Detailed Use Case: Cockburn Template (excerpt)

Use Case Name: *Order bike to user*
Goal in Context: As a user, I want to order a bike to come to my position, so that I do not have to search for a bike.
Precondition: User position can be reached by the bike
Success End Condition: The user gets a bike at their position.
Failed End Condition: The user is informed about the problem.

Description / Main Success Scenario:
Step 1: The user requests a bike to come to the users current position.
Step 2: The bike acknowledges.
Step 3: The bike drives to the user position.
Step 4: The user receives the bike.

Alternatives:
Step 1: The user requires a bike to come to a determined positions.
Step 2: The bike acknowledges
Step 3: The bike drives to the determined position.
Continue with Step 4.

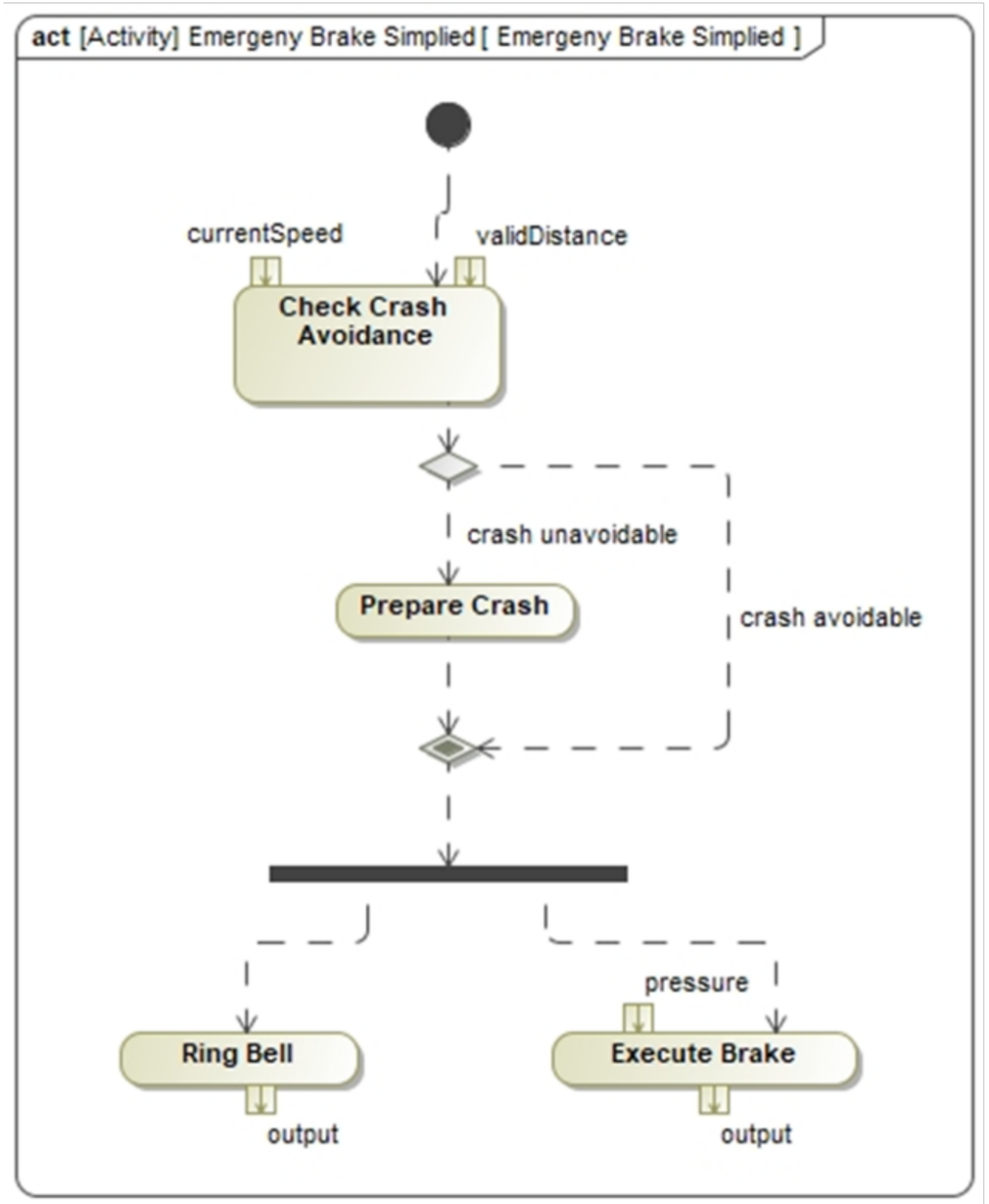
UML Activity Diagrams

An Activity Diagram visualizes:

- Activities, which specify a transformation from input to output via a sequence of actions.
- Building blocks of an activity: Actions
- Optionally: Inputs and outputs of each action
- Optionally: Who is executing which action? ("swimlanes", next slide)

Model forks in a process:

- Decision/merge node: Only one outgoing edge is taken and only if the edge's guard is fulfilled
- Fork/join node: Potentially all outgoing edges are taken

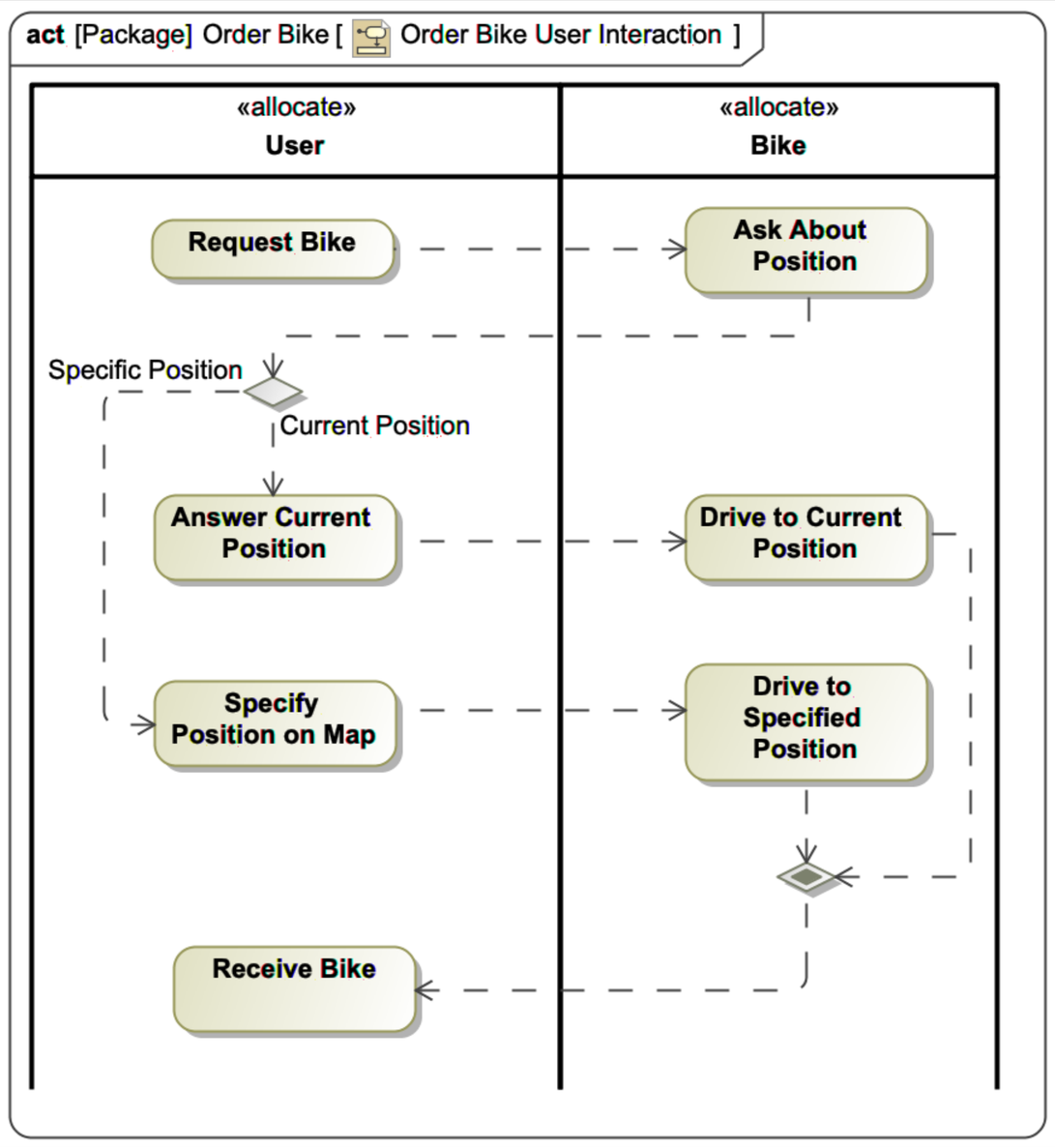


UML Activity Diagrams for Describing Use Cases

Use Case Name: *Order bike to user*
Goal in Context: As a user, I want to order a bike to come to my position, so that I do not have to search for a bike.
Precondition:
User position can be reached by the bike
Success End Condition:
The user gets a bike at their position.
Failed End Condition:
The bike could not reach the user position

Description:
Step 1: The user requests a bike to come to the users current position.
Step 2: The bike acknowledges.
Step 3: The bike drives to the user position.
Step 4: The user receives the bike.

Alternatives:
Step 1: The user requires a bike to come to a determined positions.
Step 2: The bike acknowledges
Step 3: The bike drives the to determined position.
Continue with Step 4.

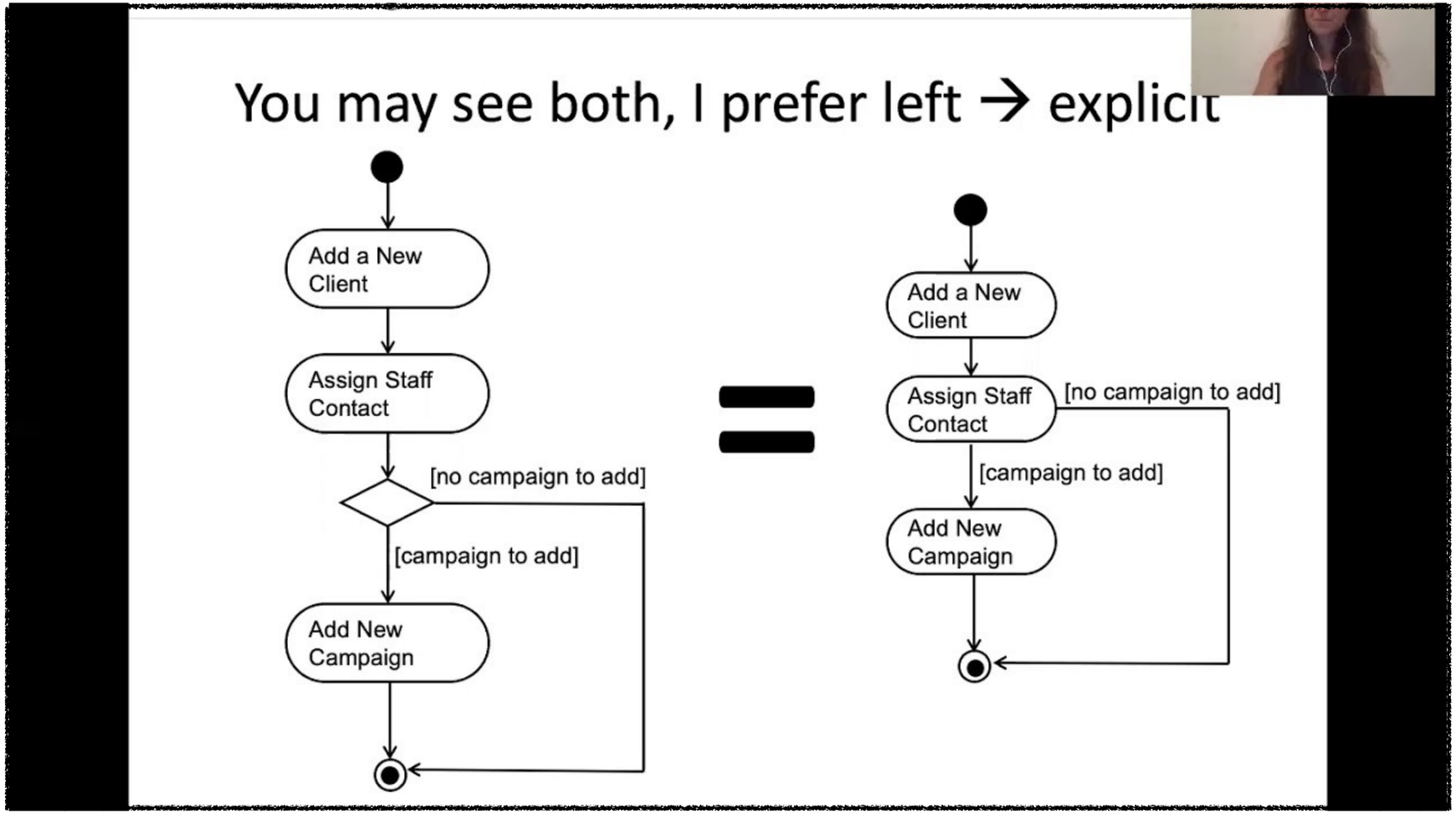


Choose your fighter:

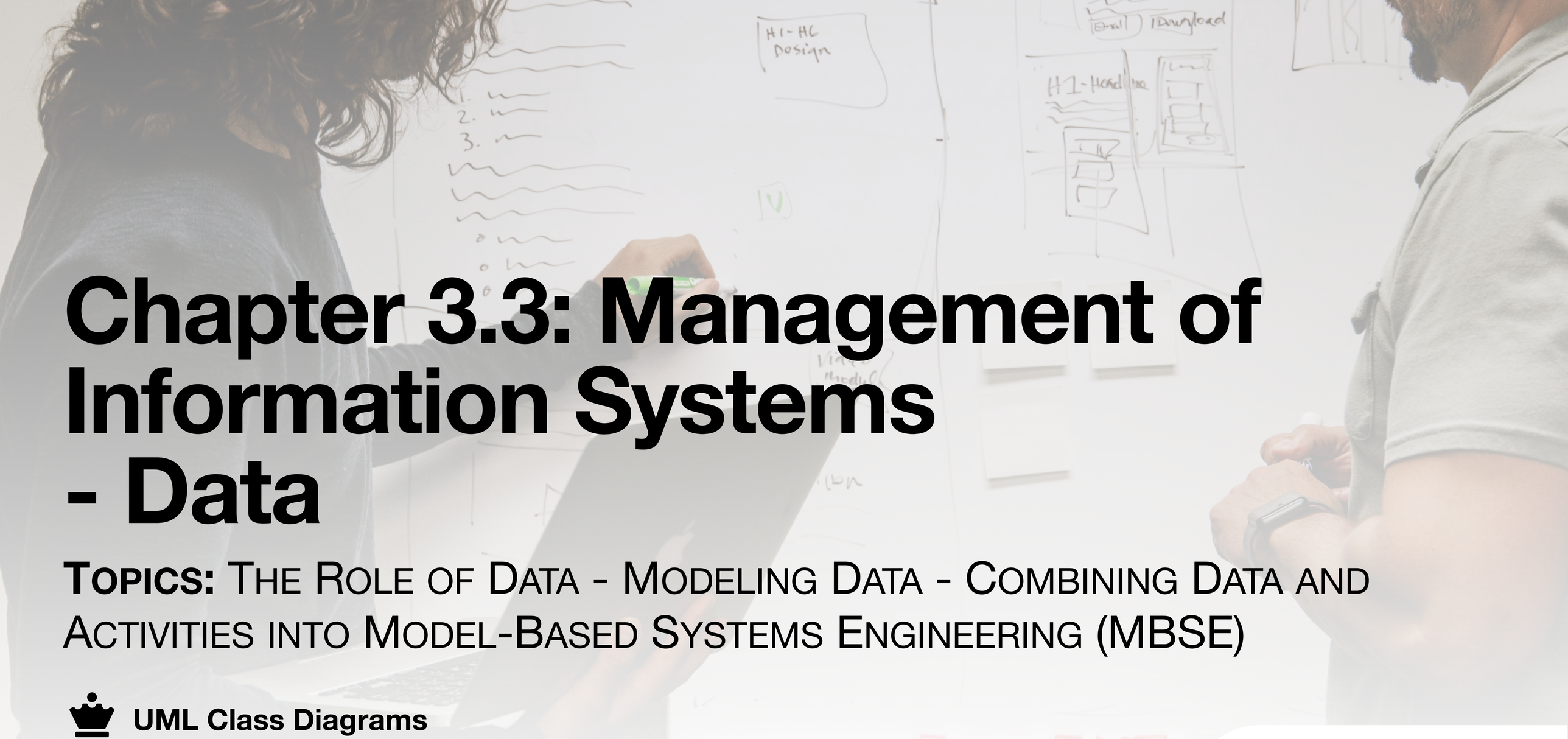
	Munich RE	URBANMAKER
OpenAI	WAELZHOLZ	?

UML Activity Diagrams: Recap

Was this too fast for you? Check the video.



Birgit Penzenstadler. Behaviour Modeling: Activity diagrams https://www.youtube.com/watch?v=m8iwrEEP7A4

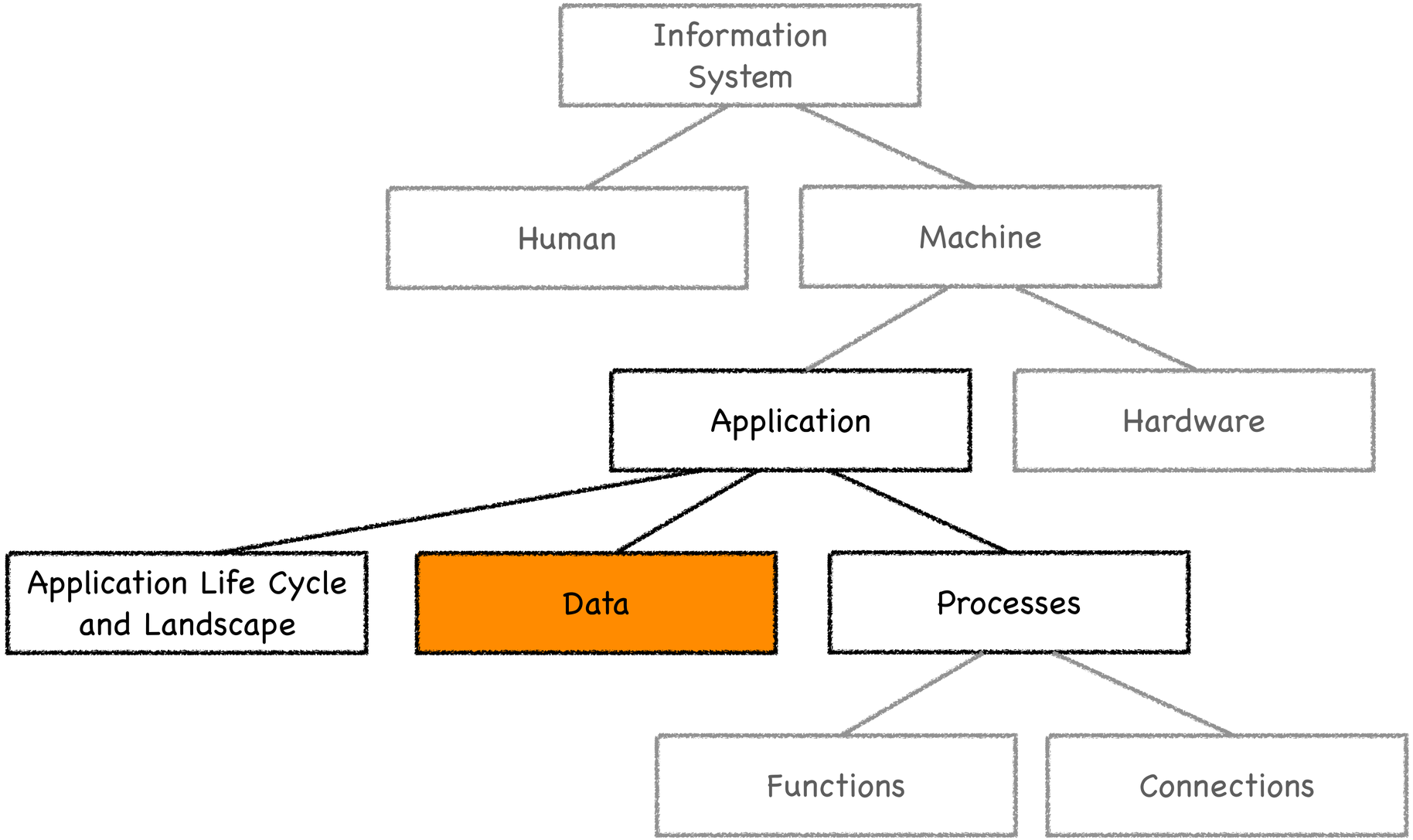


Chapter 3.3: Management of Information Systems - Data

Topics: The Role of Data - Modeling Data - Combining Data and Activities into Model-Based Systems Engineering (MBSE)

- **UML Class Diagrams**
- **Model-Based Systems Engineering**

Fachhochschule Südwestfalen
University of Applied Sciences


Information System
Human
Machine
Application
Hardware
Application Life Cycle and Landscape
Data
Processes
Functions
Connections

The Mental Model

Mental model: A mental model is the organized understanding and mental representation of knowledge about key elements of the system under consideration.

Donald A Norman. Some observations on mental models. Mental models, 1983.

- Mental Models form the fabric of communication, architecture, databases, project planning, contract negotiation ...
- Modern approaches (e.g. Domain Driven Design) re-focus on the importance of a common terminology

Evans, E., 2004. Domain driven design : tackling complexity in the heart of software. Addison-Wesley.

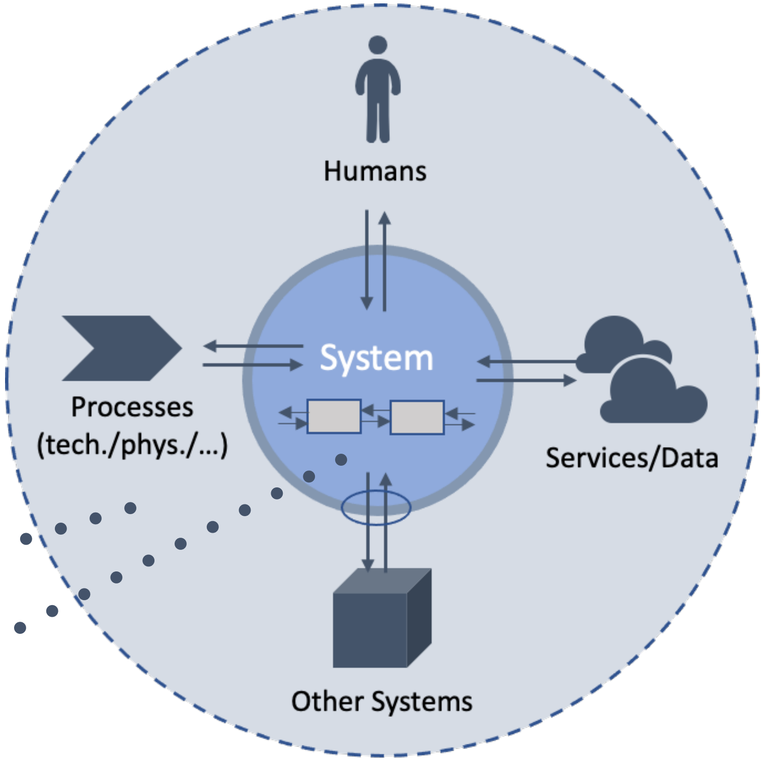
Eric Evans, 2019. What is DDD?. DDD Europe 2019. https://www.youtube.com/watch?v=pMuiVlnGqjk

Modeling: The **what** and **why**

Modeling Object:
What is modeled?

Domain Models as description of the **application domain** of the system to be built.

System Models as abstract representation of the system and its architecture.



Modeling Purpose:
What is it modeled for?

Conceptual Models for stakeholders, developers, testers, users, etc. prioritizes *understandability* for the target users over *formality*.

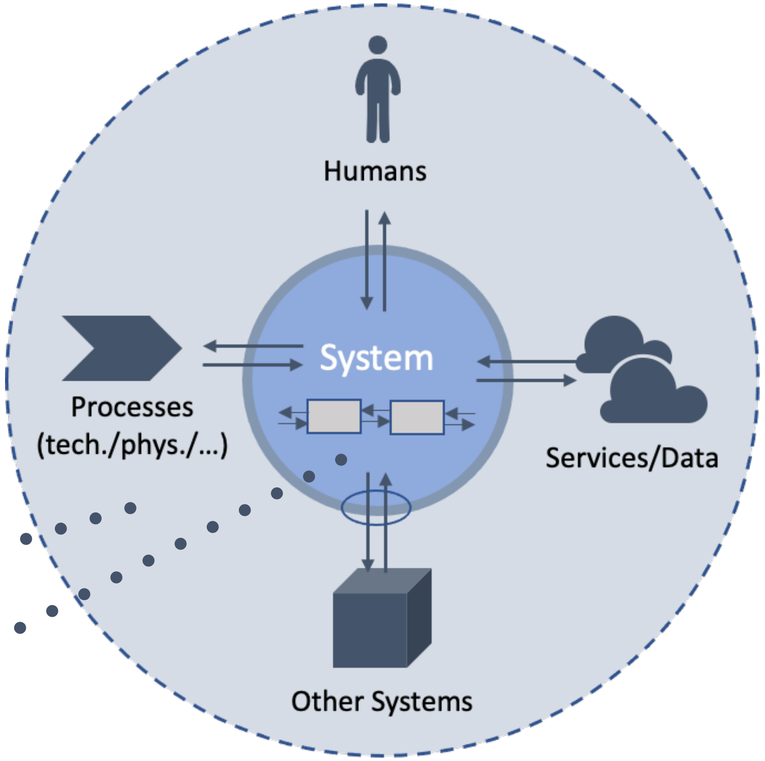
Specification Models allow to generate source code or test cases for the final system and therefore must be *formally precise*.

Modeling: The **what** and **why**

Modeling Object:
What is modeled?

Domain Models as description of the **application domain** of the system to be built.

System Models as abstract representation of the system and its architecture.

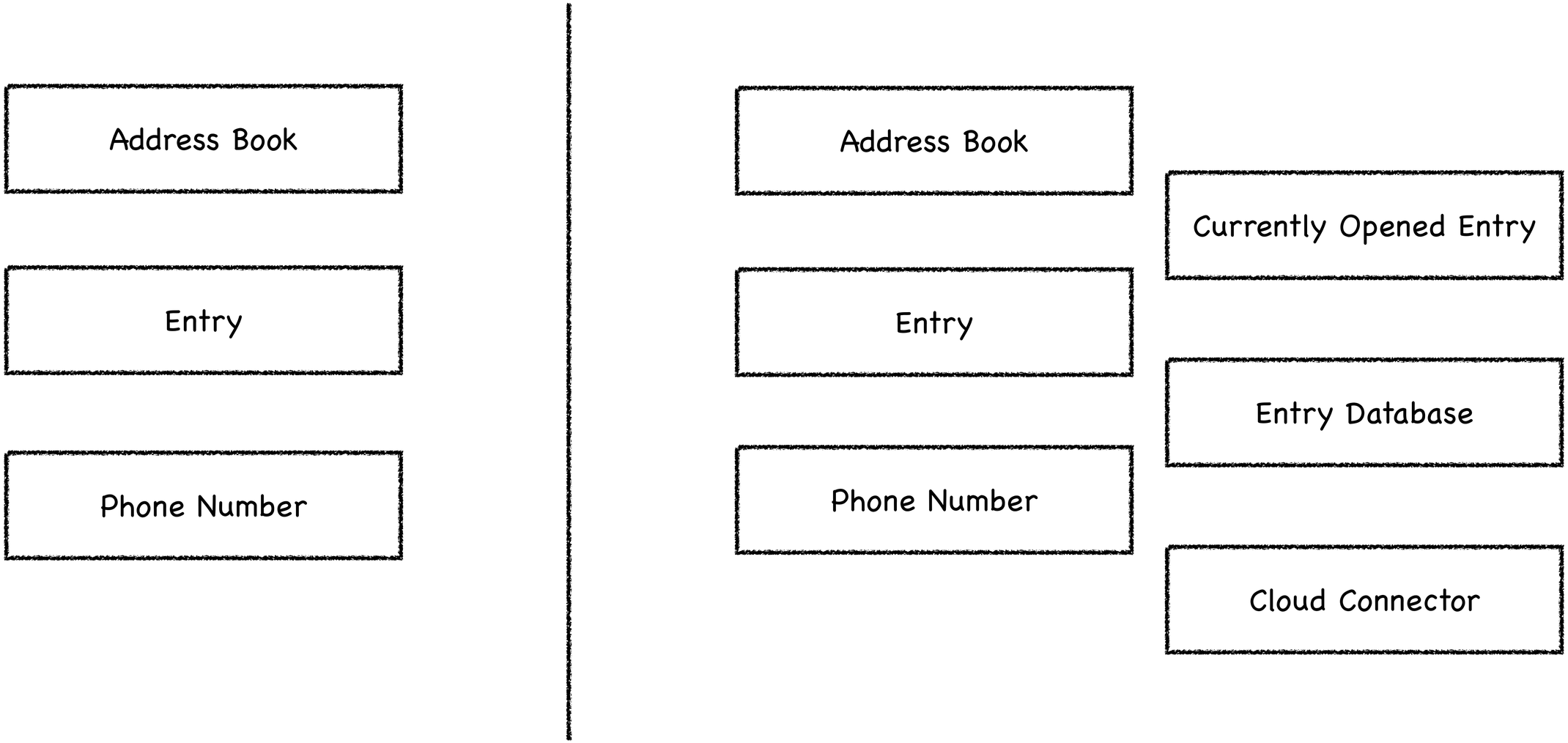


Modeling Purpose:
What is the model for?

Conceptual Models for stakeholders, developers, testers, users, etc. prioritizes *understandability* for the target users over *formality*.

Specification Models allow to generate source code or test cases for the final system and therefore must be *formally precise*.

Domain Models vs. System Models

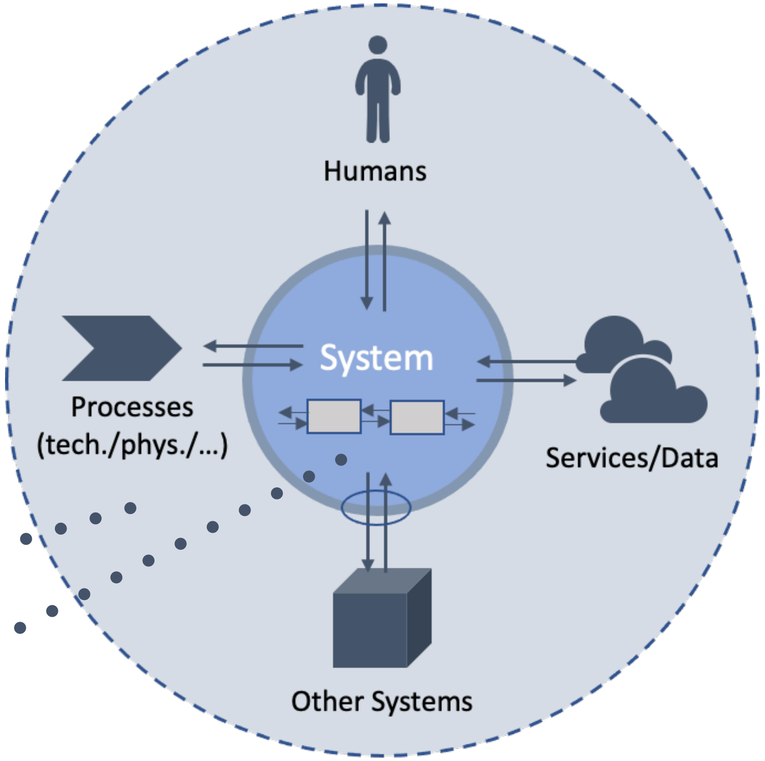


Modeling: The **what** and **why**

Modeling Object:
What is modeled?

Domain Models as description of the **application domain** of the system to be built.

System Models as abstract representation of the system and its architecture.

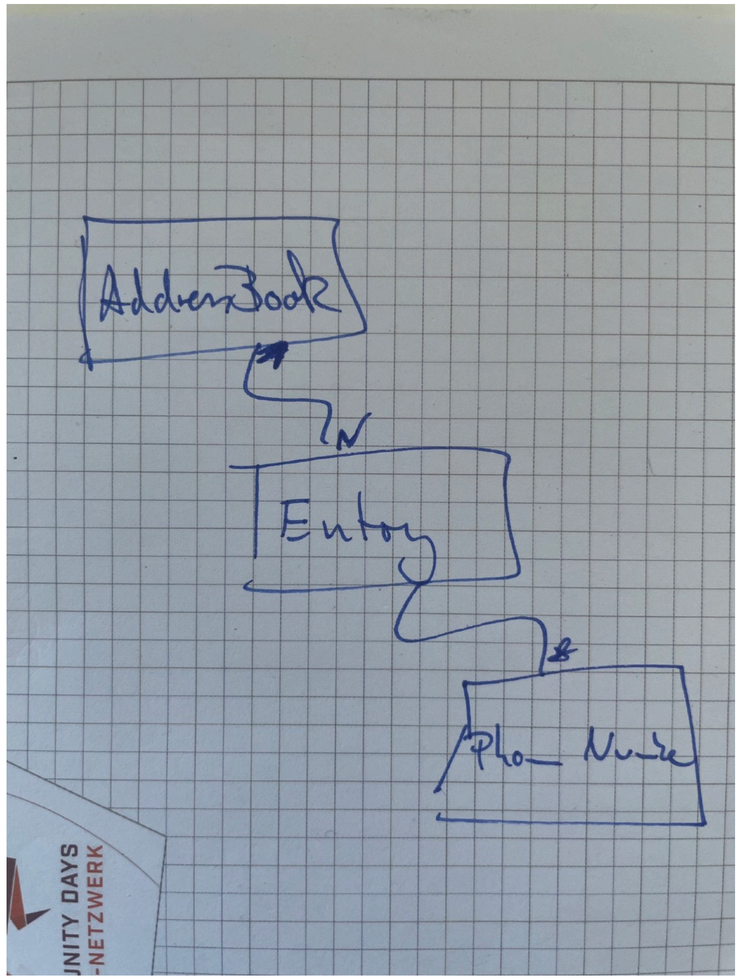


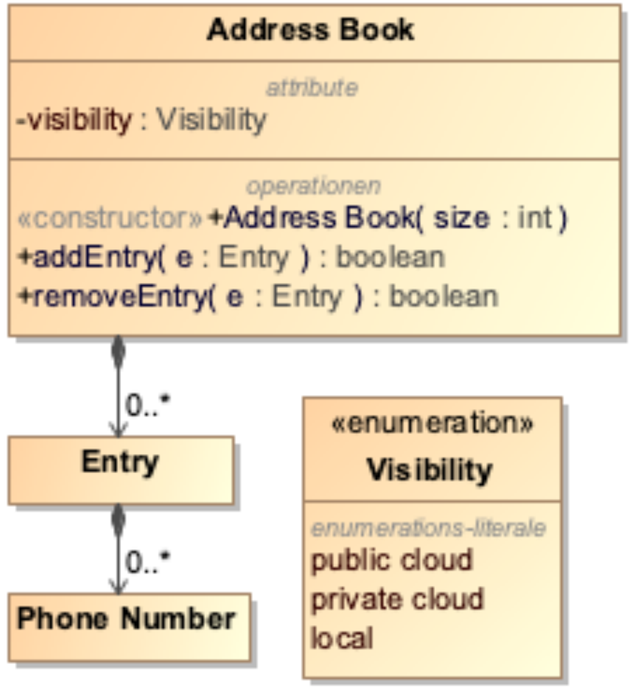
Modeling Purpose:
What is the model for?

Conceptual Models for stakeholders, developers, testers, users, etc. prioritizes *understandability* for the target users over *formality*.

Specification Models allow to generate source code or test cases for the final system and therefore must be *formally precise*.

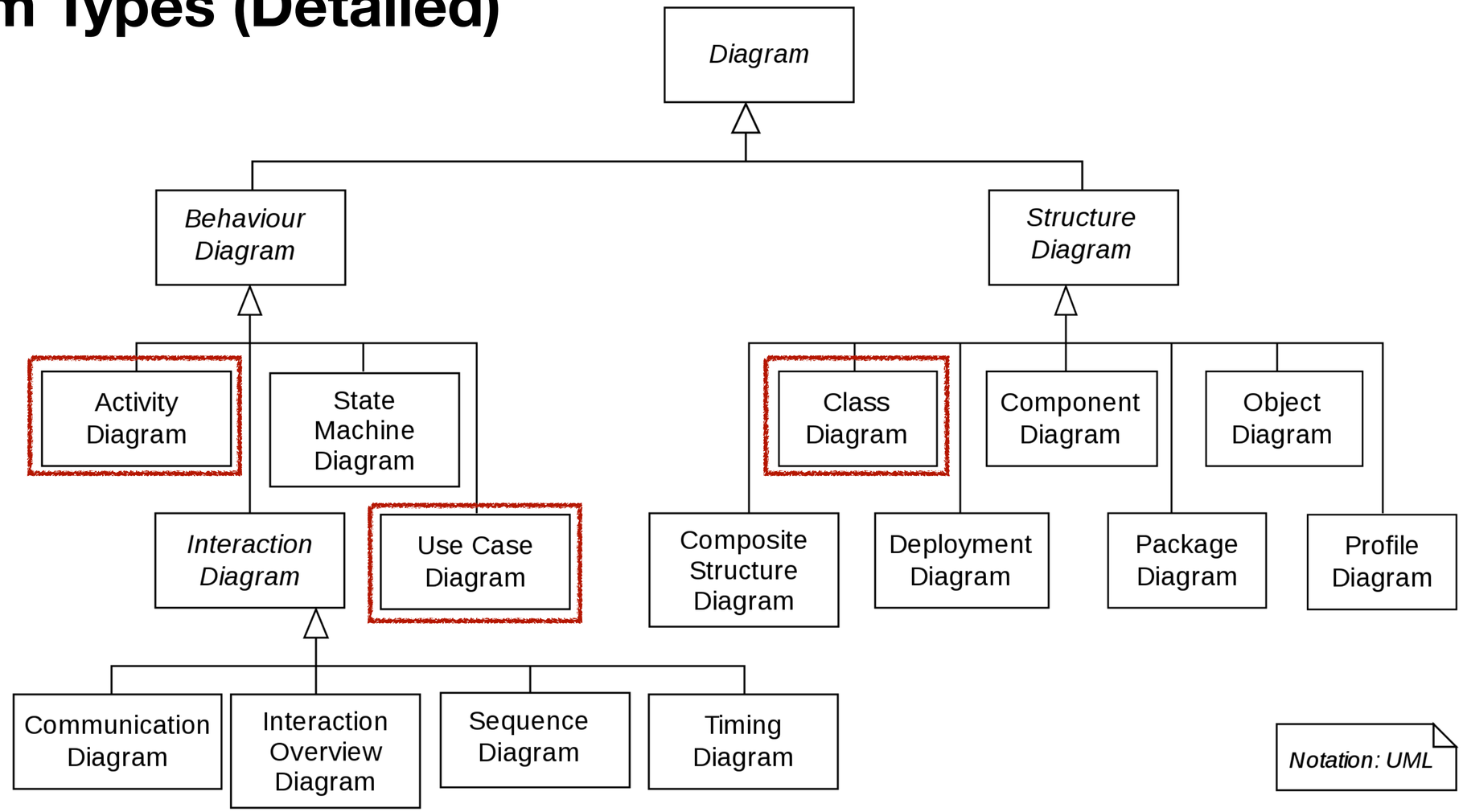
Modeling for Communication vs. Specification





The Unified Modeling Language (UML)

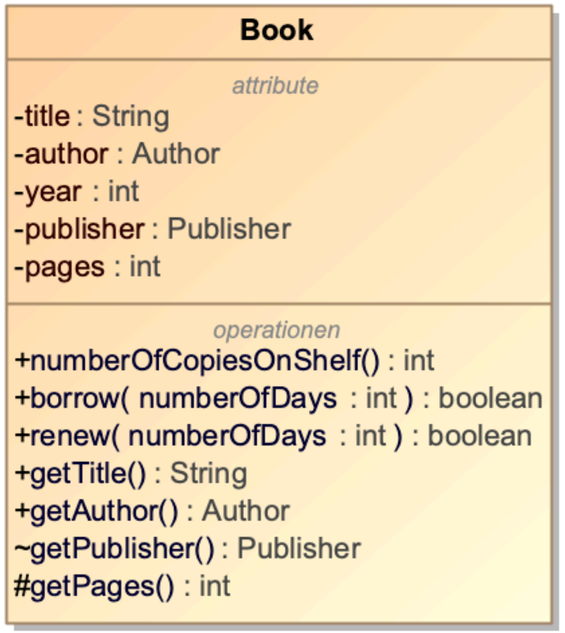
Diagram Types (Detailed)



https://upload.wikimedia.org/wikipedia/commons/thumb/e/ed/UML_diagrams_overview.svg/2560px-UML_diagrams_overview.svg.png

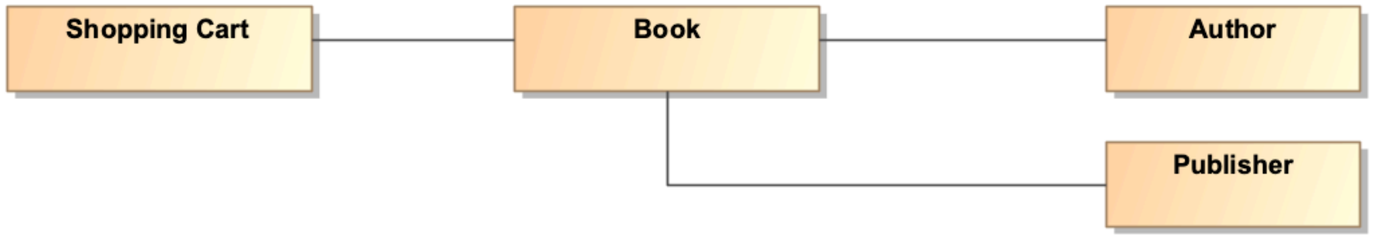
UML Class Diagrams: Classes

- *Objects* are concrete instances of *classes*,
  e.g., a specific copy of Harry Potter and the Philosopher's Stone
- Classes are generalized *types of objects*,
  e.g., the class Book
- In UML class diagrams, classes are represented through rectangles with a name
- Compartments for *attributes* and *operations* may be added
- Syntax for attributes is:
  <+/-/#/~> <name> : <type>
- Syntax for operations is:
  <+/-/#/~> <name> (<parameters>:<types>) : <return type>



UML Class Diagrams: Associations

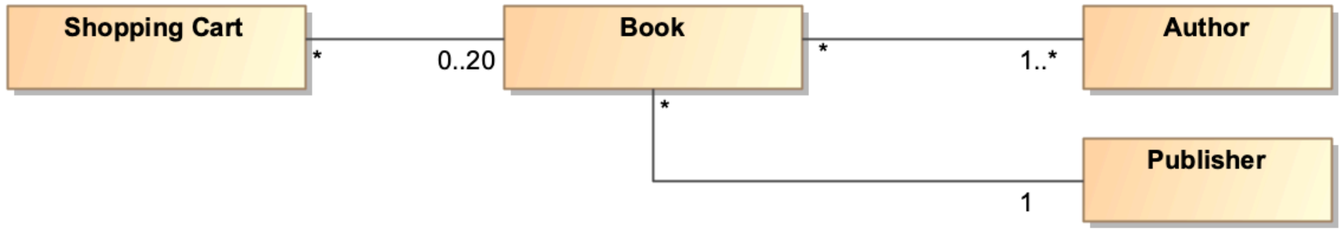
- An association connects two (or rarely more) classes.
- A *link* is an instance of an *association*.
- *Rolenames* show the roles that objects play in an association.

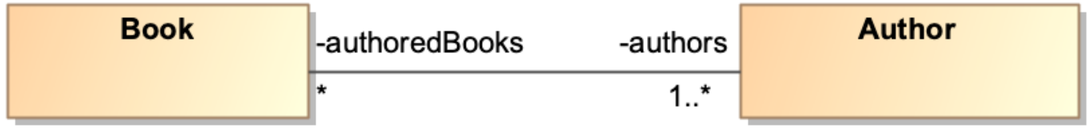


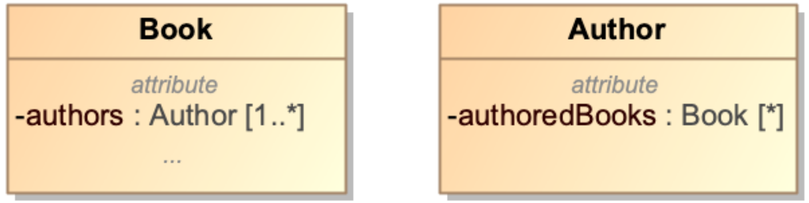
UML Class Diagrams: Multiplicities

The most common multiplicity indications:

- **0..1**: The property may or may not have a value.
- **1..1**: The property has exactly (and always) a value. Usually denoted as “1” only.
- **0..*** or shorthand *: The property has zero or more values.
- **1..***: The property one or more values.
- **n..m**: The property has at least n and at most m values.








```
public class Book {
   private Author[] authors;
   ...
}
public class Author {
 private Book[] authoredBooks;
 ...
}
```

How to Create A Data Model from a Specification

A Simple Method

Simple Method: look for noun phrases in the system description!

Then remove things which are:
▶ Redundant ▶ Attributes ▶ Outside scope ▶ Operations and events
▶ Vague ▶ On a different level of detail (*System Model* vs. *Domain Model*)

Similarly, can use *verb phrases* to identify operations and/or associations.

Books and Journals: The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

Each record company has a unique name; an email address and telephone number are also kept on each record company. A record company publishes one or more albums; every album is published by exactly one record company. An album is identified by its album ID, and other attributes are title, price, and number of songs. Each album contains one or more songs. Each song can be included on one or more albums and is performed by exactly one musician. Every song is identified by its song ID and in addition its song title and length are also kept as attributes. A musician may perform one or more songs and he/she is uniquely described by a musician ID. We also know each musician's name and address (combination of street, city and ZIP). A musician receives a separate royalty check for each of his songs yearly from the record company. Each check is identified by its check number, and we also keep track of the date and amount of each check.

What's the relationship between authors, customers, and publishers?

Generalization

Inheritance is a binary relationship:

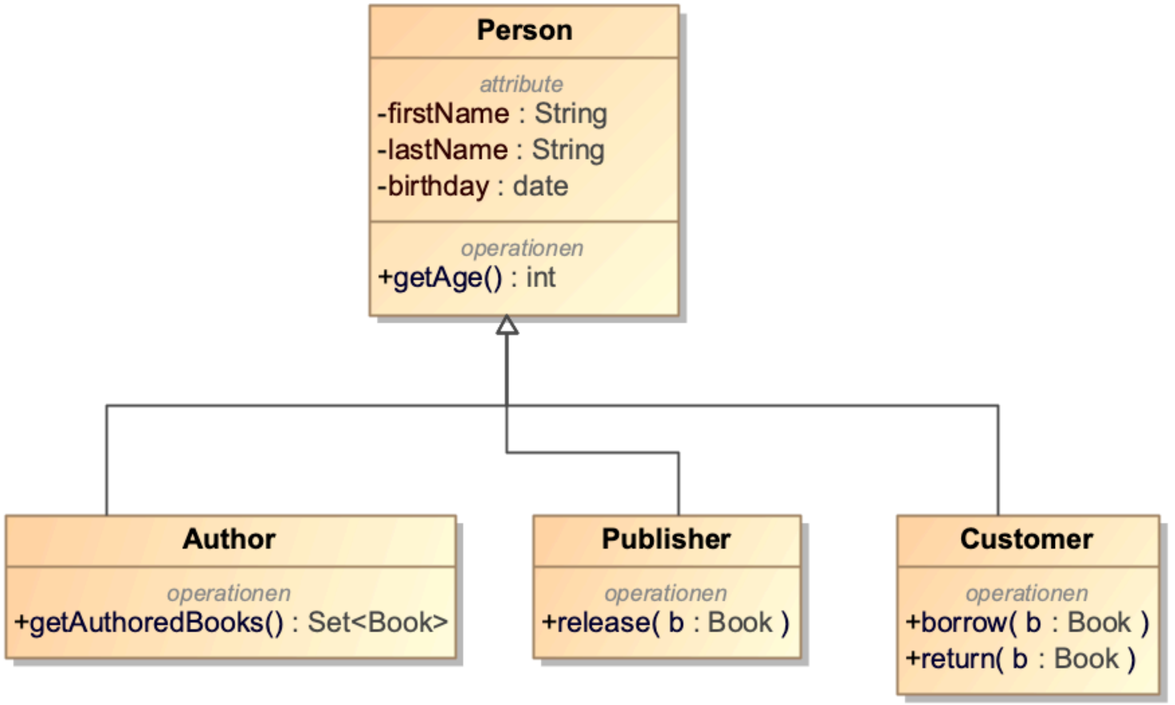
- Subtype: A subgrouping of the objects in a class that has attributes distinct from those in other subgroupings
- Supertype: A generic class that has a relationship with one or more subtypes

Attribute Inheritance:

- Subtype entities inherit values of all attributes of the supertype
- An instance of a subtype is also an instance of the supertype

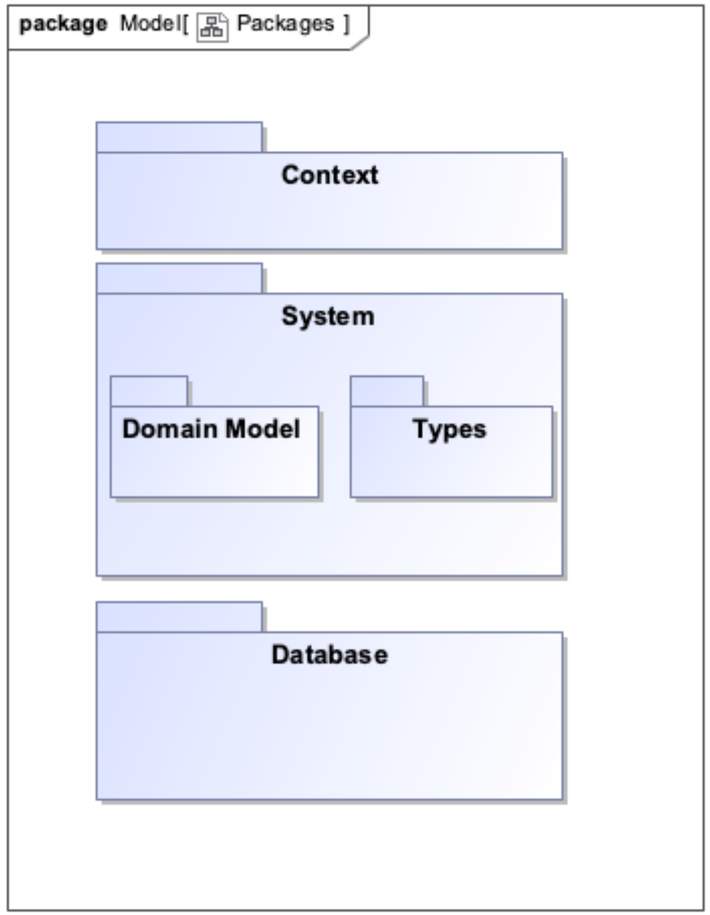
Association Inheritance:

- Associations at the supertype level indicate that all subtypes will have this association
- Associations at the subtype level only refer to the subtype



UML Class Diagrams: Packages

- Packages define regions
- Think of them as folders, defining the path to elements inside the package



UML Class Diagrams: Access Operators

- Syntax for attributes is:
  <+/-/#/~> <name> : <type>

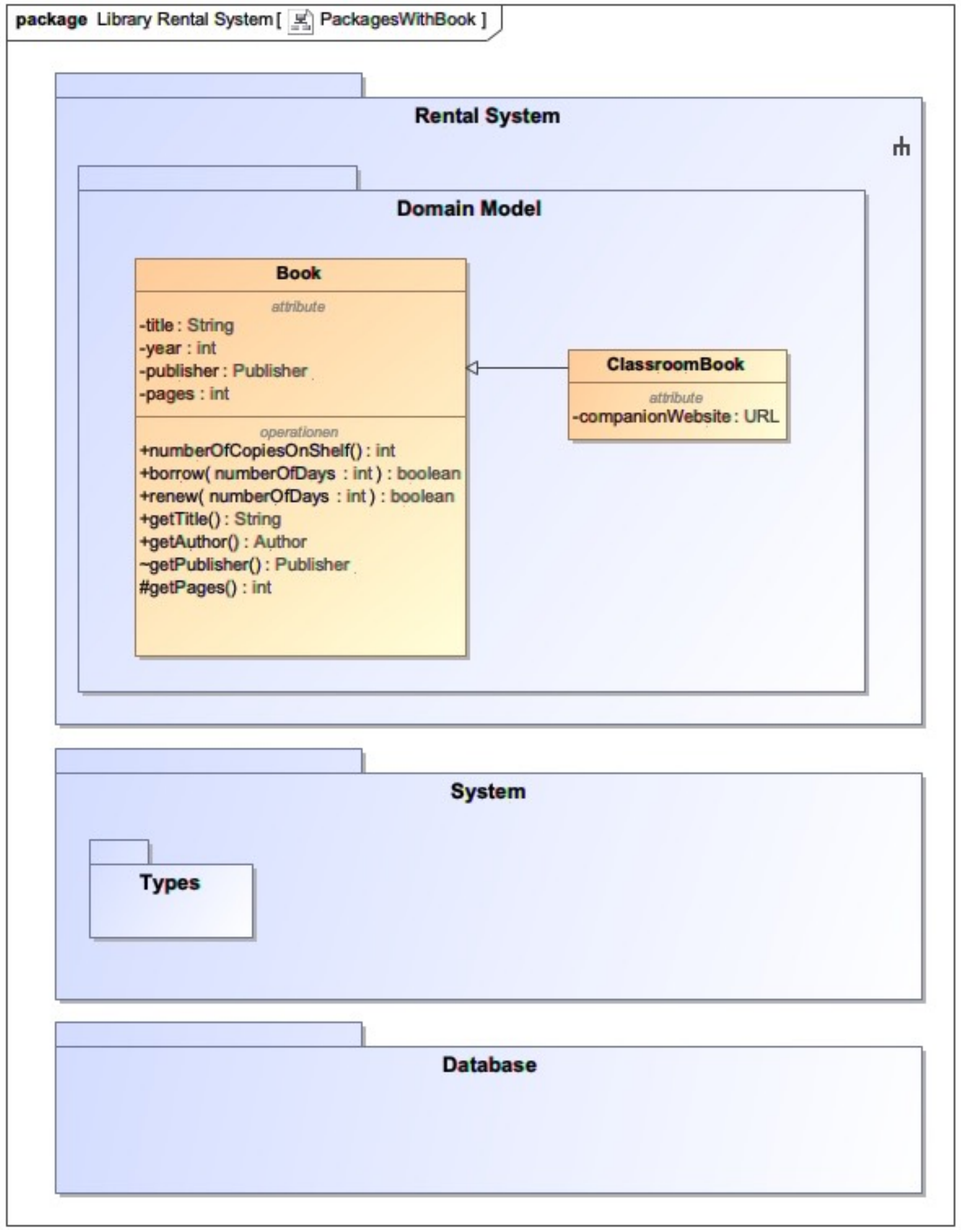
- Syntax for operations is:
  <+/-/#/~> <name> (<parameters>:<types>)
  : <return type>

+ (public): accessible from all classes with access to this package

~ (package): accessible only inside this package

(protected): accessible only to this and to all subclasses

- (private): accessible only to this class



Choose your fighter:

Virtuelles Krankenhaus VKh.NRW

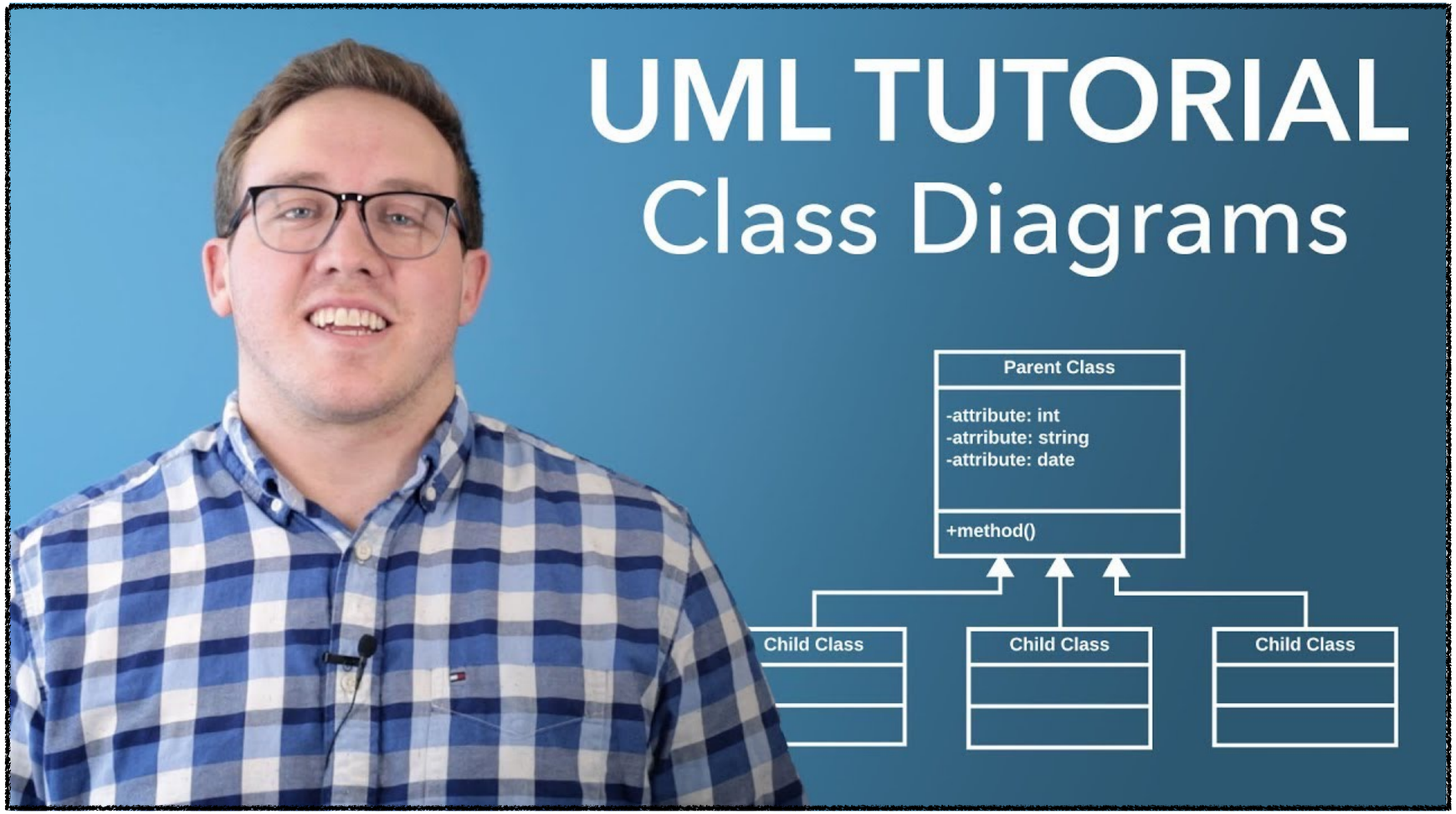
URBANMAKER

WAELZHOLZ

?

UML Class Diagrams: Recap

Was this too fast for you? Check the video.

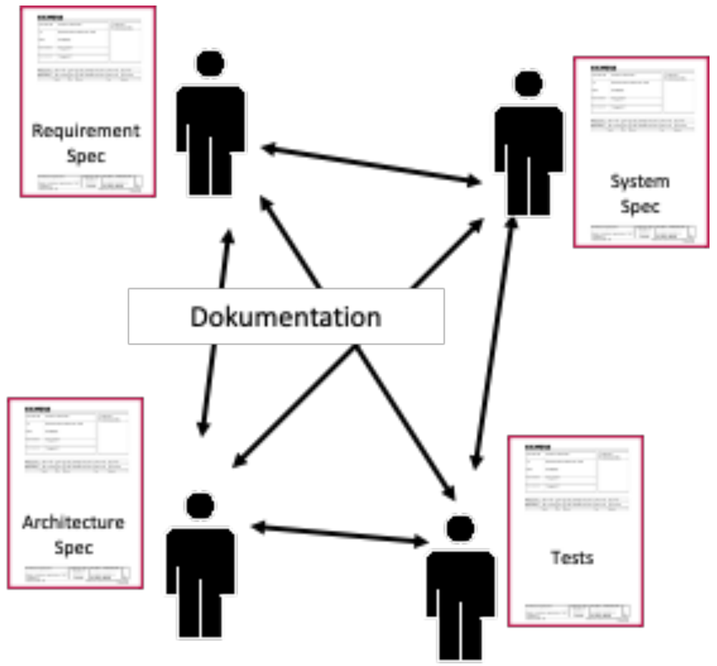


LucidChart. UML Class Diagram Tutorial
https://www.youtube.com/watch?v=UI6lqHOVHic

Model-based Systems Engineering

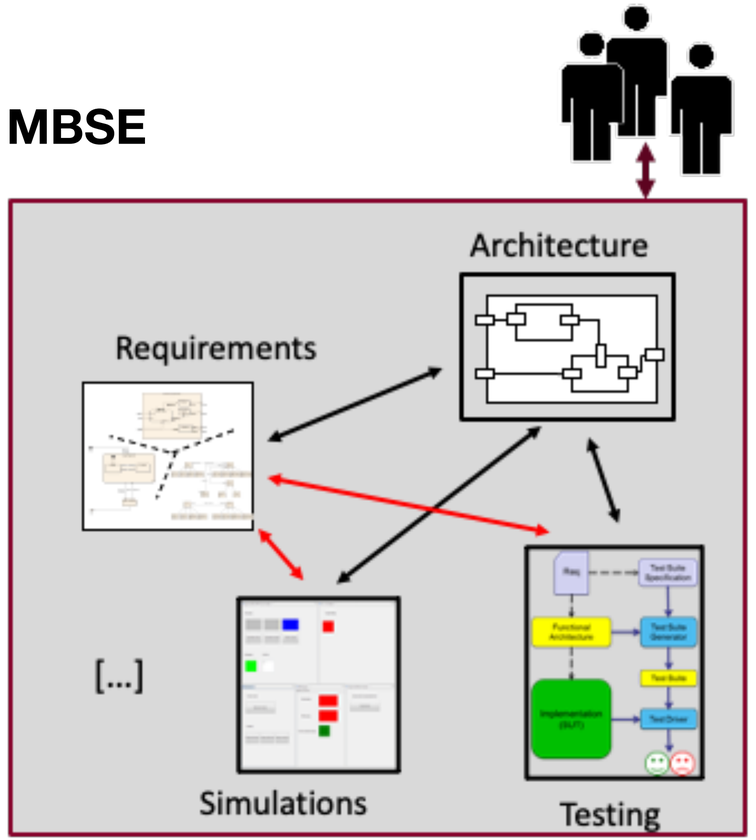
Going from Documents to Models

Document-based Engineering



Document Types and Document Structure

MBSE



Types, Relations, Traces, Information
Architecture, Representation, Usage

Of Models and Diagrams...

(System) Model:

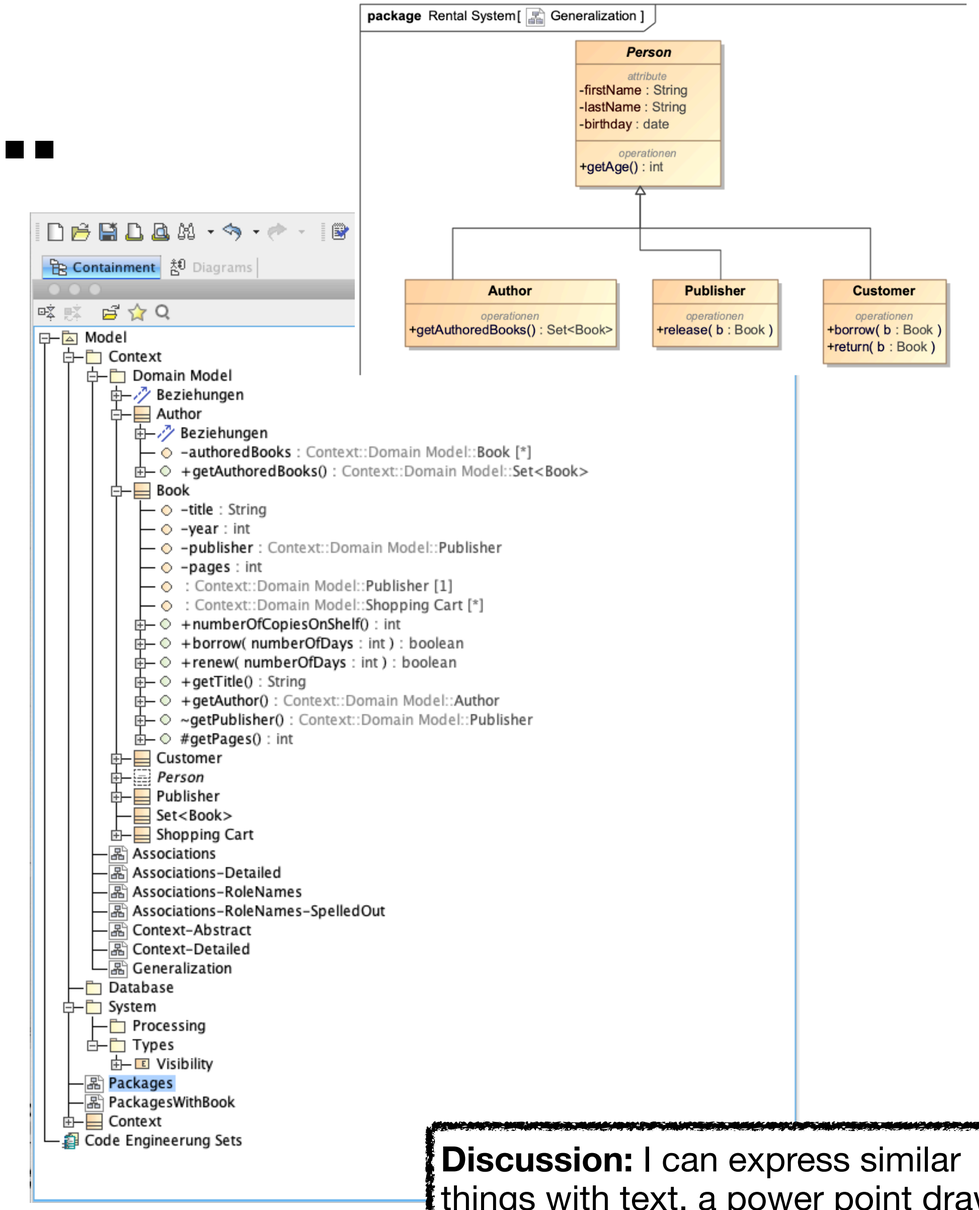
- Contains all system elements and their relations
- Is structured in a hierarchic tree (think: file system), but the tree does not necessarily represent system structure

Diagram:

- Visualizes a selection of system elements and a selection of their relations
- Provides a graphical view of a part of the model

Keep in mind:

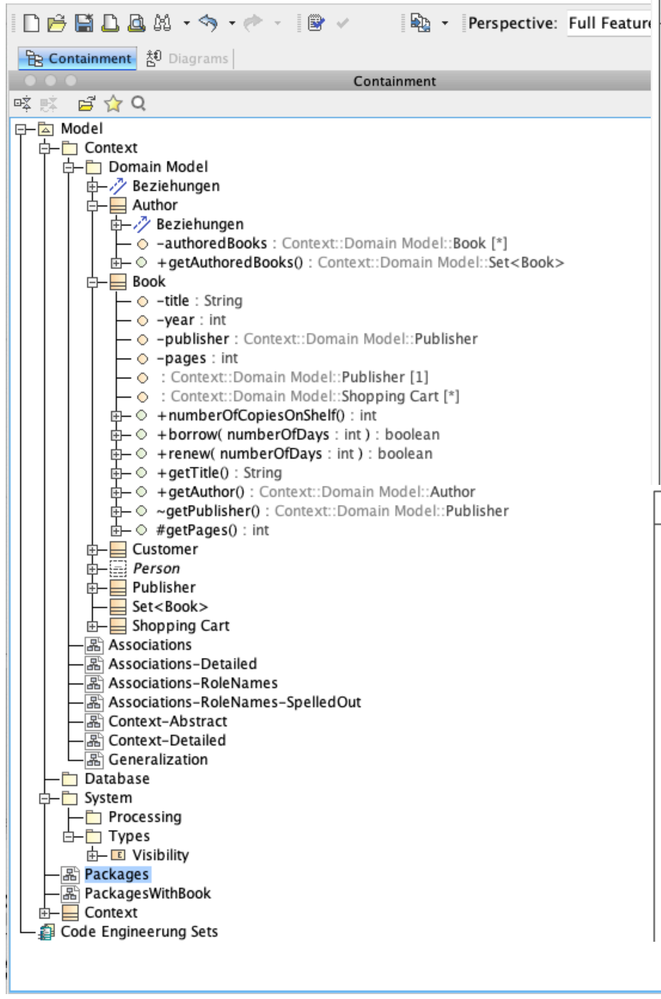
- A model element can be visualized **in many diagrams**
- Even if a model element is not shown in a diagram **it may still be part of the model**
- You can create a model **without a single diagram**

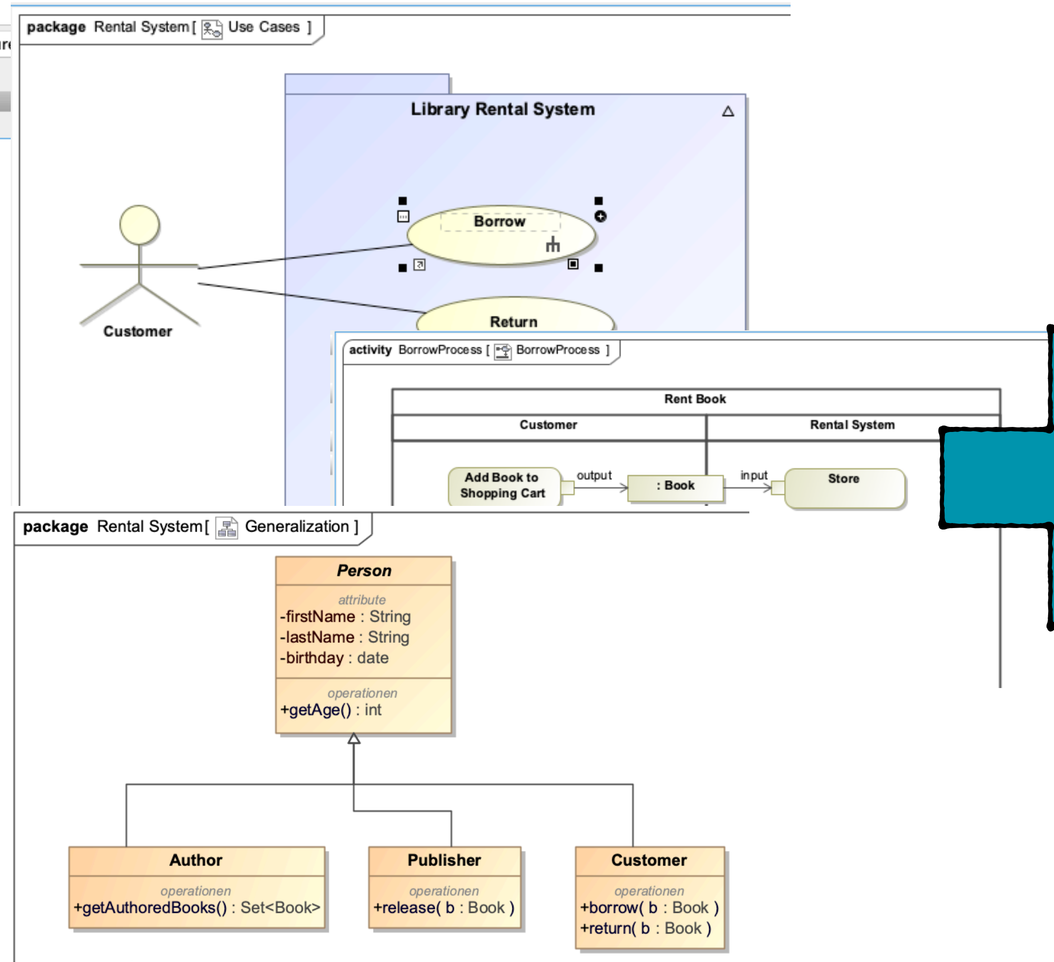


Discussion: I can express similar things with text, a power point drawing, a diagram and a model. *What does each step add?*

Model-based Systems Engineering

How it all comes together





Validate requirements

Analyze impact of change

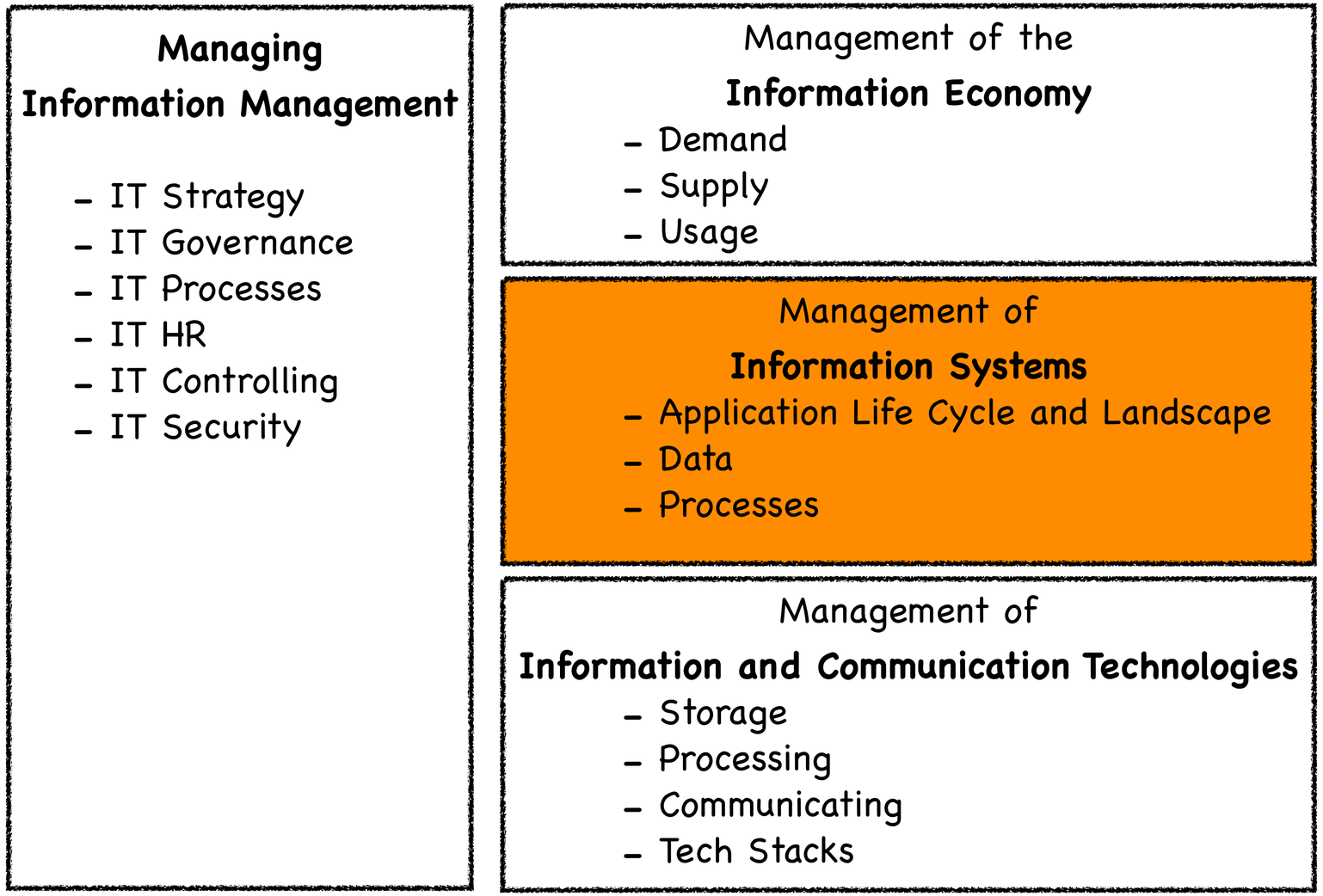
Verify design

Generate artifacts

Model-based Systems Engineering

How it all comes together: Demo

Information Management



Krcmar, Helmut. "Informationsmanagement." Informationsmanagement. Springer Gabler, Berlin, Heidelberg, 2015, p.94.