

# Software Modeling and UML

Dr. Rodrigo Spínola



Lecture 20 - Software  
Modeling and UML

**TDresearchteam**  
Technical Debt Research Team



2

## Last time

- **All software-intensive systems have an architecture**
- How much effort should you put into varies greatly
- Most of the time, the architecture is implicit
  - Choice of technology, platform
  - Still need to understand the architecture
- Key drivers are mostly non-functional
  - Runtime: capacity, performance, scalability, security
  - Non runtime: evolvability, regulatory,...



Lecture 20 - Software  
Modeling and UML

**TDresearchteam**  
Technical Debt Research Team



3

## Last time

**Architecting is Making Decisions!**

**The life of a software architect is a long (and sometimes painful) succession of suboptimal decisions made partly in the dark.**



Lecture 20 - Software Modeling and UML

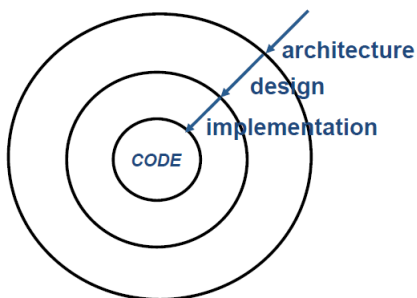
**TDresearchteam**  
Technical Debt Research Team



4

## Last time

- Architecture involves a set of strategic design decisions, rules or patterns that constraint design and code



Architecture decisions are the **most fundamental decisions** and changing them will have **significant ripple impacts**.



Lecture 20 - Software Modeling and UML

**TDresearchteam**  
Technical Debt Research Team



5

# Agenda

- Software Modeling
- UML
- UML Diagrams



Lecture 20 - Software  
Modeling and UML

**TDresearchteam**  
Technical Debt Research Team



6

# Why model software?

- Engineers have always modeled things they are planning to build
- Displays an engineered system at a particular level of abstraction
- Helps one think clearly about the system
- Crucial in communicating to others the structure of a system
- Makes working in a team possible



Lecture 20 - Software  
Modeling and UML

**TDresearchteam**  
Technical Debt Research Team



7

# UML

due to its semantic

UML is a **graphical language** for **visualizing**, **specifying**, **constructing**, and **documenting** the artifacts of a software-intensive system.

- Consists of several different diagram types
- Can be used at different abstraction levels
  - From business process to individual language statements
- **It is a language, not a method or procedure!**



Lecture 20 - Software  
Modeling and UML

**TDresearchteam**  
Technical Debt Research Team



8

## UML Diagrams

- UML defines several diagram types, divided into two general sets:
  - Structural Modeling Diagrams:
    - Used to model “things” that make up a model – the classes, objects, interfaces, and components. In addition, they are used to model the relationships and dependencies between elements
  - Behavioral Modeling Diagrams:
    - Capture the varieties of interaction and instantaneous states within a model as it “executes” over time; tracking how the system will act in a real-world environment, and observing the effects of an operation or event, including its results



Lecture 20 - Software  
Modeling and UML

**TDresearchteam**  
Technical Debt Research Team



# Structural UML

- **Class diagrams** define the basic building blocks of a model: the types, classes, and general materials used to construct a full model
- **Object diagrams** show how instances of structural elements are related and used at run-time
- **Component diagrams** are used to model higher level or more complex structures, usually built up from one or more classes, and providing a well-defined interface
- **Deployment diagrams** show the physical disposition of significant artifacts within a real-world setting



# Behavioral UML

- **Use case diagrams** are used to model user/system interactions. They define behavior, requirements and constraints in the form of scripts or scenarios
- **State machine diagrams** are essential to understand the instant-to-instant condition, or “run state” of a model when it executes
- **Communication diagrams** show the network, and sequence, of messages or communications between objects at run-time, during a collaboration instance
- **Sequence diagrams** are closely related to communication diagrams and show the sequence of message passed between objects using a vertical timeline
- **Timing diagrams** fuse sequence and state diagrams to provide a view of an object’s state over time, and messages which modify that state



# UML

- Using UML models for software maintenance
  - Scaniello et al. "On the impact of UML analysis models on source-code comprehensibility and modifiability". ACM TOSEM 2014
    - *"We carried out a family of experiments to investigate whether the use of UML models produced in the requirements analysis process helps in the comprehensibility and modifiability of source code. [...] The results of both the individual experiments and meta-analysis indicate that UML models produced in the requirement analysis process influence neither the comprehensibility of source code nor its modifiability."*
  - **In other words, UML diagrams must be kept up to date to be useful**



## Key Ideas in Software Modeling with UML

- Names are important
  - Give good names to classes in class diagram, use cases, etc.
- Provide only essential details
  - **Avoid over-modeling**
- Keep the model up to date
  - Easy for the model lose its usefulness if it's outdated



# Tool Support for UML

- Many specialized tools exist
  - Some integrated in IDEs
    - Extensions for Eclipse or other IDEs
    - Visual studio has native support for class and sequence diagrams
  - Standalone (e.g. MS Visio, Visual Paradigm)



Lecture 20 - Software  
Modeling and UML

**TDresearchteam**  
Technical Debt Research Team



**VCU**  
Computer Science  
College of Engineering



Class is  
over,  
questions?

# Software Modeling and UML

Dr. Rodrigo Spínola



Lecture 20 - Software  
Modeling and UML

**TDresearchteam**  
Technical Debt Research Team



**VCU**  
Computer Science  
College of Engineering