Software Systems

Design Lecture 1A
Activity Diagrams

Joke van Staalduinen

Credits to Klaas Sikkel

Contents

1.1 Design Lectures + Lab Sessions

1a Activity Diagrams
1b Use case models
2a Class Diagrams
2b Sequence Diagrams
3a State Machines

Unified Modeling Language

3b Version management
4a Maintainability, Software Metrics

Design Project

- Design a software system for parking houses
  - Groups of 4 persons (on Canvas)
  - Kick-off today, hour 4

Design Test

- Test questions are similar to lab session exercises
  - Signing off all D-exercises is required for admission to the test

Out task as System Developers/Computer scientists/…

Develop a Computer System

1.2 What is a system?

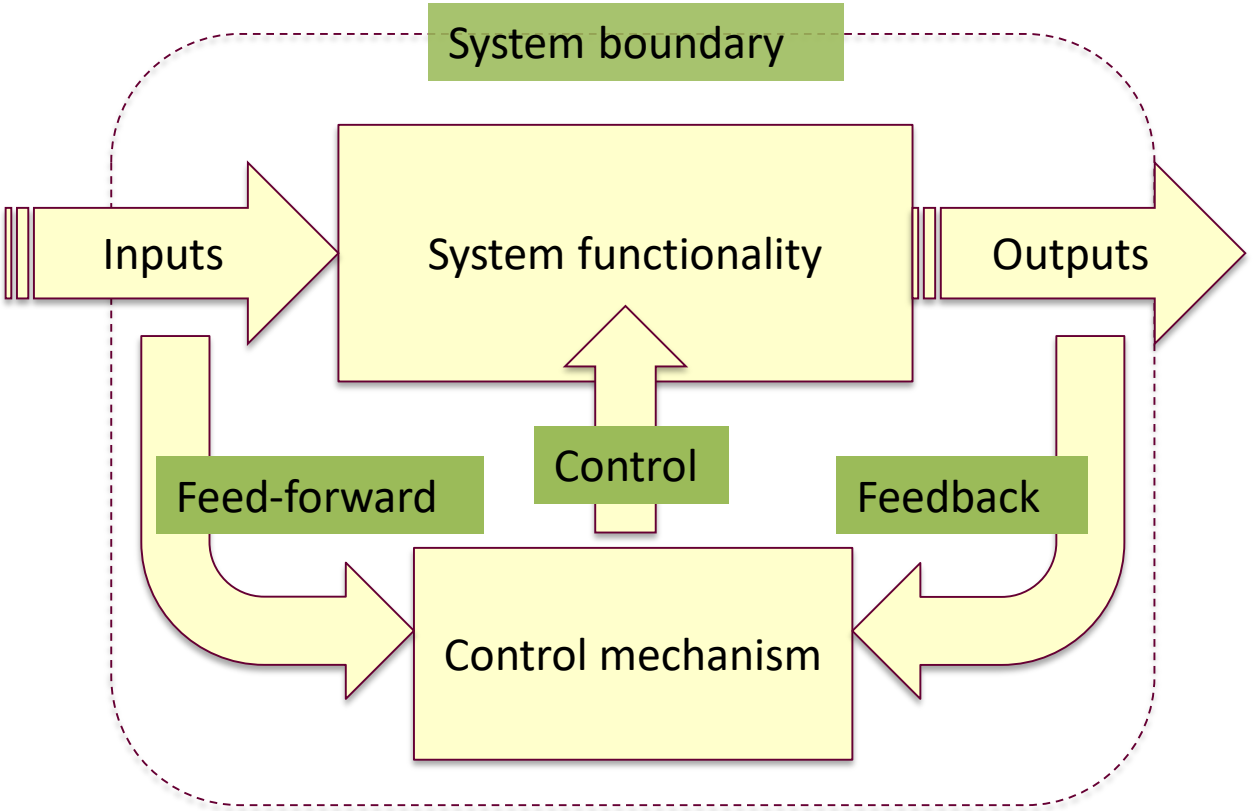
- A system is a collection of inter-related components that work together to achieve some common goal. The system changes input to output.
- The components are dependent on each other
- Each system has a boundary and can include software, hardware and people, depending on where the boundary is defined

Examples of Systems

- Digestive system
- Economic system
- Electricity system
- Clock in/out system
- Immune system
- Central heating system
- Etc

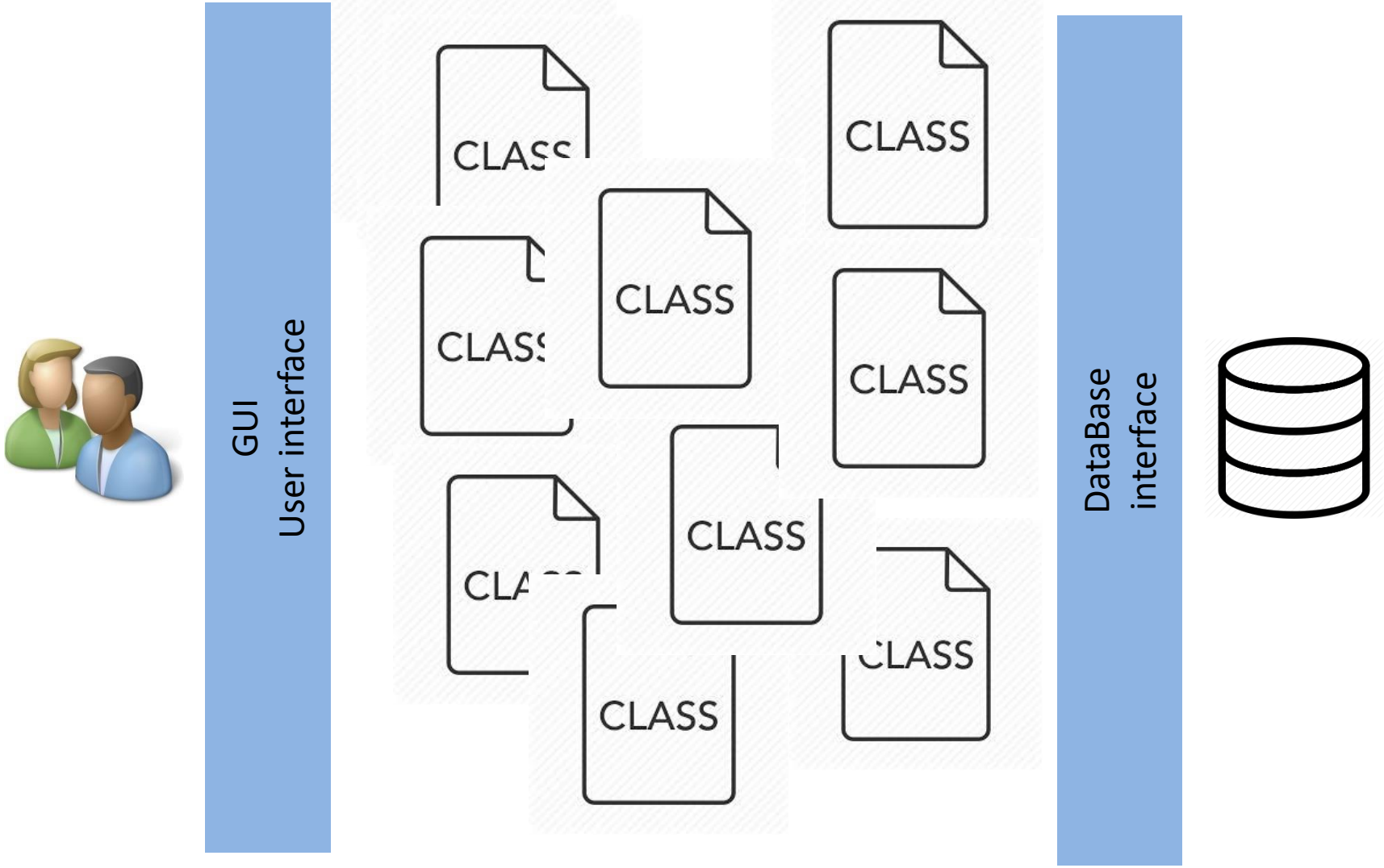
1.2 What is an Information system?

- Computer System =
- Software System
- Information System



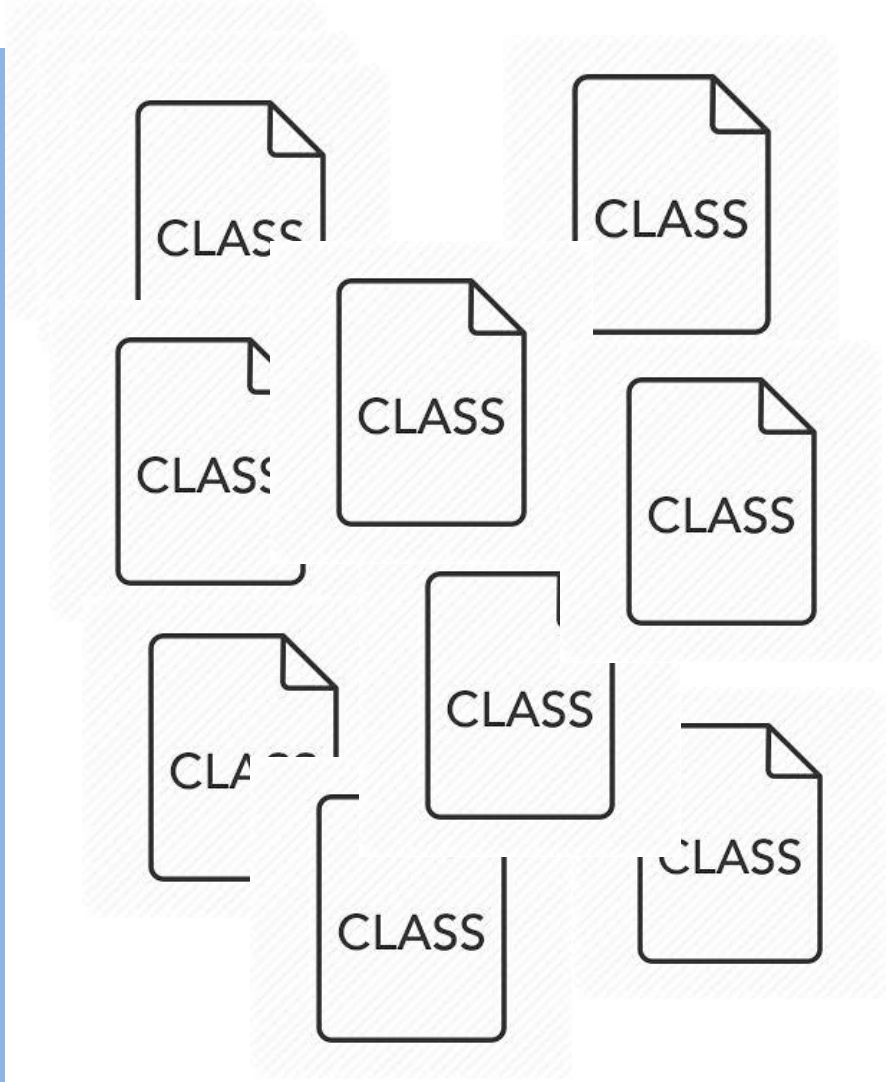
[Bennett et al., Figure 1.3]

Developing an Information System



What do we want to achieve in M2 Design?

GUI
User interface



How do we want to achieve this?

- Find out and understand the user requirements
- Model the requirements so that the developers ca create the system

Requirements Engineer

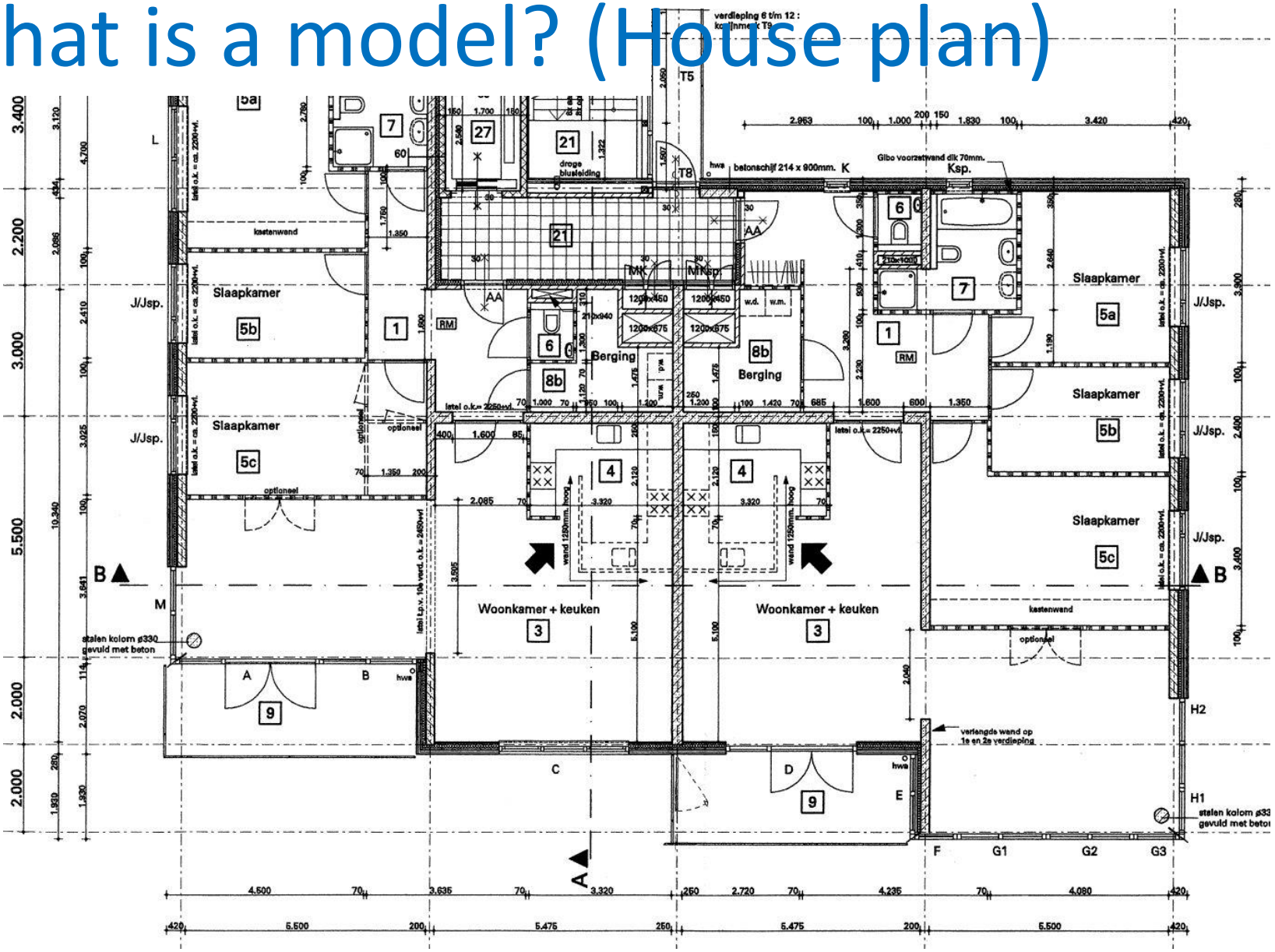
Design models

What is a model?

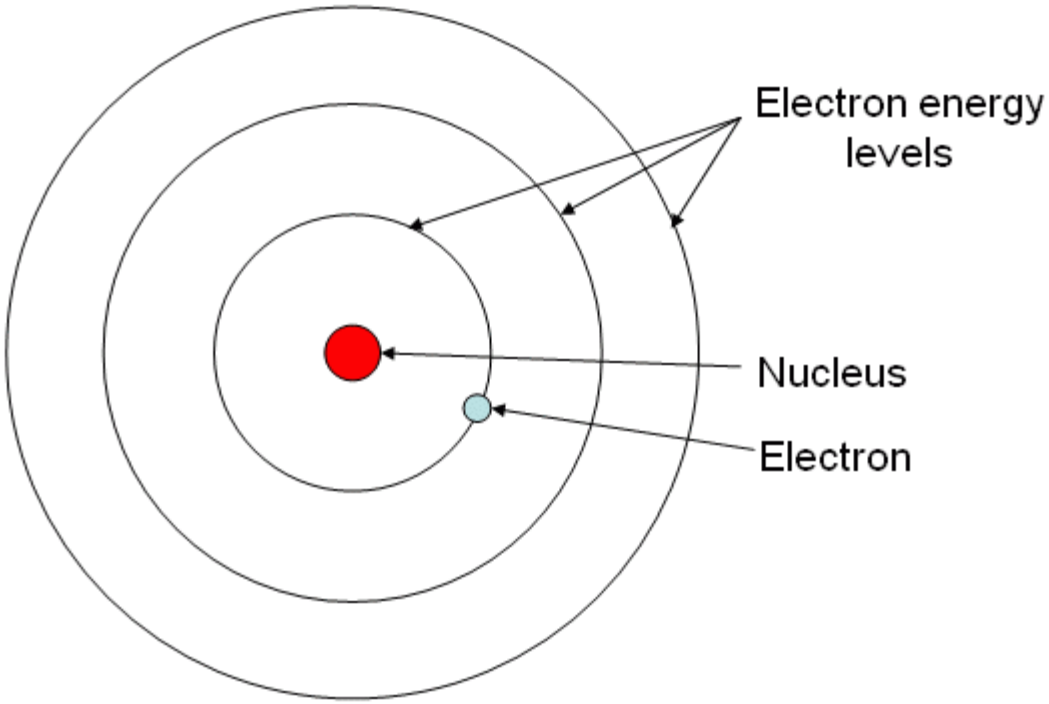
- A model is a **simplified** representation of part of the world (or part of a software system) **from a particular view**

What is a model? (Scale model)

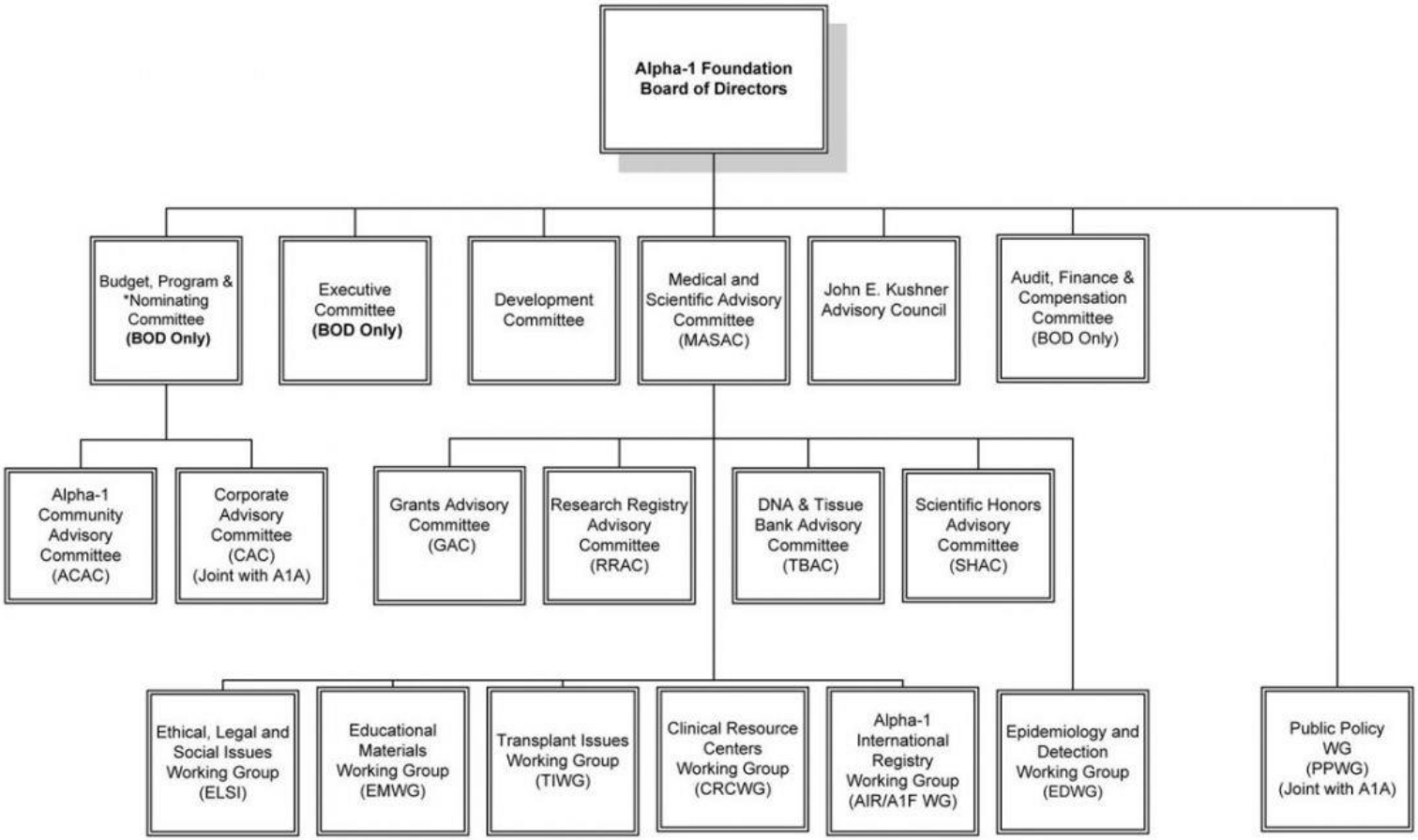
What is a model? (House plan)



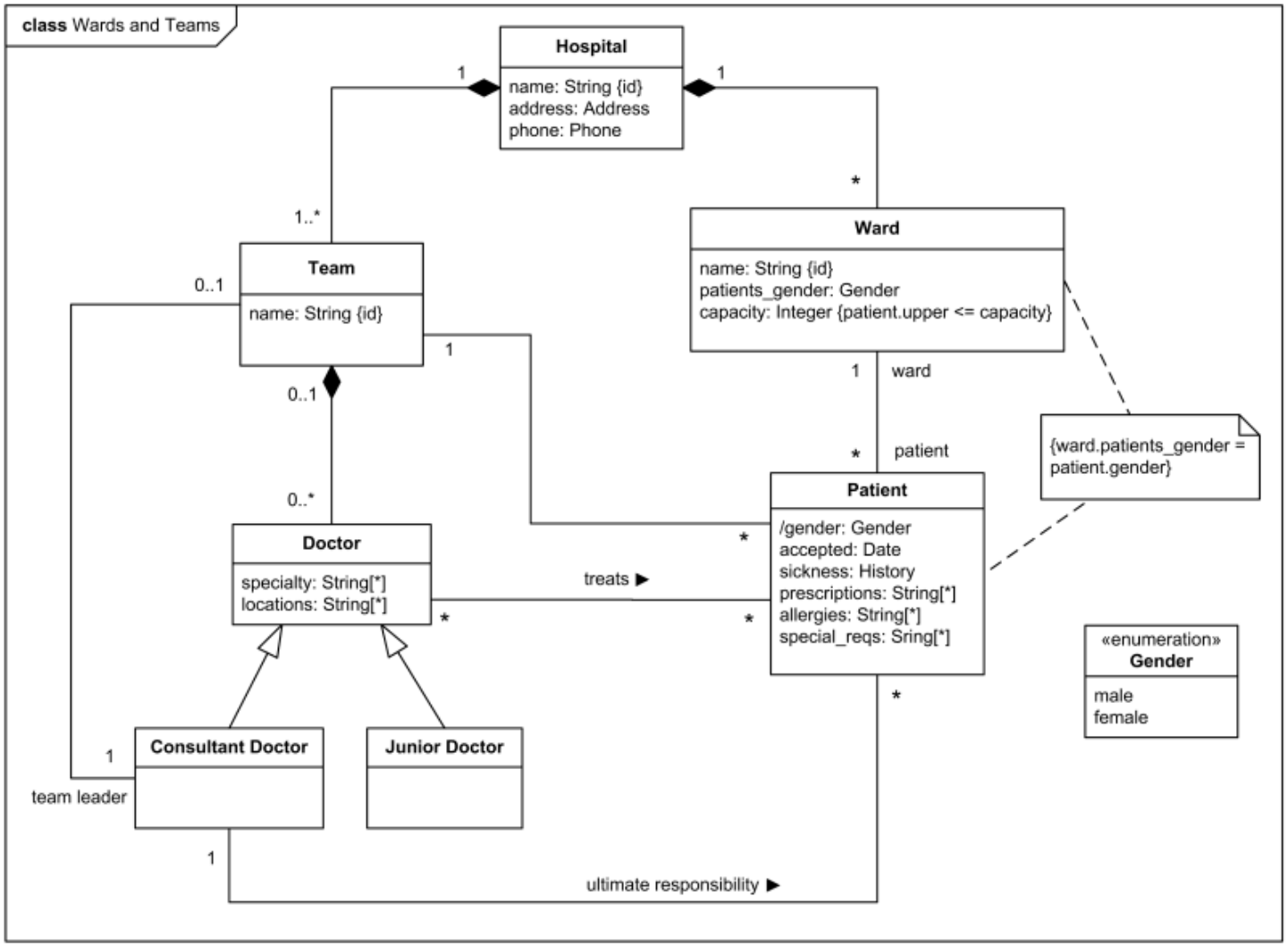
What is a model? (Universal Physics law)



What is a model? (Organization chart)



What is a model? (Class diagram)



What is a model?

- Model as (part of a) theory
  - claims universal validity (Physics)
- Model as a structured representation of reality
  - Organization Chart, Scale Model
- Model as a prescription for a system/object that is to be constructed
  - Floor Plan, Scale model

UML Model

What is a model?

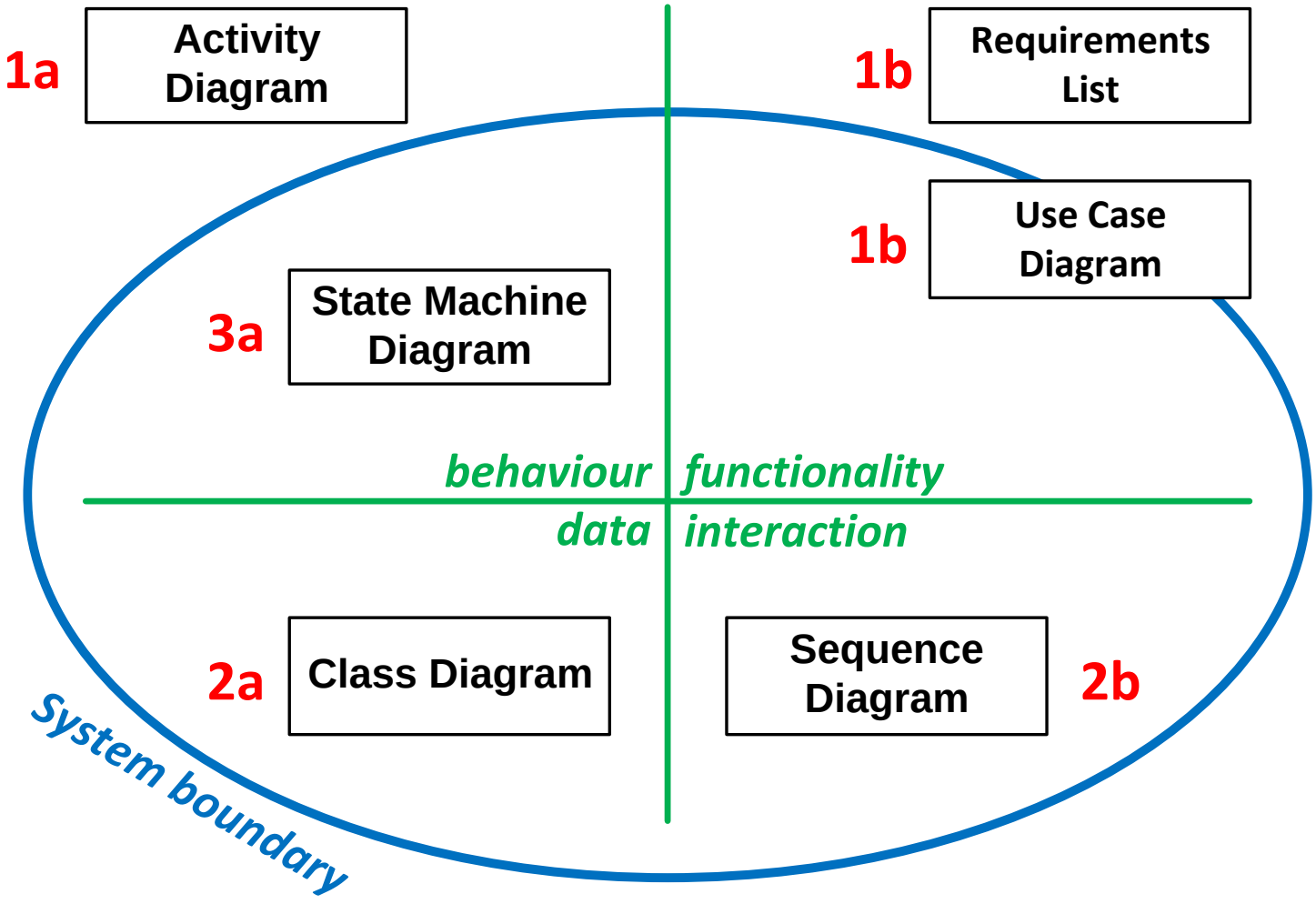
- A model is a simplified representation of part of the world (or part of a software system) from a particular view
  - Simplified: what do we discard, what do we represent?
  - Representation: which notation is to be used?
  - View: which particular aspect is modelled?

1.3 Unified Modelling Language

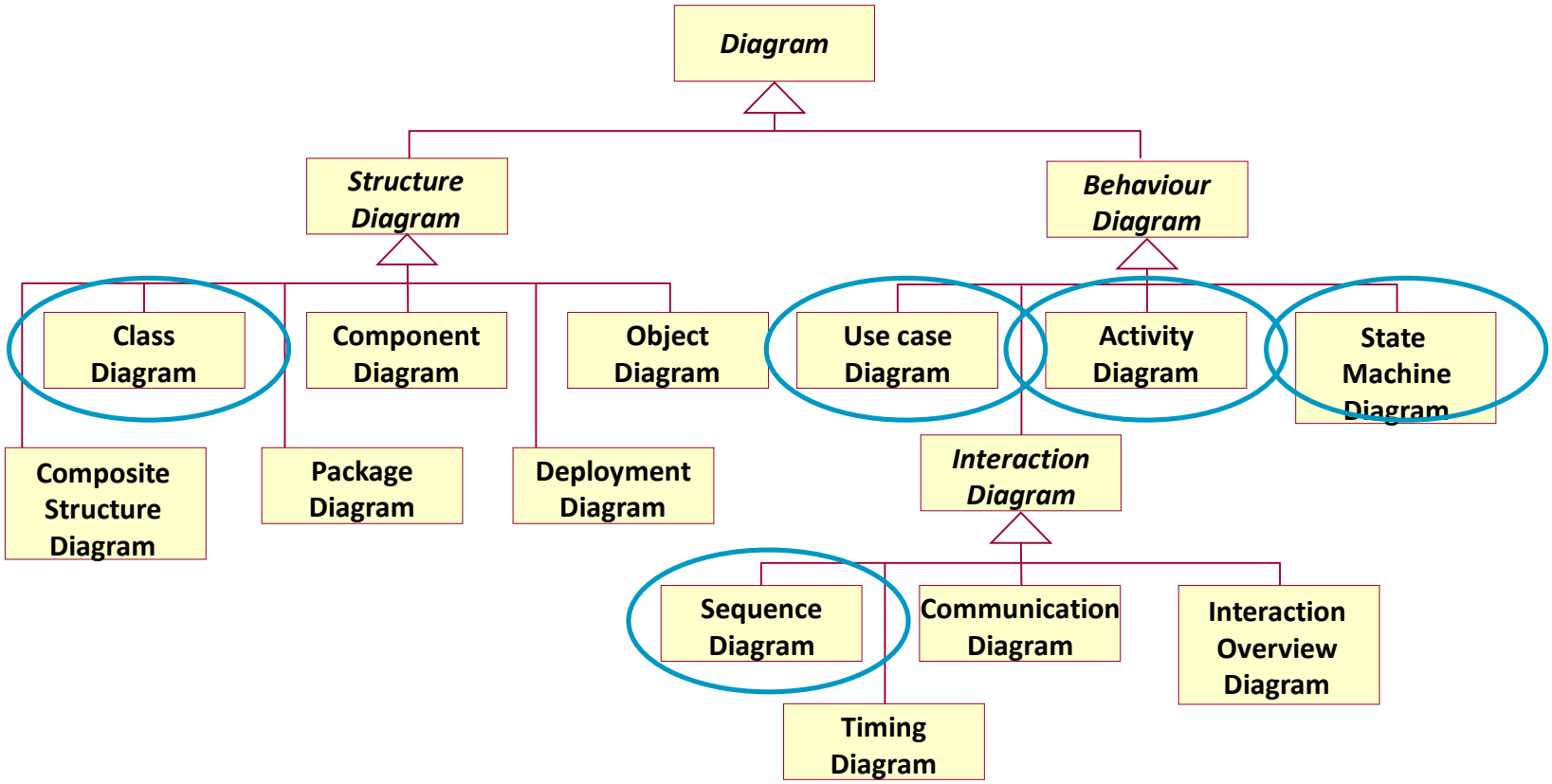
- In UML we (mostly) use diagrams as a representation to define models
- UML Diagrams are unambiguous, it is always clear what is meant

  (*In contrast to natural language, which very often leaves room for multiple interpretations*)

UML diagrams in this module



Unified Modelling Language



Literature about UML

- The slides are self-contained – everything you need to know for the lab sessions, the project, and the exam is (briefly) explained in the slides
- *If you want more explanation, you can use a book*
  - *Bennett et al.: Object 0riented Systems Analysis and Design using UML, McGraw-Hill, 4th ed., 2010*
  - *Or any other text book on UML*
  - *Or online resources (e.g. Wikipedia)*
- https://creately.com/blog/diagrams/activity-diagram-tutorial/#draw
- https://www.uml-diagrams.org/ticket-vending-machine-activity-diagram-example.html?context=activity-examples

1.4 Design (as a general concept)

[Wikipedia (Oct 2017)]:

- Design = the creation of a plan or convention for the construction of an object or a system
  (as in architectural blueprints, engineering drawings, business processes, circuit diagrams and sewing patterns)

Analysis and Design (in Softw. Eng.)

[Rumbaugh, 1997]:

- Design = Describing how the system will be constructed without actually building it

[Bennett et al, 2010]:

- Analysis = Finding out what the system should do and how the different system parts are related to one another
- Design = Modelling how the various parts of a system will work together

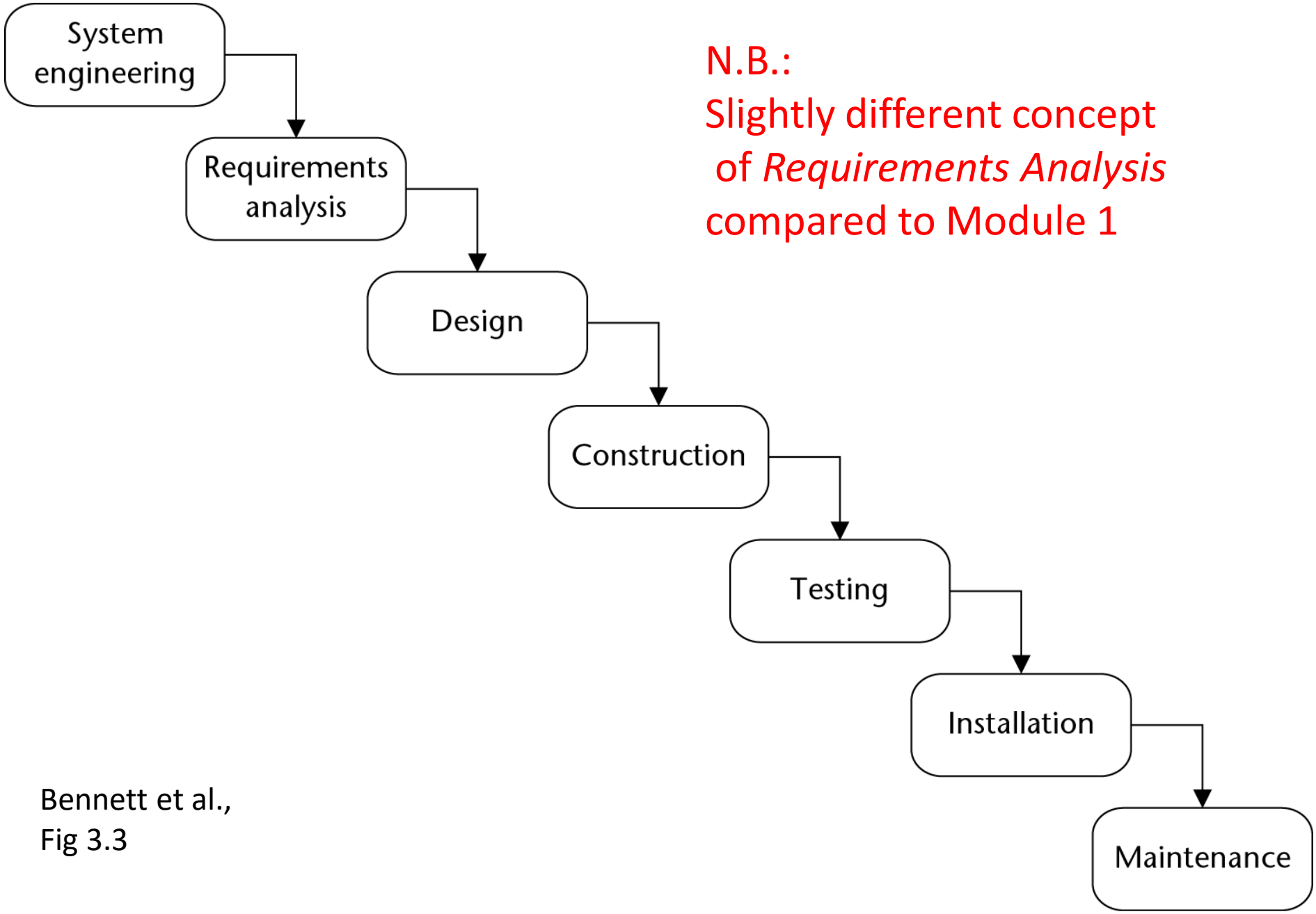
Analysis and Design (in Softw. Eng.)

The boundary between analysis and design is not well-defined:

- The *process of creating an initial design* can be regarded as part of the analysis of how a working system can be constructed

1.5 A structured approach to Software Development

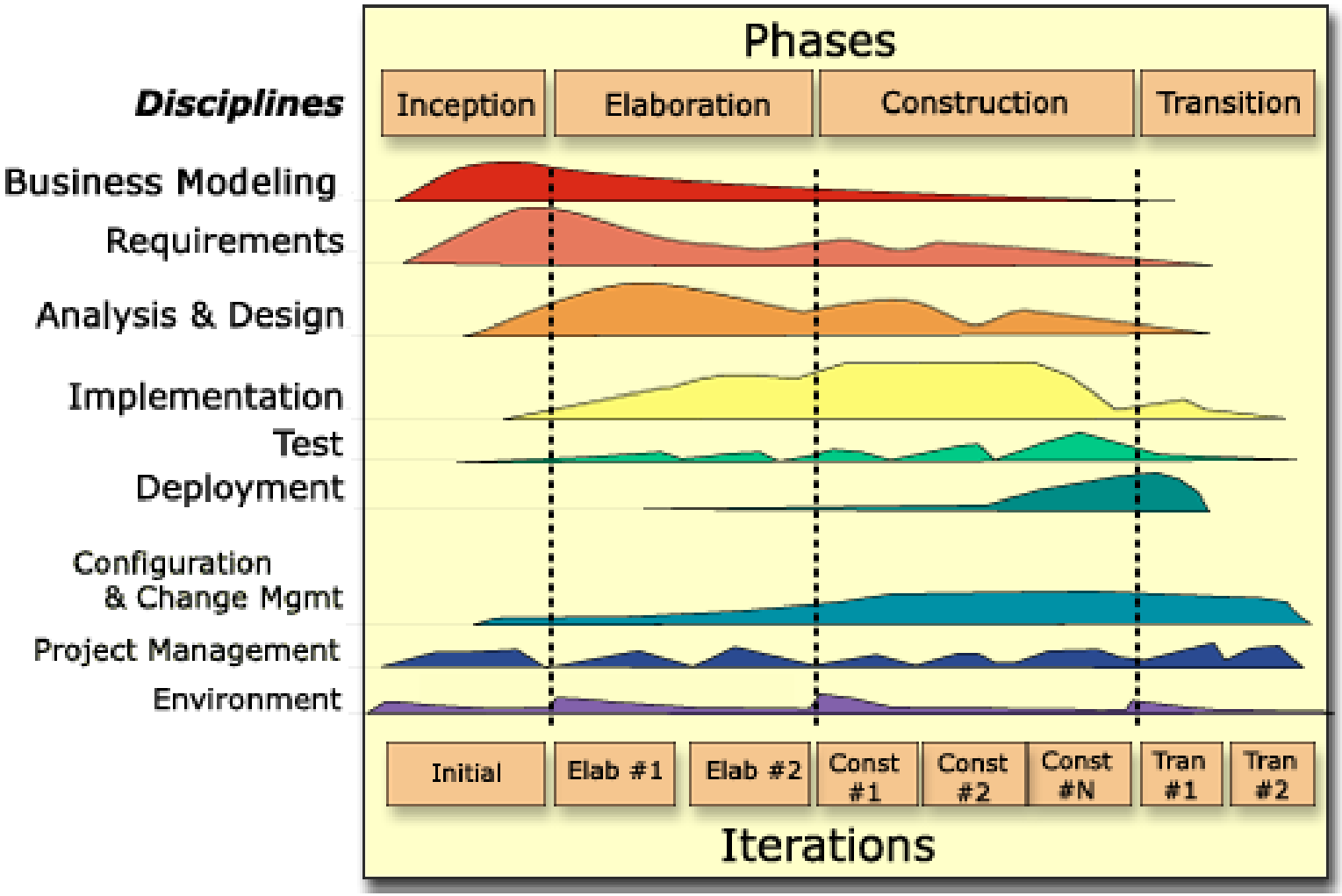
- Waterfall model vs. Agile development



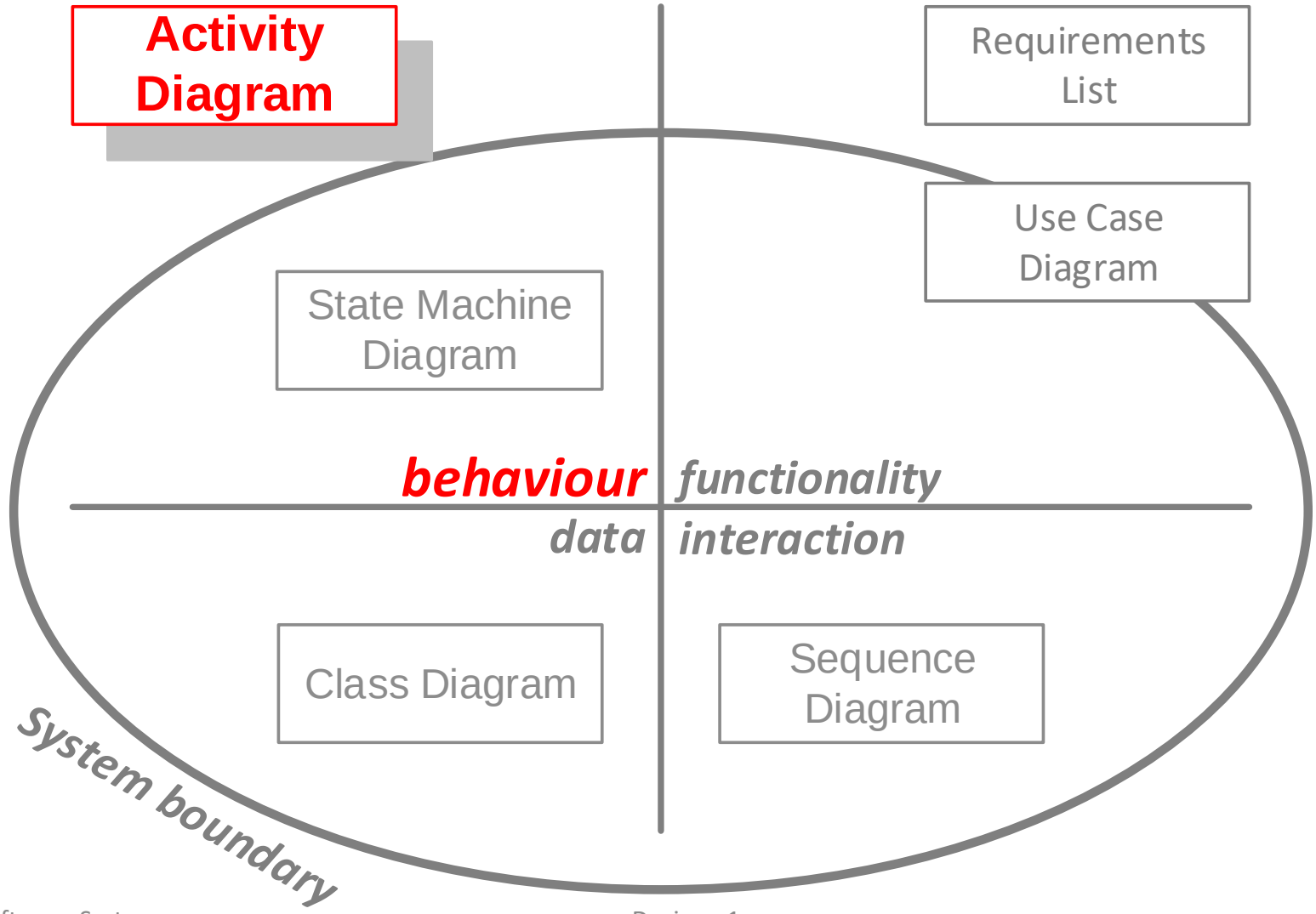
N.B.:
Slightly different concept of *Requirements Analysis* compared to Module 1

Bennett et al.,
Fig 3.3

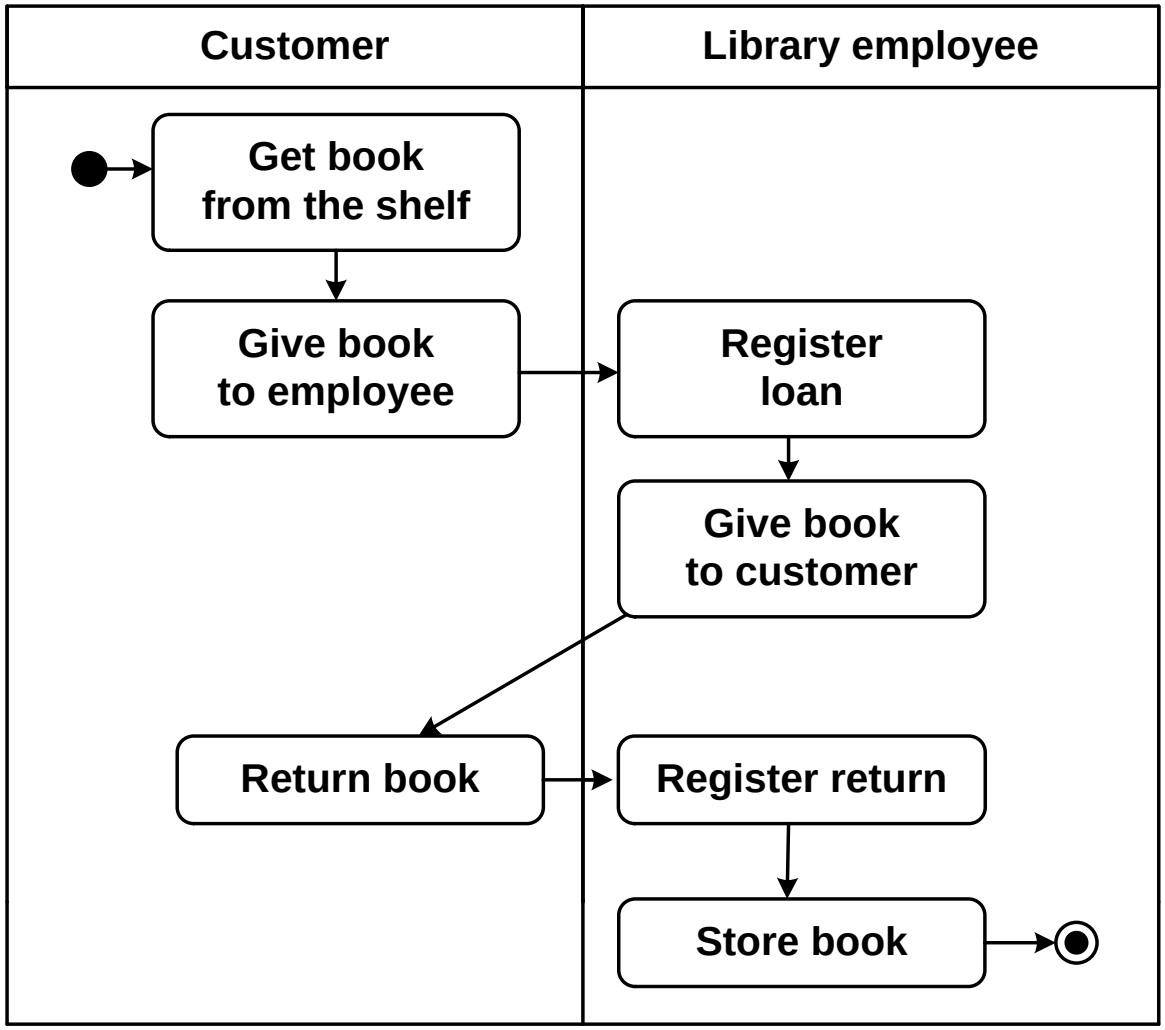
Not purely top down …



Today's topic



Example of an Activity Diagram



Purpose of an AD

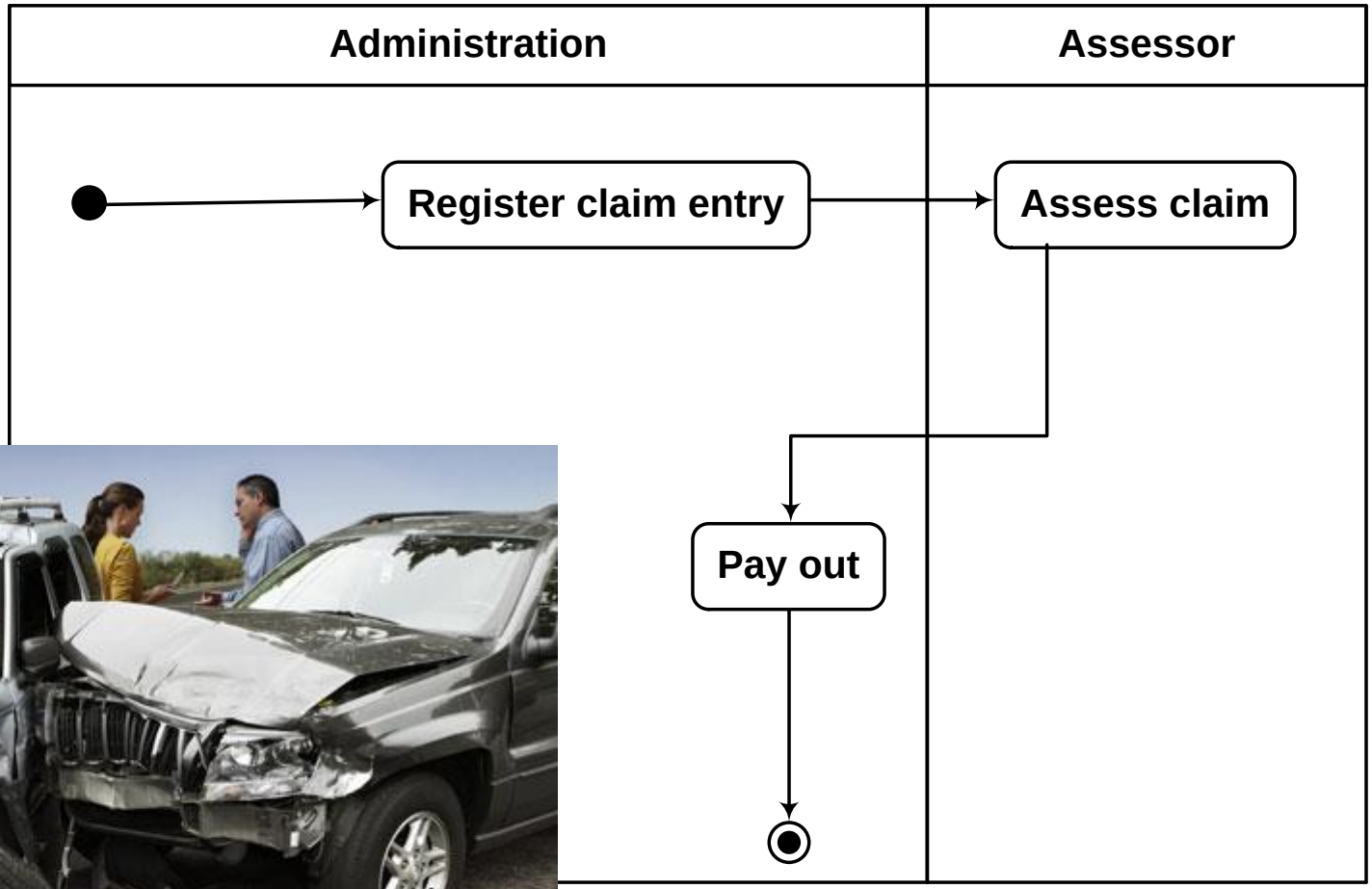
- An AD contains a coherent series of activities.
- In an Activity Diagram we include those activities that are relevant for the system to be designed
- Activities are carried out by actors. These can be persons, but also (parts of) computer systems.

What do we model with an AD

- ADs offer a general notation to describe steps in any organized process, like
  - Workflow in a company
  - Steps in an algorithm
- We use ADs to model processes in the real world, in order to derive functional requirements.
  - E.g. Business processes that need computer system support

Simple example

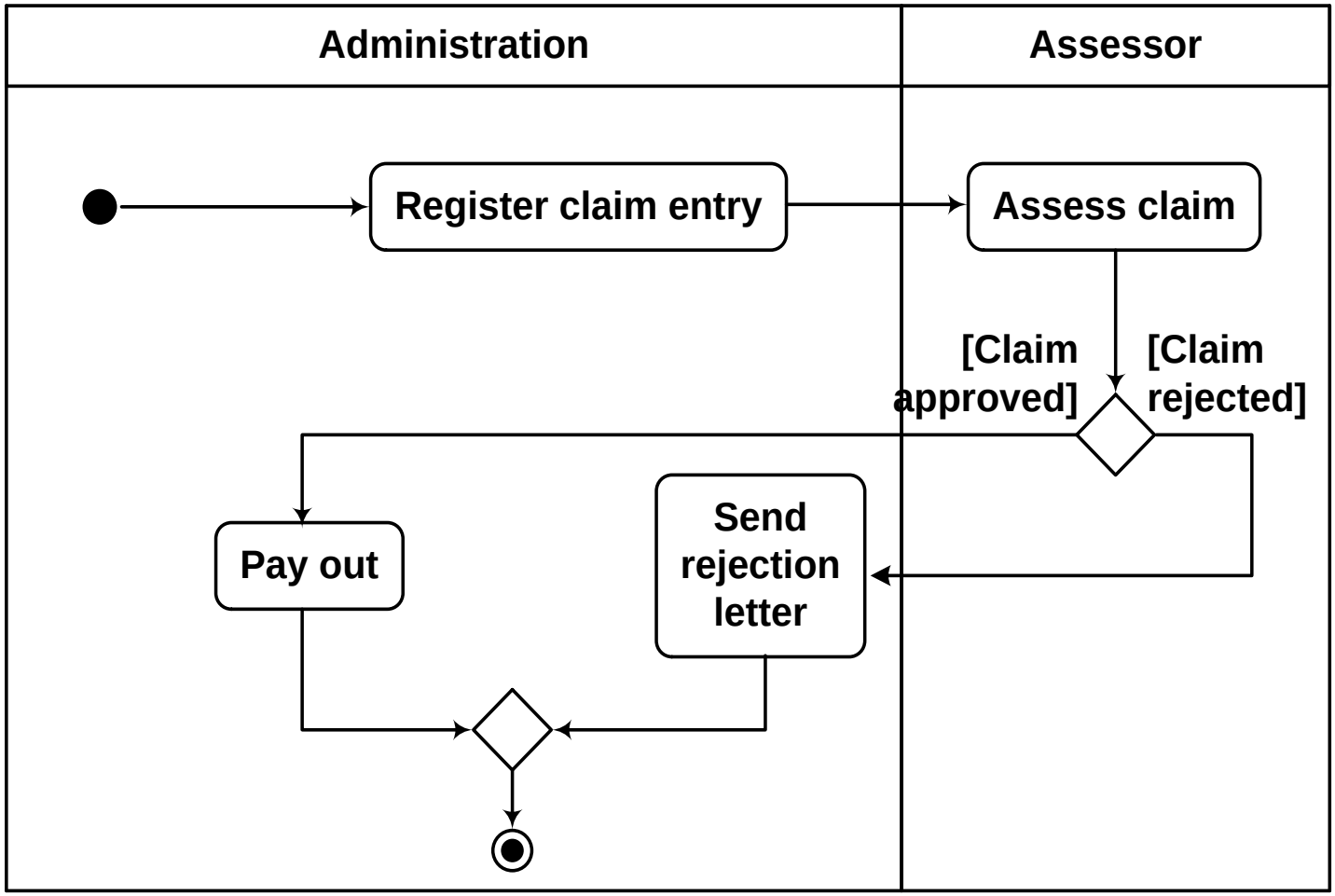
(Insurance company)



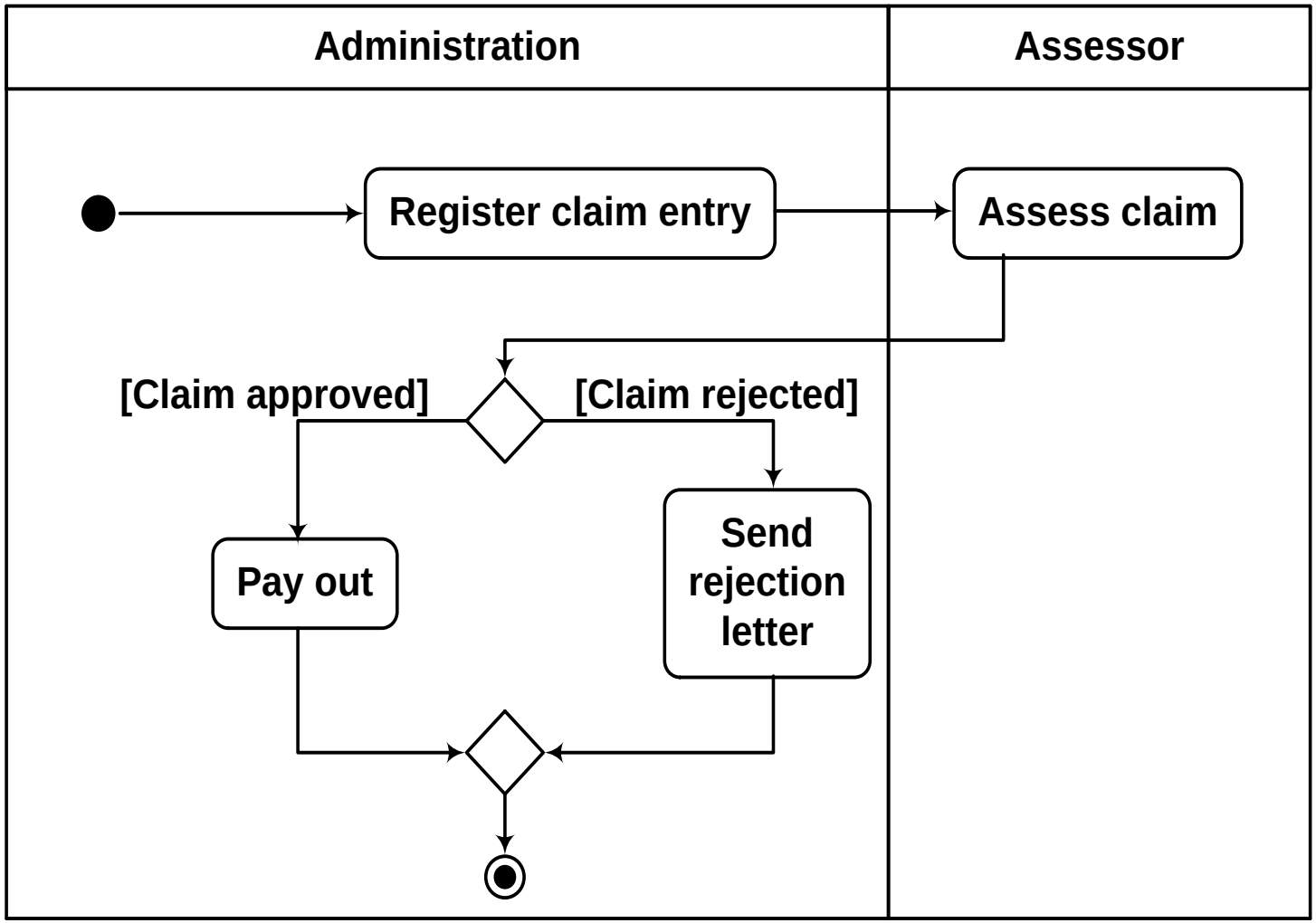
Elements of an AD (1)

- Activities
  (always formulated as imperative sentences: “Register...”)
- “Swimlanes” labelled with the name of the actor who carries out the activities
- Control flow (arrows) indicating the order in which the activities are carried out
- Start symbol and stop symbol

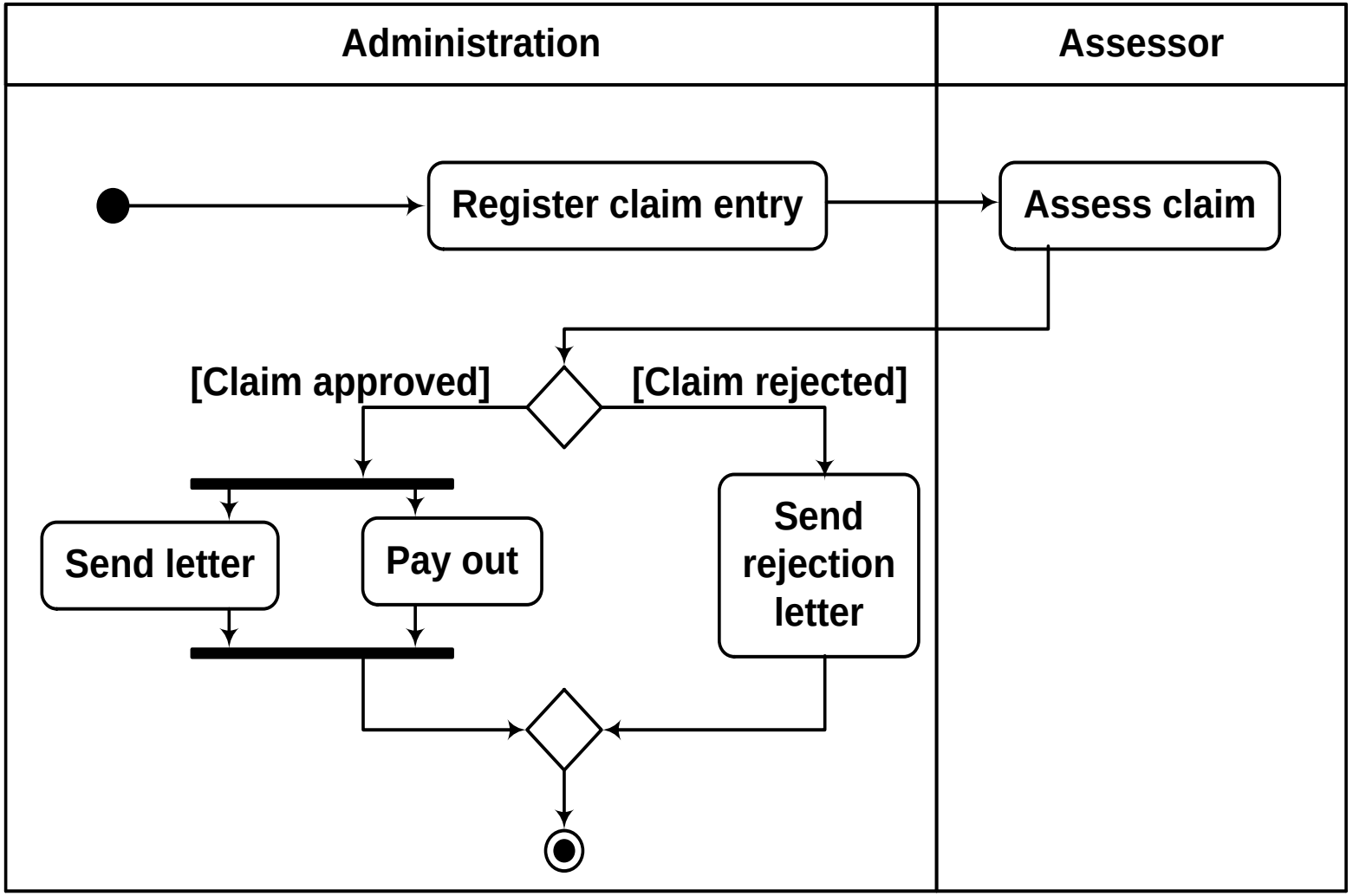
Branch and *Merge*



(essentially the same diagram)



Fork and *Rejoin*



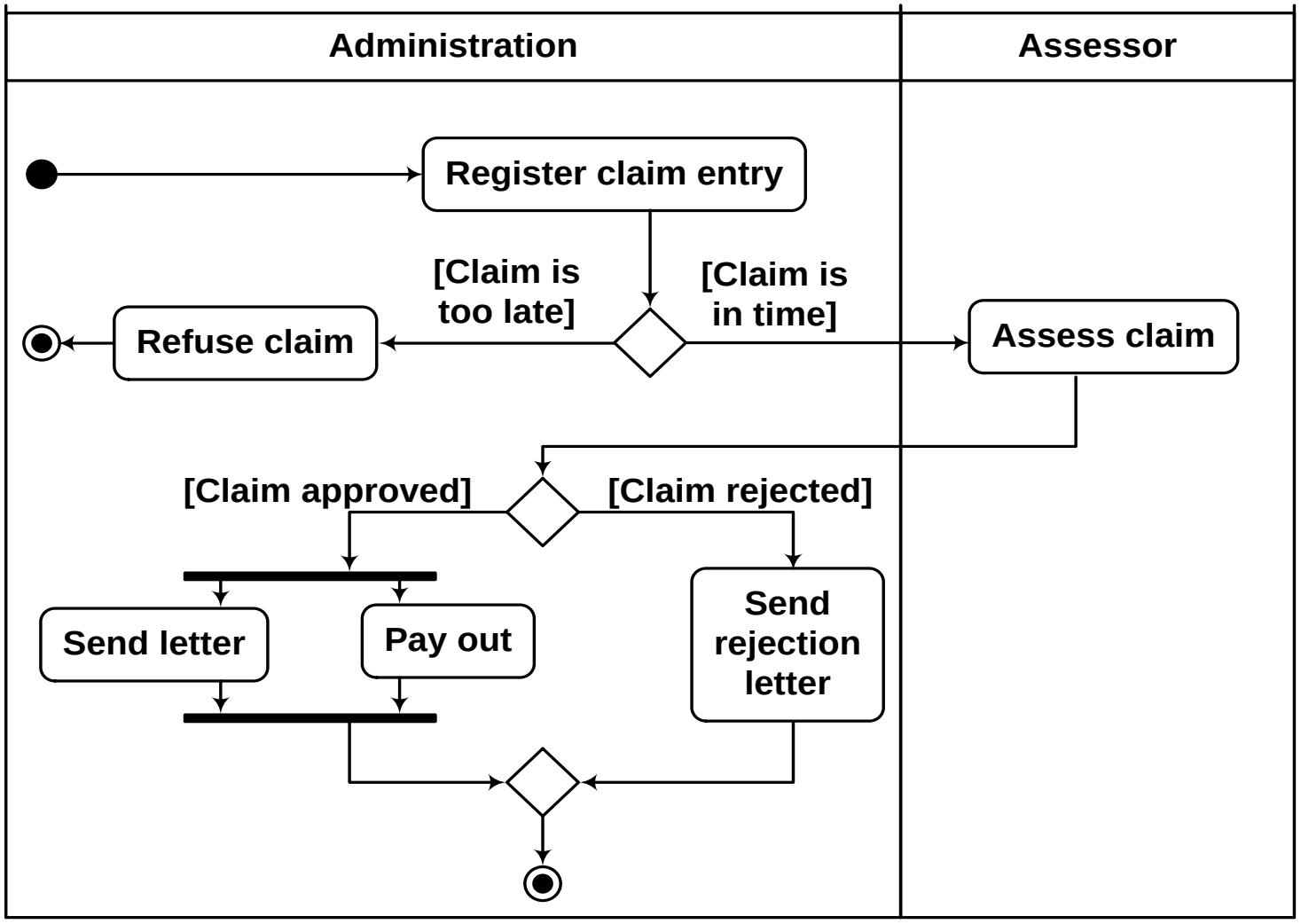
Elements of an AD (2)

- Branch: There are various paths to continue. ***One*** of these is executed. The conditions determine which one.
- The different paths of a branch come together in a merge.
- Fork: There are various paths to continue. ***Each*** of these is executed separately. The order in which activities in different paths are executed is not determined.
- The different paths of a fork come together in a rejoin.

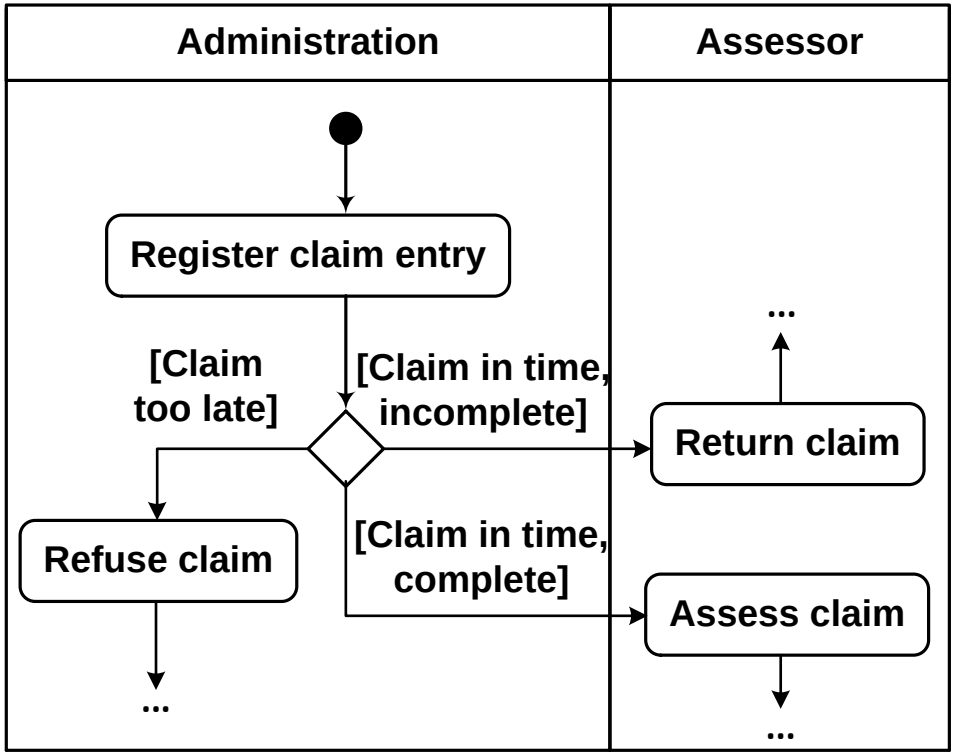
Elements of an AD (2, contd.)

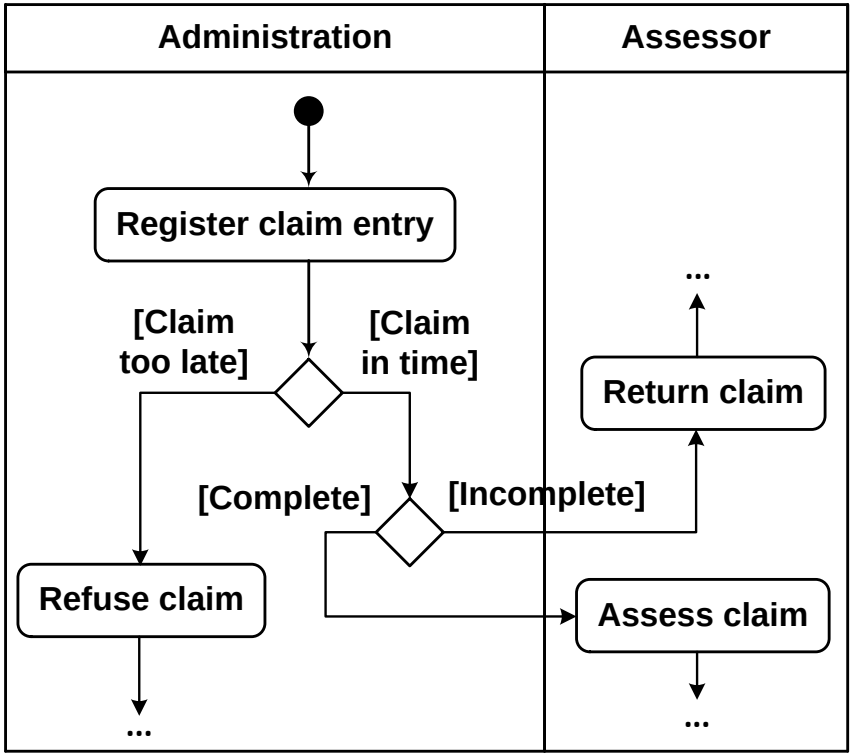
- In which swimlane you put a branch, (merge, fork, rejoin), does not matter.
- In which swimlane you put an activity does matter, as the swimlane defines the actor

Multiple end symbols...

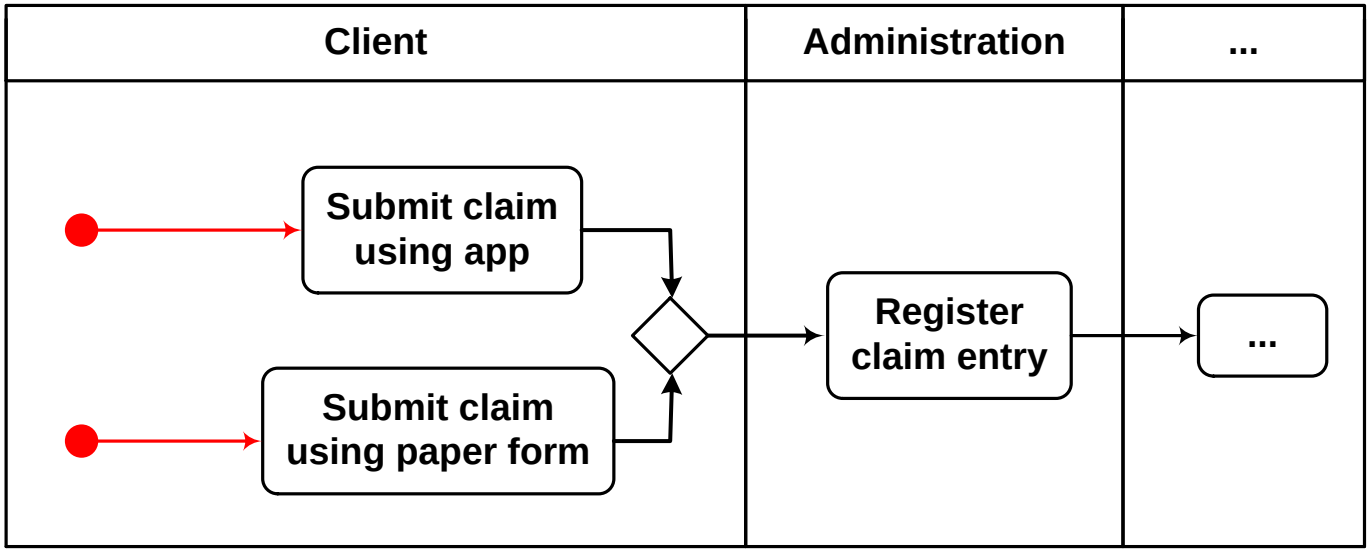


Branch into more than two paths…



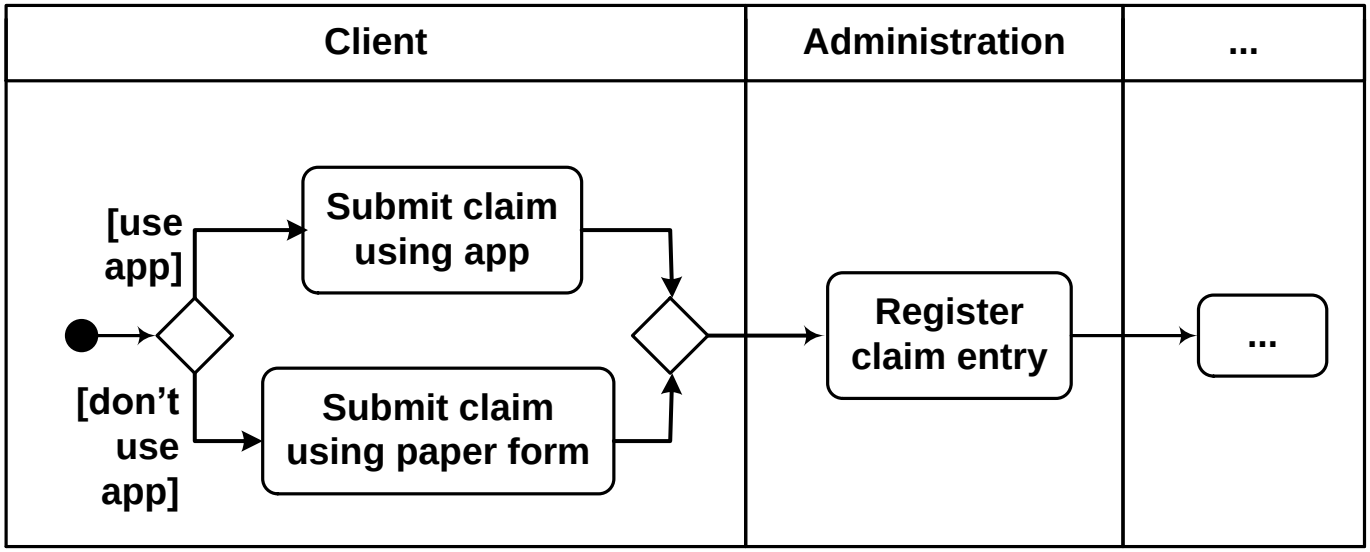


Single start symbol!

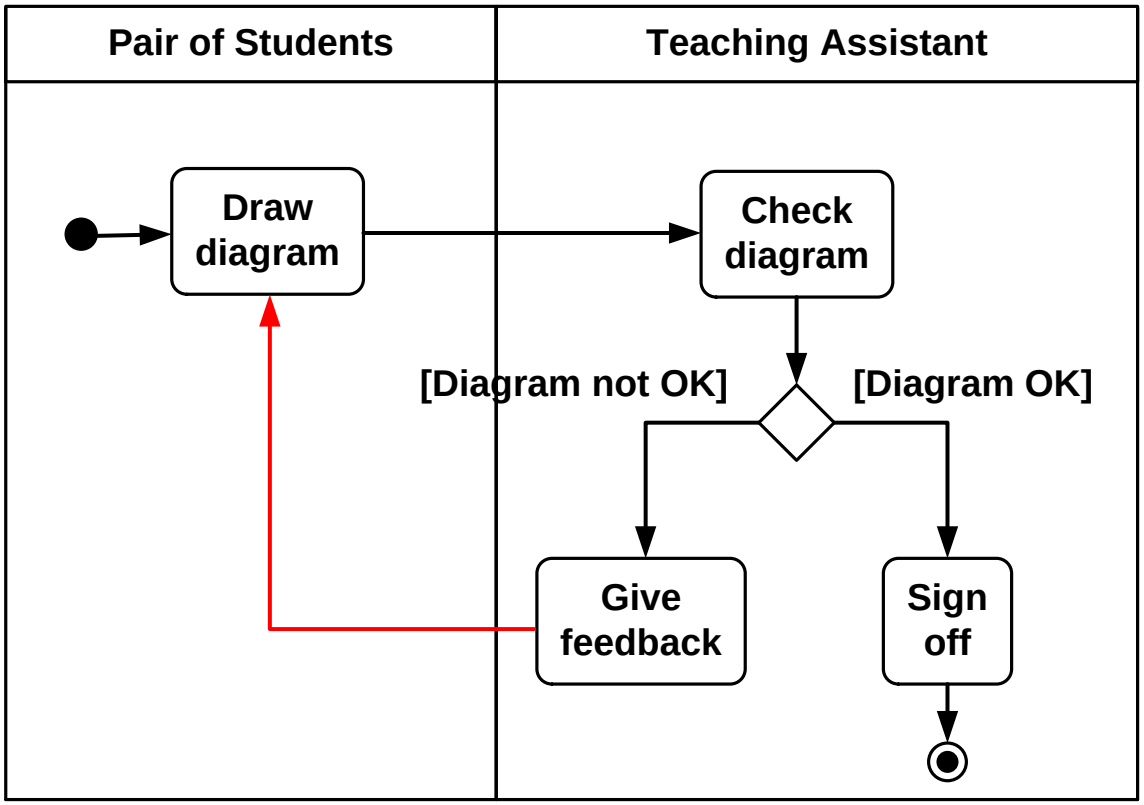


incorrect

Single start symbol!

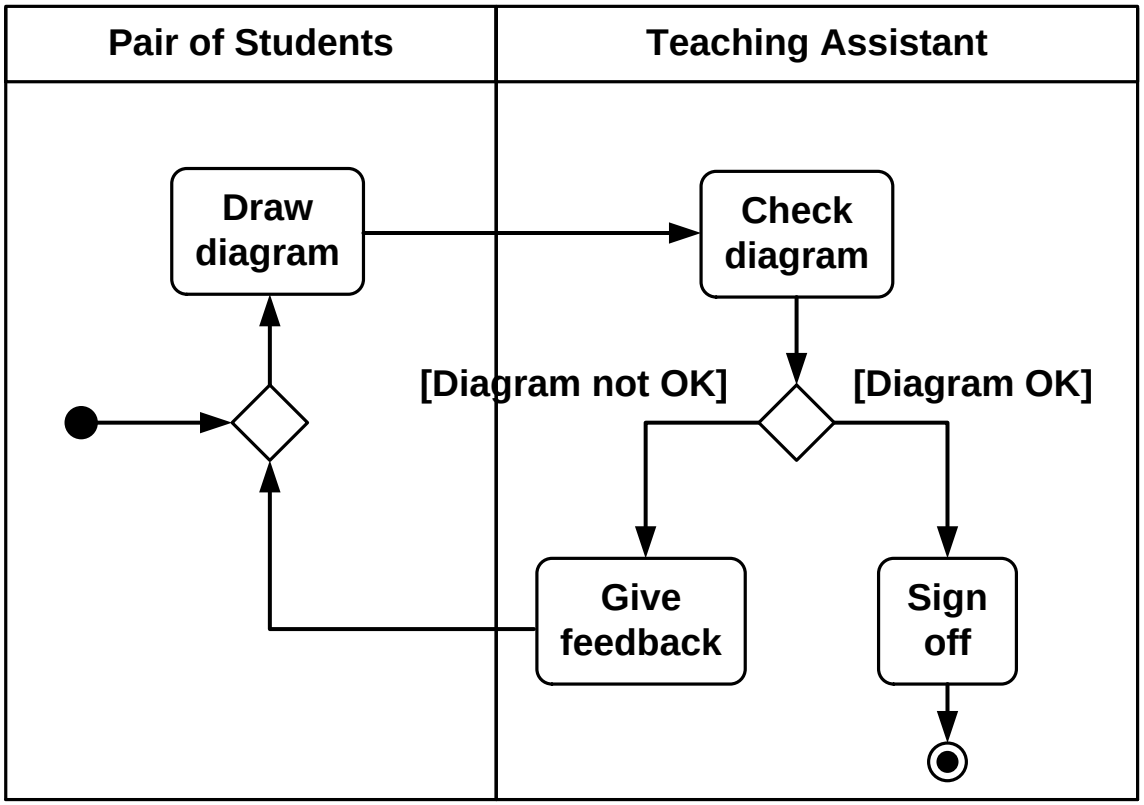


Loops



poorly modelled

Loops



Better model, separating activities and control flow

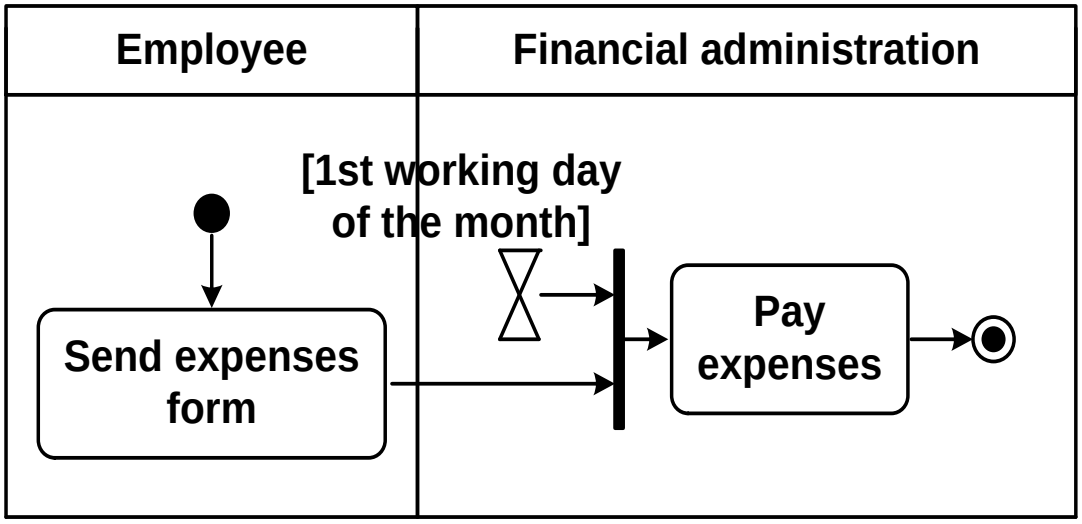
Elements of an AD (3)

- Sometimes it is more convenient not to merge the paths of a branch, but to use multiple end symbols
- However, there is always a single start symbol

- It is possible to introduce loops in an AD

- Some activities only can start on/after a specific moment in time. This is indicated with the clock symbol and a rejoin.
  - (see next slide)

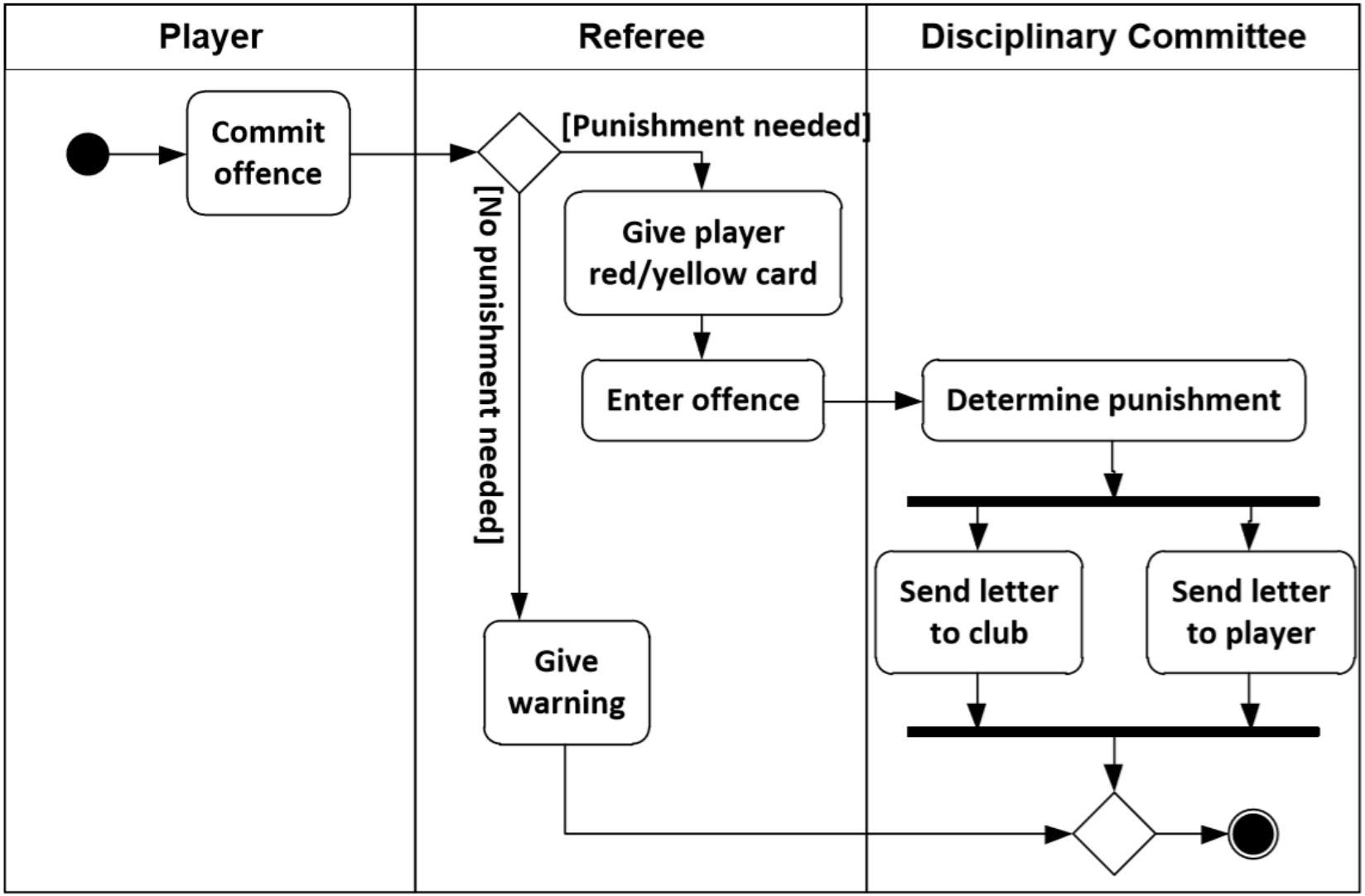
Using the clock symbol



Exercise: make an activity diagram

The Dutch Sports Association wants a system for registration of offences committed by players.

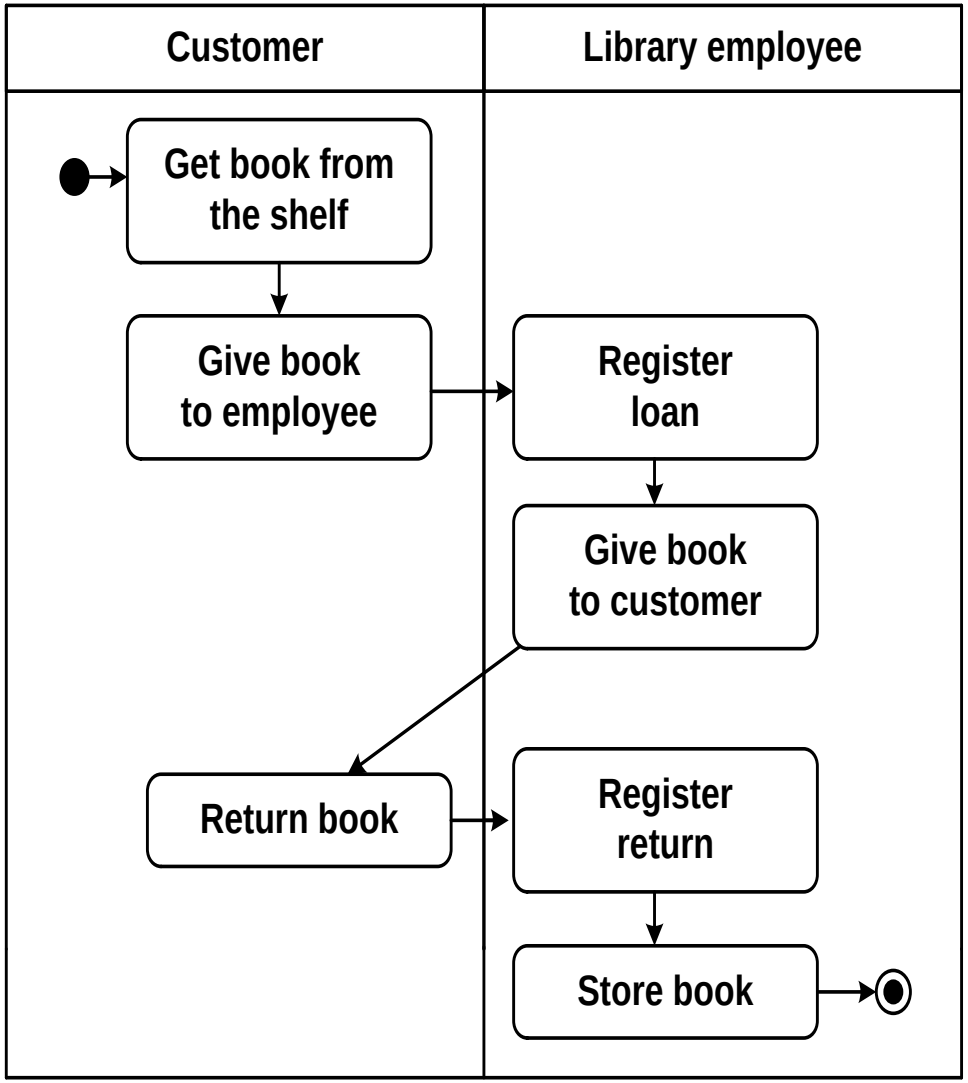
1. **When a player commits an offence during a match, the referee can give a warning or show a yellow / red card.**
2. **When a player gets a yellow / red card, after the match the referee enters the offence and the type of card into the system. Warnings are not registrered.**
3. **The Disciplinary Committee decides on a punishment.**
4. **The Disciplinary Committee informs both the player and the club by letter about the punishment.**


Player
Referee
Disciplinary Committee
Commit offence
[Punishment needed]
[No punishment needed]
Give player red/yellow card
Enter offence
Determine punishment
Give warning
Send letter to club
Send letter to player

An AD describes a single case

An AD is a (visual representation of a) model – hence a simplification of the real world

- For simplicity, we model a single case
  - example: borrowing a book (usually, people borrow several books)



An AD shows the normal flow of events

An AD is a (visual representation of a) model – hence a simplification of the real world.

- Exceptions to the normal flow are not included in the model.
  - Example: a library member looses a book. What then?

- ***What is normal, what is an exception?***

  If a special case is explicitly mentioned in the case description, it should be included in the model
  - Example: a reminder is sent when a book is not returned in time.

Different ADs for different processes

- Sometimes it makes sense to construct a model that consists of various separate ADs
  - Example: borrowing a book and membership renewal are different processes. So we make different ADs.
    (Even though they are combined in practice – in most cases, membership is renewed when an attempt to borrow a book fails)

Design exercises

- Laboratory exercises (in pairs) ***No answers given***
  - must be signed off
- Recommended exercises ***Answers given on Canvas***
  - Help you to prepare for the test
- Example tests (Canvas) ***Answers given on Canvas***
  - Help you to prepare for the test

  The test requires skills (not factual knowledge), which you can acquire only by practicing

UML Drawing tool

- Lab sessions: Visual Paradigm (UML drawing tool)
  - See docs.ia.utwente.nl > Visual Paradigm for how to use it

- Test: paper and pencil

About lab sessions

- Participation in lab sessions is expected
  - You get feedback
  - You want to get the assignments signed off

- If you have digested the theory *(i.e. actively followed the lecture or studied the slides)* it should be possible to complete all exercises in the lab session
  - Otherwise, finish it at home
  - Should be signed of one week after the session

Administration

- ***Who has no access to Software Systems on Canvas? Please contact me NOW !***

- Issues with group composition:
  - Remaining issues will be solved during the Project Kick-off, hour 4 (11:45-12:30)