Architectural Design

Something about Software Architecture Visual Modeling

ISEP / LETI / ESOFT

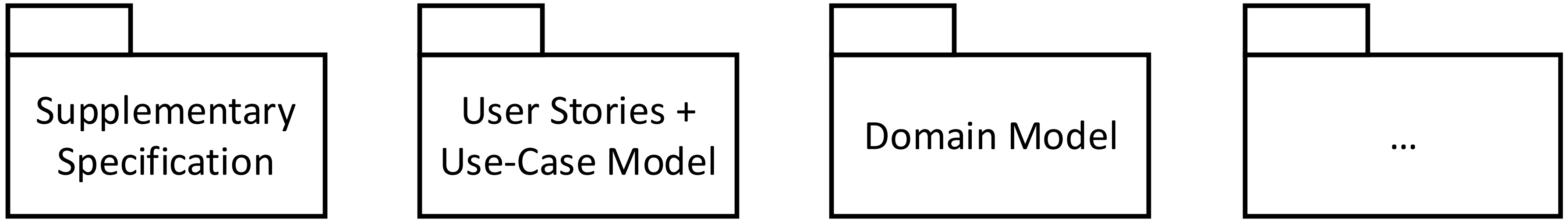
Topics

- From Requirements to Design
- Architectural Design
  - C4 Model
  - 4+1 View Model
- MyDemo System Example

From Requirements to Design

From Requirements to Design (1/2)

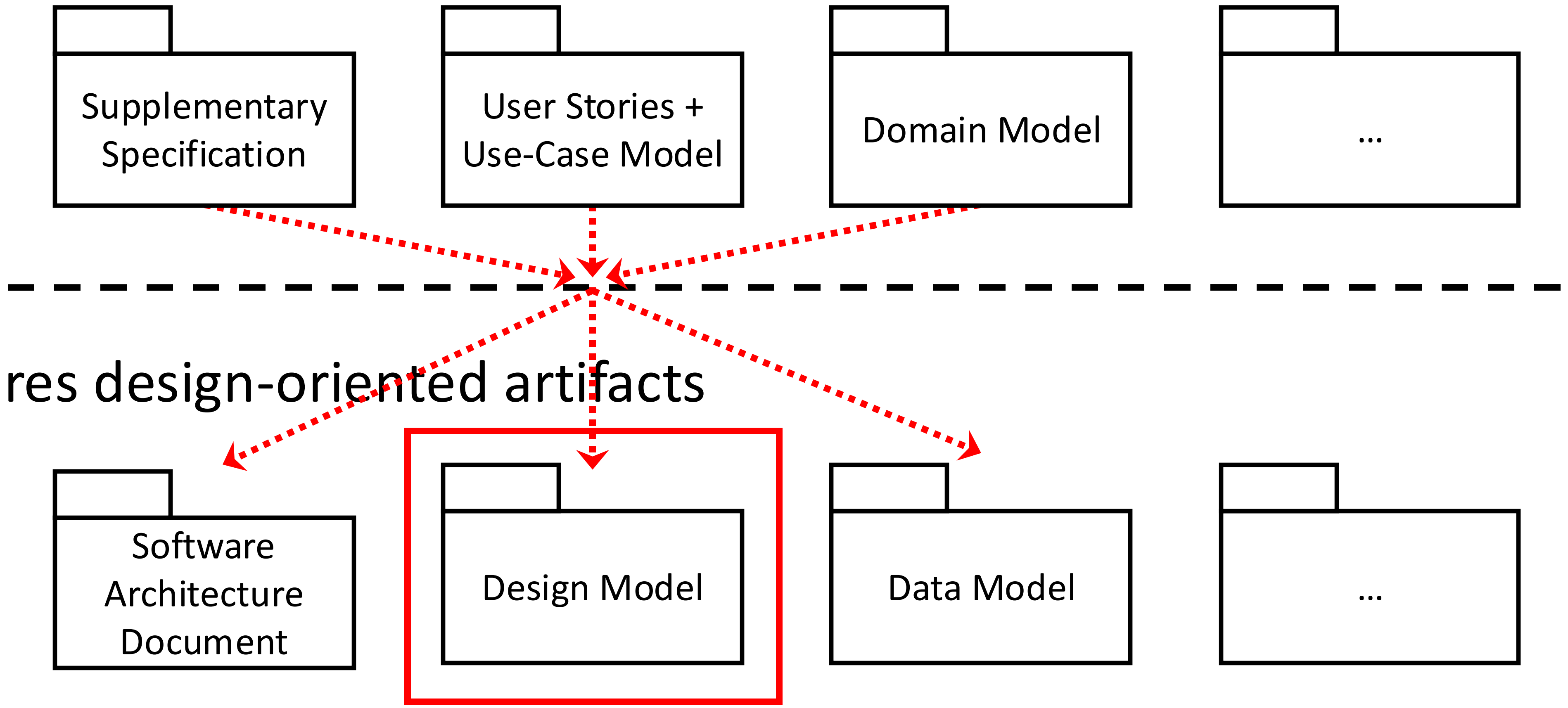
- Requirements-driven set of artifacts



From Requirements to Design (2/2)

- Requirements-driven set of artifacts

- Inspires design-oriented artifacts


Supplementary Specification
User Stories + Use-Case Model
Domain Model
...
Software Architecture Document
Design Model
Data Model
...

Design as a SW Activity

- Aims to outcome a **conceptual solution** that allows to fulfil software requirements, comprising artifacts from a coarser to a finer granularity
- How?
  - Applying best practices, principles and patterns
    - Responsibility-driven Design
    - Modularity
    - GRASP, SOLID and GoF
    - Architectural patterns
    - (others)
  - Adopting formal notations (e.g., UML) on developed artifacts
- The architectural description must fundamentally serve to **reason about the system**, and not just to describe it.

Architectural Design

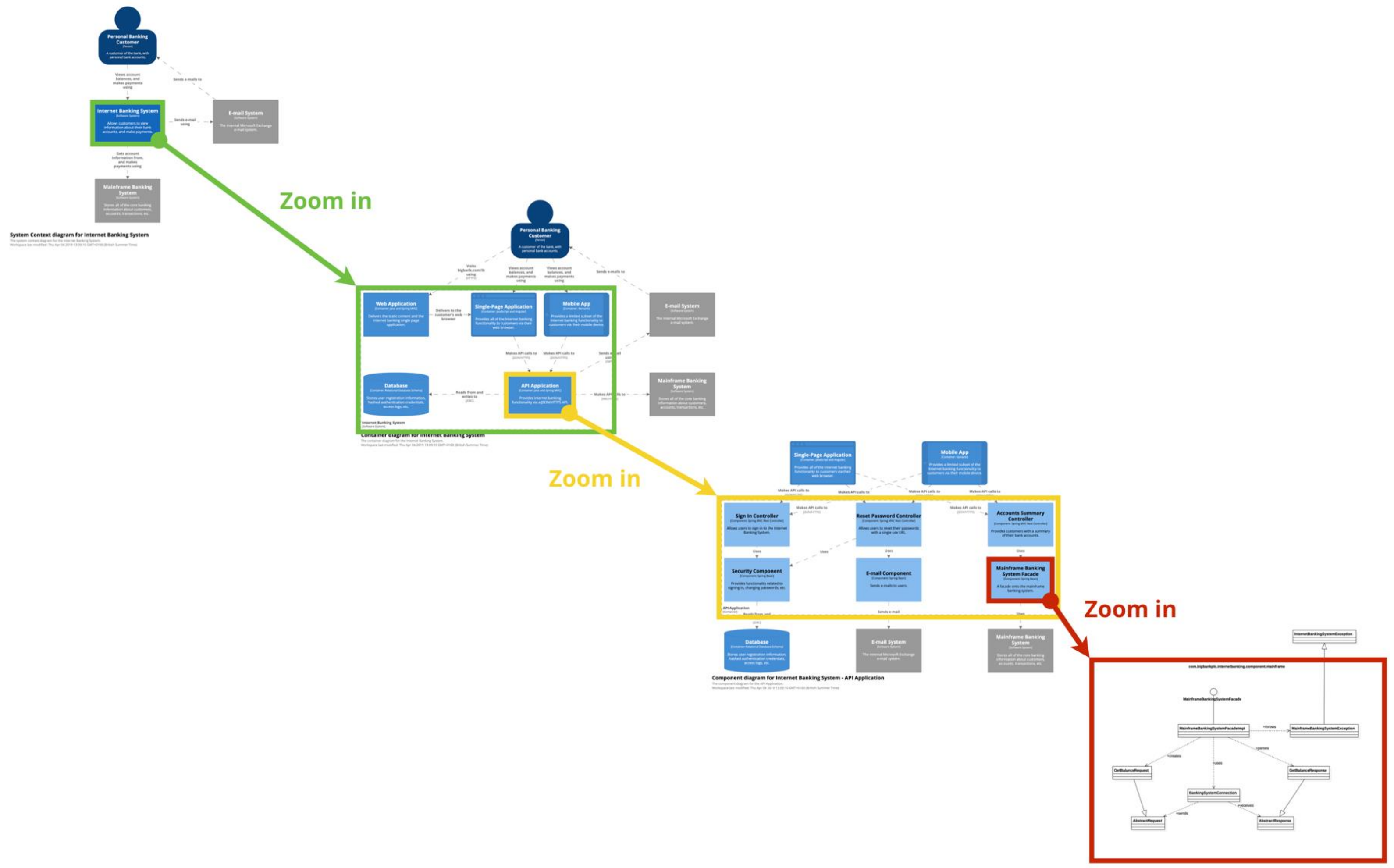
Designing with Distinct Levels of Granularity

- From Architectural Design (cf. C4 Model)
  - System/Context Perspective (Level 1 of C4) → More abstract/coarser
  - Containers/Applications (Level 2) → Less abstract/coarser than L1
  - Components (Level 3) → Even less abstract/coarser than L2

- To Detailed Design
  - Code (Level 4) → The most detailed design (the finest granularity)

- Suggested approach: **OO Design** (cf. further slides)

The C4 Model (*levels*)


Personal Banking Customer
Internet Banking System
E-mail System
Mainframe Banking System
System Context diagram for Internet Banking System
Zoom in
Web Application
Single-Page Application
Mobile App
Database
API Application
Container diagram for Internet Banking System
Zoom in
Sign In Controller
Reset Password Controller
Accounts Summary Controller
Security Component
E-mail Component
Mainframe Banking System Facade
Component diagram for Internet Banking System - API Application
Zoom in

Level 1
Context

Level 2
Containers

Level 3
Components

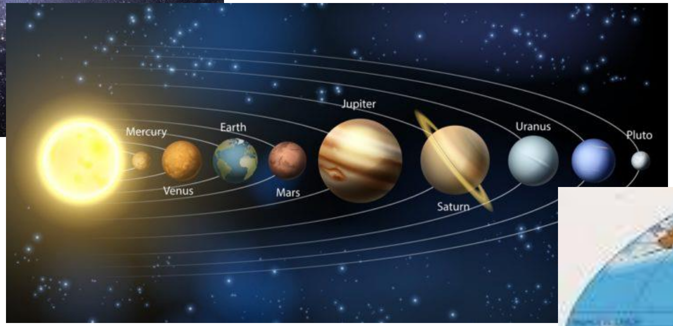
Level 4
Code

The C4 Model (*levels*) – Metaphor

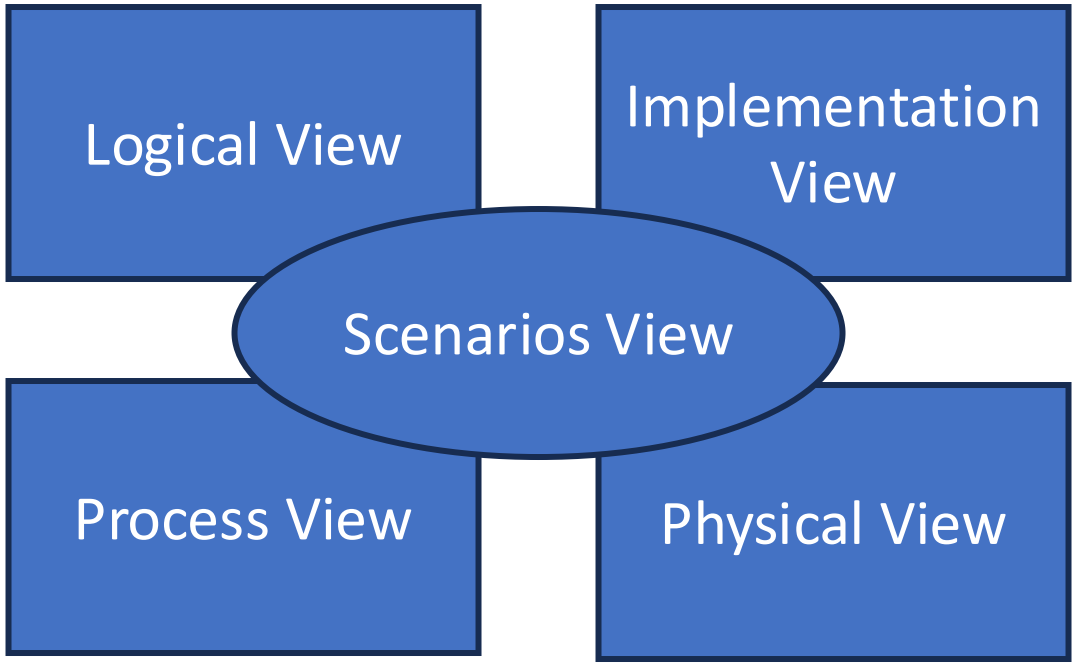
coarse grain

zoom in

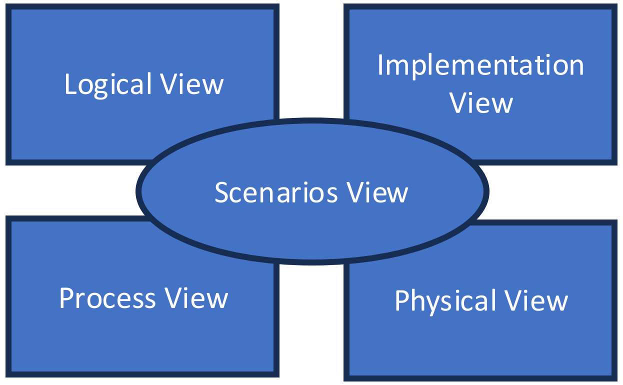
fine grain



The 4+1 Model (*views*) (1/2)

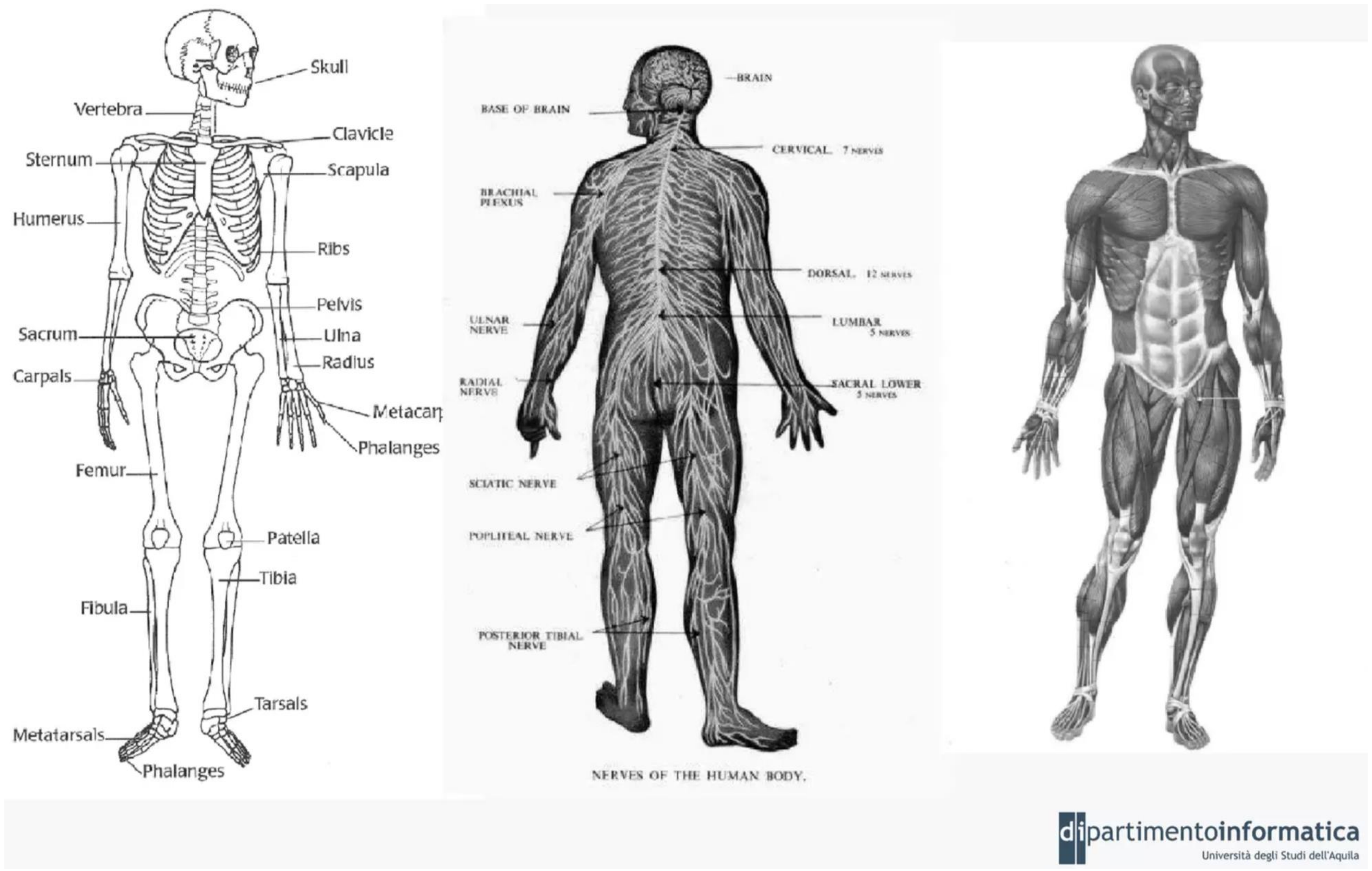


The 4+1 Model (*views*) (2/2)



- **Logical** (or Structural) **View** – concerned with the decomposition of the system in (logical) modules, components and objects to primarily accomplish functional requirements
- **Process** (or Behavioral) **View** – concerned with concurrency and synchronization aspects to primarily accomplish non-functional requirements such as performance and availability
- **Implementation** (or Development) **View** – concerned with the static organization of the software (as the developer sees it)
- **Physical** (or Deployment) **View** – concerned with mapping the software (parts) to the hardware
- **Scenarios** (or Use Case) **View** – concerned with showing how all elements (of the other views) work together using a small set of scenarios

The 4+1 Model (*views*) – Metaphor

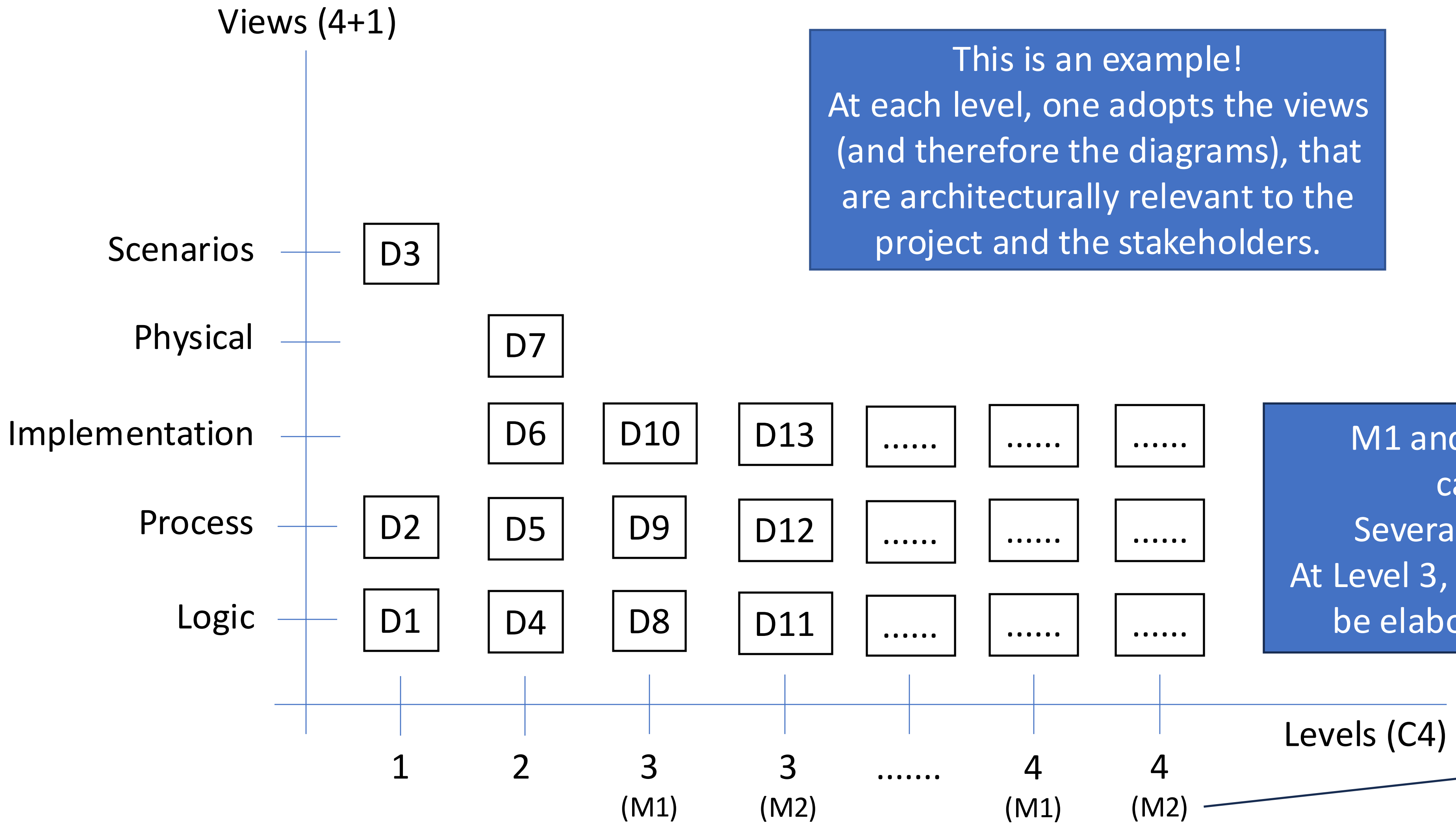


Views and UML (possible mapping/usage)

View	UML Diagram
Logical View	Class Diagram, Object Diagram, Component Diagram, Package Diagram, Composite Structure Diagram
Process View	Activity Diagram, State Machine Diagram, Sequence Diagram, Timing Diagram, Interaction Overview Diagram
Implementation View	Component Diagram, Package Diagram
Physical View	Deployment Diagram
Scenarios View	Use Case Diagram

Combining 4+1 with C4

There are no mandatory diagrams.
Only relevant or irrelevant.



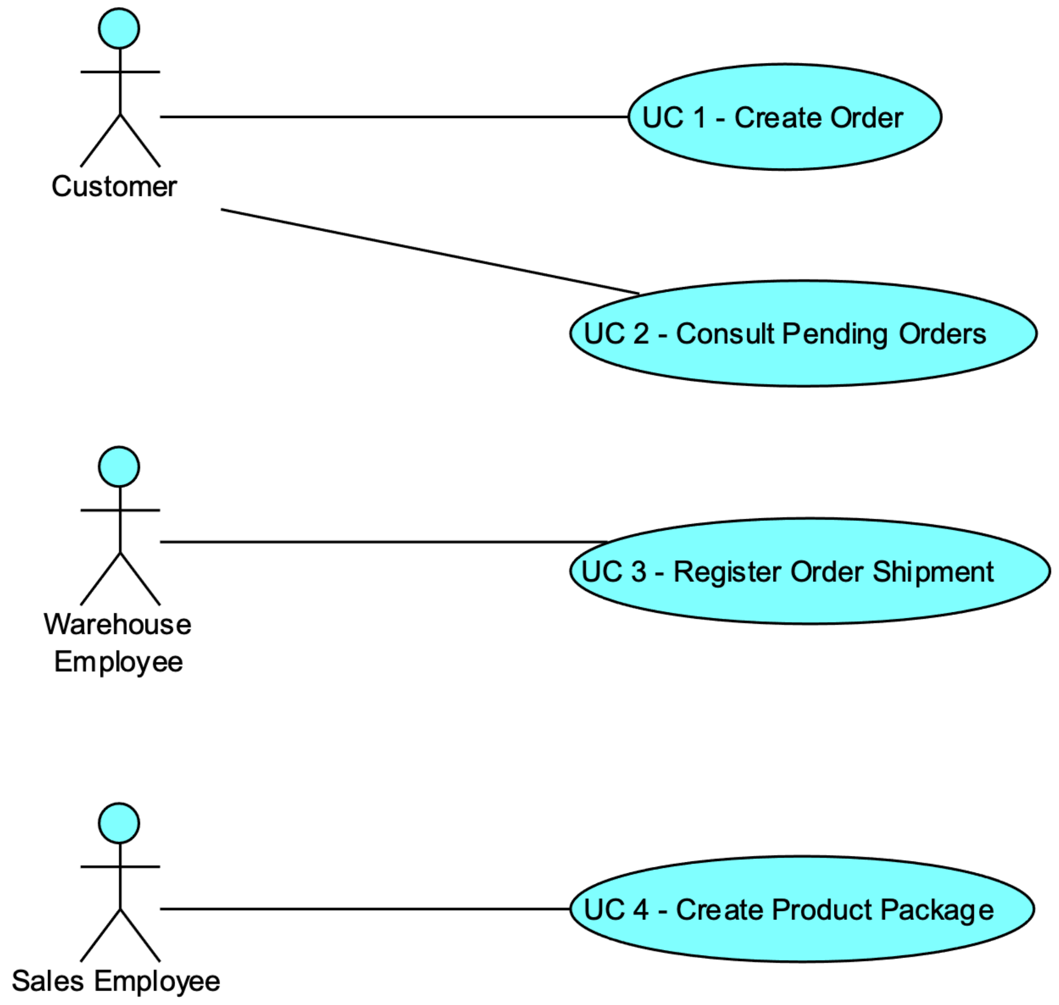
This is an example!
At each level, one adopts the views (and therefore the diagrams), that are architecturally relevant to the project and the stakeholders.

M1 and M2 are two containers captured at Level 2.
Several containers might exist.
At Level 3, new views/diagrams should be elaborated for each container.

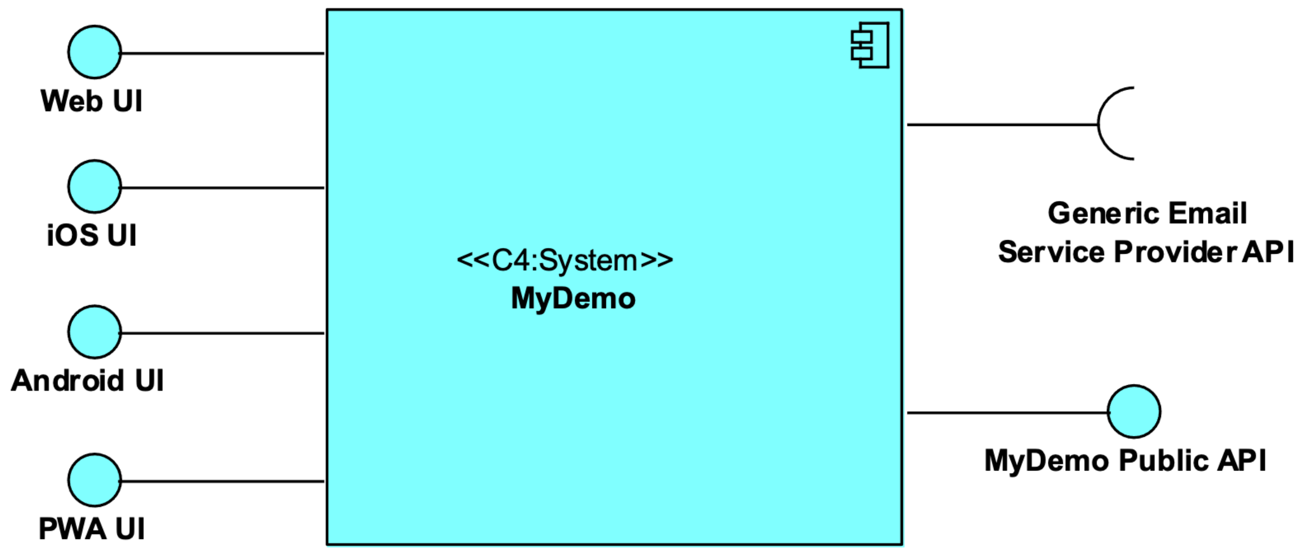
MyDemo System

Example based on a typical software architecture

Level 1 – Scenarios View


Customer
UC 1 - Create Order
UC 2 - Consult Pending Orders
Warehouse Employee
UC 3 - Register Order Shipment
Sales Employee
UC 4 - Create Product Package

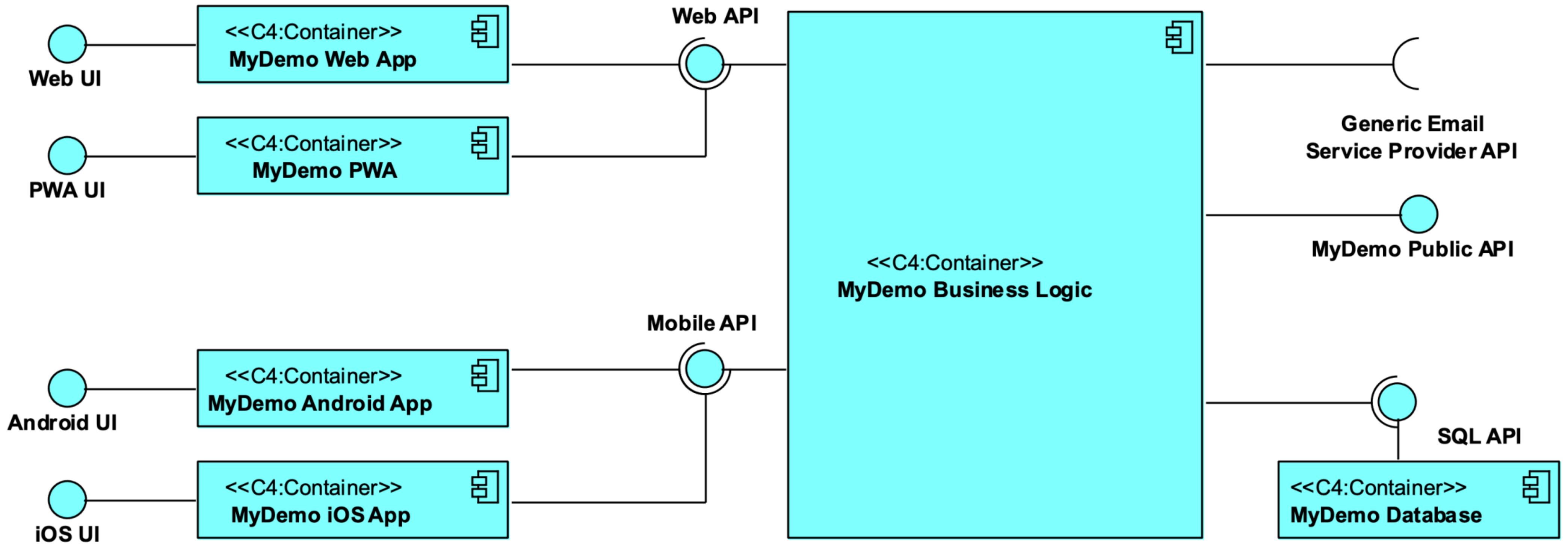
Level 1 – Logical View



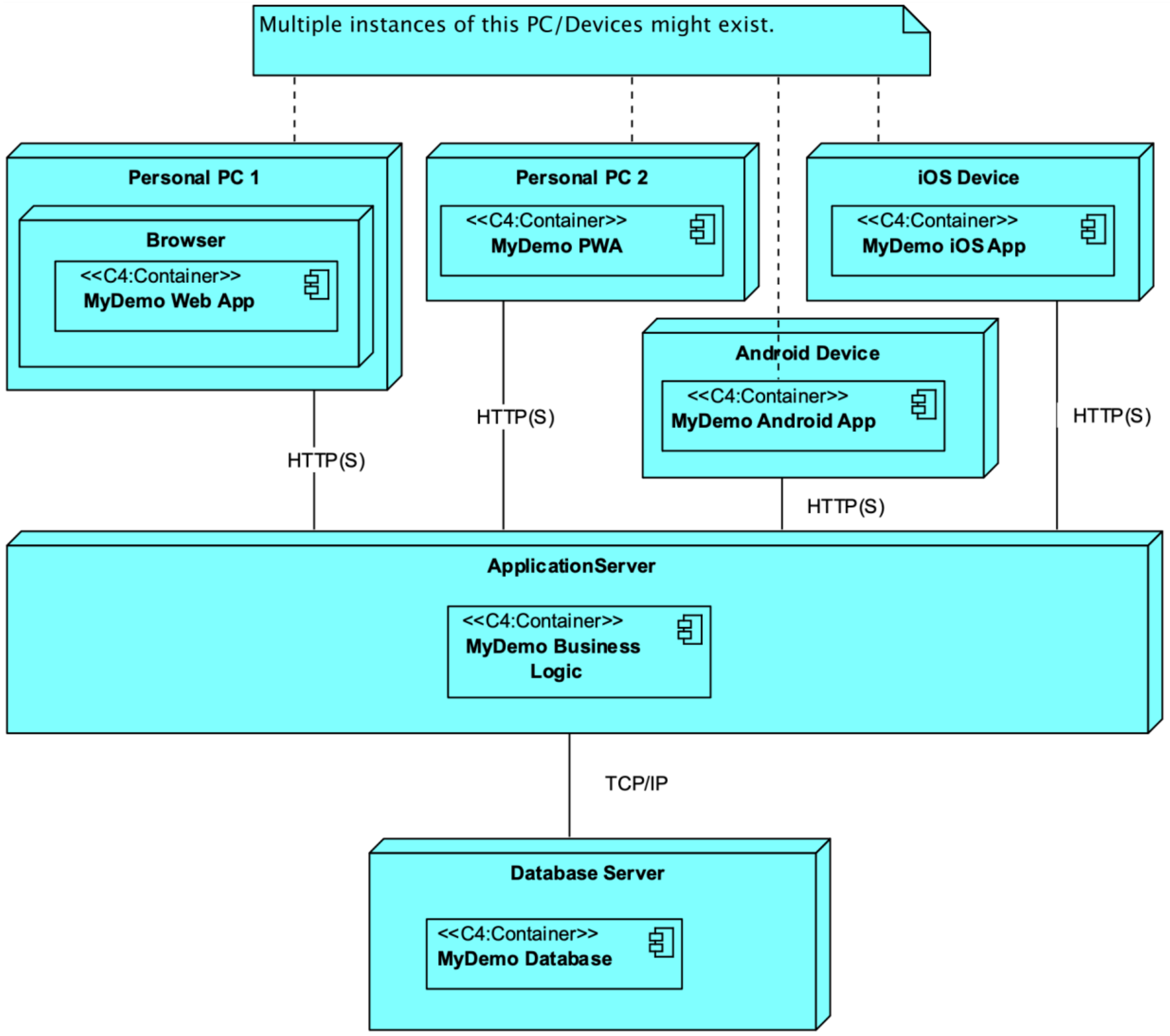
- Provides
  - 4 User Interfaces (UI) for its actors
  - 1 API for external systems usage

- Requires (makes use) of
  - 1 Generic Email Service Provider API

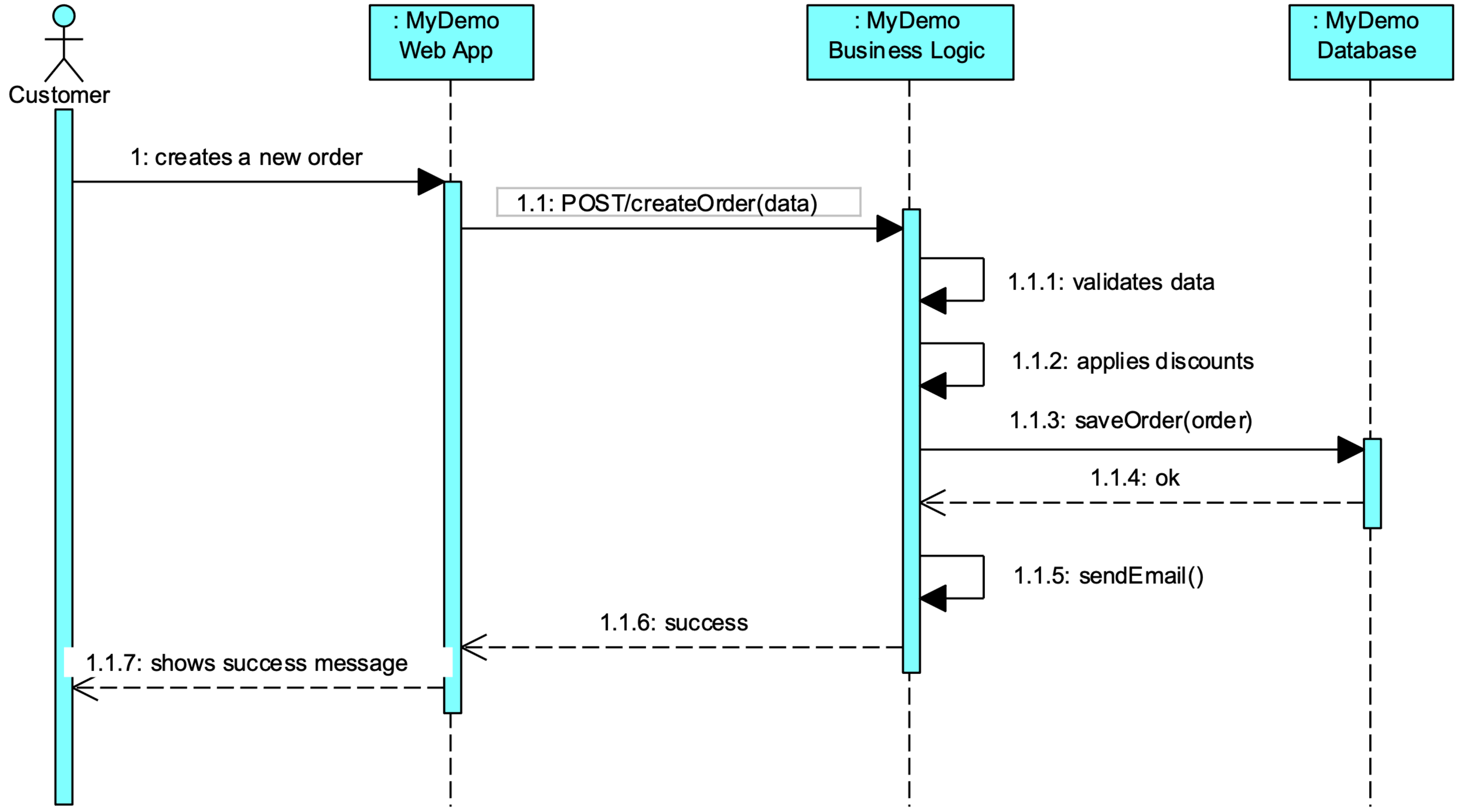
Level 2 – Logical View



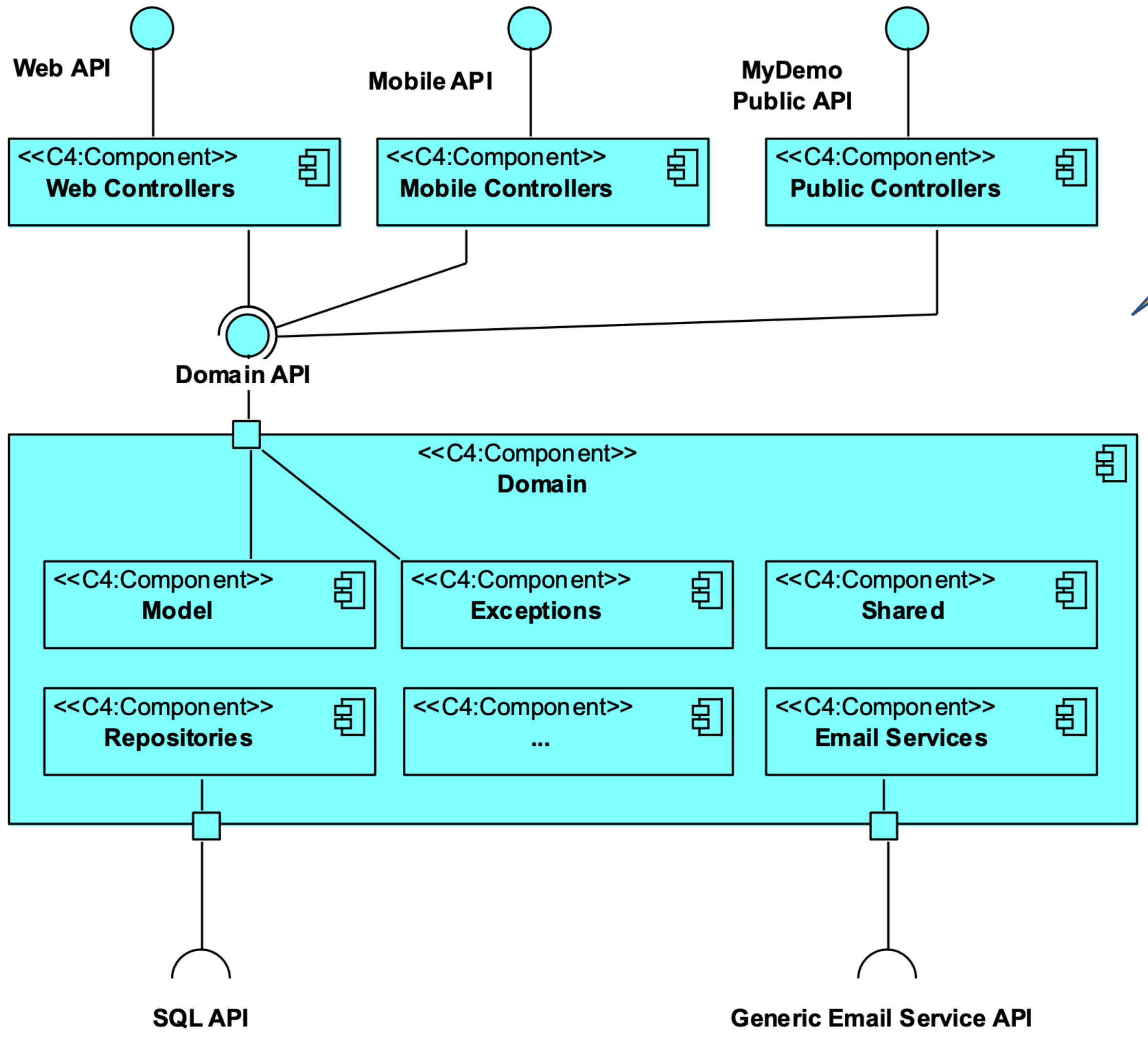
Level 2 – Physical (Deployment) View



Level 2 – Process View (for UC Create Order)

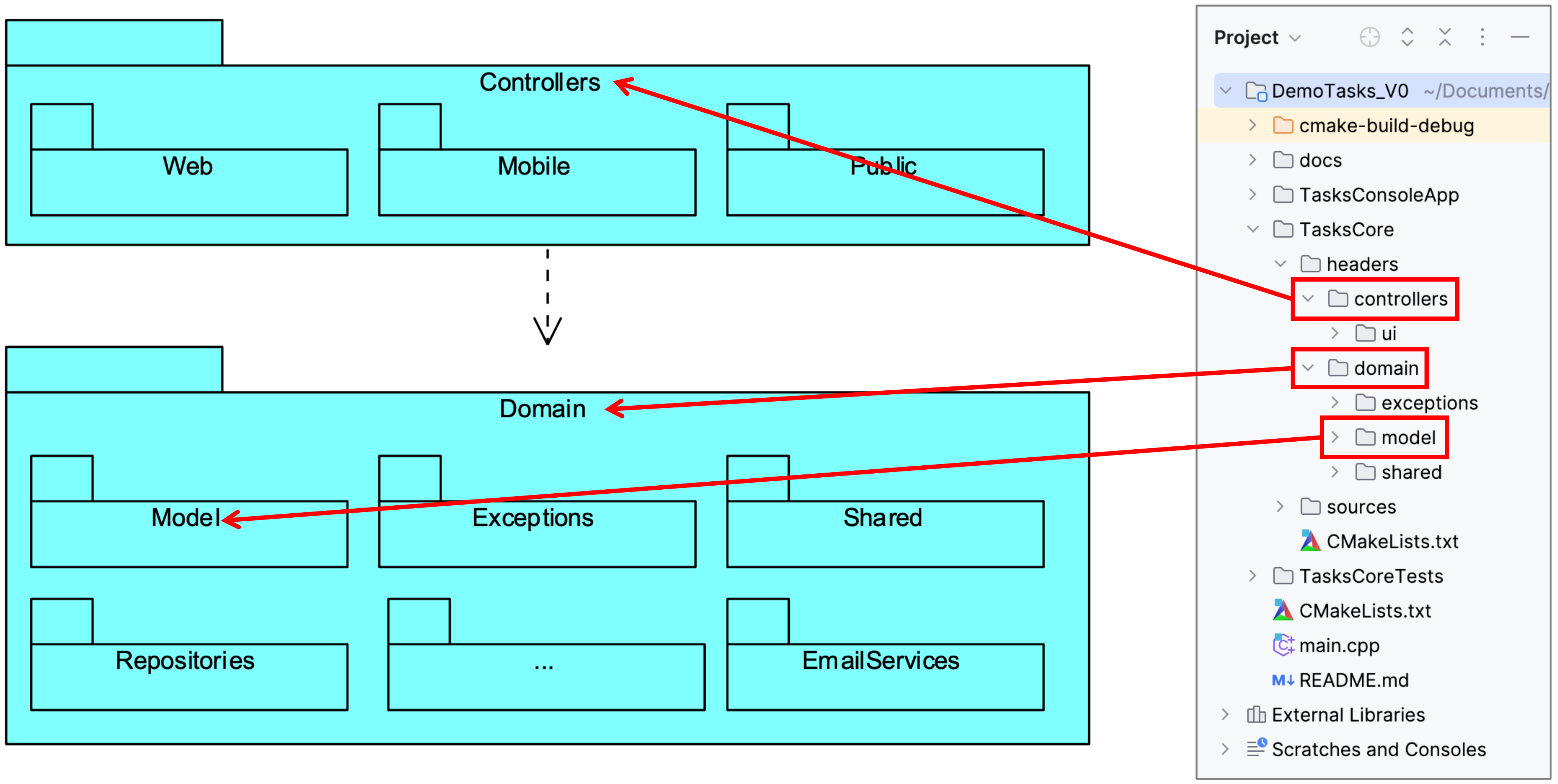


Level 3 – Logical View (for Business Logic Container)


Web API
Mobile API
MyDemo Public API
<<C4:Component>> Web Controllers
<<C4:Component>> Mobile Controllers
<<C4:Component>> Public Controllers
Domain API
<<C4:Component>> Domain
<<C4:Component>> Model
<<C4:Component>> Exceptions
<<C4:Component>> Shared
<<C4:Component>> Repositories
<<C4:Component>> ...
<<C4:Component>> Email Services
SQL API
Generic Email Service API

The focus of ESOFT is on developing a Business Logic Container

Level 3 – Implementation View (for Business Logic Container)



Project

- DemoTasks_V0 ~/Documents/
  - cmake-build-debug
  - docs
  - TasksConsoleApp
  - TasksCore
    - headers
      - controllers
        - ui
      - domain
        - exceptions
        - model
        - shared
    - sources
    - CMakeLists.txt
  - TasksCoreTests
  - CMakeLists.txt
  - main.cpp
  - README.md
- External Libraries
- Scratches and Consoles

Level 3 / Level 4 Views

- UI Containers
  - MyDemo Web App
  - MyDemo PWA
  - MyDemo iOS App
  - MyDemo Android App

$\}$ Lectured on LETI-DSSMV course unit

- Database-related Containers $\longrightarrow$ Lectured on LETI-BDAMD course unit

- Business Logic Containers $\longrightarrow$ Lectured throughout this course unit

Summary

- In the context of UML and architectural design, both C4 and 4+1 models are valuable approaches for comprehensively modeling and documenting a system.
- In the C4 Model, the system is structurally decomposed into containers and components. Each diagram describes a different level of detail, from a coarse granularity to a fine granularity.
- In the 4+1 Model, each view serves a specific purpose, collectively providing a holistic understanding of the system’s structure, behavior, deployment, development and user interaction.

Bibliography

- Fowler, M. (2003). UML Distilled (3rd ed.). Addison-Wesley. ISBN: 978-0-321-19368-1
- Larman, C. (2004). Applying UML and Patterns (3rd ed.). Prentice Hall. ISBN: 978-0-131-48906-6
- Brown, S. The C4 model for visualizing software architecture. Available on: https://c4model.com
- Kruchten, P. (1995). Architectural Blueprints—The "4+1" View Model of Software Architecture. IEEE Software 12 (6). Available on: https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf
- 4 + 1 Views in Modeling System Architecture with UML