4+1 Architectural View Model

By Wei Liem Tan

Contact: weiliem.tan@unimelb.edu.au

Introduction

- → Originally created by Philippe Kruchten in 1995
- → Architecture is separated into five views to examine different parts from different perspectives, easing overall viewing
- → Each of the views highlight parts of the system, while intentionally suppressing other parts

What is a view model?

- → A view model in software engineering is a framework.
- → It defines a coherent set of views to be used in the construction of a system architecture or software architecture.
- → A view describes the system in the viewpoint of different stakeholders, such as end-users, developers and project managers.
- → Enables human to comprehend very complex system.

Why use this model?

- → To better organize design efforts and separation of concern.
- → Provides a way for developers and architects to prioritize modelling concerns.
- → Different stakeholders can examine different parts of the system that are of interest to them.
  - → Developers interested in software components.
  - → System Admins/System Engineers interested in deployment, hardware specification and network configuration.

Different views of 4+1 view model

→ Four views:

1. Logical View: The logical view describes the functional requirements—what the system should provide in terms of services to its users.
2. Process View: The process view deals with the dynamic aspects of the system, explains the system processes and how they communicate. i.e.: Interaction between front-end and back-end.
3. Development View: The development view illustrates a system from a programmer's perspective and is concerned with software management.
4. Physical View: The physical view depicts the system from a system engineer's point of view. It describes the mappings of software onto hardware. i.e.: Deployment, hardware specifications

→ Extra view – Scenarios: Also known as the use-case view. Illustrated with use cases to describe sequences of interactions between objects and between processes.

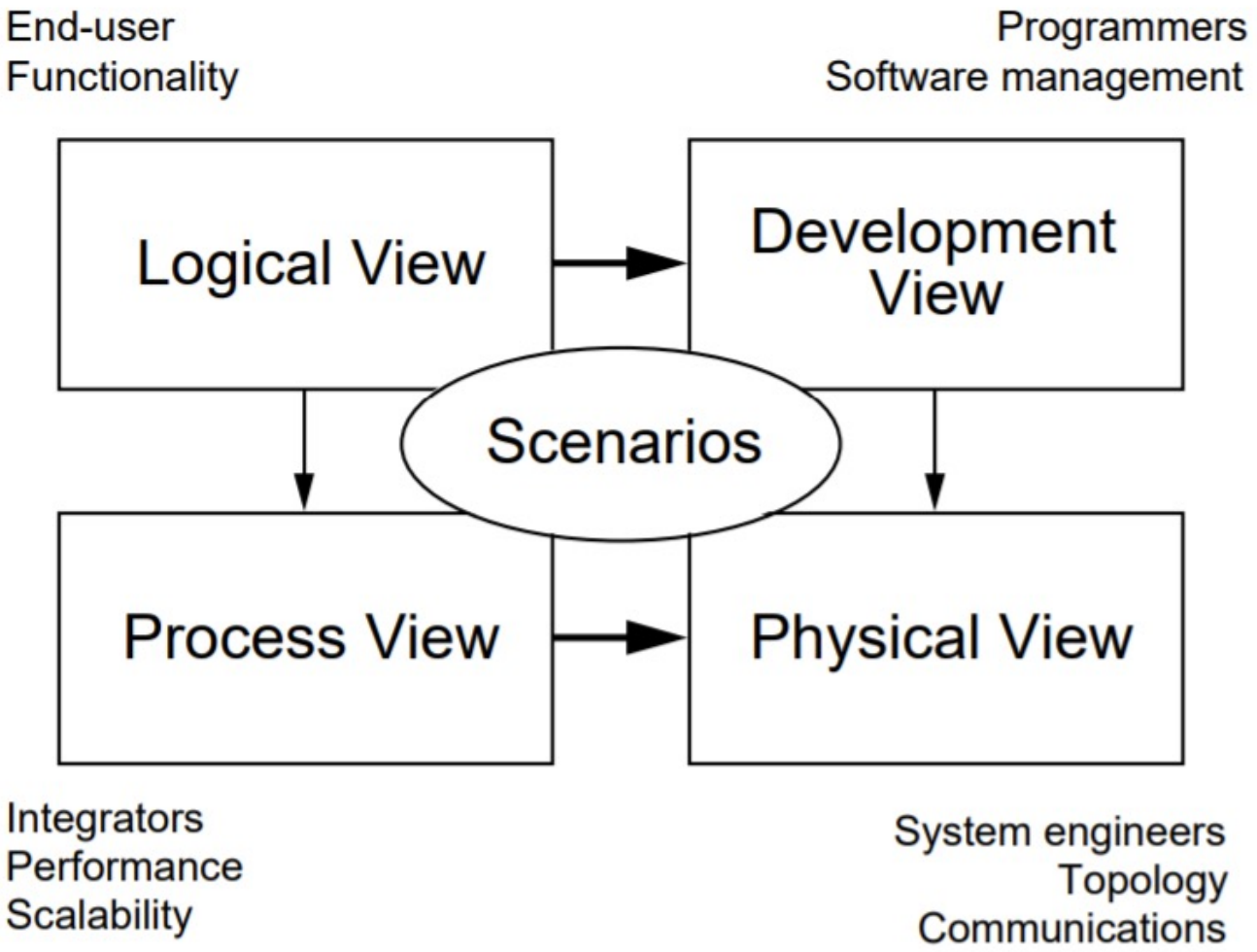


Figure 1 — The “4+1” view model

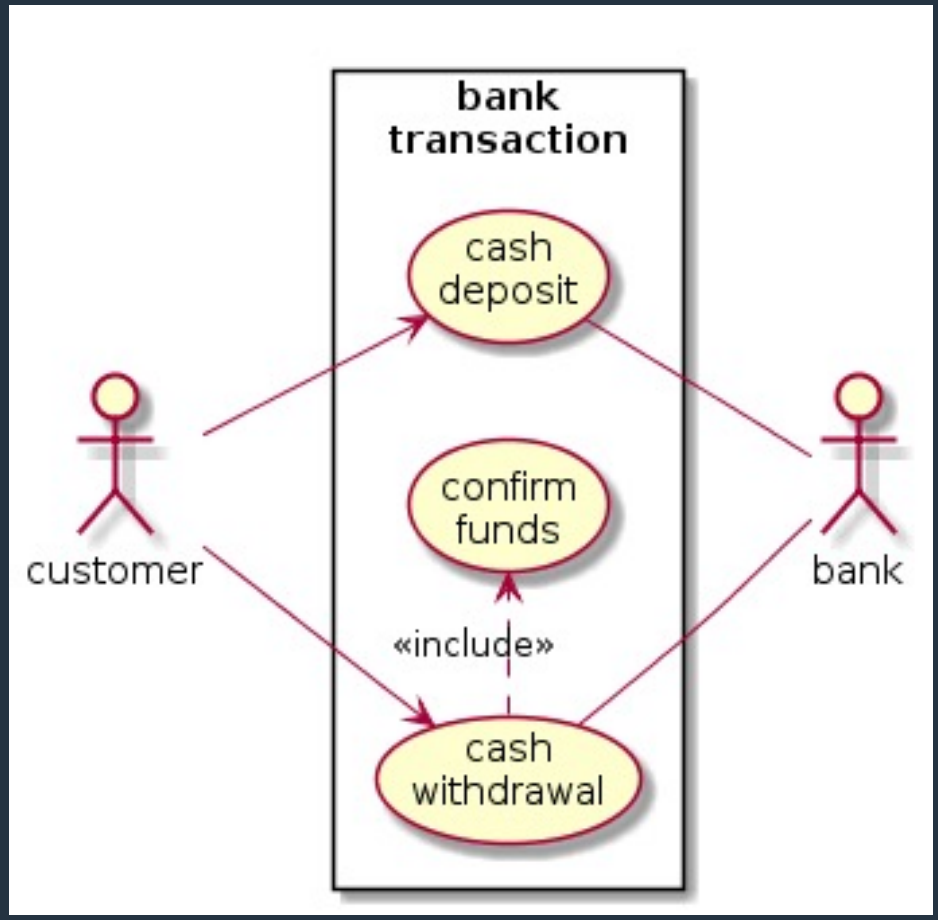
Different types of UML diagrams

- → UML has many types of diagrams, each of them can be categorized into one of the views.
- → Behavior diagrams: A type of diagram that depicts behavioral features of a system or business process. This includes activity, state machine, and use case diagrams as well as the four interaction diagrams.
- → Interaction diagrams: A subset of behavior diagrams which emphasize object interactions. This includes communication, interaction overview, sequence, and timing diagrams.
- → Structure diagrams: A type of diagram that depicts the elements of a specification that are irrespective of time. This includes class, composite structure, component, deployment, object, and package diagrams.
- → More details: http://agilemodeling.com/essays/umlDiagrams.htm

Scenario / Use Case View

- → Viewer: End-users
- → Purpose: To represent the scenarios that tie four views together; Forms the reason as to why all the other views exist; To test that the system is complete and consistent
- → Diagrams: Use-case diagrams

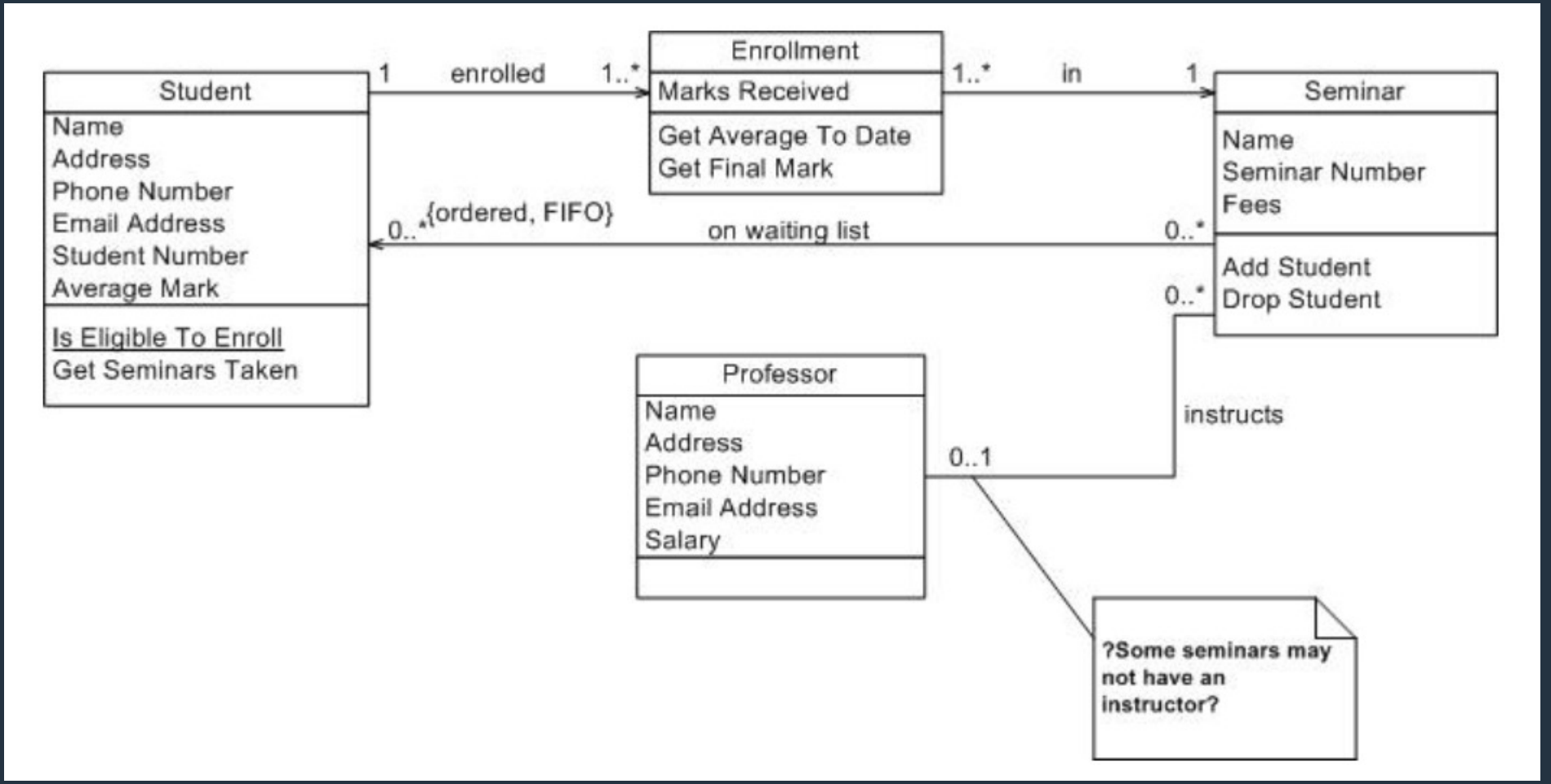
Use Case Diagram



Logical View

- → Viewer: End-users
- → Purpose: describes the functional requirements—what the system should provide in terms of services to its users.
- → Decomposed into a set of key abstractions taken (mostly) from the problem domain.
- → Shows components of the system and their interactions / relationships.
- → Diagrams: Class, Object, Package, State Machine, Composite Structure.

Conceptual Class Diagram / Domain Diagram

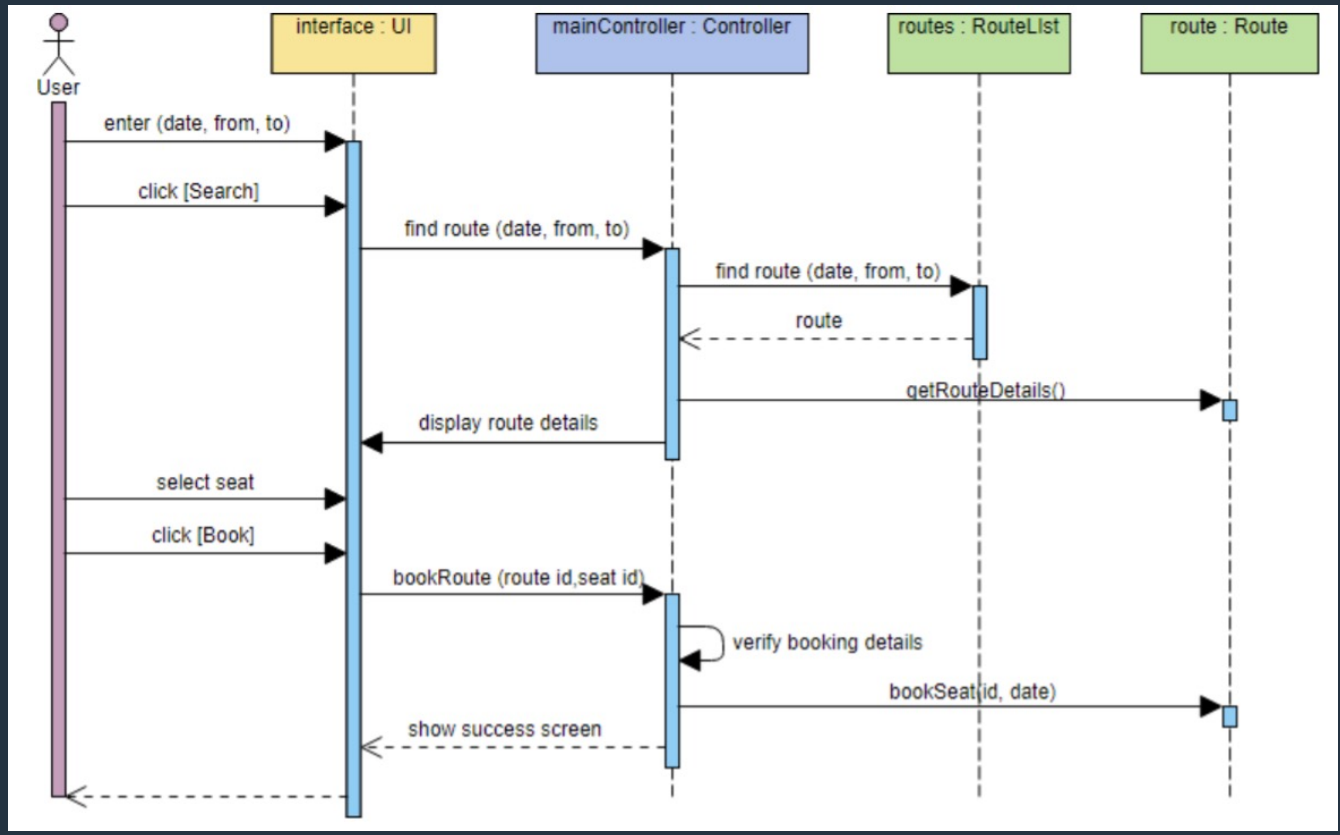


Process View

- → Viewer: Integrators
- → Purpose: Process decomposition; Shows the workflow of the system and how different processes interact with each other;
- → Diagrams: Sequence, Communication, Activity, Timing, Interaction Overview

Sequence Diagram

→ The process architecture can be represented at various levels of abstraction such as interactions between systems, subsystems and objects etc. based on the need.

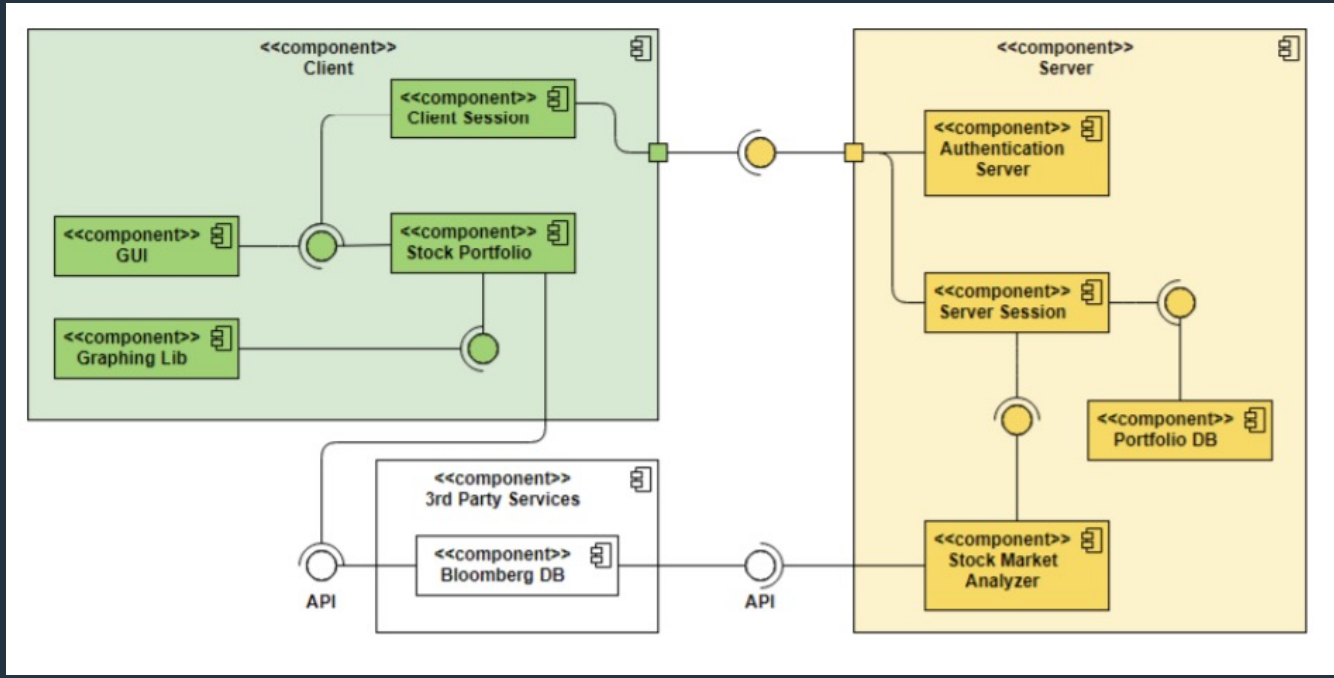


Development View

- → Viewer: Developers
- → Purpose: Illustrates a system from a programmer's perspective and is concerned with software management; building block view of the system, e.g. Packages Used, Execution Environments, Class Libraries and Sub systems utilized.
- → Diagrams: Component, Database

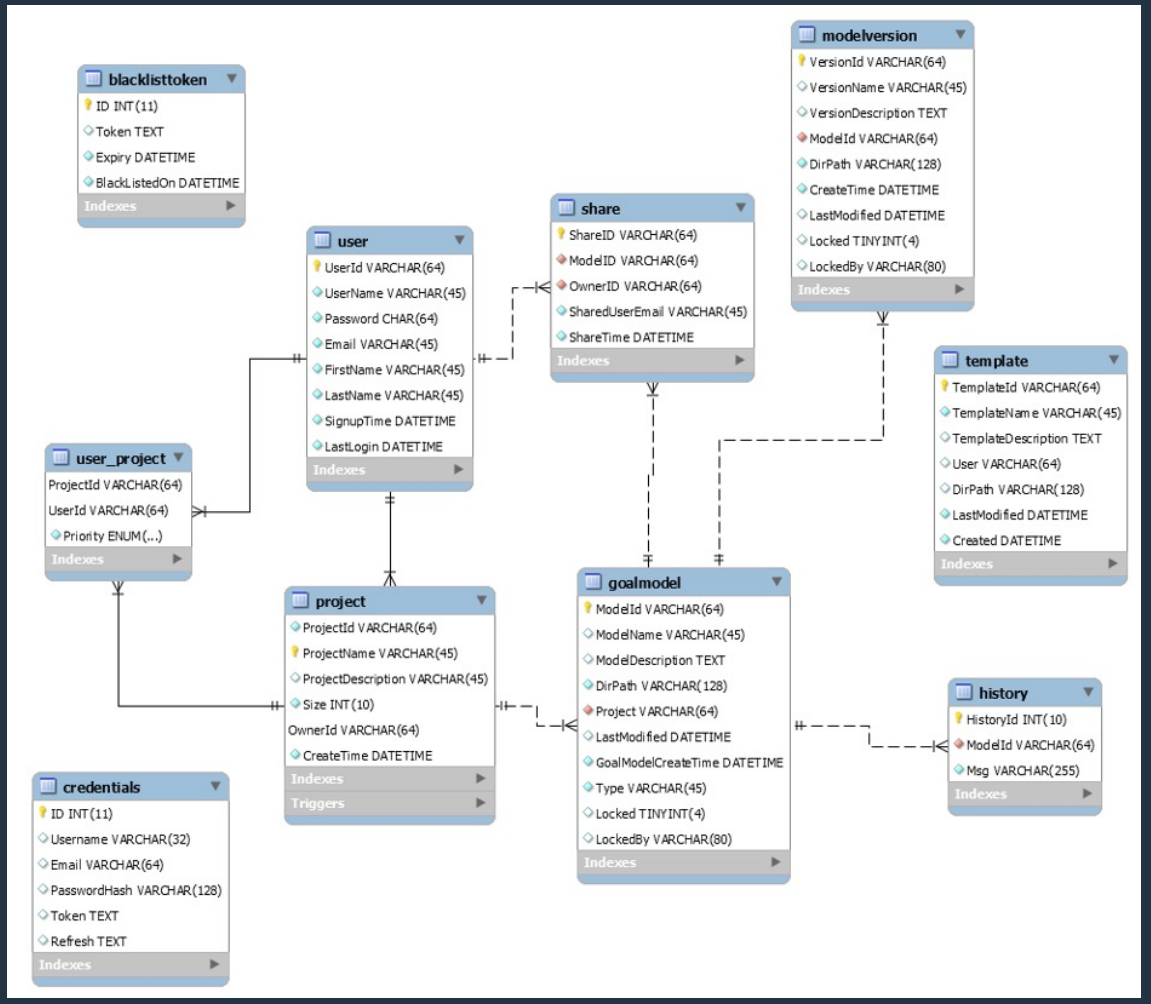
Component Diagram

→ Component diagram depicts how components are wired together to form larger components of software systems.



Database Model

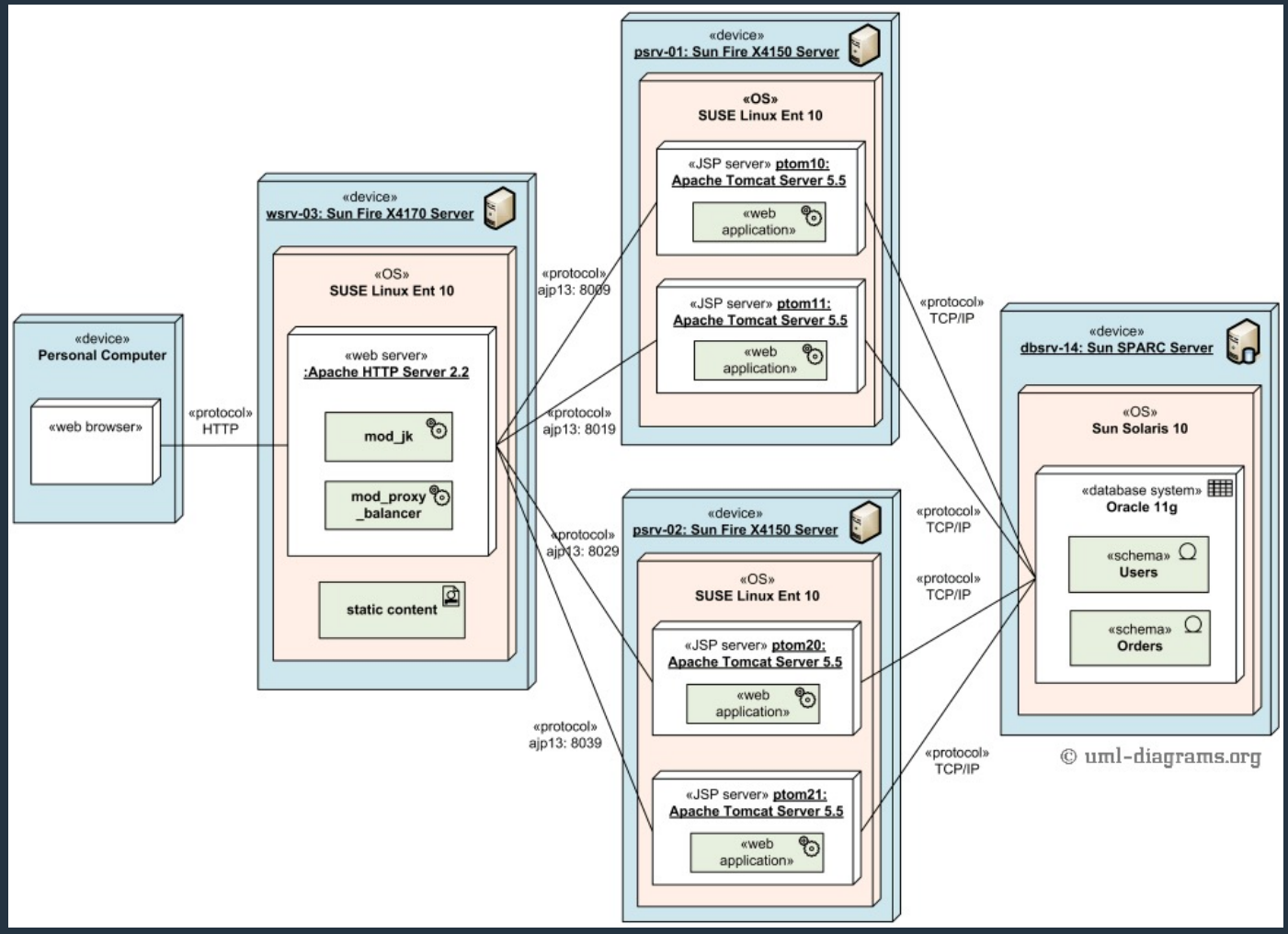
- → A database model is a type of data model that determines the logical structure of a database.
- → It fundamentally determines in which manner data can be stored, organized and manipulated.



Deployment View

- → Viewer: System Engineers / System Admins
- → Purpose: Shows mapping of software onto hardware; Shows the system's execution environment; Concerned with deployment, network configuration etc.
- → Diagram: Deployment

Deployment Diagram



Guideline to modelling your system

- → Identify key requirements that best describes your system, try to capture that while modelling your architecture.
- → For each views, decide which diagrams to create that best represent the details of the system. It is not necessary to draw all the diagrams, pick the ones that are useful for your system.
- → Review important diagrams with your team. When your system's architecture is accurately depicted in diagrams, that is when flaws are exposed. Utilize what you learnt in Software Modelling and Design.

Thank you!

If there's any questions, contact me at: weiliem.tan@unimelb.edu.au