Final Lecture – UML Design: A Comprehensive Summary

Traian Florin Șerbănuță

2025

Goal

Consolidate UML knowledge, reflect on modeling choices, and connect diagrams, design patterns, and evaluation techniques.

Agenda

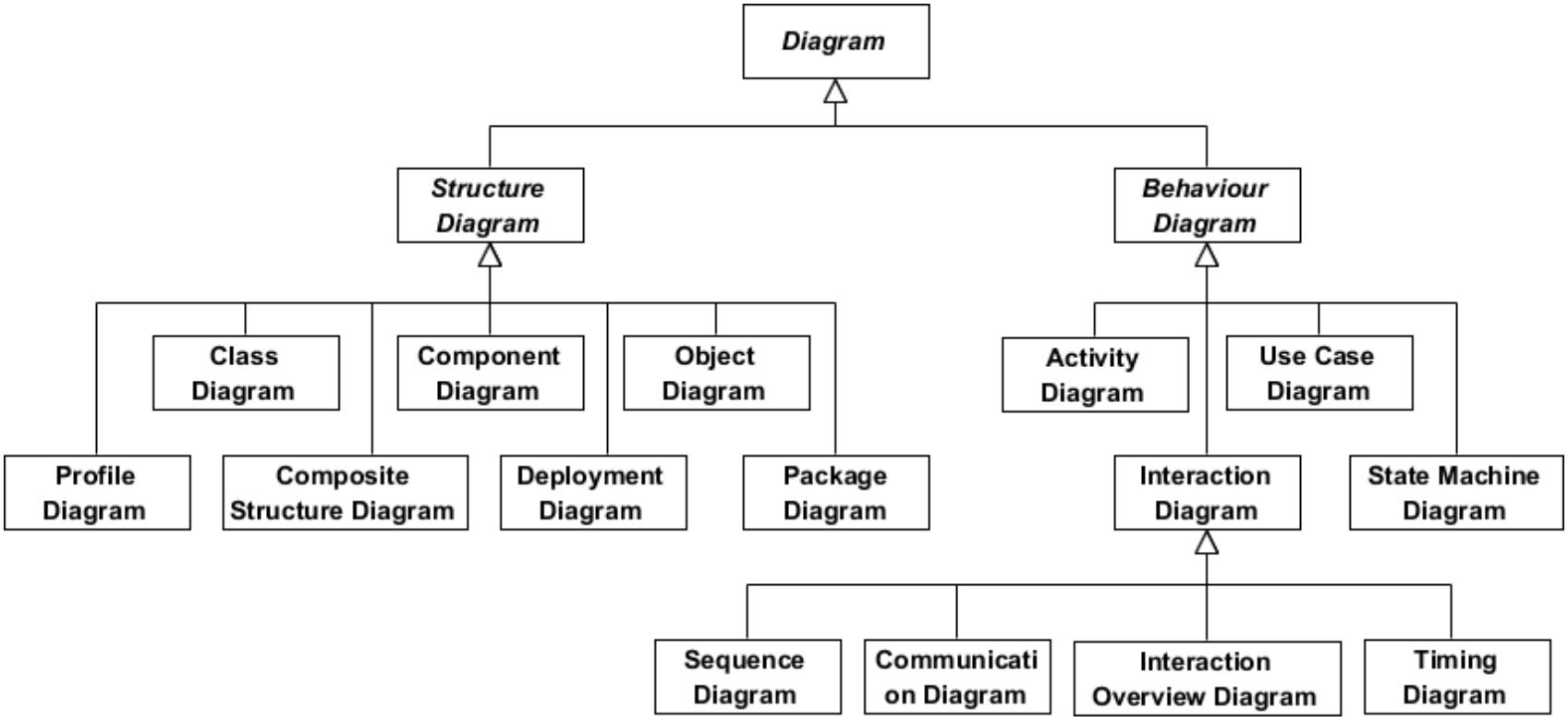
1. Why UML still matters
2. UML diagram landscape (overview)
3. Behavioral modeling recap
4. Structural modeling recap
5. Design patterns & architecture recap
6. Model evaluation & quality
7. Reflection & discussion exercises

Foundations & Behavioral View

Why UML?

- Common visual language for software design
- Supports **analysis → design → implementation → testing**
- Bridges communication gaps between stakeholders
- Enables reasoning before code exists
  UML is not about drawing diagrams — it is about thinking structurally and behaviorally.

UML Diagram Landscape



Behavioral Modeling – What We Learned

Use Case Diagrams

- Capture system *goals*, not implementation
- Define system boundary and actors

Sequence & Communication Diagrams

- Describe object interactions over time
- Message consistency with class operations

State Diagrams

- Model lifecycle and event-driven behavior
- Excellent for controllers, protocols, devices

Activity Diagrams

- Model workflows and business logic
- Emphasize concurrency and control flow

Interactive Discussion

Prompt

- Which behavioral diagram did you find *most useful*?
- Which one felt *least intuitive*?
- Where did diagrams help clarify misunderstandings?

Students discuss in small groups $\rightarrow$ short plenary feedback.

Structure, Design & Evaluation

Structural Modeling – What We Learned

Class & Object Diagrams

- Static structure vs runtime snapshots
- Responsibility-driven design

Package Diagrams

- Manage complexity
- Support layering and modularization

Component & Deployment Diagrams

- Architectural view
- Mapping software to hardware

Composite Structure Diagrams

- Internal structure of complex classifiers

UML Meta-Model & Profiles

- UML defined using MOF (M3 $\rightarrow$ M2 $\rightarrow$ M1 $\rightarrow$ M0)
- Profiles customize UML for domains
- Stereotypes, tagged values, constraints
  Profiles adapt UML without changing the language itself.

Design Patterns Recap

Categories

- **Creational**: Builder, Singleton, Factory Method
- **Structural**: Bridge, Adapter, Decorator, Proxy, Composite
- **Behavioral**: Iterator, Observer, State, Visitor, Mediator

Key Lesson

- Patterns encode *proven design decisions*
- UML helps document and communicate patterns

Evaluating & Testing UML Models

Evaluation Dimensions

- Syntactic
- Semantic
- Pragmatic
- Organizational
- Technical

Techniques

- Reviews & checklists
- Cross-diagram consistency
- OCL constraints
- Simulation & model-based testing

Interactive Exercise (Group Task)

Choose one of the following

1. A UML diagram type you used in a project
2. A design pattern from the course
3. A modeling mistake you now recognize

Discuss

- What problem does it solve well?
- When would it *not* be appropriate?
- How would you explain it to a junior engineer?

Each group shares **one insight**.

Final Reflection

- UML is a *toolbox*, not a checklist
- Choose diagrams intentionally
- Prefer clarity over completeness
- Models should evolve with understanding
  Good models ask good questions — not just provide answers.

Closing & Course Takeaways

- You can now **read, critique, and design** UML models
- You understand how diagrams relate and complement each other
- You can evaluate model quality and apply patterns consciously

Thank you for the engagement throughout the course!