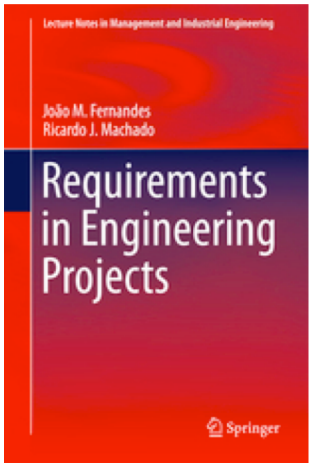
UML Modeling

João M. Fernandes
Universidade do Minho
jmf@di.uminho.pt, www4.di.uminho.pt/jmf

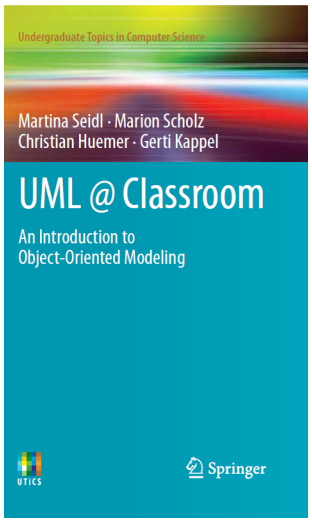
Literature



Requirements in engineering projects
João M. Fernandes and Ricardo J. Machado

Springer Publishing, 2016

ISBN 978-3-319-18596-5



UML @ Classroom:
An introduction to object-oriented modeling
Martina Seidl, Marion Scholz, Christian Huemer and Gerti Kappel

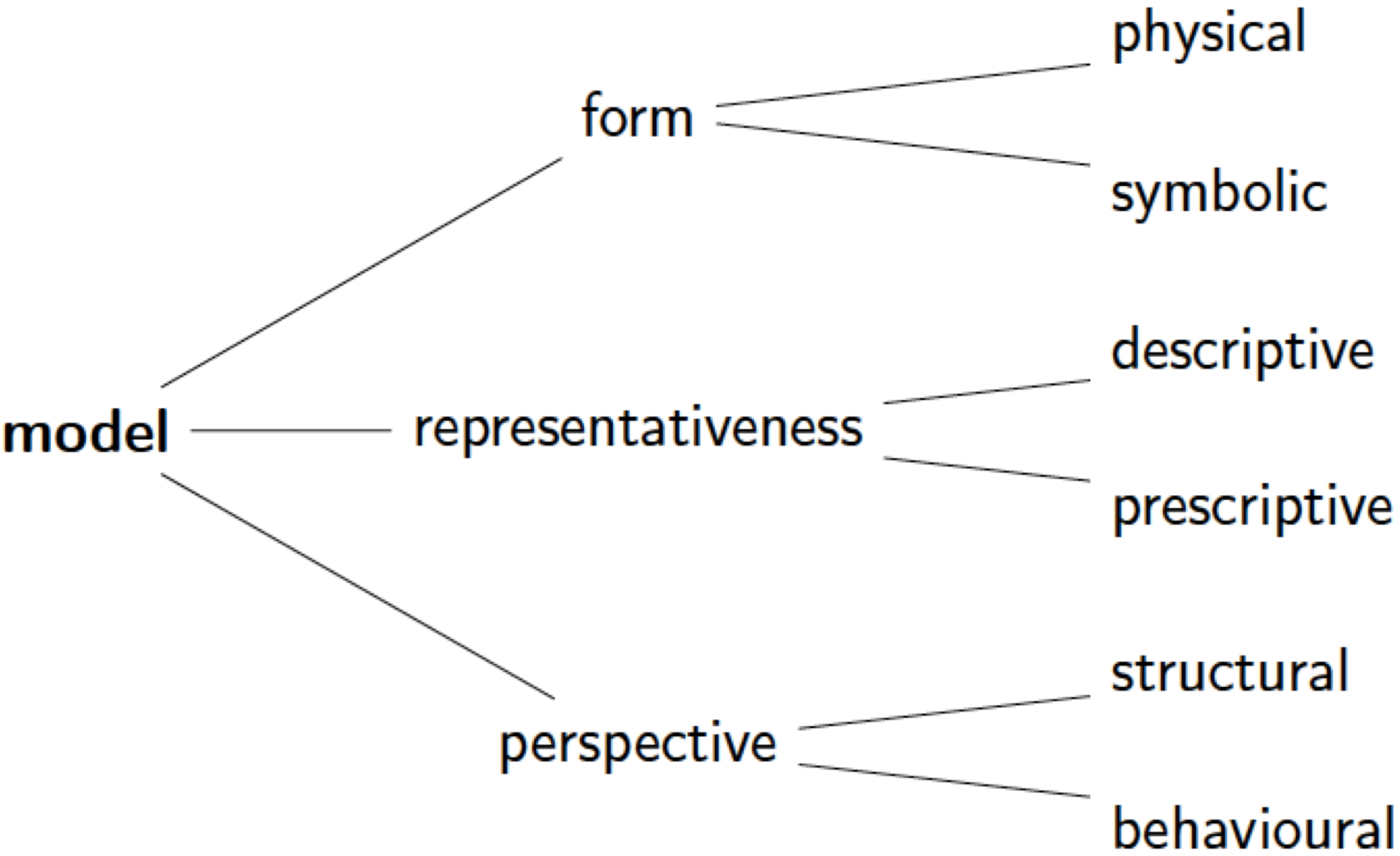
Springer Publishing, 2015

ISBN 978-3-319-12741-5

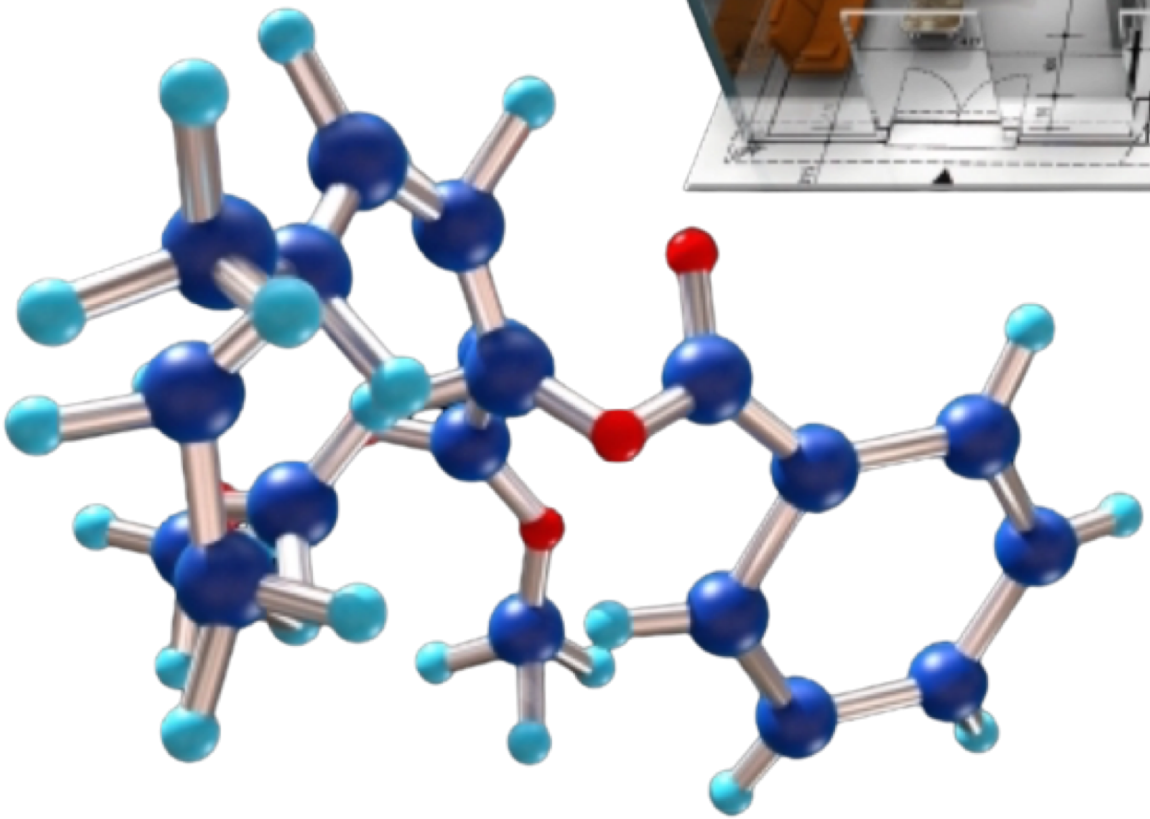
Modeling

- Modeling is an essential ingredient in all engineering fields
- It is a highly creative task
- Modeling is the process of:
  - identifying adequate concepts
  - selecting abstractions to construct a model that appropriately reflects a given universe of discourse
- Modeling permits the cost-effective use of the model in place of the real-world object/process for some **purpose**
- If it is to be useful, a model must not represent all aspects of reality
- Modeling is related to abstraction, simplification, and formalization
- The inverse operation to abstraction is refinement or concretization

Modeling: Dimensions



Modeling: Physical models



Modeling: Symbolic models

- A symbolic model (or mathematical) uses logical and quantitative relationships involving the dimensions of the system
- In electrical engineering, models are a set of equations that represent the electrical circuits by applying some laws
- An example is Ohm's law: V = R x I
- Typically, a symbolic model does not resemble the system it represents, but is fundamentally arbitrary or conventional
- Arbitrariness is the absence of any necessary connection between a linguistic form and its meaning
- A class is represented in UML by a rectangle, but this is a pure convention
- The word 'strawberry' has no relation to the fruit itself; it is a pure convention

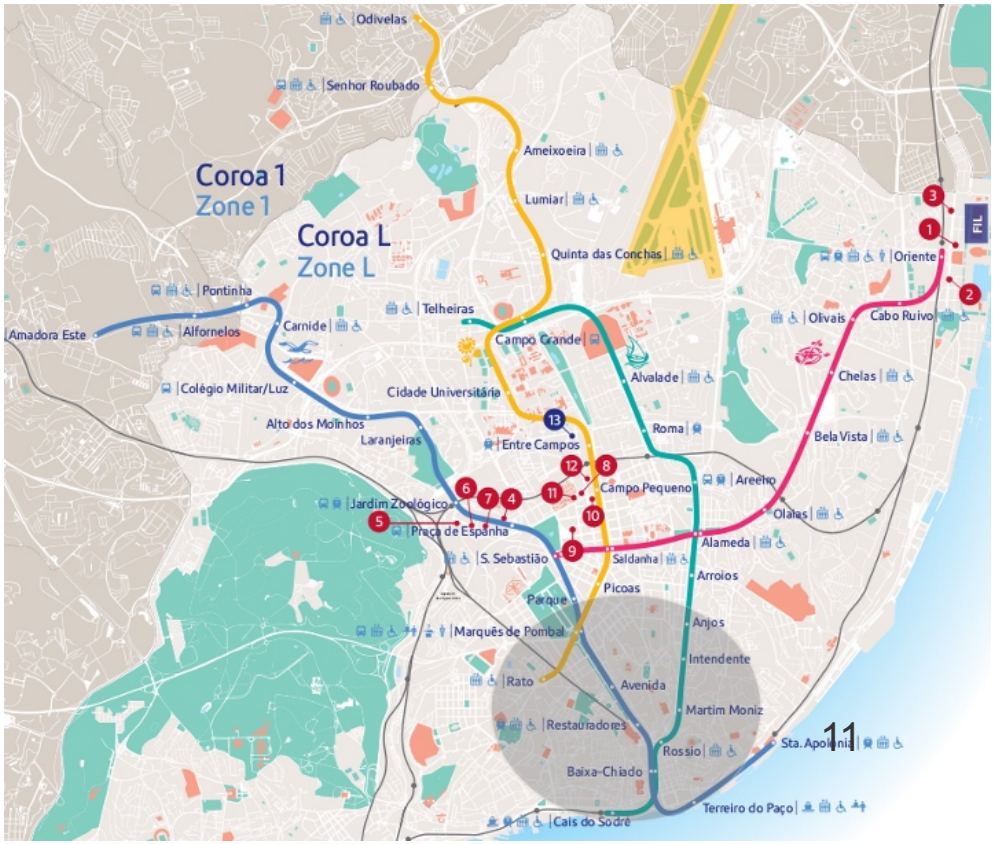
Modeling

symbolic model

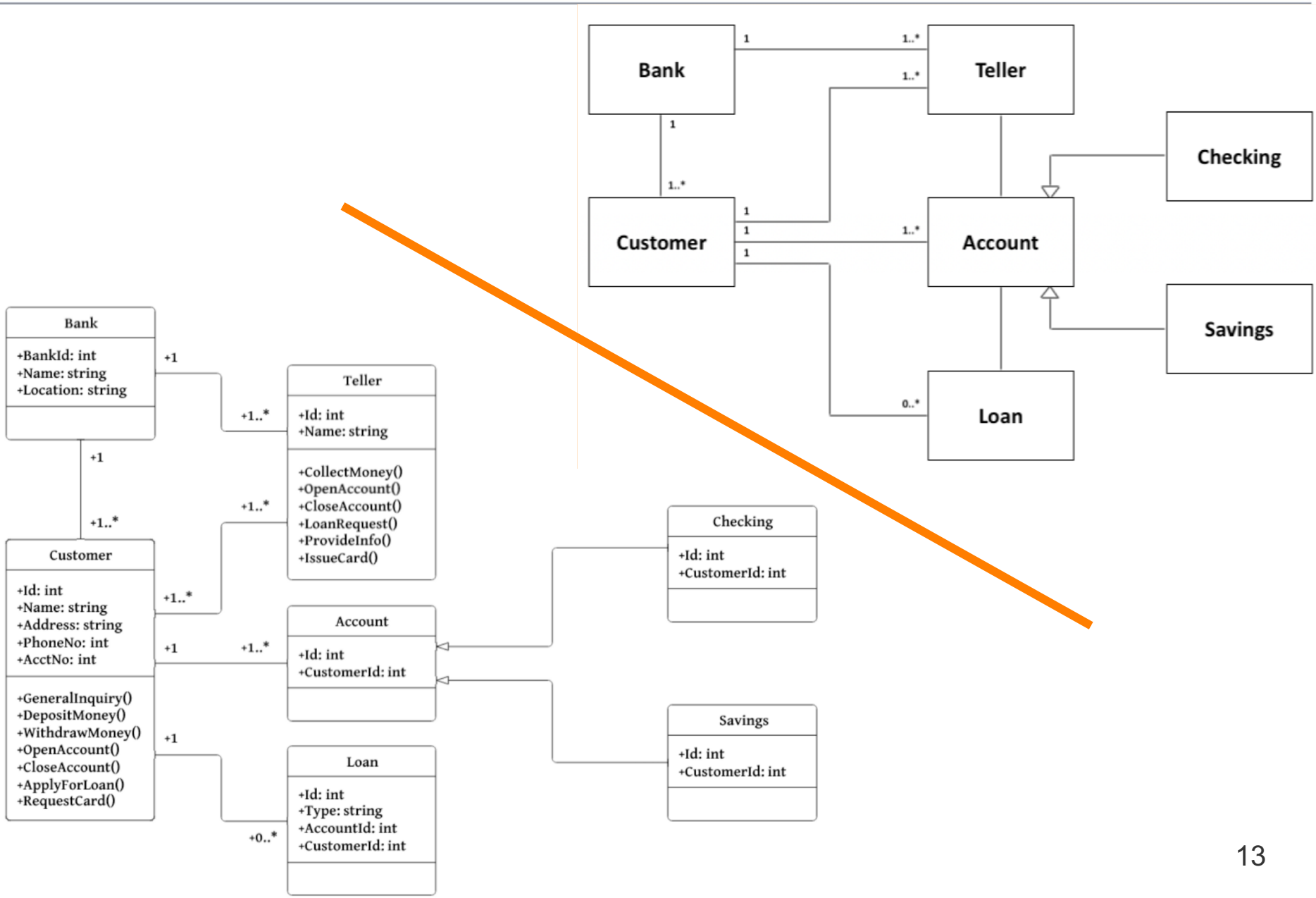
physical model

reality

Modeling: Topological and topographic maps



Modeling: Abstraction



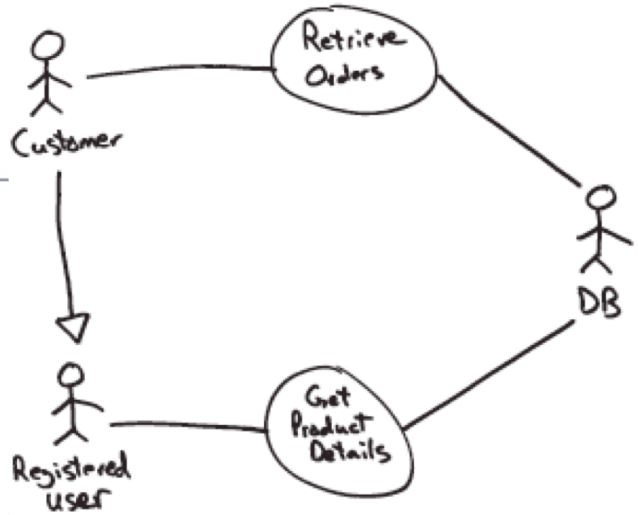
Modeling: Model properties

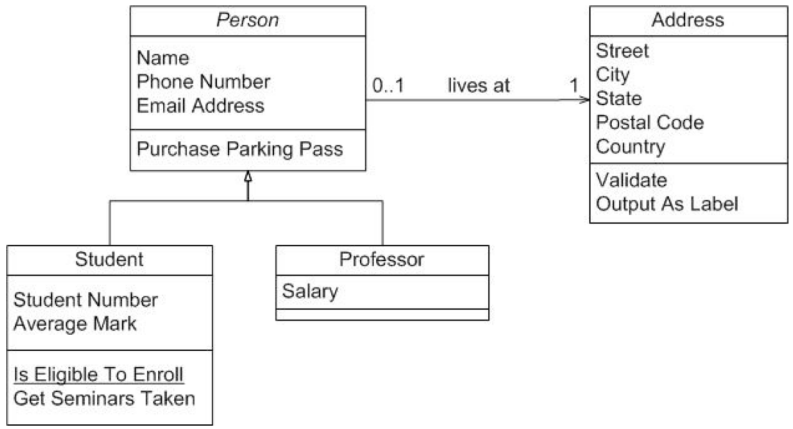
Models are simpler, safer and cheaper than reality:

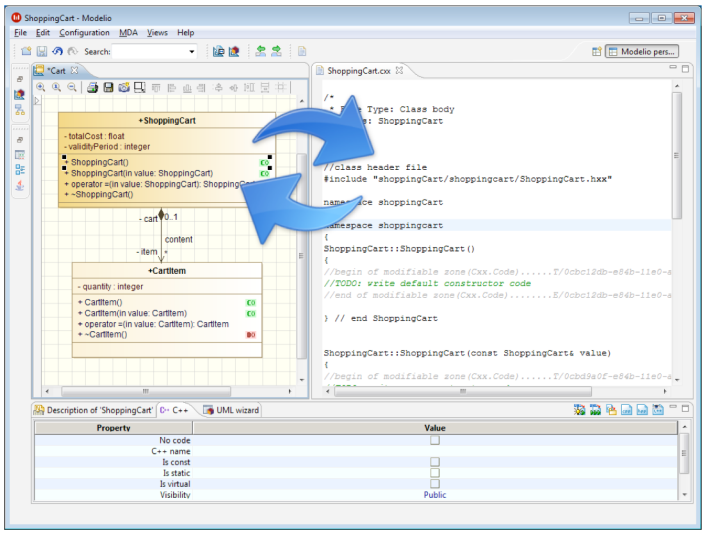
- **Abstraction**. A model is a reduced description of the system
- **Understandability**. By removing details that are irrelevant for a given viewpoint, models allows one to more easily understand some of the system properties
- **Accuracy**. For the properties of interest, a model provides a true-to-life representation of the system
- **Reasoning**. A model helps in correctly analyzing and reasoning about the interesting properties of the system, either through experimentation or formal analysis
- **Inexpensiveness**. A model must be drastically cheaper to construct and analyze than the system

UML

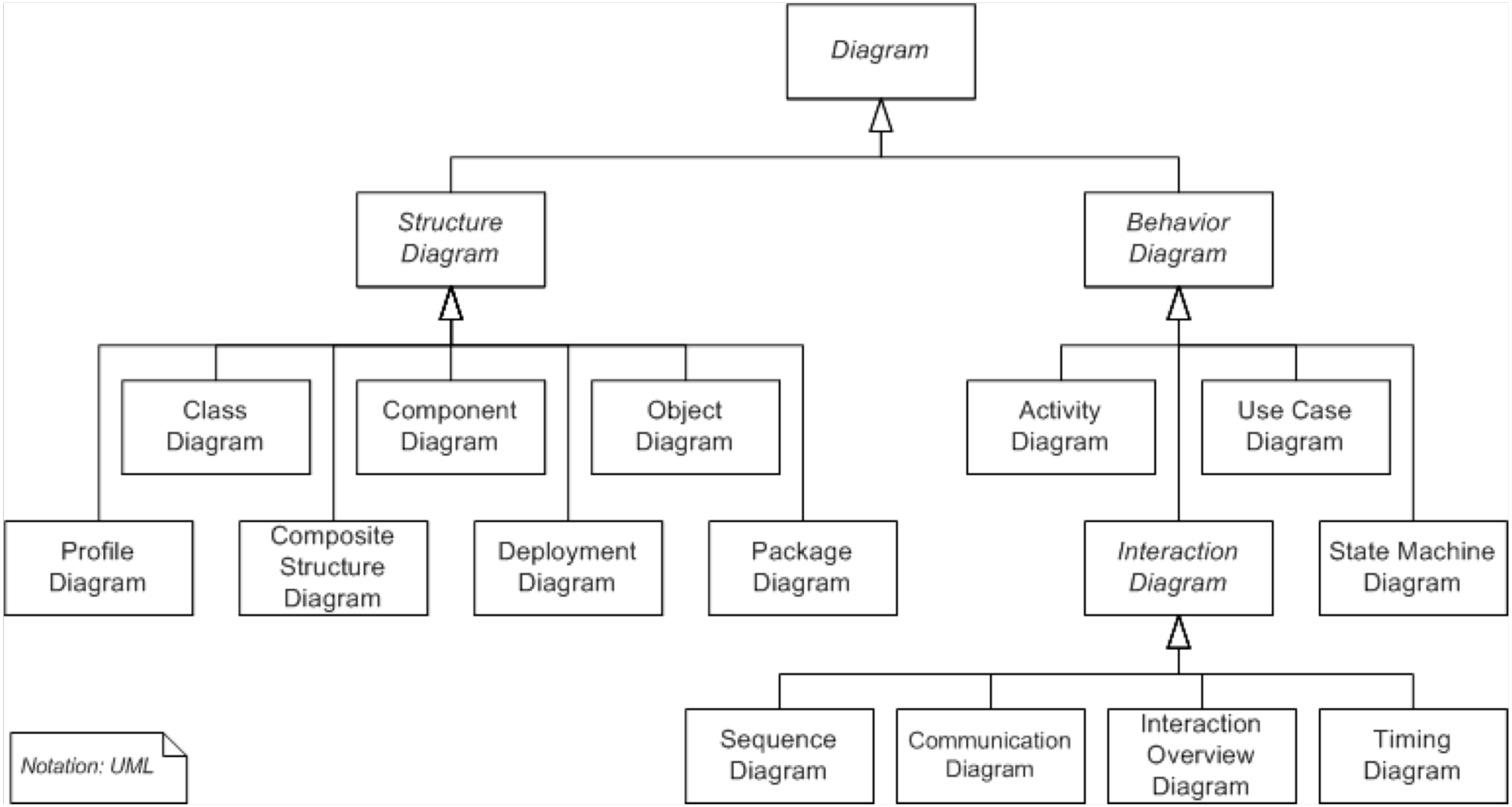
- Software engineers use the UML in 3 modes:
  - sketch (informal design)
  - blueprint (model-driven development)
  - programming language
- In **sketch**, the UML is used to help communicate some aspects of a system
- UML **blueprints** are detailed design artefacts used to produce code
- These models must contain sufficient detail to enable developers to create ready-to-run systems without having to make design decisions
- When all the system can be specified in the UML, we reach UML as **programming language**







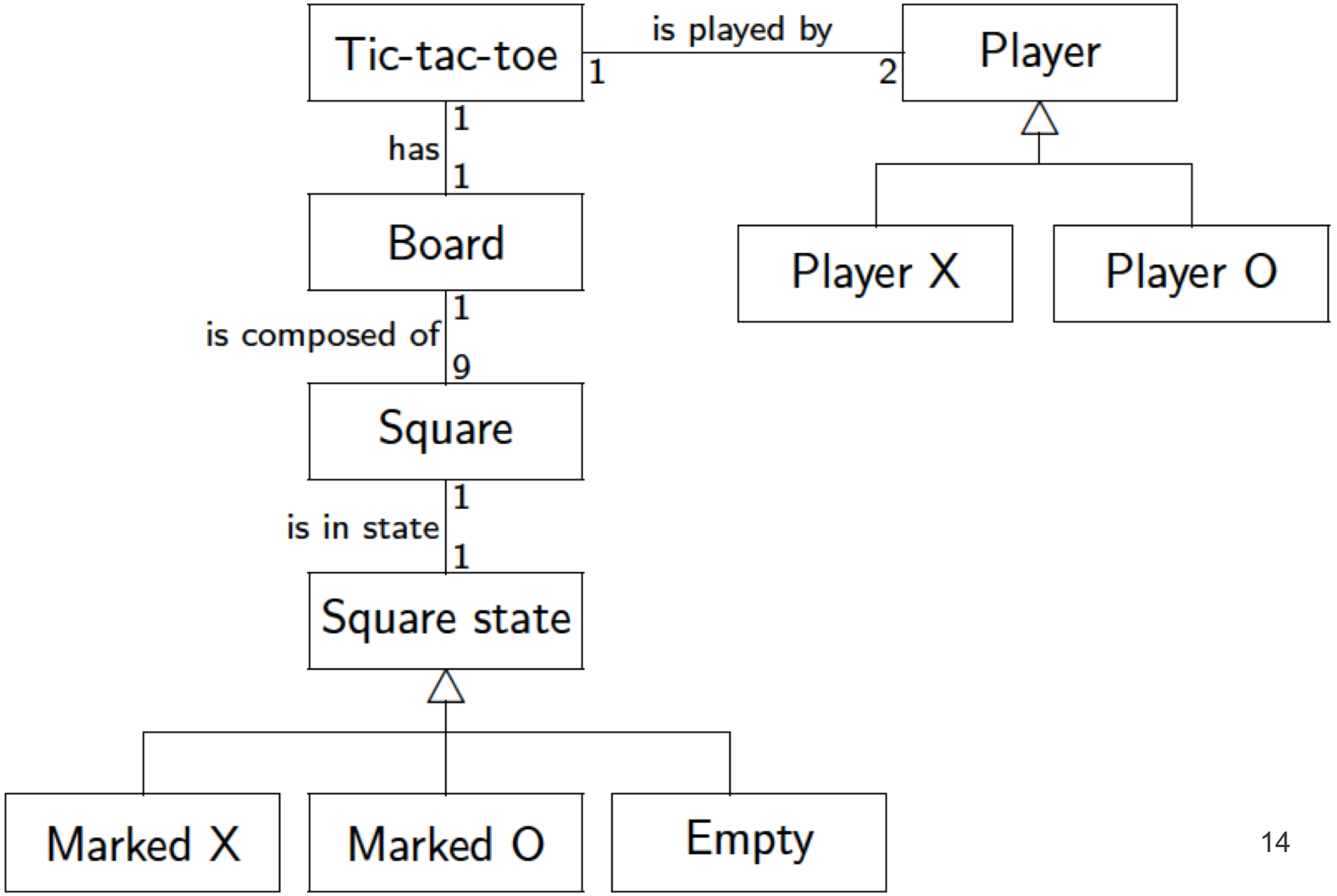
UML: Diagrams



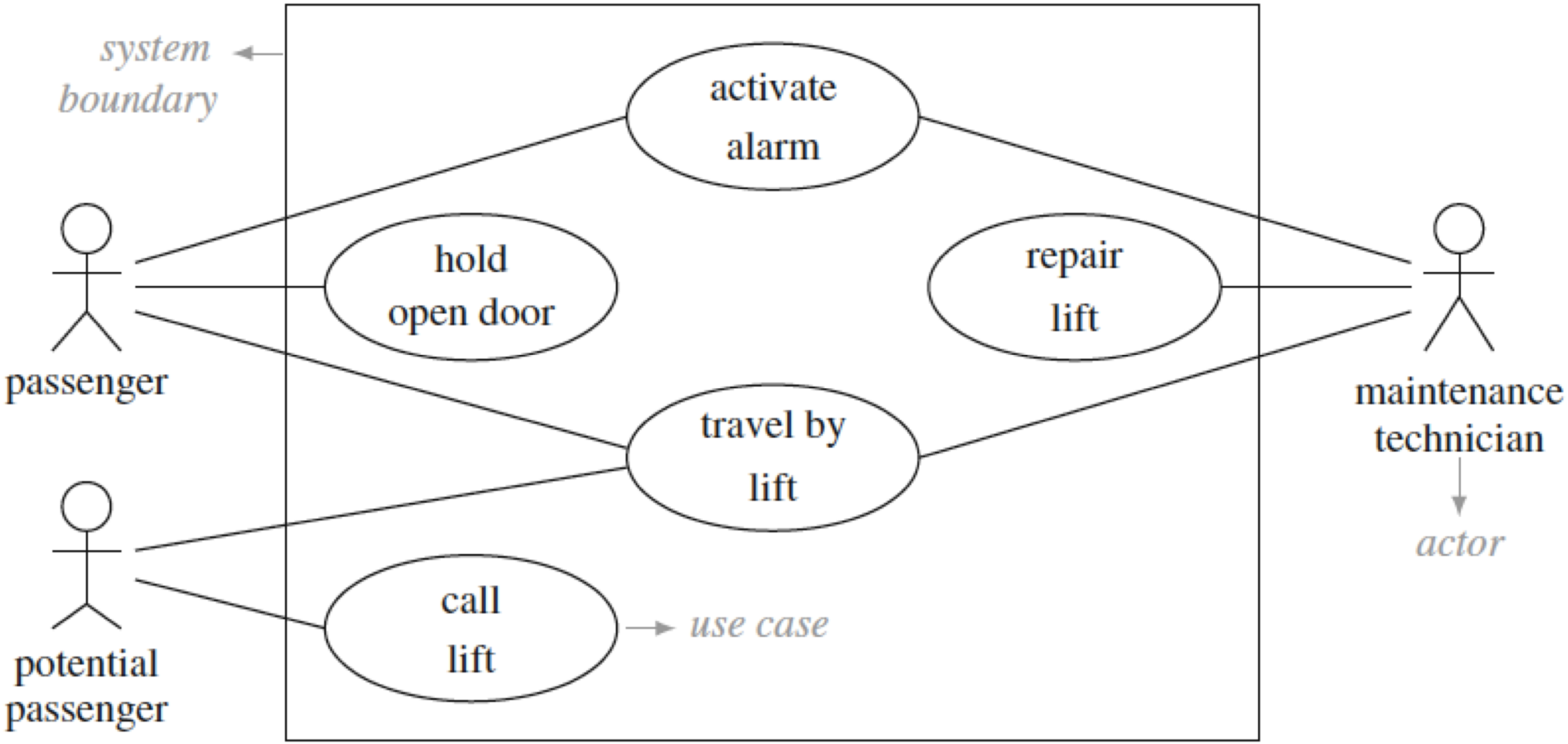
Essential UML

Model	Purpose
Domain	Describe the vocabulary, concepts of the domain and characteristics of the systems that can be developed for the considered domain
Use case	Describes the proposed functionalities of a given system
Interaction	Show how the various objects or entities collaborate, emphasising the flow of control and data among them
Class	Present a set of concepts, types and classes and the respective relations
State	Specify the behaviour of an entity or indicate the various states (or modes) through which it transits throughout its life
Activity	Show the control flow among the activities of a process

Essential UML: Domain model



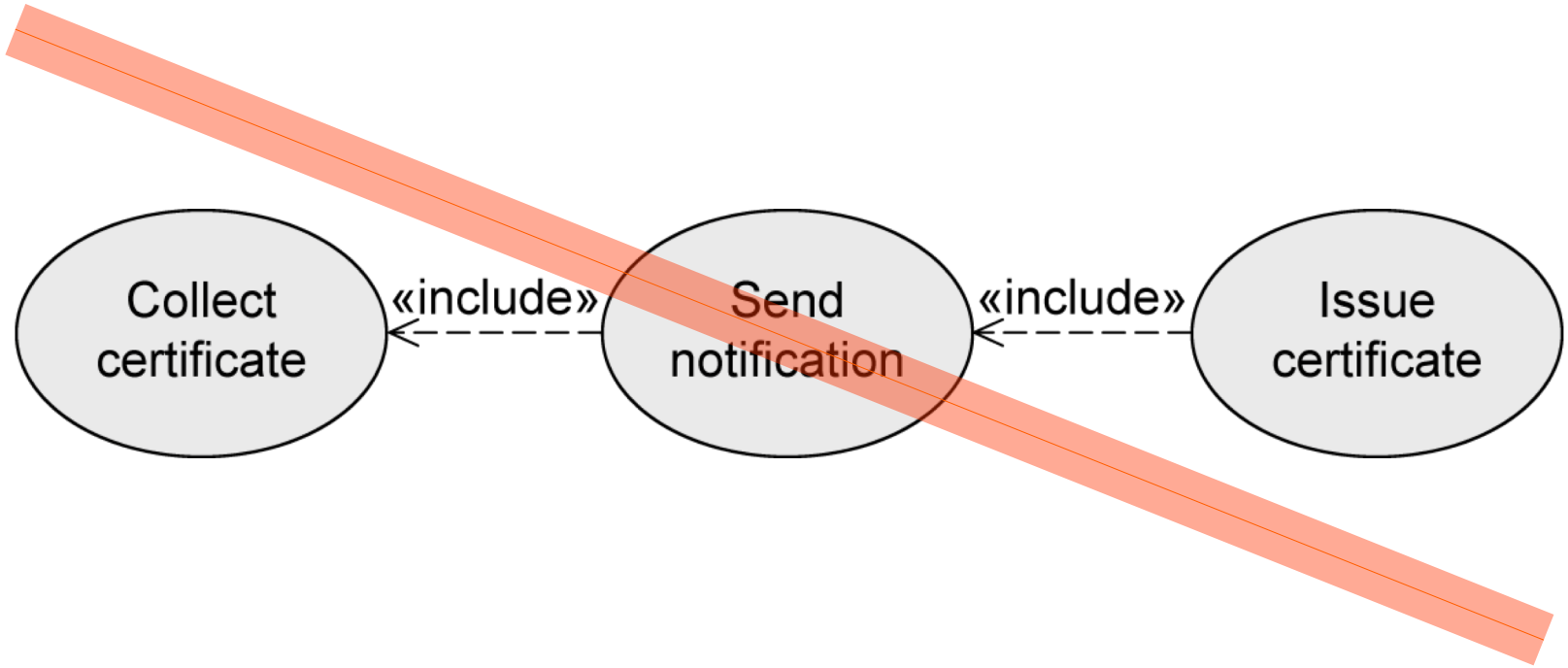
Essential UML: Use case model



Essential UML: Use case model

Typical errors to avoid (1/5)

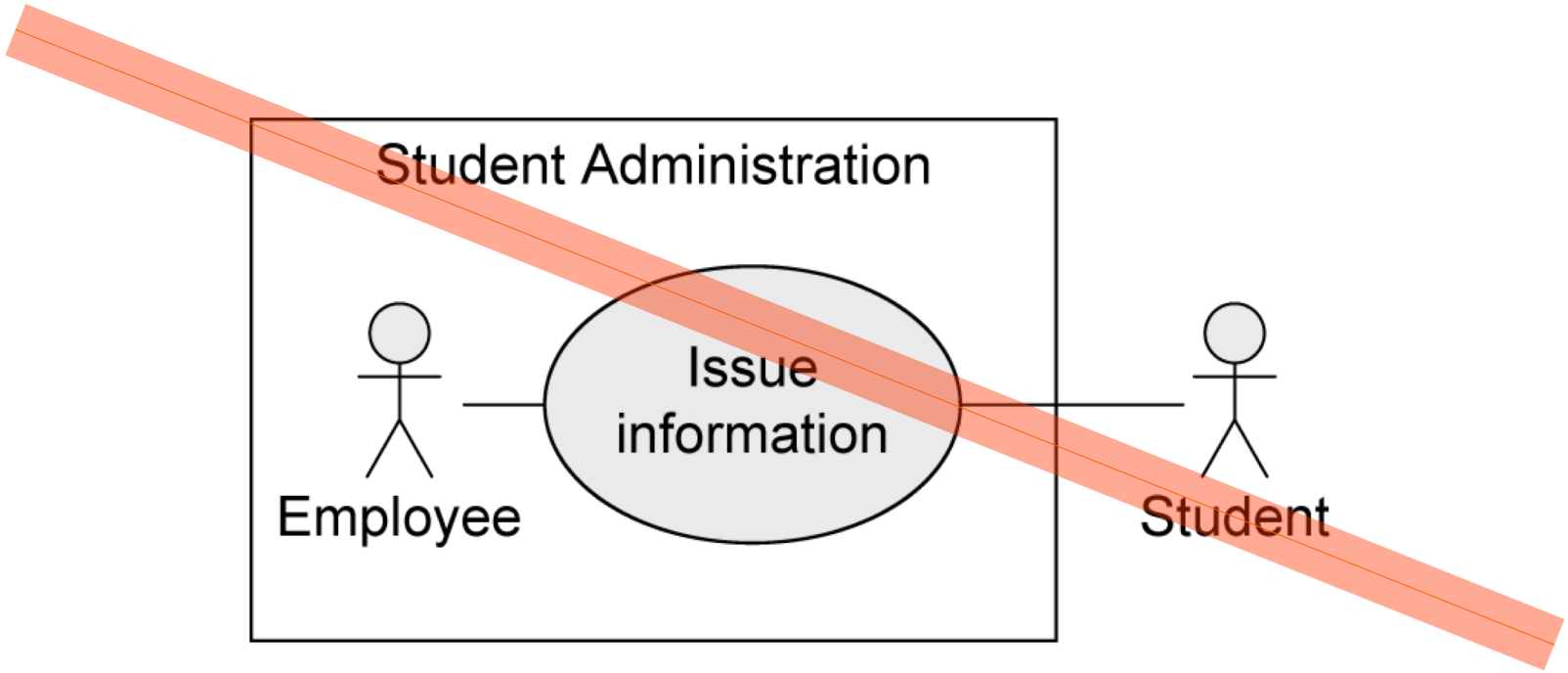
- Use cases do not model processes/workflows



Essential UML: Use case model

Typical errors to avoid (2/5)

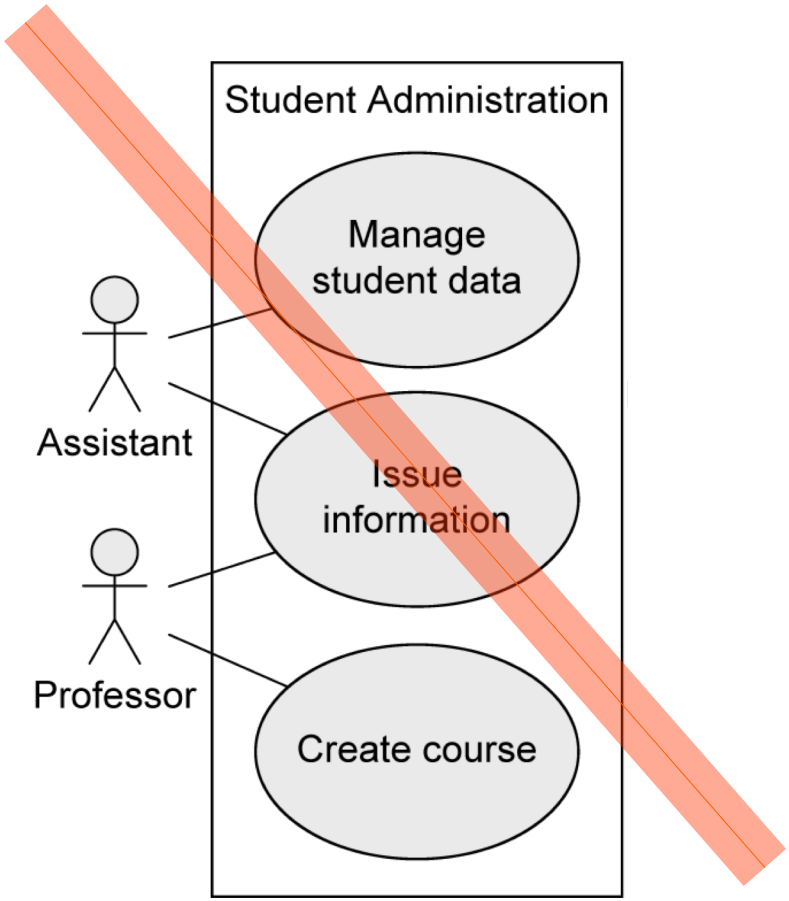
- Actors are not part of the system
- Actors are positioned outside the system boundaries

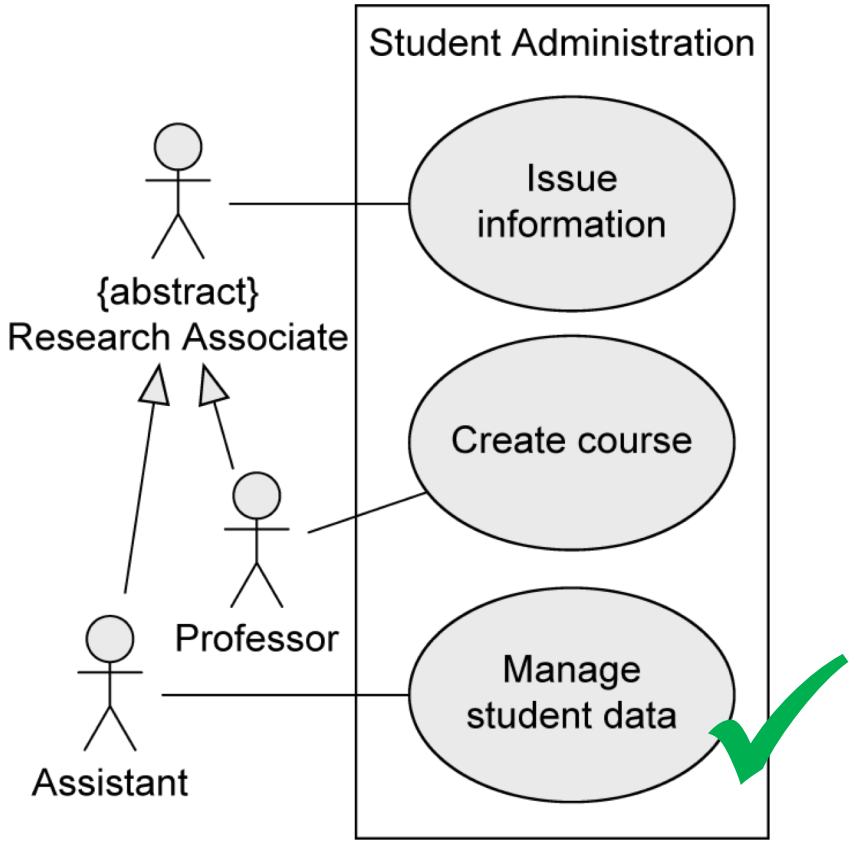


Essential UML: Use case model

Typical errors to avoid (3/5)

- Use case **`Issue information`** needs **EITHER** actor **`Assistant`** **OR** actor **`Professor`** for execution

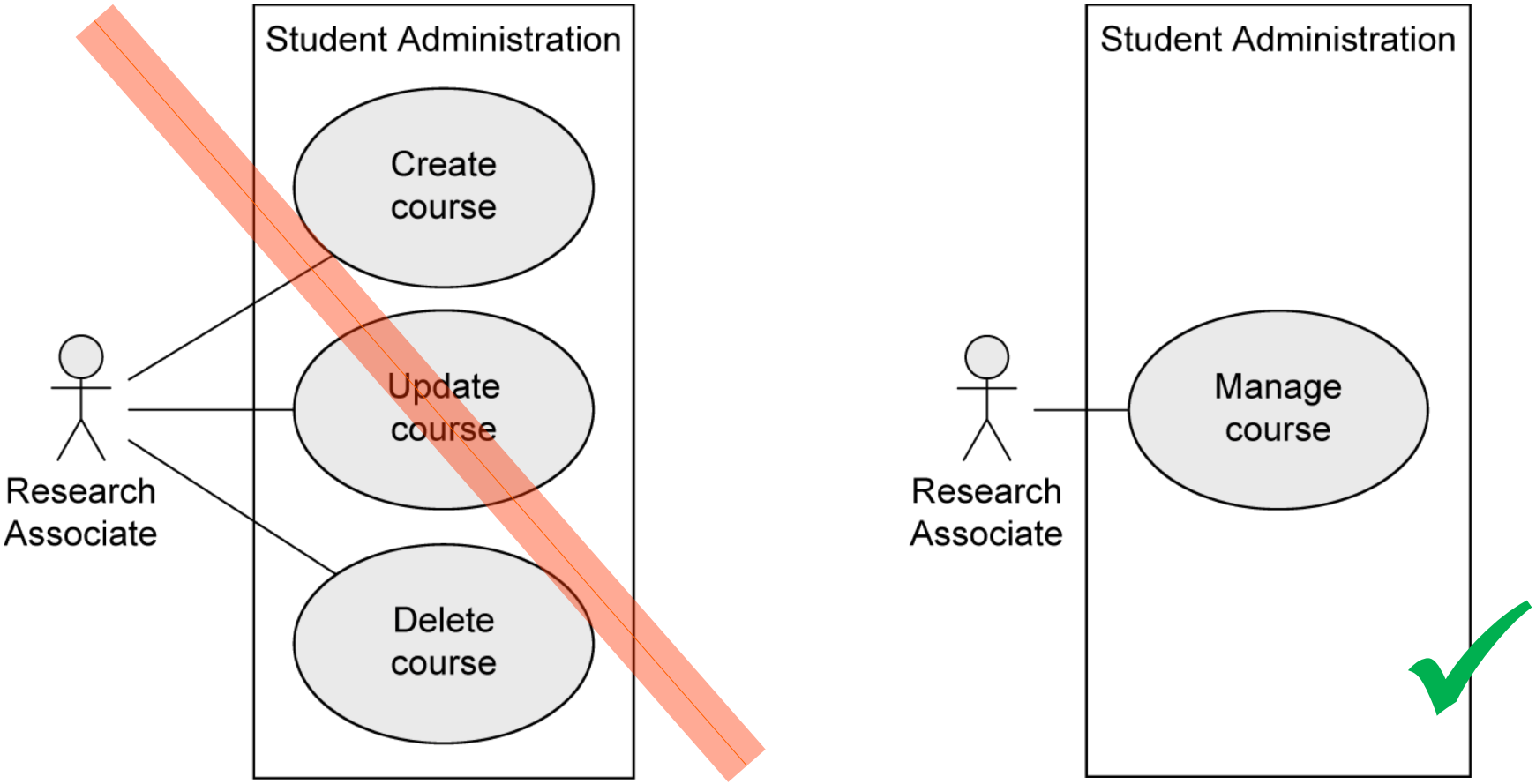




Essential UML: Use case model

Typical errors to avoid (4/5)

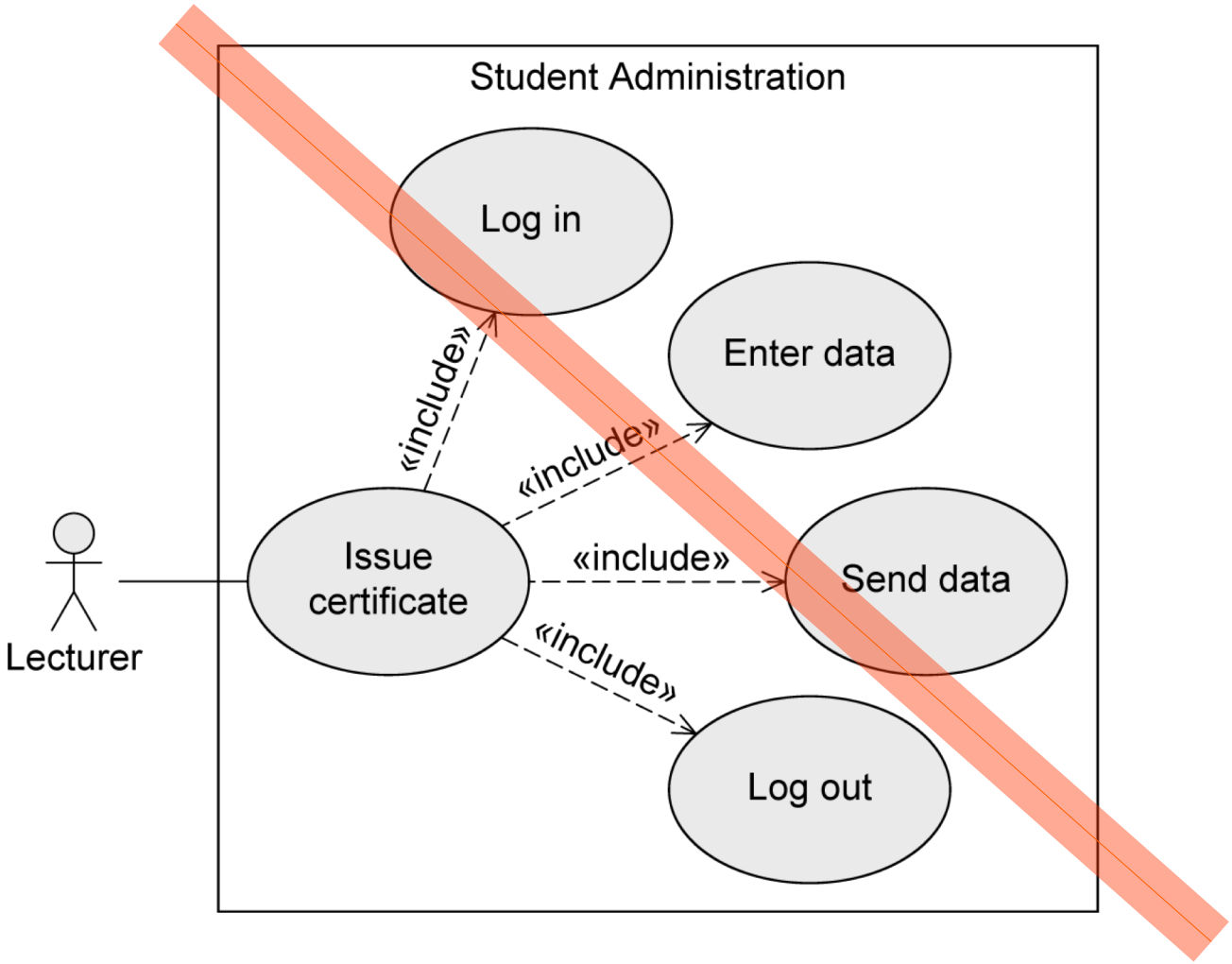
- Many small use cases with the same objective may be grouped to form one (more abstract) use case

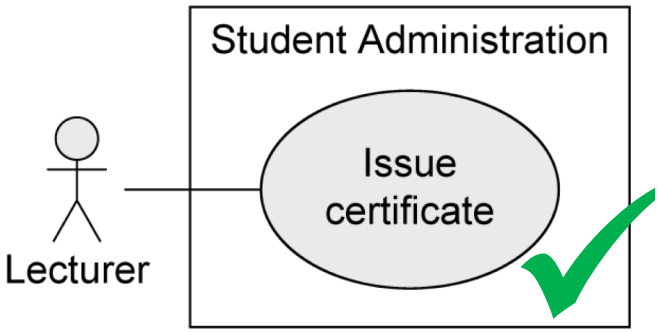


Essential UML: Use case model

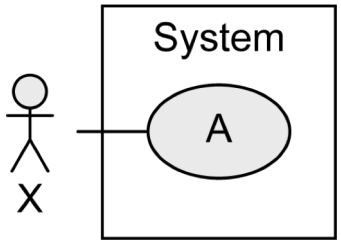
Typical errors to avoid (5/5)

- The various steps are part of the use cases, not separate use cases themselves! ➔ NO functional decomposition

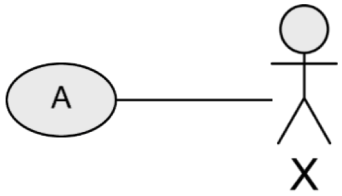
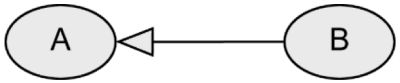
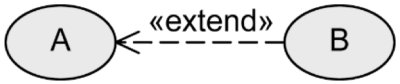
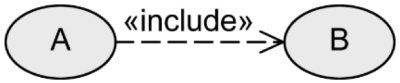




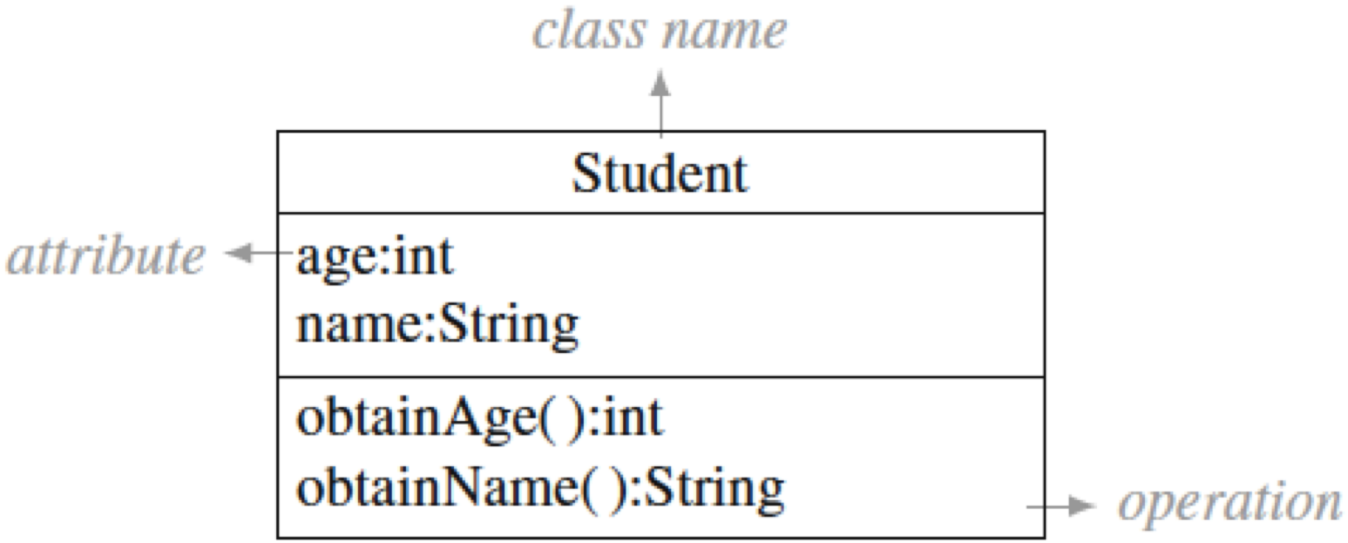
Essential UML: Use case model

Name	Notation	Description
System		Boundaries between the system and its users
Use case		Unit of functionality of the system
Actor		Role of the users of the system

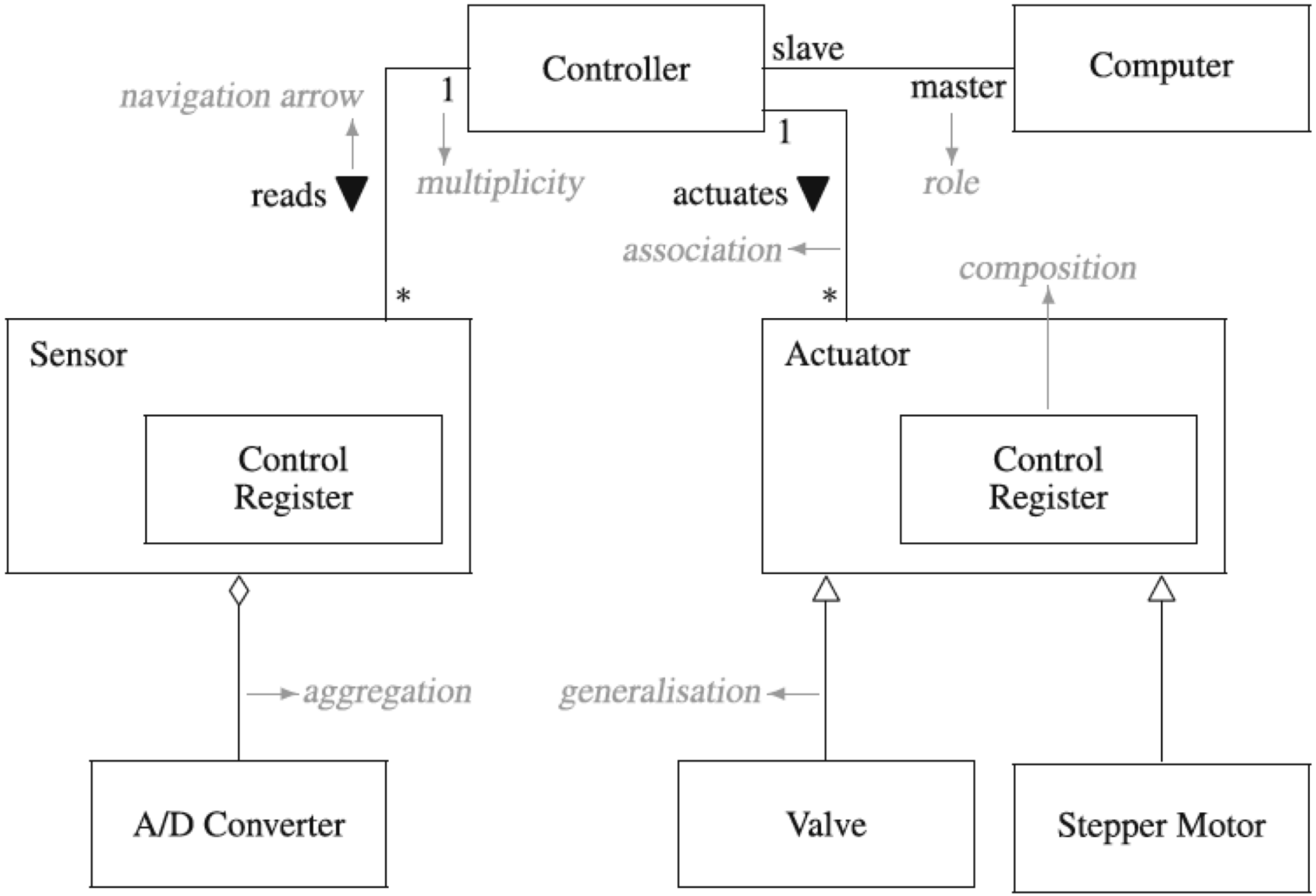
Essential UML: Use case model

Name	Notation	Description
Association		Relationship between use cases and actors
Generalization		Inheritance relationship between actors or use cases
Extend relationship		B extends A: optional use of use case B by use case A
Include relationship		A includes B: required use of use case B by use case A

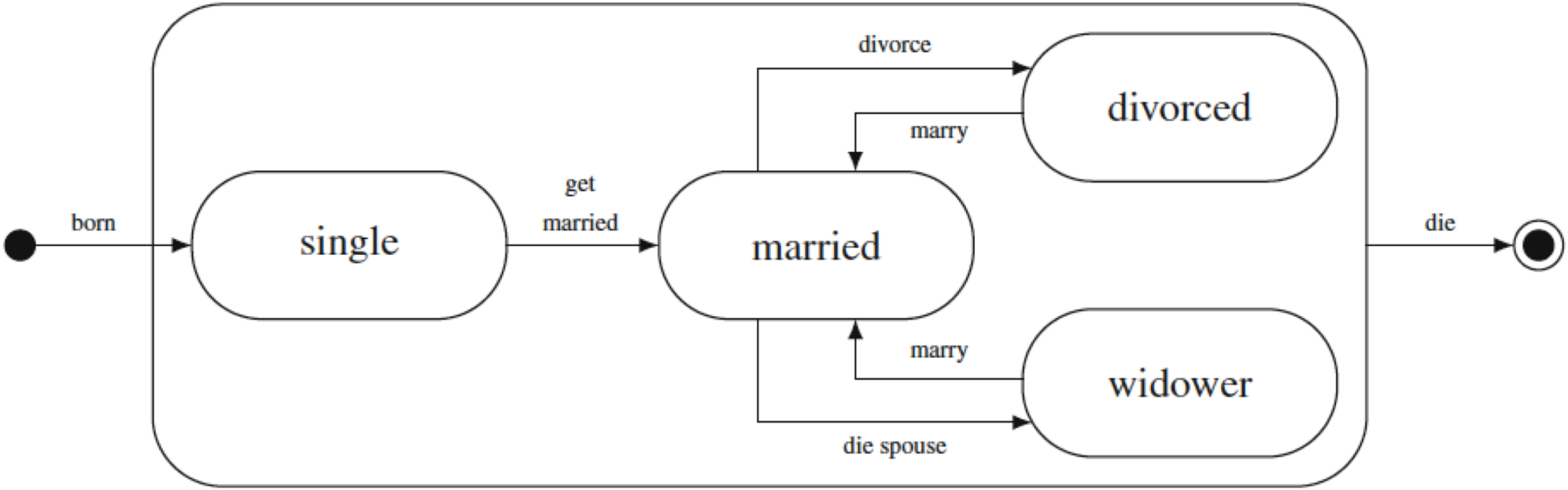
Essential UML: Class model (1/2)



Essential UML: Class model (2/2)



Essential UML: State model



Essential UML: Activity model