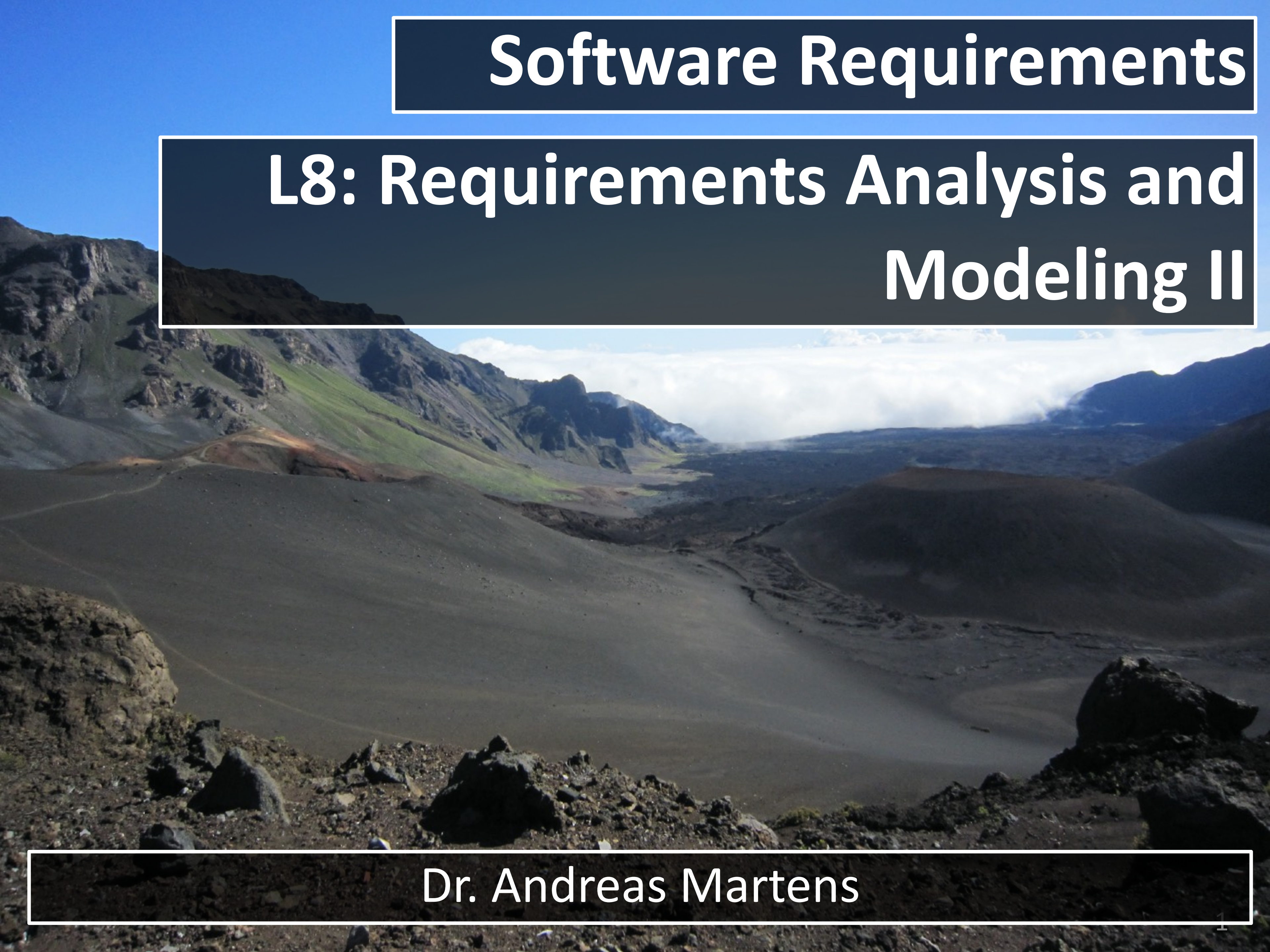
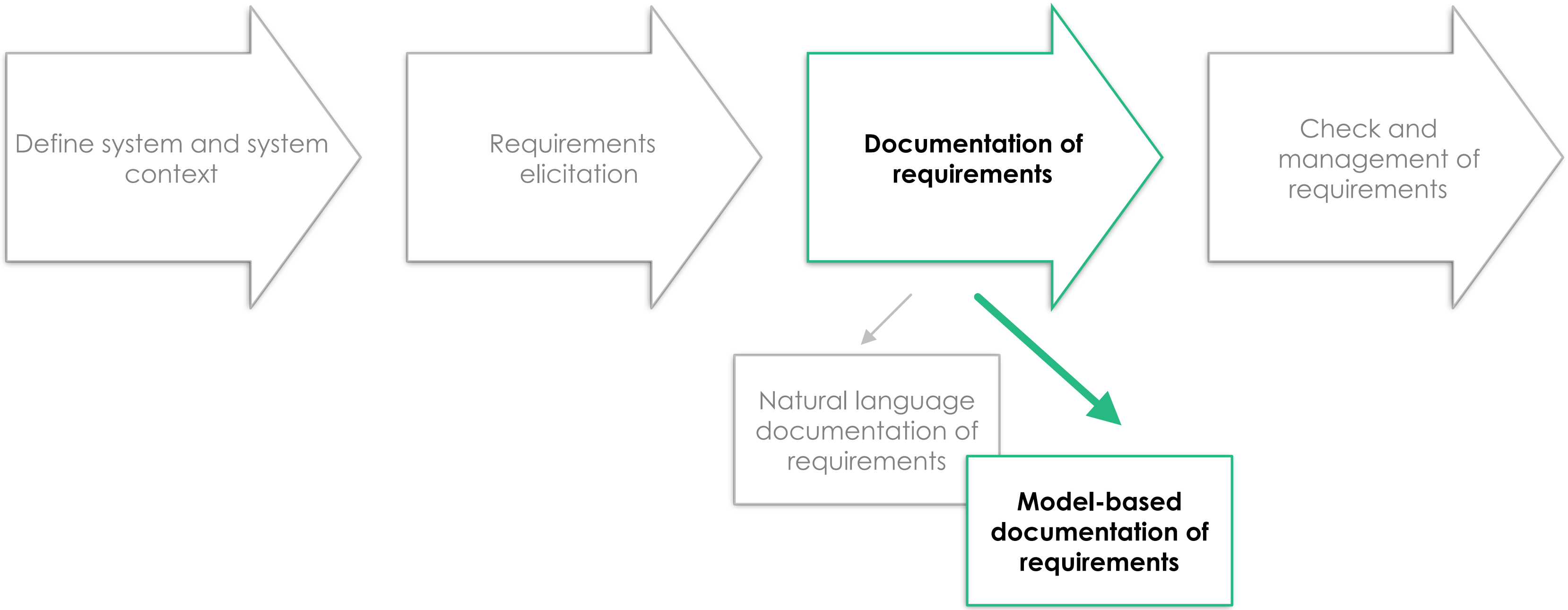


Software Requirements

L8: Requirements Analysis and Modeling II

Dr. Andreas Martens

Requirements engineering process



Define system and system context

Requirements elicitation

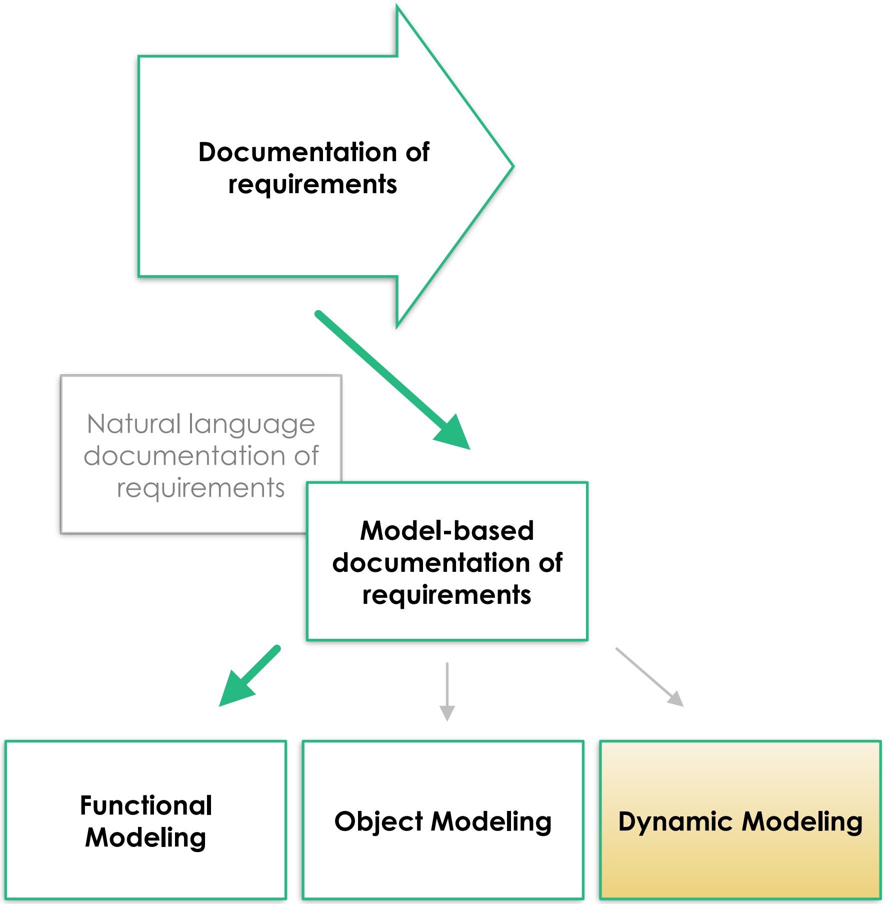
Documentation of requirements

Check and management of requirements

Natural language documentation of requirements

Model-based documentation of requirements

Overview



Documentation of requirements

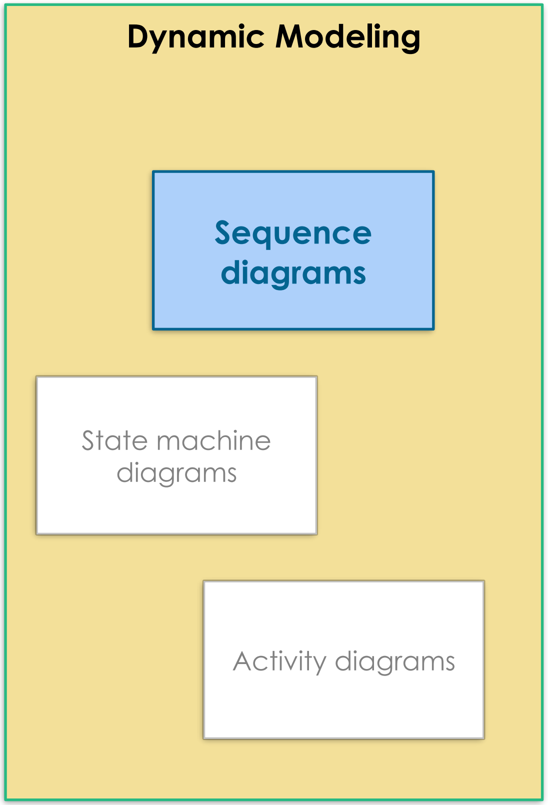
Natural language documentation of requirements

Model-based documentation of requirements

Functional Modeling

Object Modeling

Dynamic Modeling



Dynamic Modeling

Sequence diagrams

State machine diagrams

Activity diagrams

Dynamic modeling

Describe the components of the system that have interesting dynamic behaviour

Why do we need modeling of the dynamics?

- IT systems are about **flow** of activities, data and workflow states
- Common pattern: If - then - else

How to do this?

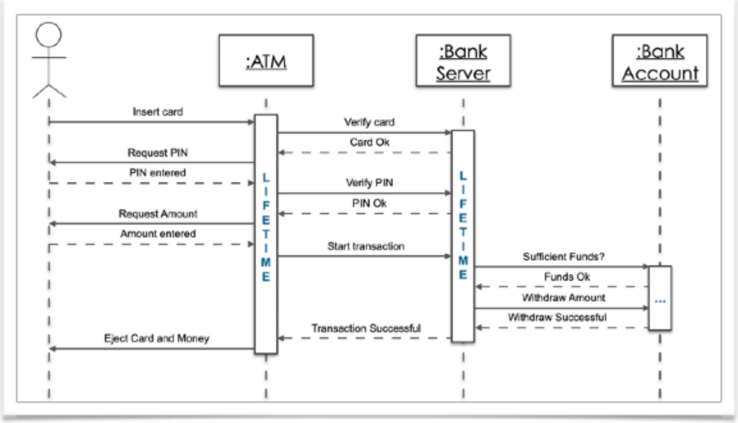
- **Sequence diagrams**: For the interaction between different objects
- **State diagrams**: One state diagram for each class with interesting dynamic behaviour
- **Activity diagrams**: graphically represents the networking of elementary actions and their connections with control and data flows

Dynamic modeling - Sequence Diagrams

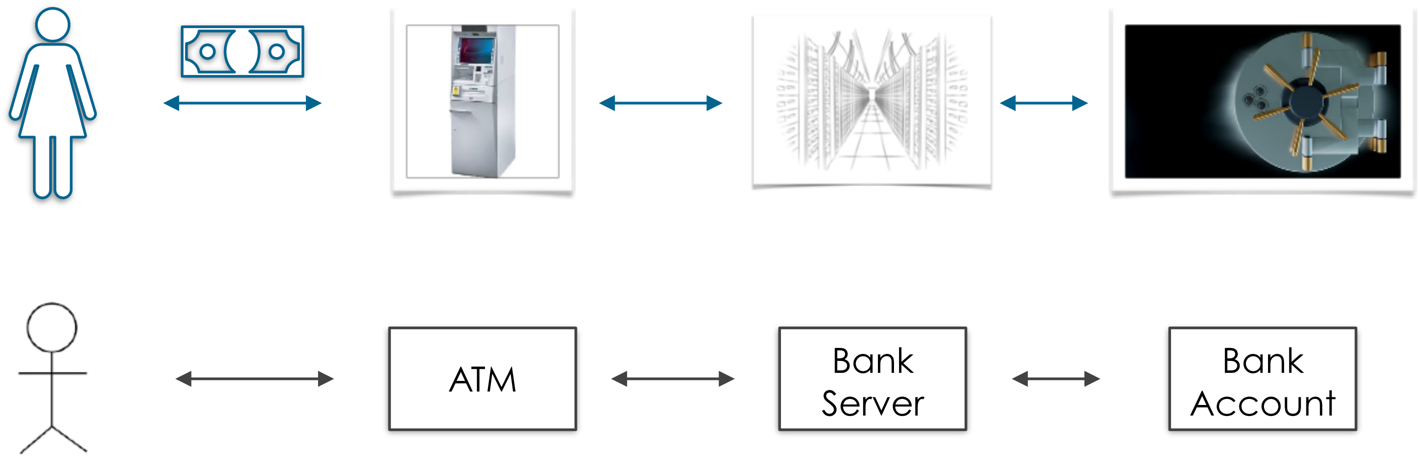
Search for interacting objects in a story

- Start with **flow of events** in scenarios and use case descriptions
- These are messages between two objects
- An event always has a **sender** and a **receiver**

From the flow of events create a **sequence diagram** (of events)



Example: Sequence Diagrams

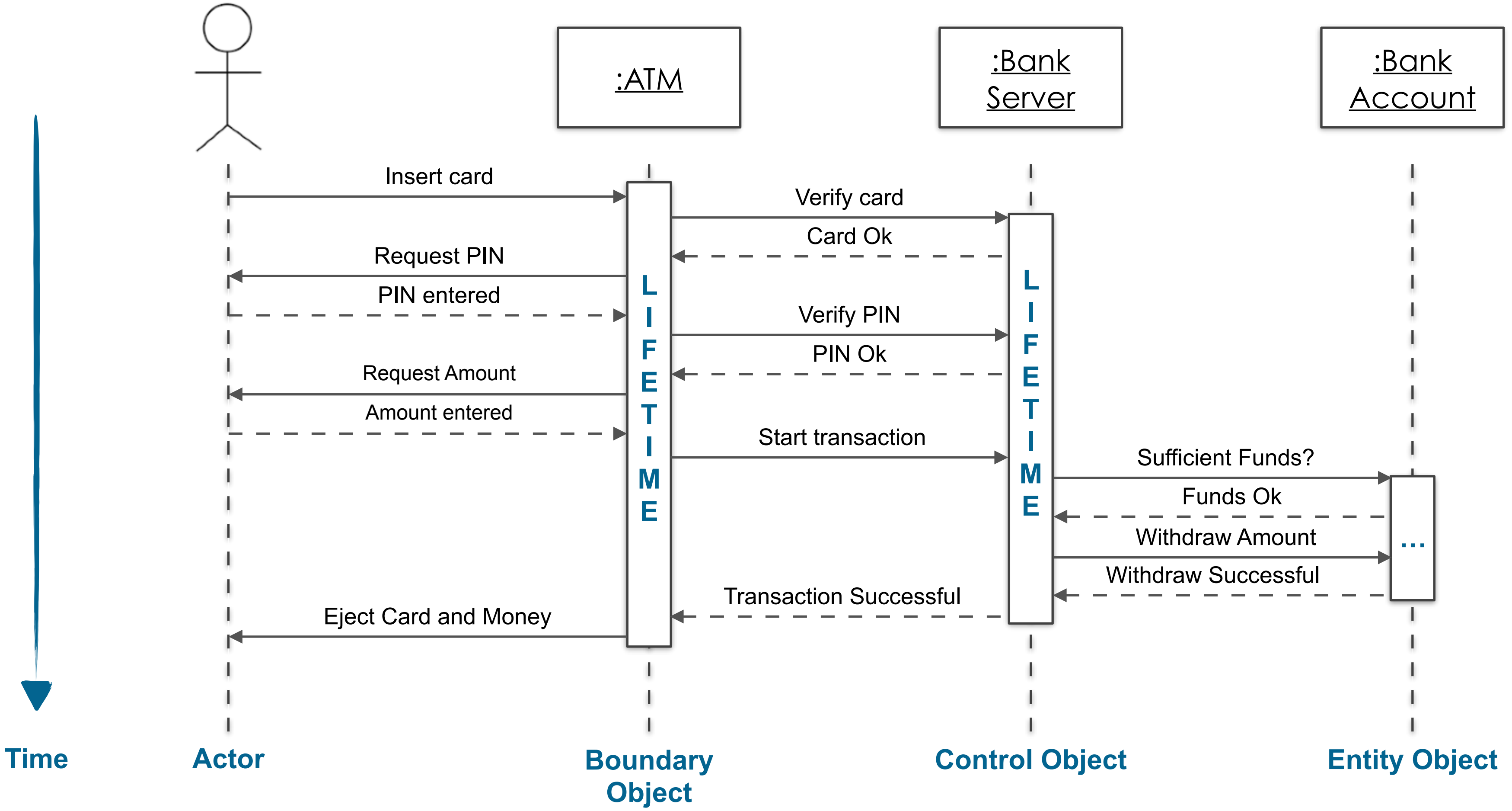


ATM

Bank Server

Bank Account

Example: Sequence Diagrams



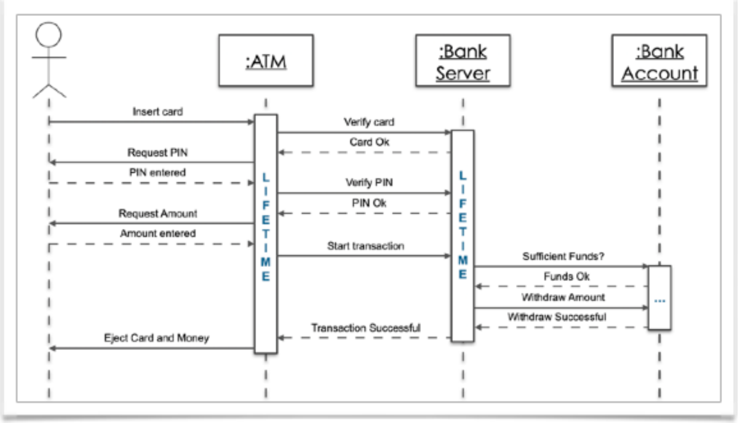
Properties

Derived from use cases

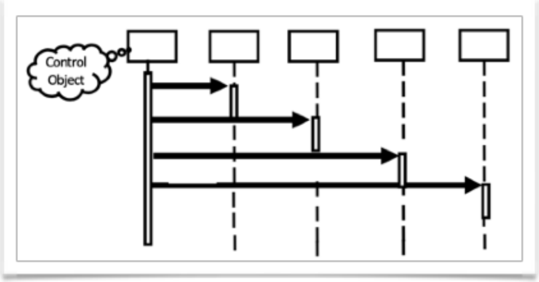
Their structure helps us to determine how **decentralized** the system is

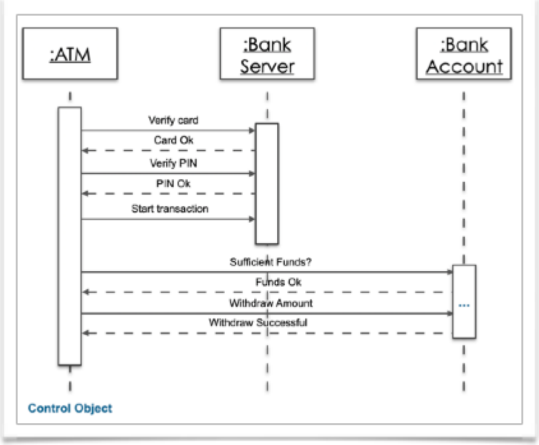
Sequence diagrams help to identify:

- The **temporal relationship** between objects over time
- **Sequence of operations** as a response to one ore more events



Fork Sequence Diagrams





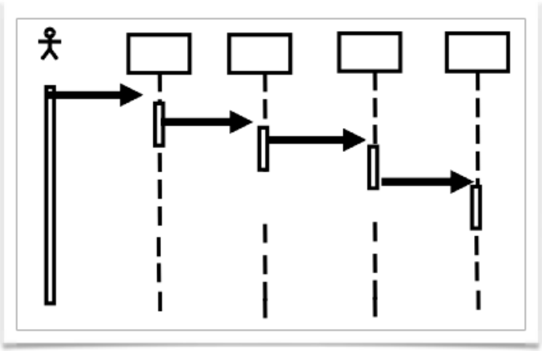
Fork Sequence Diagrams: The dynamic behaviour is placed in a single object

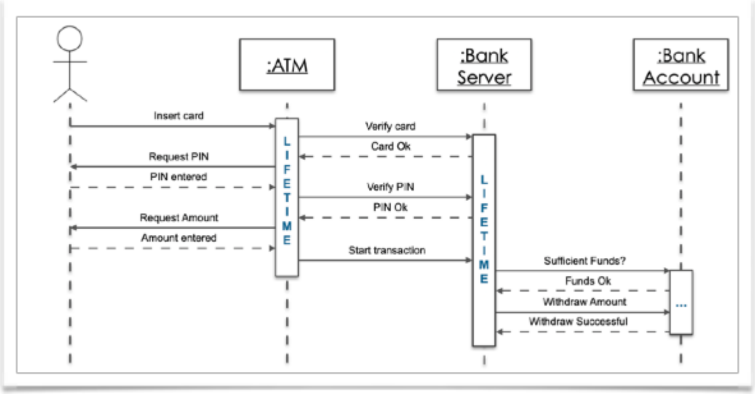
- This is usually a control object
- It knows all objects and uses them for direct requests and commands

Stair Sequence Diagrams

Stair Sequence Diagrams: The dynamic behaviour is distributed

- Each object delegates responsibility to other objects
- Each object knows only a few other objects (which objects can help with a specific behaviour)





Fork or stair?

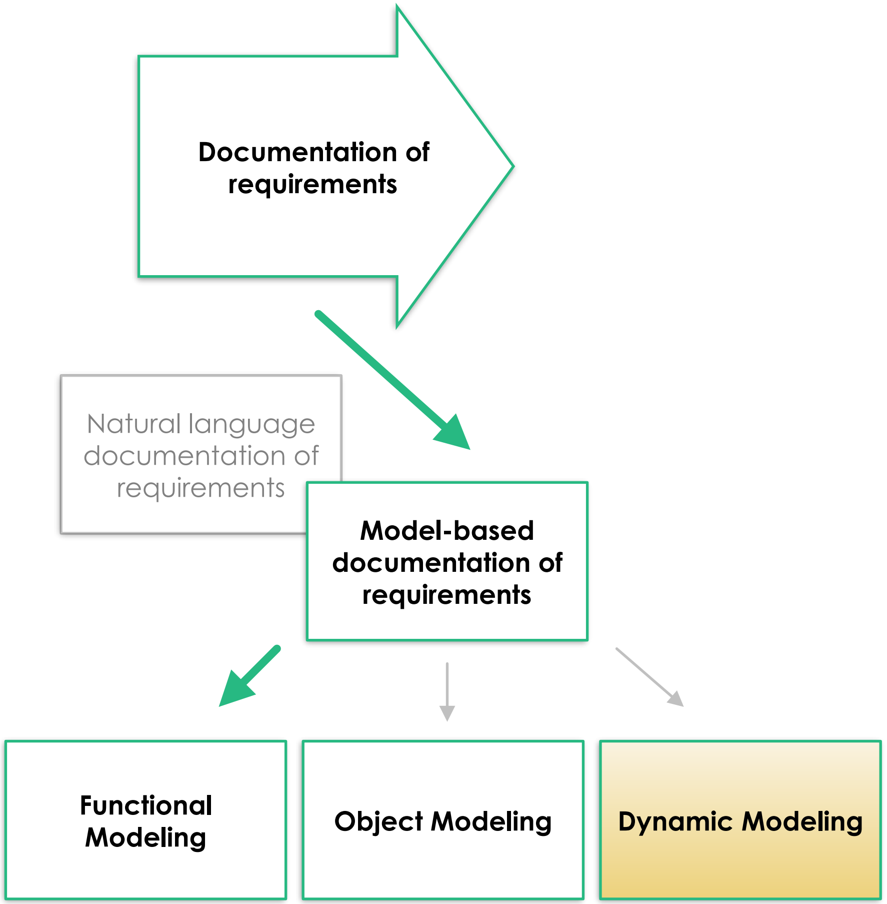
Use the fork if…

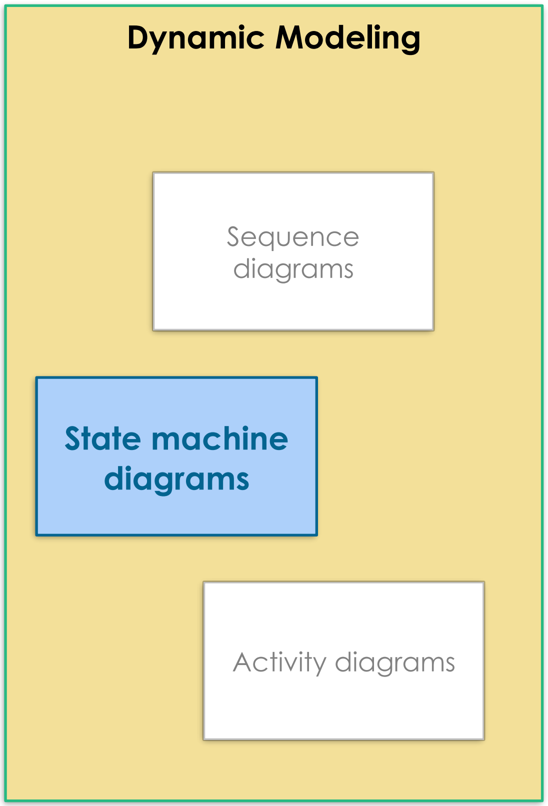
- You need **centralized** control structure
- The operations can **change order**
- New operations are expected to be **added**

Use the stair if…

- You need a **decentralized** control structure
- The operations have a **strong connection**
- The operations must be performed in the **same order**

Next step


Documentation of requirements
Natural language documentation of requirements
Model-based documentation of requirements
Functional Modeling
Object Modeling
Dynamic Modeling


Dynamic Modeling
Sequence diagrams
State machine diagrams
Activity diagrams

State Machine Diagram

State Machine Diagram

- A **notation** for a state diagram that describes the response of an object to the receipt of outside stimuli (events)

State machine

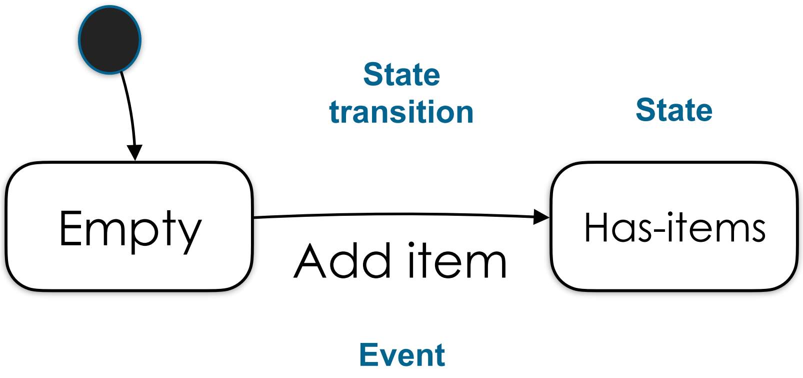
- A **model** of behavior composed of a finite number of states, transitions between these states, and actions
- A **Moore Machine** is a special type of state machine, where the output depends only on the state
- A **Mealy Machine** is a special type of state machine, where the output depends on the condition, event, action of the transition, and the state

State Machine Diagram

State Machine Diagrams help to identify changes to an individual object over time

State: An abstraction of the attributes of a class

- *Event*(*attr*) [condition]
- State has duration

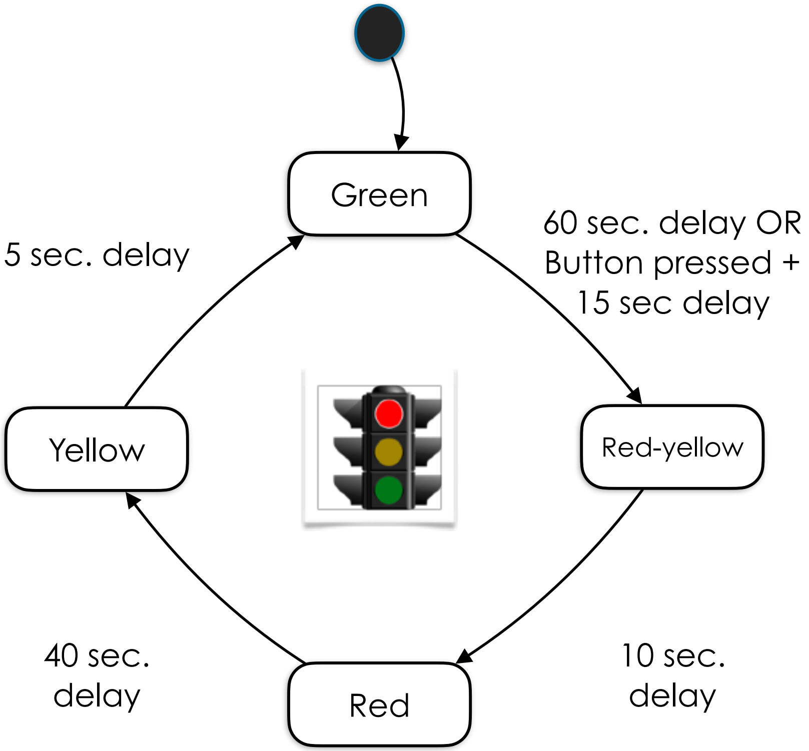


Example: State Machine Diagram

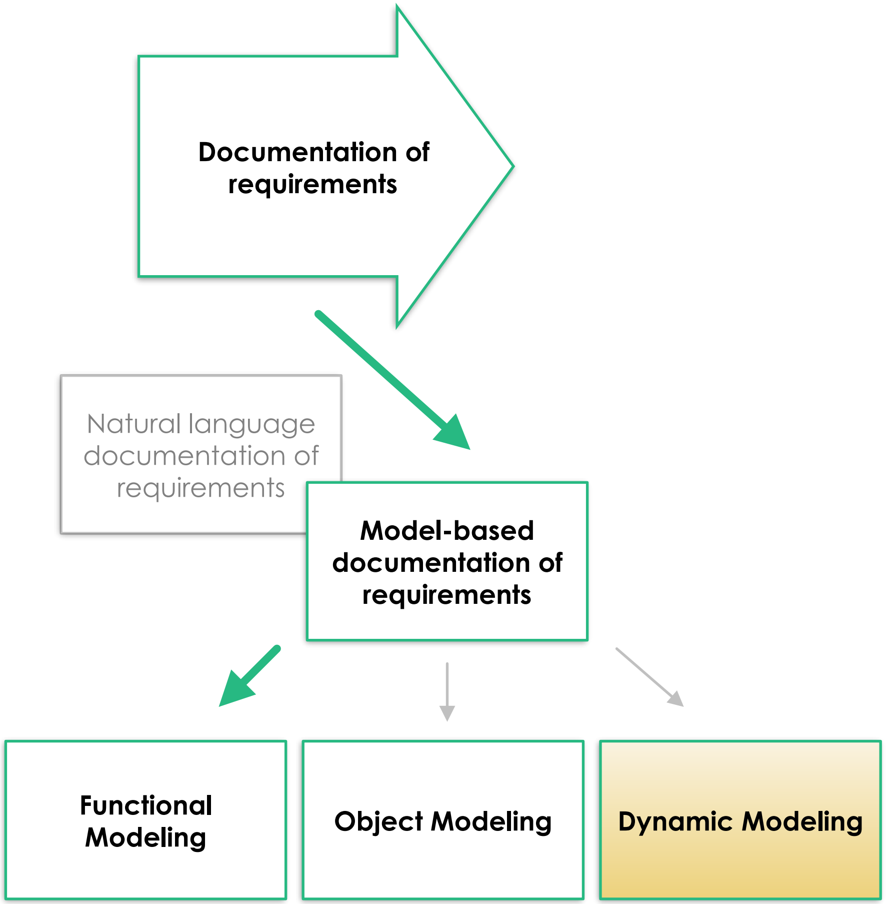
Business requirement: *„Traffic light should control the crossing of a road“*

Real requirements, specified in the workshops:

- Traffic light turns red-yellow after 60 seconds
- Traffic light turns also red-yellow if a button pressed (after 15 sec. delay)
- After 10 seconds traffic light turns red
- After 40 seconds traffic light turns yellow
- After 5 seconds the traffic light turns green again



Overview


Documentation of requirements

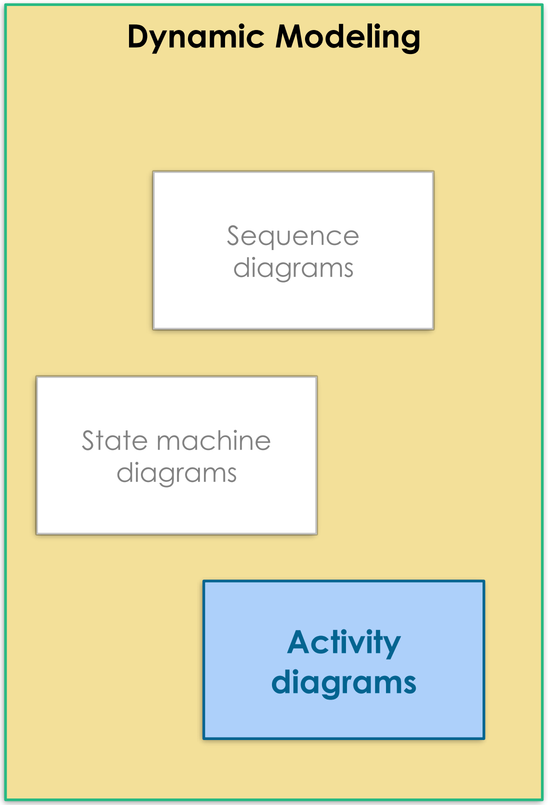
Natural language documentation of requirements

Model-based documentation of requirements

Functional Modeling

Object Modeling

Dynamic Modeling


Dynamic Modeling

Sequence diagrams

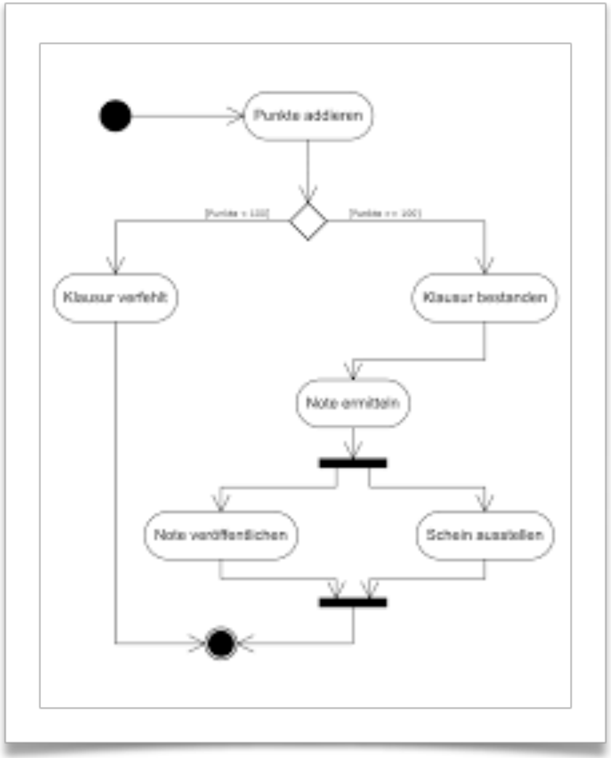
State machine diagrams

Activity diagrams

Activity Diagram

Activity Diagram

- Modelling of activity nodes and control flows between the activity nodes
- graphical representations of workflows of stepwise activities with support for choice, iteration and concurrency
- Synchronization bars in activity diagrams allow the modeling of concurrent control and object flows. Alternative control and object flows can be described by decision nodes.

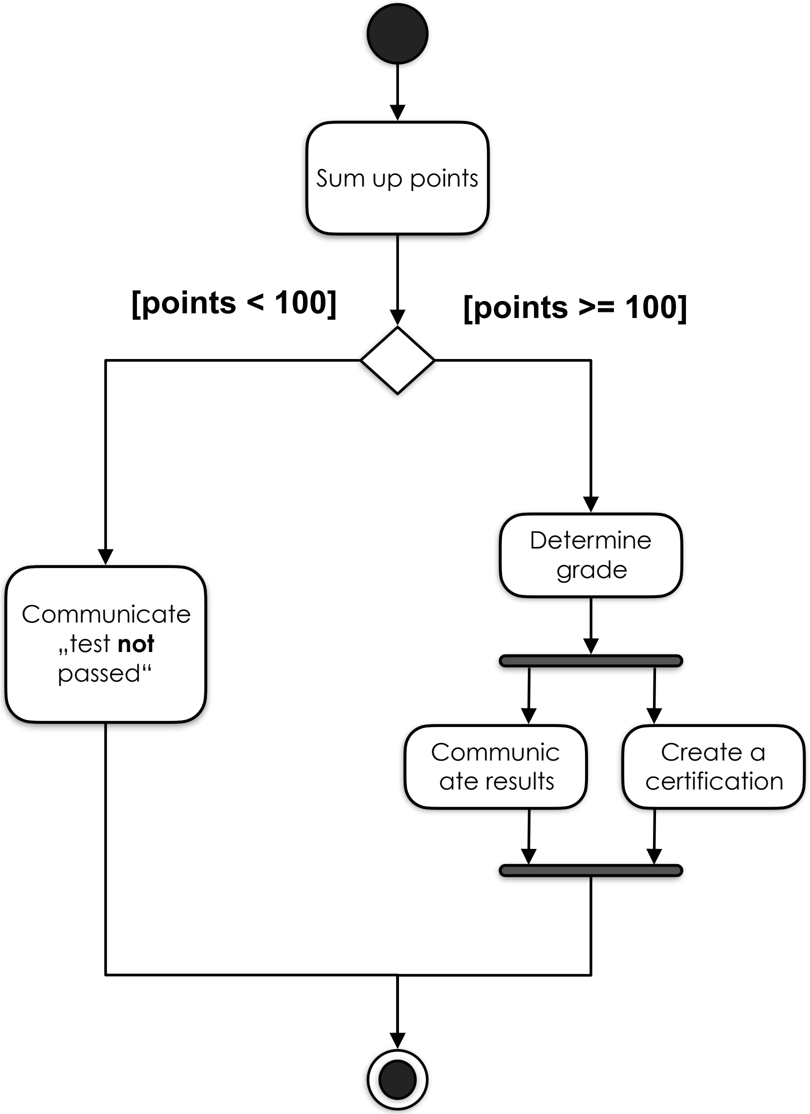


Example: Activity Diagram

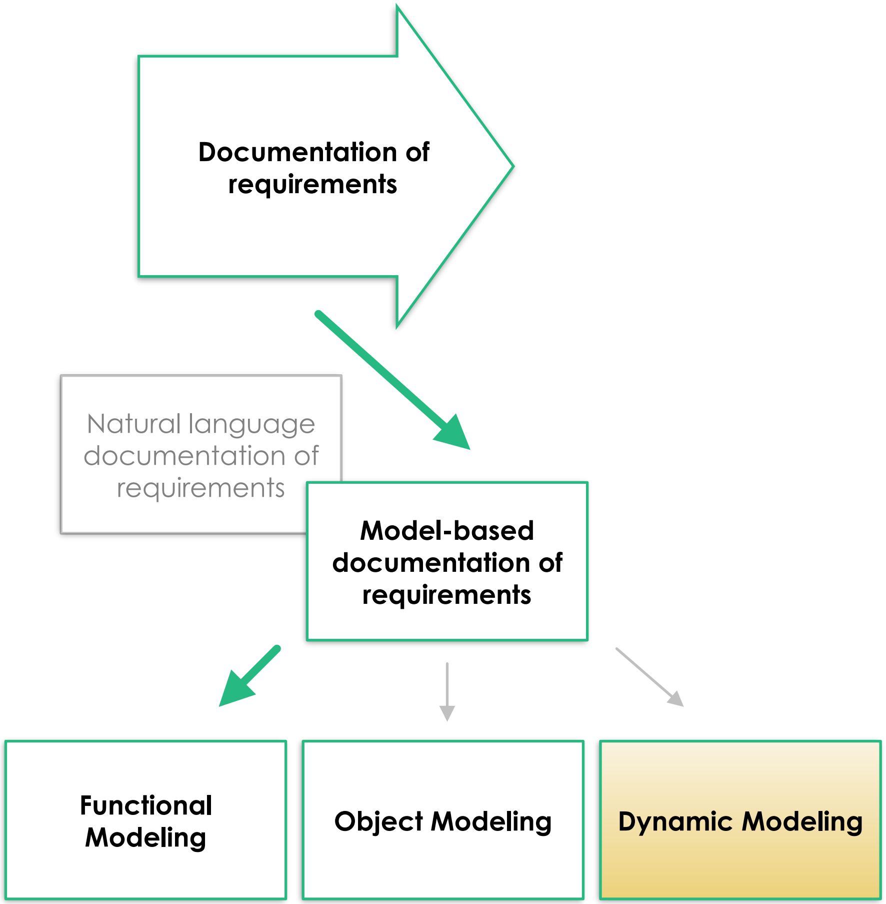
Business requirement: *„Review and communicate the results of an exam“*

Real requirements, specified in the workshops:

- Check all tasks of the exam and sum up points
- The test is passed if one have achieved at least 100 points
- Create an examination certificate when passing the exam
- It contains the grade that is still to be calculated
- Communicate results to the test participants



Analysis Example



Documentation of requirements

Natural language documentation of requirements

Model-based documentation of requirements

Functional Modeling

Object Modeling

Dynamic Modeling

Analysis Let us do analysis for a toy example

Business says: „*I want to have something like that*"

1. Understand business
2. Identify requirements
3. Define Use Cases
4. Build diagrams
   - Functional modeling
   - Object modeling
   - Dynamic modeling

1. Understand business

What do we know about the business?

- We are talking about a technical toy (not for baby’s and seniors)
- For children (trigger) but also for adults (payer)
- This is a remote-controlled vehicle (normally drives on the ground)
- It's moving at high speed (makes fun)
- It is electrically driven (needs batteries, can have high acceleration)
- The toy is mainly played outdoors
- …

2. Identify requirements

Wat is required?

- It should be a car
- A person wants to use a remote control to drive the toy car
- The vehicle should be able to drive straight ahead and also turn
- Appropriate controls should be available on the remote control
- The speed of the vehicle should be high
- The reaction time should be very short

- If the battery in the remote control is empty, the vehicle should stop
- The vehicle should be of robust construction
- Only certain frequencies may be used for signal transmission
- The range of the control should be sufficient
- It'd be cool if it could swim, too

3. Define Use Cases

Use Case 1: Move car forward

- Entry condition: The car is not moving
- Flow of events:

    1. Driver turns power on

- Exit condition: Car moves forward

Use Case 2: Turn car

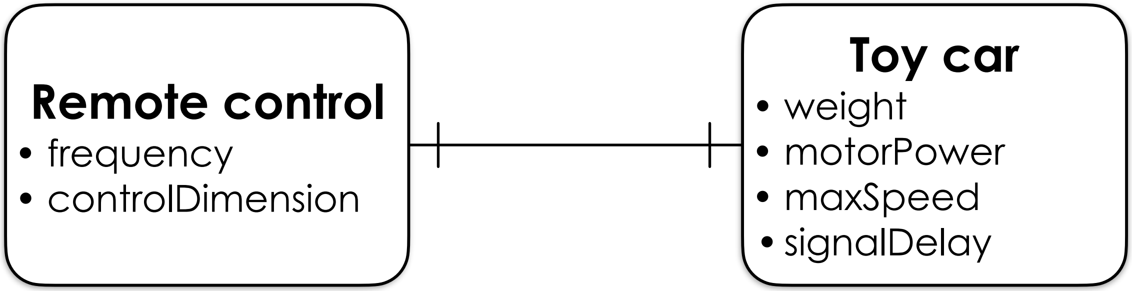
- Entry condition: The car is moving
- Flow of events:

    1. Driver operates steering

- Exit condition: Car turns

4. Build Diagram

Entity Relationship Diagram



Remote control

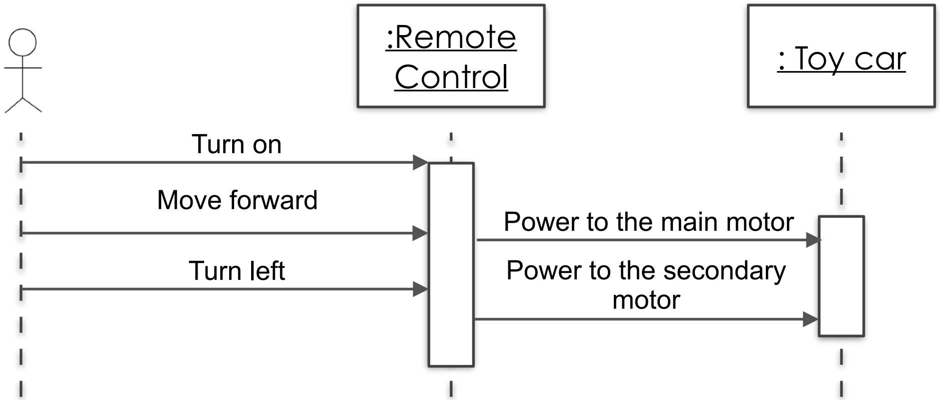
- frequency
- controlDimension

Toy car

- weight
- motorPower
- maxSpeed
- signalDelay

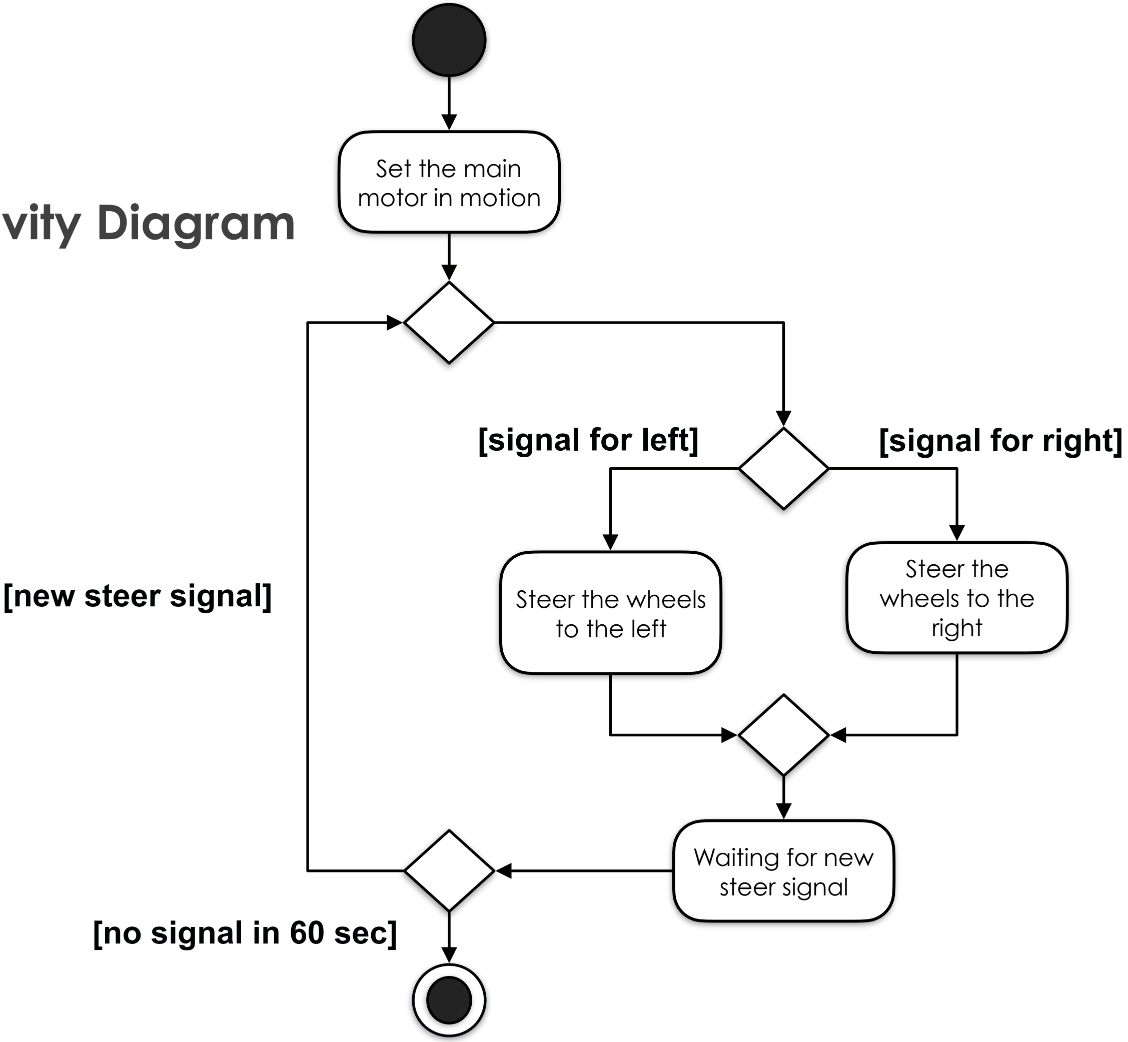
4. Build Diagram

Sequence Diagram

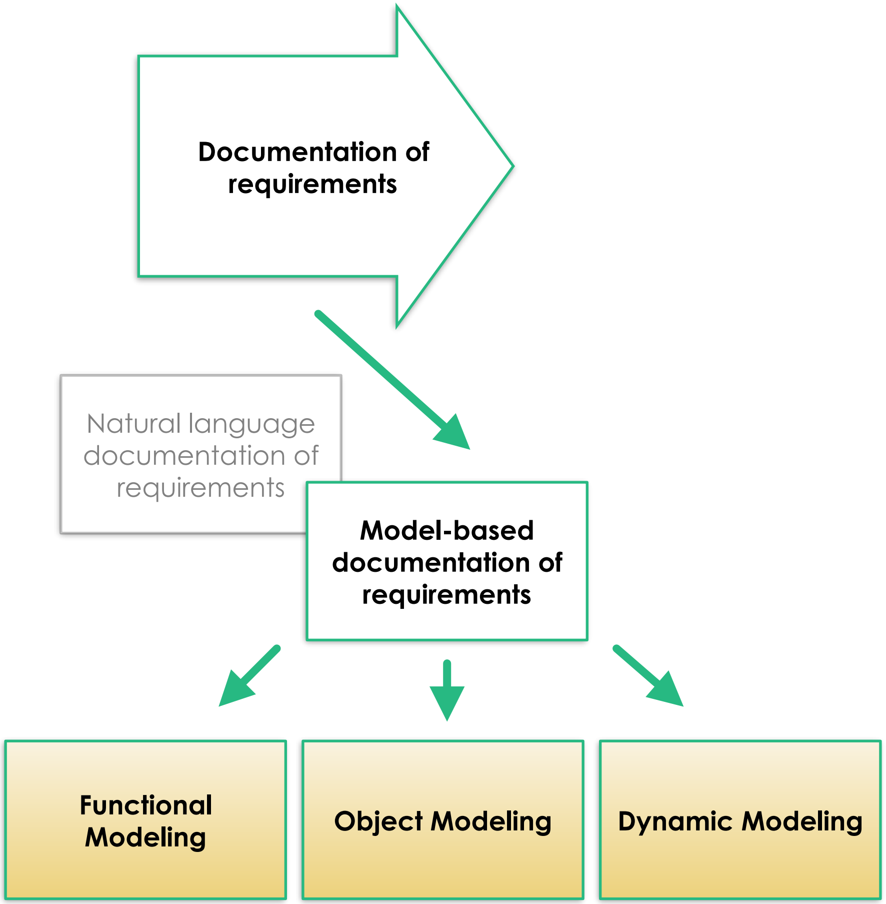


4. Build Diagram

Activity Diagram


Set the main motor in motion
[signal for left]
[signal for right]
Steer the wheels to the left
Steer the wheels to the right
[new steer signal]
Waiting for new steer signal
[no signal in 60 sec]

Summary



Documentation of requirements

Natural language documentation of requirements

Model-based documentation of requirements

Functional Modeling

Object Modeling

Dynamic Modeling