Software Design and Architecture

[Software Architecture] – Chapter 02, L03

Lecture Outlines

- Designing the Architecture with the **4+1view** model
  - ✓ The **logical** view
  - ✓ The **process** view
  - ✓ The **development** view
  - ✓ The **physical** view
  - ✓ Plus … The **user** view

Designing the Architecture

- When designing the architecture of a software system, there is no single view that can present all system aspects
- Instead, software designers can organize the description of their architectural decisions using a collection of different views (why?)
- Specific views can provide partial representation of the software architecture to a specific stakeholder (e.g. system users, analysts, developers, etc.)

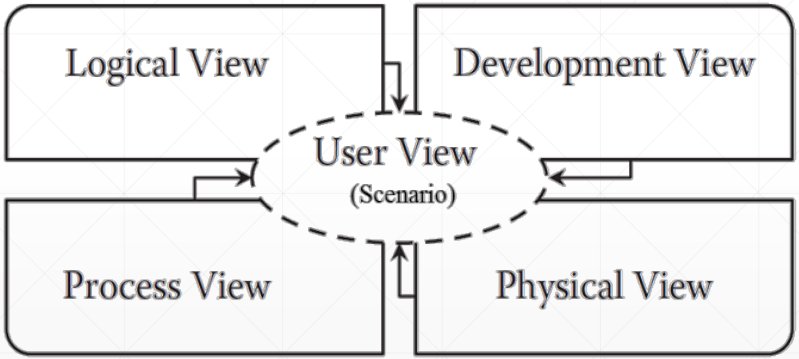
Extra:

- In software architecture, **multiple views are used** because no single view can capture all aspects of a complex system.
- This approach helps **manage complexity** by breaking the system into understandable perspectives.
- It ensures that **different stakeholders' concerns** (e.g., developers, users, managers) are addressed.
- This makes it easier to **develop, analyze, and maintain** the software effectively.
- Each view targets a **different area** (e.g., functional, structural, behavioral).

A popular model is usually used for designing different architectural views is the **4+1 view** model

The 4+1 View Model

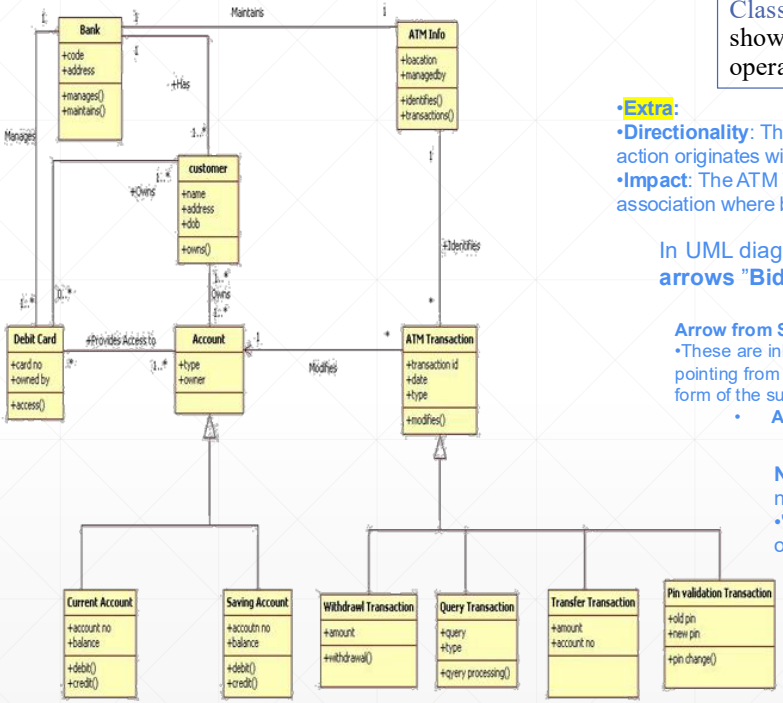
- The **4+1 view** model was originally introduced by Philippe Kruchten in 1995.
- The model provides four essential views: the **logical** view, the **process** view, the **physical** view, and the **development** view.
- To ensure consistency among all 4 views, the user's perspective is captured through several use cases, as part of the **user** view



The logical View

- The logical view specifies system decomposition into conceptual entities (such as objects or classes) and the connection between these entities
- The view basically is an abstraction of the system's functional requirements
- The logical view is usually modeled using:
  - Static diagrams such as: **class/object** diagram
  - Dynamic diagrams such as: **interaction** overall diagram, **sequence** diagram, **communication** diagram, **state** diagram, and **activity** diagram
- The usual stakeholders of the logical view are the end-users, analysts, and designers.

*ATM Example – Logical View (Class Diagram)



Class diagrams :
show the classes of the system, their inter-relationships, and the operations and attributes of the classes.

•**Extra**:
•**Directionality**: The arrow points from ATM Transaction to Account, showing that the action originates with the ATM Transaction.
•**Impact**: The ATM Transaction directly affects the Account, unlike a bidirectional association where both classes might equally influence each other.

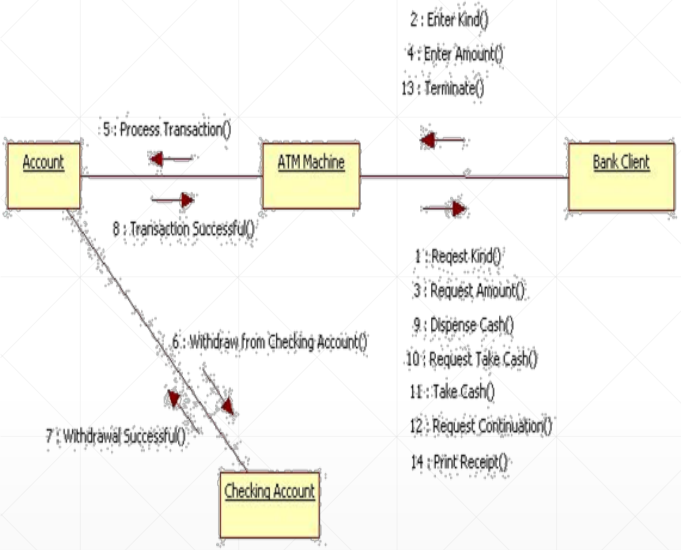
In UML diagrams, bidirectional associations can have **two arrows** "**Bidirectional Relationship**" or **no arrows**:

Arrow from Subclass to Superclass:
•These are inheritance relationships shown with a line ending in a triangular arrowhead, pointing from the subclass to the superclass. They indicate that the subclass is a specialized form of the superclass. For example:
- **Account** (superclass) to **Current Account** and **Saving Account** (subclasses).

Names above arrows (e.g., Maintains, Modifies, Provides Access to) describe the nature of the relationship from one class to another. For example:
•"Maintains" on the arrow from Bank to ATM Info implies that each Bank maintains one or more ATMs.

Multiplicities (the numbers near the lines) indicate the number of instances of one class that can be associated with another class. For example, "1..*" means "one to many" and "1" means "exactly one.

*ATM Example – Logical View (Collaboration Diagram)



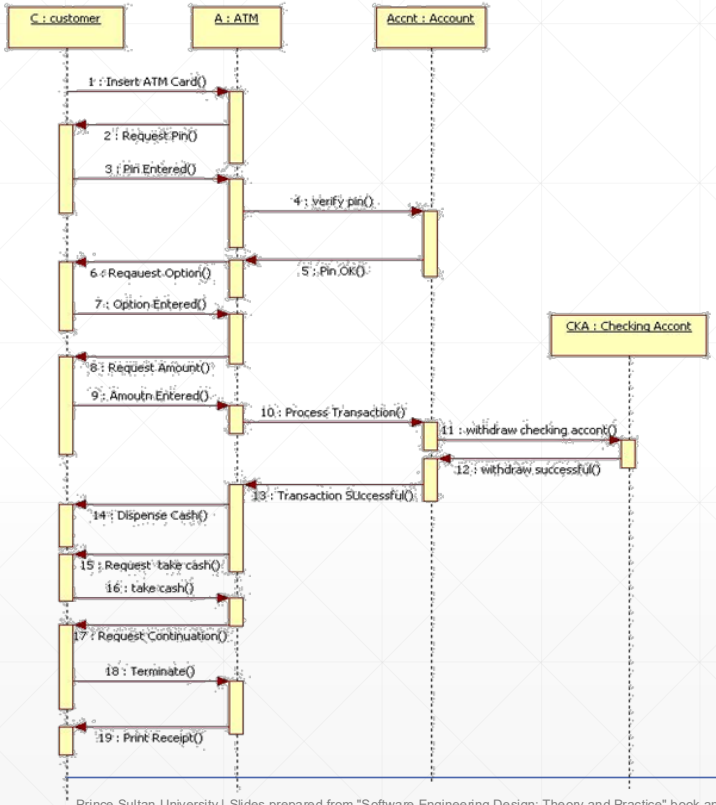
Extra: Explanation of the previous communication Diagram

- **Bank Client:** Represents the user of the ATM.
- **ATM Machine**: Handles transactions and connects client with accounts.
- **Account:** General representation of a bank account.
- **Checking Account**: a specialized account object participating in the ATM transaction scenario

Bank Client **inserts card**.

1. ATM Machine **requests PIN**.
2. Bank Client **enters PIN**.
3. ATM Machine **requests the amount** to be withdrawn or deposited.
4. Bank Client **enters the amount**.
5. ATM Machine **Process transaction with Account/Checking Account**.
6. ATM Machine **requests to take cash** (for withdrawals).
7. Withdraw successfully: The *banking system* confirmed your money is deducted.
8. Validates transaction(Successful)
9. Dispense cash: The *ATM machine hardware* actually pushes the cash out.
10. Request take cash
11. Take cash: You physically grab it.
12. Requests continuation
13. Terminates session
14. print receipt

Collaboration diagram: is more focused on showing the communication between main entities



*ATM Example – Logical View (Sequence Diagram)

Sequence diagram: used to model interactions between design units together with the messages exchanges and the time-order of the messages

Extra:

- **Class Diagram (Structure)** → shows the **structure** of the system (objects and relationships).
- **Sequence Diagram (Behavior)**→ shows the **behavior over time** (order of interactions) So it:
  - ✓ Show the **order of interactions** between objects.
  - ✓ Explain **system behavior step by step**.
  - ✓ Clarify **who does what** in a use case.

* Example taken from: http://www.programsformca.com/2012/03/uml-diagrams-library-mgmt-system.html

The Process View

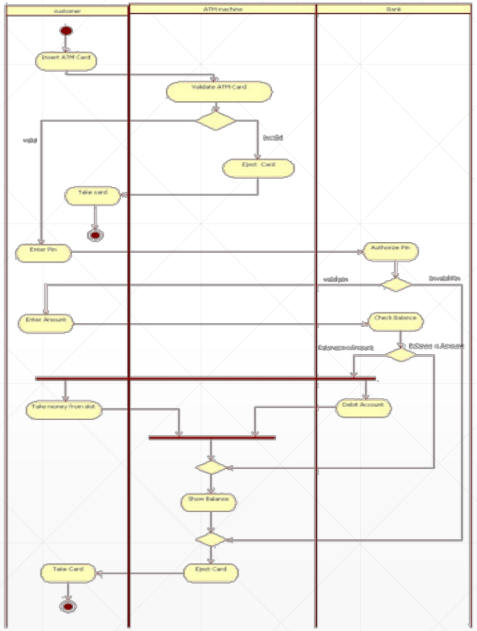
- The process view focuses on the dynamic aspects of the system, i.e., its execution time behavior.
- This view is an abstraction of processes or threads
- Shows process synchronization and concurrency.
- It presents how all execution units are organized at run-time
- It contributes to non-functional requirements and quality attributes such as scalability and performance requirements.
- The UML activity diagram and interaction overview diagram support this view.
- The stakeholders of this view are usually the developers and integrators

Extra:

•**Concurrency**: Looks at how tasks that run at the same time interact and coordinate.

•**Synchronization**: Manages how processes or tasks share resources or data without interfering with each other.

*ATM Example – Process View (Activity Diagram)



Activity diagram: helps to describe the flow the system, and explore operations and processes

Note: click on the diagram to view an online larger size diagram

* Example taken from: http://www.programsformca.com/2012/03/uml-diagrams-library-mgmt-system.html

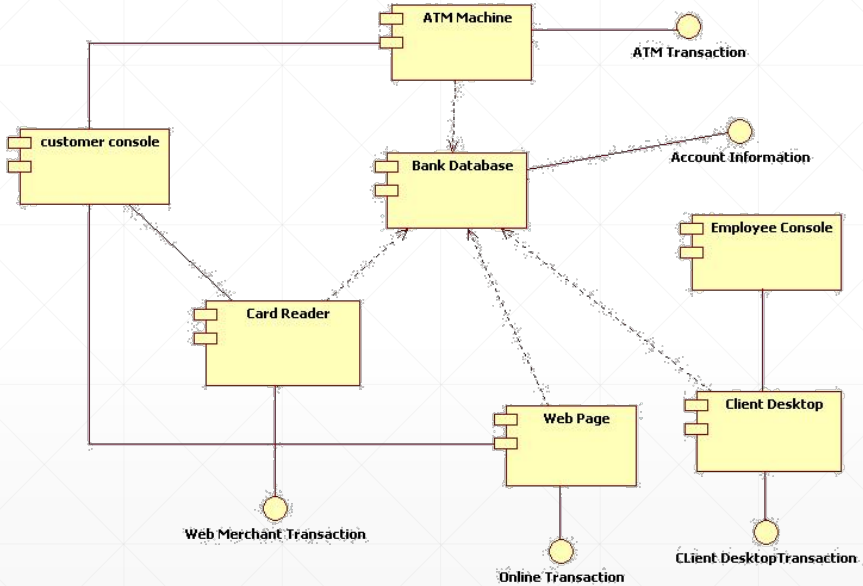
The development View

- The development view describes the software static organization of the system modules. (from implementation point of view)
- Modules such as class library, sub-system, or packages (building blocks that group classes)
- UML diagrams such as package diagrams and component diagrams are often used to support this view.
- This view can be used to analyze the system from the perspective of how logical components map to physical files and directories.
- The stakeholders of this view can be programmers and software project manager

Extra:
The development View means how source code is organized into **modules, packages, subsystems, and libraries**.

*ATM Example – Development View (Component Diagram)

Component diagram: shows components of a system and required interfaces, ports, and relationships between them



Full banking system

* Example taken from: http://www.programsformca.com/2012/03/uml-diagrams-library-mgmt-system.html

The Physical View

- The physical view represents the deployment aspects of software systems (from configuration and installation point of view)

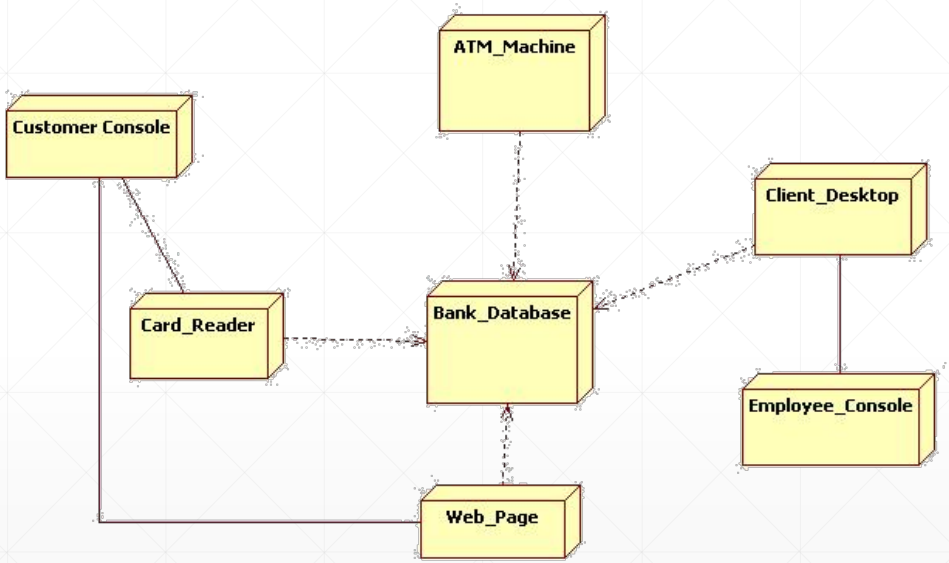
- The physical view can be modeled in UML using:
  - ✓ Deployment diagrams

- The physical view is appropriate to evaluate the system's:
  - ✓ Availability: for example, specifying the need for redundant nodes to increase the system's availability
  - ✓ Performance and scalability: for example, required processor speed, bandwidth, etc. to meet performance requirements.
  - ✓ Security: for example, required firewalls

- The stakeholders of this view are system installers, system administrators, and system engineers

*ATM Example – Physical View (Deployment Diagram)

Deployment diagram:
A diagram used to model the physical structure of a software system



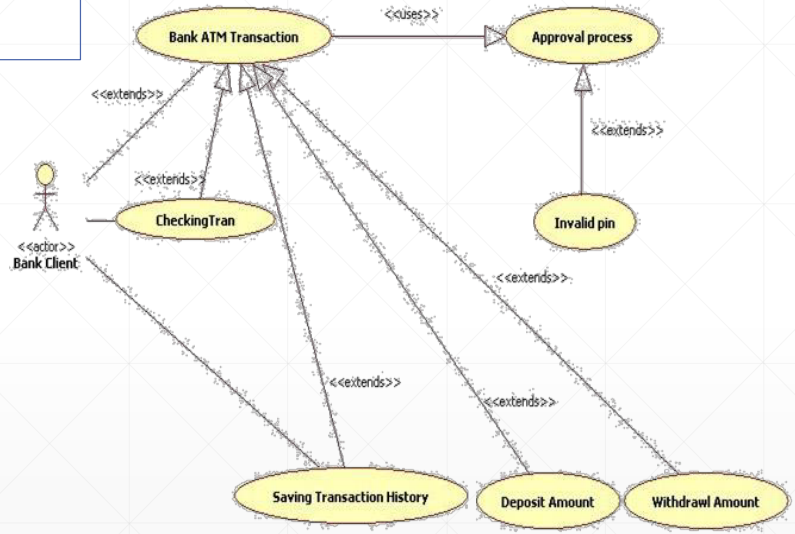
* Example taken from: http://www.programsformca.com/2012/03/uml-diagrams-library-mgmt-system.html

The User View (or Scenario view)

- The scenario view describes the functionality of the system, i.e., how user will use the system and how the system provides services to the users.
- This view provides a foundation for other 4 views and allow them to work together seamlessly and logically.
- It helps designers discover architectural elements during design and helps to validate the architectural design afterward.
- So, the scenario view helps to make the software architecture consistent and validated against functional and non-functional requirements.

*ATM Example – User View (Use case diagram)

Use case diagram:
Behavioral diagram used to capture, specify, and visualize required system behavior



* Example taken from: http://www.programsformca.com/2012/03/uml-diagrams-library-mgmt-system.html

Extra:

In the 4+1 View Model

1. Logical View (what the system should do)

- **Focuses on** functionality for end-users.
- **Typical diagrams**:
  - **Class Diagram** (static structure).
  - **Sequence Diagram** (dynamic behavior).
  - **Collaboration Diagram** (object interactions).

2. Process View (how the system works at runtime)

- **Focuses on** concurrency, control flow, and processes.
- **Typical diagrams:**
  - **Activity Diagram** (workflow, actions).
  - **Interaction Overview Diagram** (high-level control of interactions).
  - **State Machine Diagram** (object life cycles).

3. Development View (how the system is built)

- **Focus on** Software organization in code.
- **Typical diagrams:**
  - **Package Diagram.**
  - **Component Diagram.**

4. Physical View (how **software (logical components)** is mapped to **hardware (servers, devices)).**

- **Focus on where components run** (on which hardware or node).**)**
- **Typical diagrams:**
  - **Deployment Diagram**

5. The "+1" User (Scenario) View

- **Focuses on** capture **user scenarios and requirements** to drive the other four views
- **Typical diagrams**:
  - **Use cases Diagram, Textual scenarios**

The 4+1 View Model (Summary)

*	Logical	Process	Development	Physical	Scenario
Description					
Stake holder					
Consider					
UML – Diagram					

* This table is taken from: https://www.tutorialspoint.com/software_architecture_design/architecture_models.htm

Summary

- Designing the Architecture with the **4+1view** model
  - ✓ The **Logical** view
  - ✓ The **process** view
  - ✓ The **development** view
  - ✓ The **physical** view
  - ✓ Plus … The **user** view