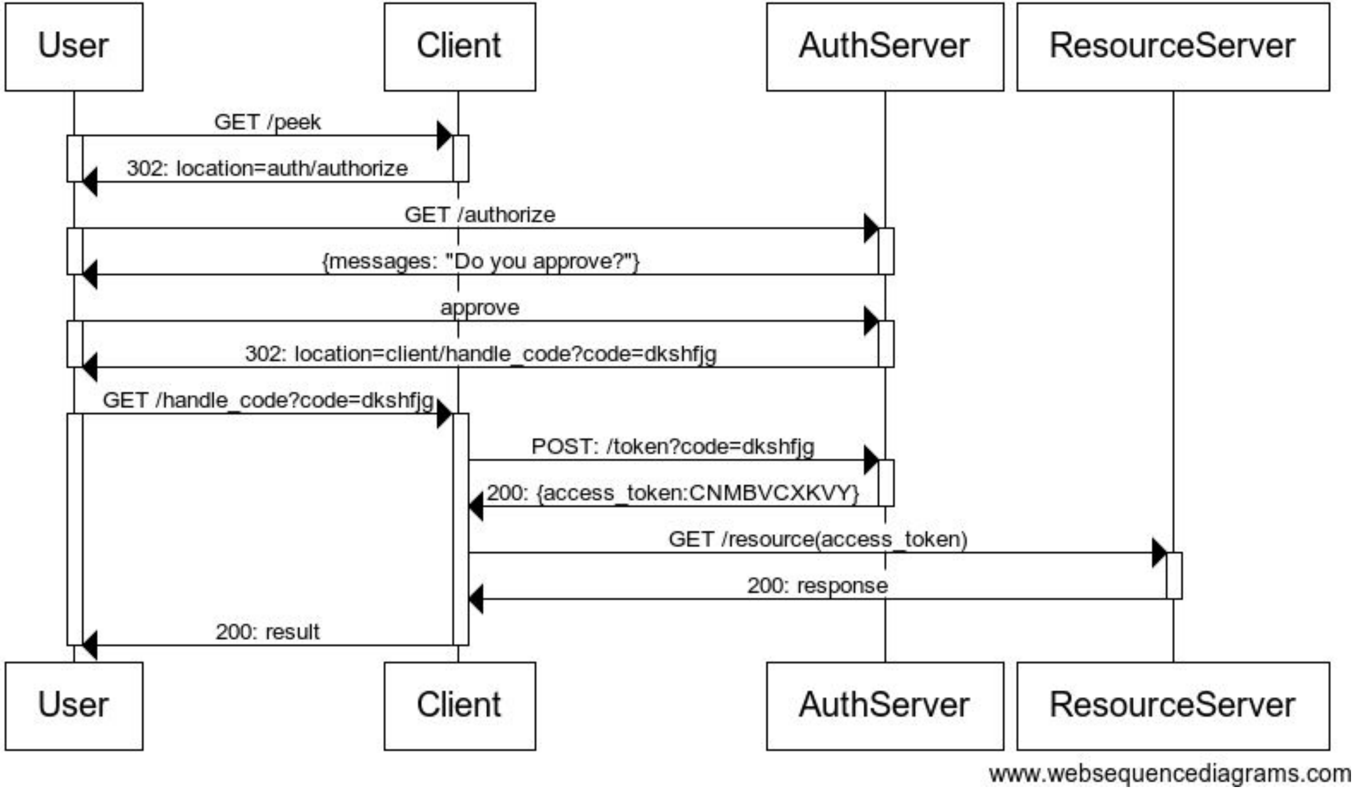


Deploy Diagram

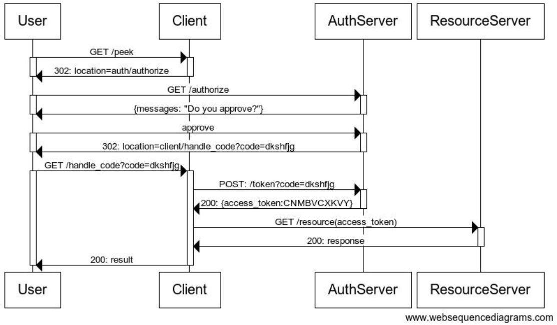


- ❏ Part of Structure Diagrams.
- ❏ "Physical" Deployment.
- ❏ Hardware & Cloud Infrastructure
- ❏ Correlation and dependence

Sequence Diagram


User
Client
AuthServer
ResourceServer
GET /peek
302: location=auth/authorize
GET /authorize
{messages: "Do you approve?"}
approve
302: location=client/handle_code?code=dkshfjg
GET /handle_code?code=dkshfjg
POST: /token?code=dkshfjg
200: {access_token:CNMBVCXKVY}
GET /resource(access_token)
200: response
200: result
User
Client
AuthServer
ResourceServer
www.websequencediagrams.com

Sequence Diagram


User
Client
AuthServer
ResourceServer
GET /peek
302: location=auth/authorize
GET /authorize
{messages: "Do you approve?"}
approve
302: location=client/handle_code?code=dkshfjg
GET /handle_code?code=dkshfjg
POST: /token?code=dkshfjg
200: {access_token:CNMBVCXKVY}
GET /resource(access_token)
200: response
200: result
User
Client
AuthServer
ResourceServer
www.websequencediagrams.com

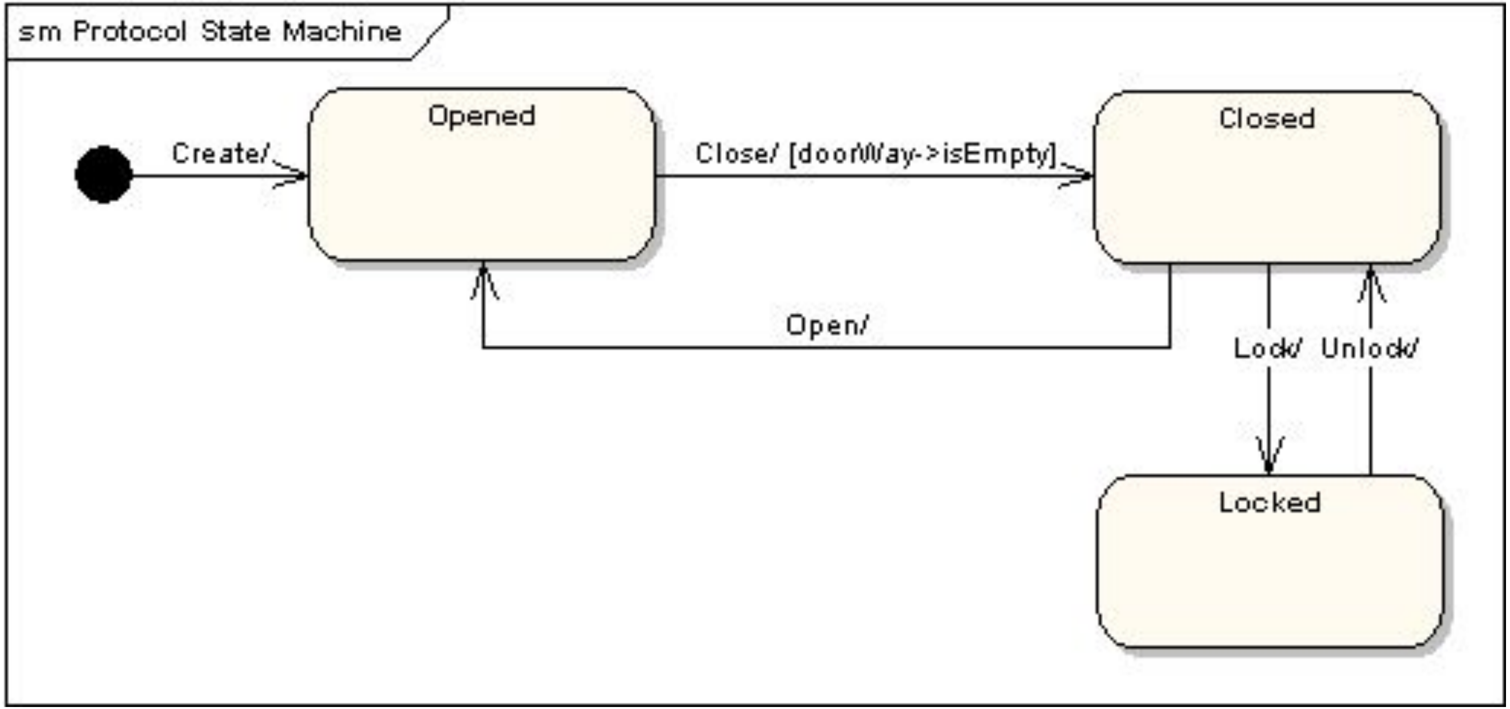
□ Part of BEHAVIOR Diagrams.

□ Time & Sequence

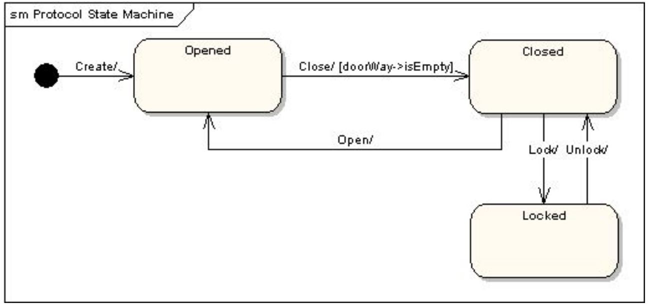
□ Message Exchange

□ Event / Event Scenarios

State Diagram



State Diagram



□ Part of BEHAVIOR Diagrams.

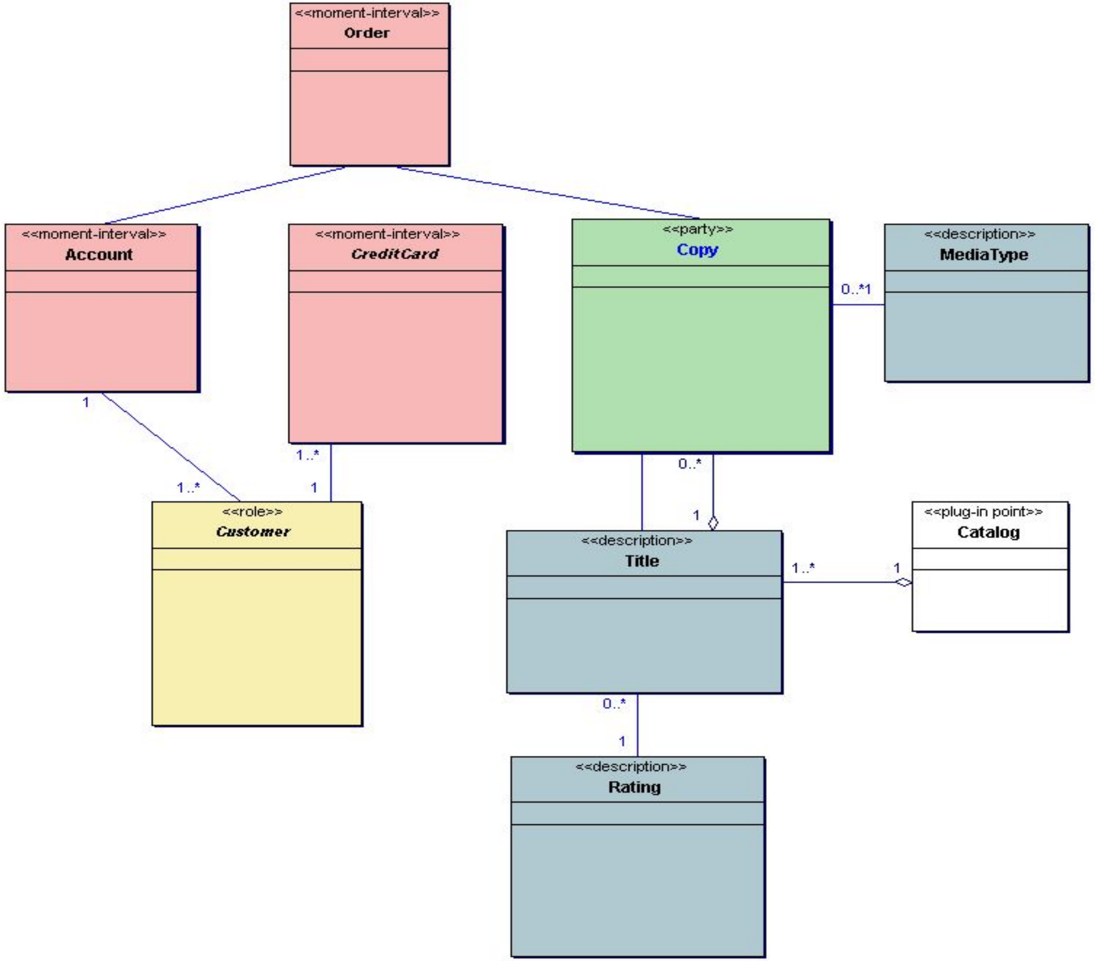
□ Finite Automaton

□ State Machines (FSM)

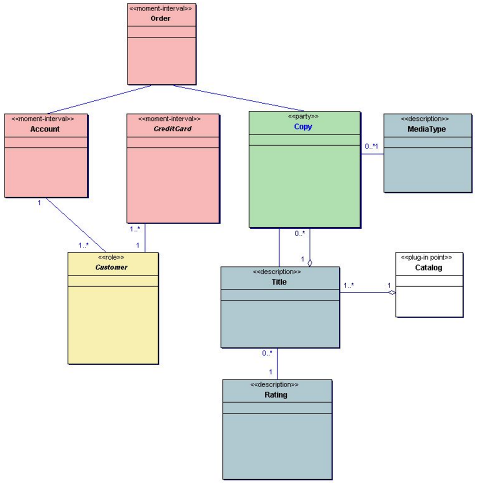
□ Events & States

UML Modeling in Color

Role	Moment, Interval
Description	Party, place, thing



UML Modeling in Color



□ Pink (Moment-interval) – Represents a moment or interval of time.

□ Yellow (Role) – A way of participating in the above activity.

□ Blue (Description) – A catalog like description which classifies objects.

□ Green (Party, Place or Thing) – Something that is uniquely identifiable

DOs

DON’Ts

- □ **DOs**
  - □ Class Diagram with your CORE domain.
  - □ Package/Class diagram for the whole system: BIG Picture.
  - □ Sequence/Deploy for specific and complicated flows: i.g: oauth 2.0

- □ **DONTs**
  - □ Class Diagram for all Classes
  - □ Package diagrams for all diagrams
  - □ Sequence diagram for all user stories
  - □ Reverse Engineer UML diagram

Remember <u>understand</u> the PROBLEM <u>Think</u> about the solution is what matters, <u>UML</u> is WAY to express it.

Design a Billing / Ticket System

Exercise

- ❑ You will have events, all events will a venue, price, picture and relation to similar events.
- ❑ Events will have duration, participants with could join multiple events.
- ❑ System need to: register, pay, list, export(PDF) events, mail, cancel events.

UNIFIED MODELING LANGUAGE™

UML

Diego Pacheco