ECE 811 – SOFTWARE ENGINEERING

SOFTWARE ARCHITECTURE VIEWS –STUDY GUIDE/REVISION

---

1. INTRODUCTION TO SOFTWARE ARCHITECTURE VIEWS

1. **Definition:** Software architecture is the **high-level structure** of a system, comprising components, relationships, and principles.
2. **Architecture Views** are **representations** of a system from different perspectives (e.g., structural, behavioural, deployment).
3. **Purpose:** Different stakeholders (developers, testers, managers) need different views to understand the system.
4. **Why Are Views Important?**
    - **Simplify complexity** by breaking down the system.
    - **Improve communication** among stakeholders.
    - **Guide development, testing, and maintenance.**
    - **Ensure alignment** with business and technical goals.

2. THE 4+1 VIEW MODEL (PHILIPPE KRUCHTEN, 1995)

A widely used framework for documenting software architecture:

View	Description	Stakeholders	Key Artifacts
Logical View	Shows functional components and interactions.	Developers, Architects	Class diagrams, component diagrams
Process View	Describes runtime behaviour, concurrency, and threads.	System engineers	Activity diagrams, sequence diagrams
Development View	Code organization, modules, and dependencies.	Developers, DevOps	Package diagrams, build scripts
Physical View	Deployment on hardware, servers, and networks.	Operations, Sysadmins	Deployment diagrams, network maps
Scenarios (+1)	Use cases or user stories that validate the architecture.	All stakeholders	UML use case diagrams, user flows

Example: E-Commerce System

1. **Logical View:** Product, Cart, Payment classes.
2. **Process View:** Order processing workflow.
3. **Development View:** Microservices in Java/Python.
4. **Physical View:** AWS cloud deployment.

5. **Scenarios:** "User checks out a product."

3. OTHER KEY ARCHITECTURE VIEWS

3.1 Static vs. Dynamic Views

Type	Focus	Examples
Static	Structure (components, modules)	Class diagrams, package diagrams
Dynamic	Runtime behavior (interactions, flows)	Sequence diagrams, state machines

3.2 C4 Model (Simon Brown)

A simpler alternative for modern software:

1. **Context:** System scope and external interactions.
2. **Containers:** Applications/services (e.g., web app, database).
3. **Components:** Modular parts within containers.
4. **Code:** Class-level implementation.

3.3. Data View (Database-Centric)

- Focuses on **data storage, flow, and schema**.
- Used in **data-intensive systems** (e.g., banking, analytics).
- **Artifacts:** ER diagrams, data flow diagrams.

4. CHOOSING THE RIGHT VIEWS

- **For Enterprise Systems:** 4+1 View Model.
- **For Microservices:** C4 Model + Deployment View.
- **For Embedded Systems:** Process View + Physical View.

Trade-offs

- **Over-documentation** slows agility.
- **Under-documentation** risks misunderstandings.

5. CASE STUDY: UBER'S ARCHITECTURE VIEWS

1. **Logical View:** Ride-matching algorithms.
2. **Process View:** Real-time driver-passenger coordination.
3. **Development View:** Microservices (Go, Node.js).
4. **Physical View:** Kubernetes clusters globally.

5. **Scenarios:** "User requests a ride."

6. KEY TAKEAWAYS

1. **No single view captures everything**—combine views for clarity.
2. **Tailor views to stakeholder needs** (e.g., managers vs. devs).
3. **Update views as the system evolves** (avoid "architectural drift").

7. FURTHER READING

- Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice*.
- Kruchten, P. (1995). *The 4+1 View Model of Architecture.*
- Brown, S. (2018). *"The C4 Model for Visualising Software Architecture."*

SHORT REVIEW QUESTIONS

1. Which view maps software to **hardware nodes**?
2. Why is the **Scenarios view** (+1) critical?
3. Compare **Logical vs. Process Views**.
4. When would you use the **C4 Model** instead of 4+1?

Answers:

1. **Physical View.**
2. **Validates all other views with real-world usage.**
3. **Logical = Structure; Process = Runtime behavior.**
4. **For simpler, modern systems (e.g., microservices).**