

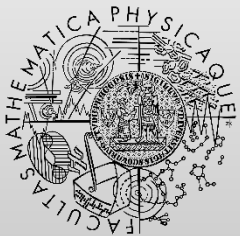
# UML: Unified Modeling Language

<http://d3s.mff.cuni.cz>

Department of  
Distributed and  
Dependable  
Systems



*Pavel Parízek*



FACULTY  
OF MATHEMATICS  
AND PHYSICS  
Charles University

# What is UML

- General-purpose graphical notation for modeling software systems
  - with formal semantics
- Many different aspects (viewpoints)
  - architecture of the system
  - processes (behavior)
  - states and transitions
  - interaction of components
- Levels of abstraction
  - conceptual
  - implementation

# Basic perspective on usage

- Creating nice large and complex diagrams
  - Various aspects of software systems
- But there is formal semantics too
  - Allows for validation, reasoning about models, and generating code
- Relatively wide adoption
  - Who: business analysts, designers, architects
- Supported by many CASE tools and IDEs
  - CASE = computed-aided software engineering

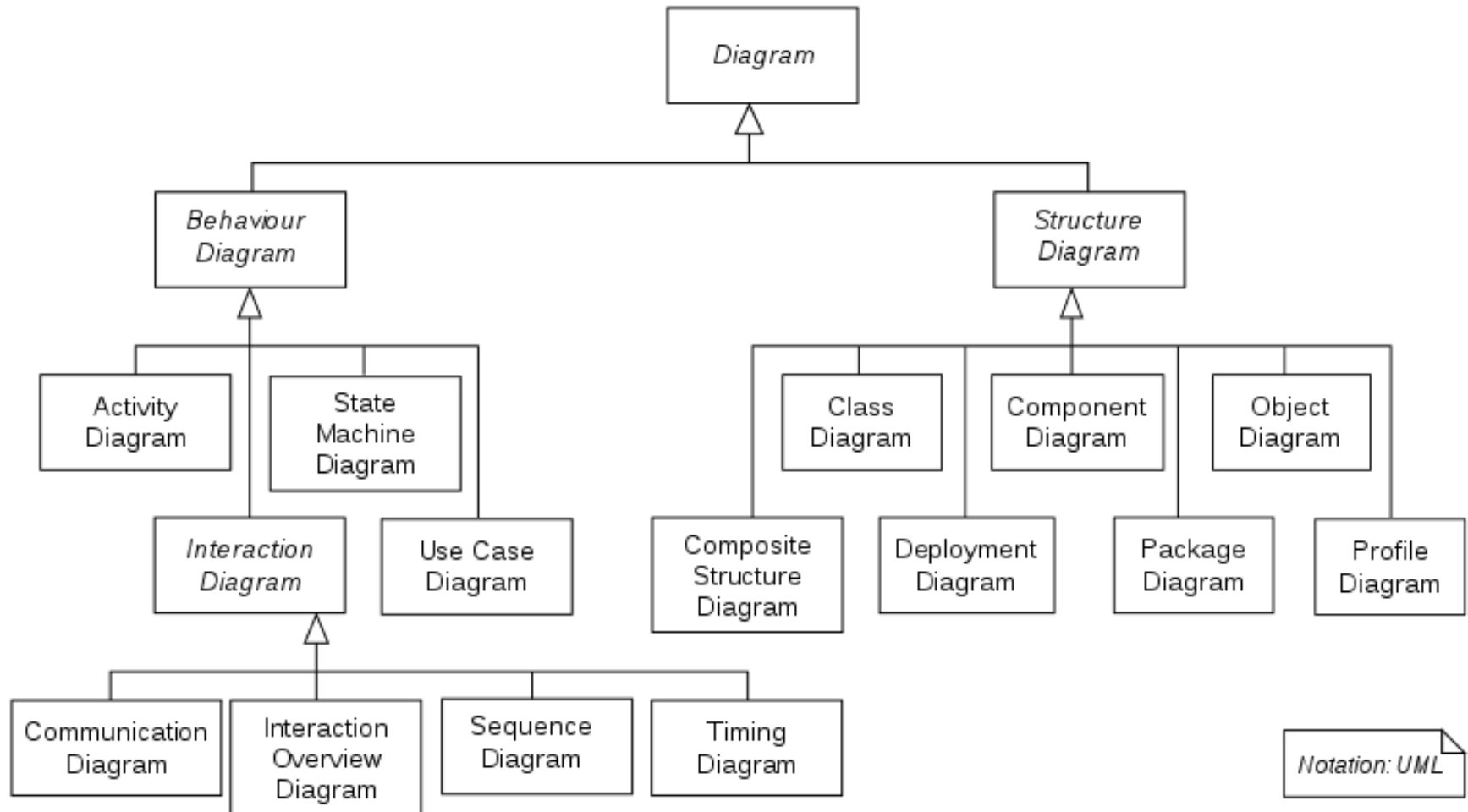
# Official information

- Maintainers
  - Object Management Group (OMG)
- Industry standard
  - ISO/IEC 19501
- Resources
  - Official website (home page): <http://uml.org/>
  - Specification: <https://www.omg.org/spec/UML>
  - [https://en.wikipedia.org/wiki/Unified\\_Modeling\\_Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language)

# UML diagrams

- Structure
  - Class diagram
    - [https://en.wikipedia.org/wiki/Class\\_diagram](https://en.wikipedia.org/wiki/Class_diagram)
  - Component diagram
  - ...
- Behavior
  - Use case diagram
    - [https://en.wikipedia.org/wiki/Use\\_case\\_diagram](https://en.wikipedia.org/wiki/Use_case_diagram)
  - Activity diagram
    - [https://en.wikipedia.org/wiki/Activity\\_diagram](https://en.wikipedia.org/wiki/Activity_diagram)
  - Sequence diagram
    - [https://en.wikipedia.org/wiki/Sequence\\_diagram](https://en.wikipedia.org/wiki/Sequence_diagram)
  - State machine
  - ...

# UML diagrams – complete schema



Source: [https://en.wikipedia.org/wiki/Unified\\_Modeling\\_Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language)

# Tools

- Free tools for creating UML diagrams
  - draw.io: <https://app.diagrams.net/>
  - PlantUML: <https://plantuml.com>, <https://www.planttext.com/>
- Microsoft Visio
  - <https://www.microsoft.com/cs-cz/microsoft-365/visio/flowchart-software>
- Enterprise Architect
  - <https://sparxsystems.com/>
  - [https://en.wikipedia.org/wiki/Enterprise\\_Architect\\_\(software\)](https://en.wikipedia.org/wiki/Enterprise_Architect_(software))
- Plugins for IDEs
  - IntelliJ (UML Generator)
    - <https://plugins.jetbrains.com/plugin/15124-uml-generator>
  - Visual Studio (Class Designer)
    - <https://learn.microsoft.com/en-us/visualstudio/ide/class-designer/designing-and-viewing-classes-and-types?view=vs-2022>

# Class diagrams

- Purpose: modeling structure of the system
  - Classes that represent sets of objects (real-world entities from a given domain) with the same characteristics (properties, features, constraints)
  - Various relationships between the objects
- Used at two levels
  - Conceptual (domain): where the domain entities and relations are captured
  - Implementation: which maps directly to source code in a programming language

# Class diagrams – elements

- Classes
  - Basic information (name)
  - Attributes (fields)
    - name, type, multiplicity (number of values)
  - Operations (actions)
  - Endpoints for associations
- Relationships
  - Association
  - Composition
  - Aggregation

# Class diagrams – associations

- Association ends labeled with
  - Relationship meaning (semantics)
  - Multiplicity (cardinality):  $0/1..N$
- Binary
- N-ary
  - three or more endpoints
- Association classes

# Class diagrams – part-of relationships

- Composition
  - Parts are not shared with other owners
  - Individual parts cannot exist without their owners
- Aggregation
  - Parts may be shared with other owners

# Class diagrams – inheritance

- Specialization
  - Generalization
- 
- Expected semantics
    - As in common programming languages

# Class diagrams – operations

- Operation = action that can be performed on class instances
- May be annotated with
  - pre-condition
  - post-condition
  - special body-condition over the result

# Component diagram

- Purpose
  - How the components are connected together
- Entities
  - Components (with names)
  - Provided interfaces
  - Required interfaces
  - Bindings (connectors)
- Resources
  - [https://en.wikipedia.org/wiki/Component\\_diagram](https://en.wikipedia.org/wiki/Component_diagram)

# Activity diagram

- Purpose
  - Modeling various processes (computations)
- Entities
  - Actions (transitions)
  - Decisions (choices)
  - Concurrency (fork/join)
- Resources
  - [https://en.wikipedia.org/wiki/Activity\\_diagram](https://en.wikipedia.org/wiki/Activity_diagram)

# Sequence diagram

- Purpose
  - Modeling interaction between objects and processes in terms of events over time
- Entities
  - Objects (processes)
  - General timeline
  - Procedure calls
  - Network messages
  - Scope of execution
- Resources
  - [https://en.wikipedia.org/wiki/Sequence\\_diagram](https://en.wikipedia.org/wiki/Sequence_diagram)

# UML profiles

- Extensions for specific domains
- SysML
  - Target domain: large systems, not just software
  - [https://en.wikipedia.org/wiki/Systems\\_modeling\\_language](https://en.wikipedia.org/wiki/Systems_modeling_language)
  - <https://sysml.org/>

# Validating (against) UML models

- Translation into formal specification language
  - Alloy, Maude, VDM, Z, ...
- Model-based testing (MBT)