PXL – Digital
421280 Software Analysis
Model based requirements documentation
SUC diagrams en descriptions

Week 09 – period 01

Luc Doumen

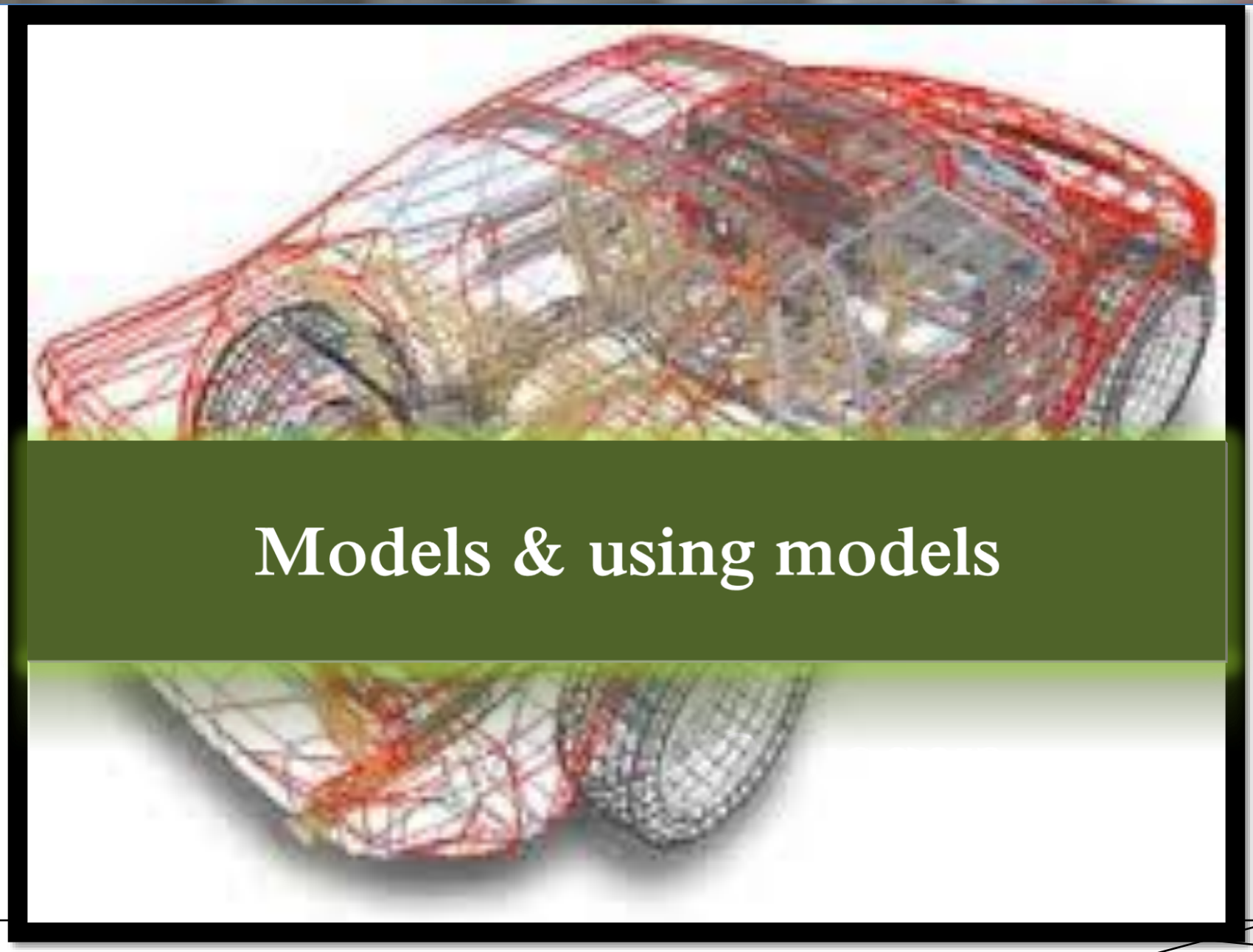
Nathalie Fuchs

DE HOGESCHOOL MET HET NETWERK

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be

Content

- Models & using models
- Topic overview
  - Goal models
  - Use models (system use cases)
  - 3 perspectives on requirements
    - Data/structural: ERM, Class models
    - Functional: Data Flow Diagram, Activity diagram
    - Behavioral: State charts
  - Sequence diagram
- Key learning points
- Questions & answers


Models & using models

Models

- A model is an abstraction of existing reality or a plan for reality to be created
- 3 important properties of models
  - Representation: map reality
  - Reduction: reduce the represented reality
  - Pragmatic: constructed for a special purpose
- In RE conceptual modeling languages are used defined by a given syntax and semantics
  - Cf. cheat sheets

Using models

- Information in pictures is quicker to understand and memorize
- Allow targeted modeling of one perspective
- Appropriate abstractions of reality can be specified by the defining the modeling language for the particular purpose

In practice: a combination of natural language and requirements models

Most frequently used models (1)

- **System Use Case Diagram (+ Description) ➔ !!!**
  - Represents a user's interaction with the system
- Entity Relationship Diagram ➔ 1TIN
  - Describes the data or information aspects of a business domain or its process requirements
- Class Diagram (OO) ➔ 1TIN
  - Describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects

Most frequently used models (2)

- Data Flow Diagram (cf. context diagram, 1TIN)
  - Shows the detailed functionality
- **Activity diagram ➔ !!!**
  - Shows the overall flow of control
- **State chart ➔ !!!**
  - Shows the behavior
- Sequence Diagram ➔ 3TIN Bachelor project + reader
  - Shows how processes operate with one another and in what order

Source: research by J. Hofmans, Improve Quality Services


Topic overview

Topic Overview

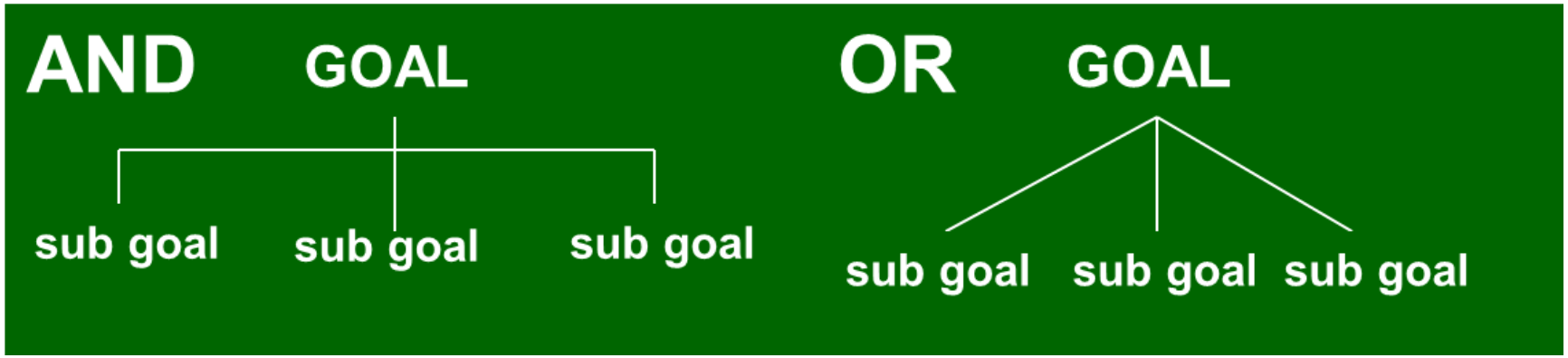
- Goal models
- System use case models (+ descriptions)
- Three perspectives on requirements
  - Data perspective
    - ERM, Class models (UML)
  - Functional perspective
    - Data flow diagrams, Activity diagrams (UML)
  - Behavioral perspective
    - State charts
- Sequence diagrams

Topic Overview

- **Goal models**
- System use case models (+ descriptions)
- Three perspectives on requirements
  - Data perspective
    - ERM, Class models (UML)
  - Functional perspective
    - Data flow diagrams, Activity diagrams (UML)
  - Behavioral perspective
    - State charts
- Sequence diagrams

Goal Models

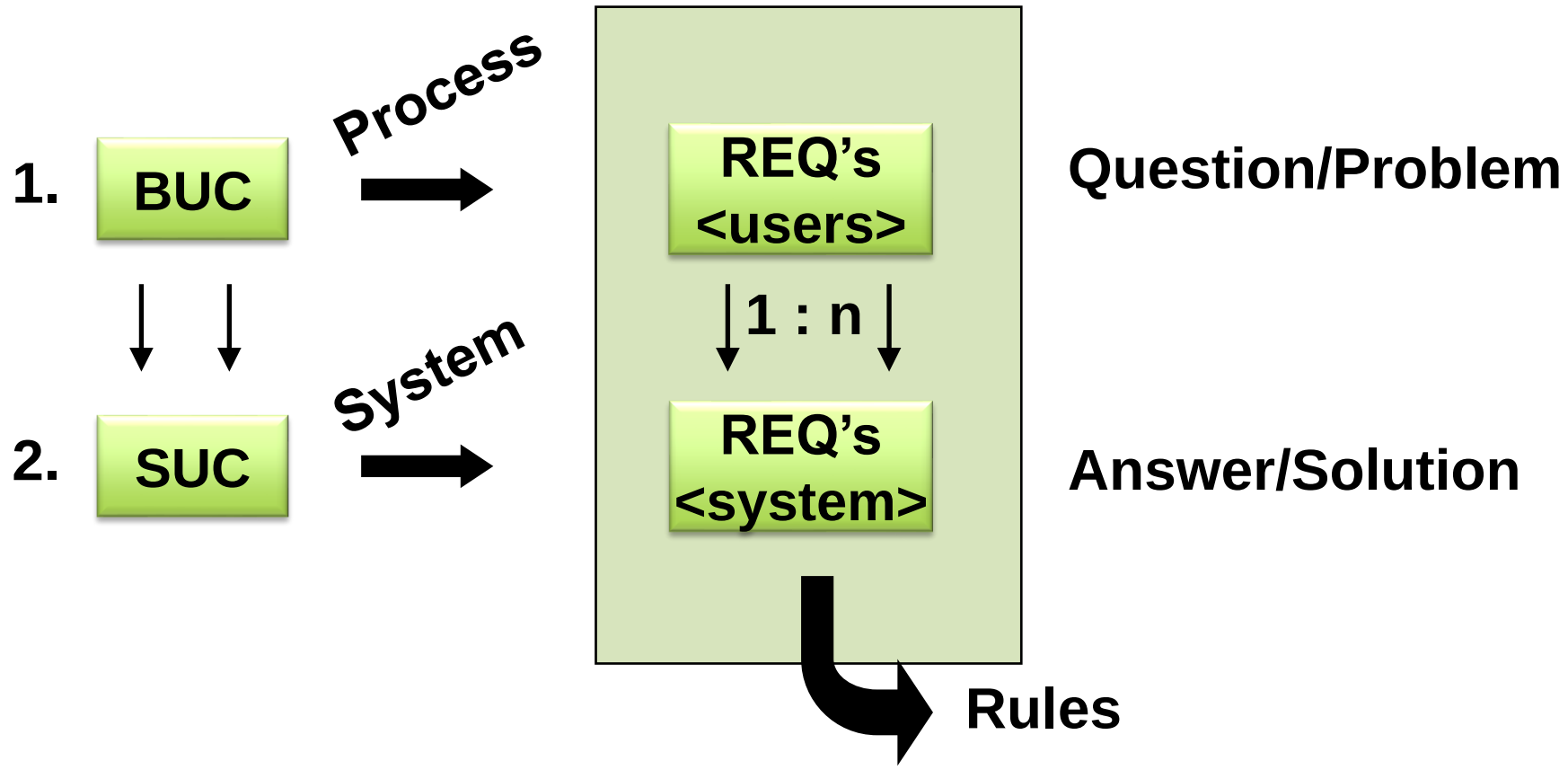
- A goal is a intentional description of one or more stakeholders about a wanted characteristic property of the system
- Often in natural language, but also in terms of models
  → and/or trees
- Decomposition in sub-goals:



Topic Overview

- Goal models
- **System use case models (+ descriptions)**
- Three perspectives on requirements
  - Data perspective
    - ERM, Class models (UML)
  - Functional perspective
    - Data flow diagrams, Activity diagrams (UML)
  - Behavioral perspective
    - State charts
- Sequence diagrams

Use Case levels



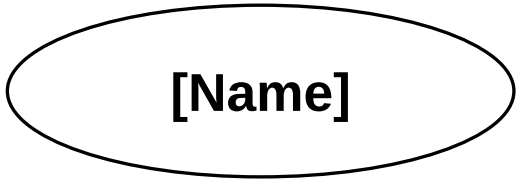
System Use Case Models + Description

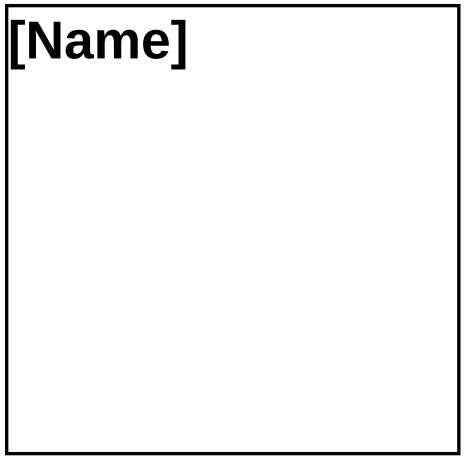
- Use cases help to examine and document the functionality from the **perspective of users** of the system
- Two concepts:
  - System use case diagrams
  - System use case specifications (= descriptions)

System Use Case Diagrams

- From an outside point of view
- Showing relationship with the context, and (some) relationships among the functionality
- Elements:
  - Use cases
  - Actors in the context
  - System boundary
  - Relationships between the elements

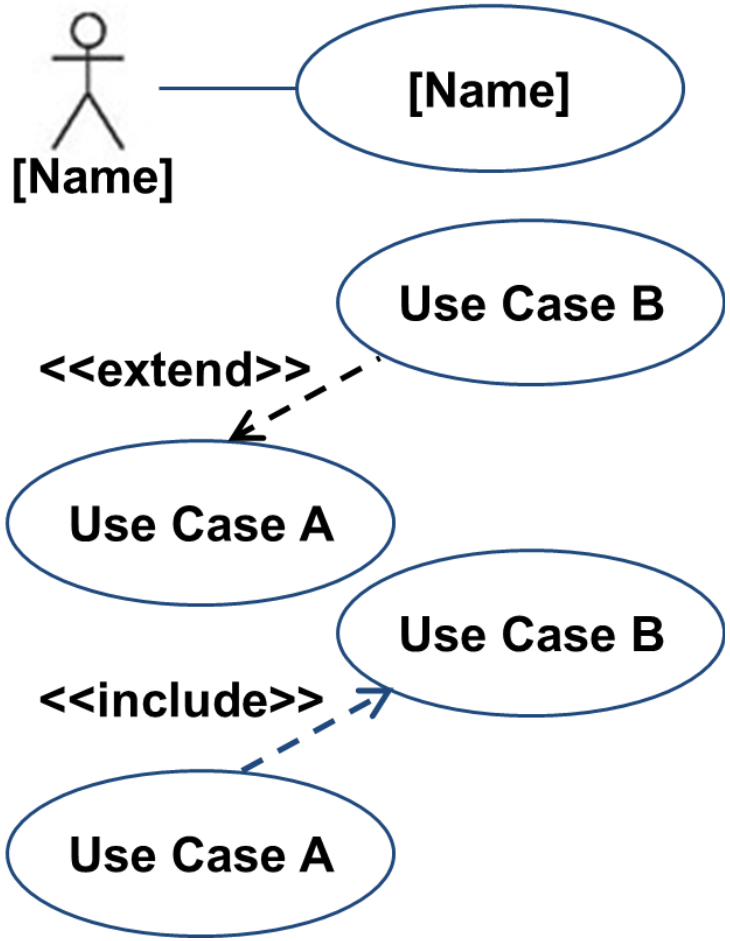
SUC: Elements (1)





- ***Use Case*** (verbs)
- ***Actor***: persons and/or other systems that interact with the system
- ***System Boundary***: determines what's inside (the use cases) and outside (persons and other systems)

SUC: Elements (2): Relations



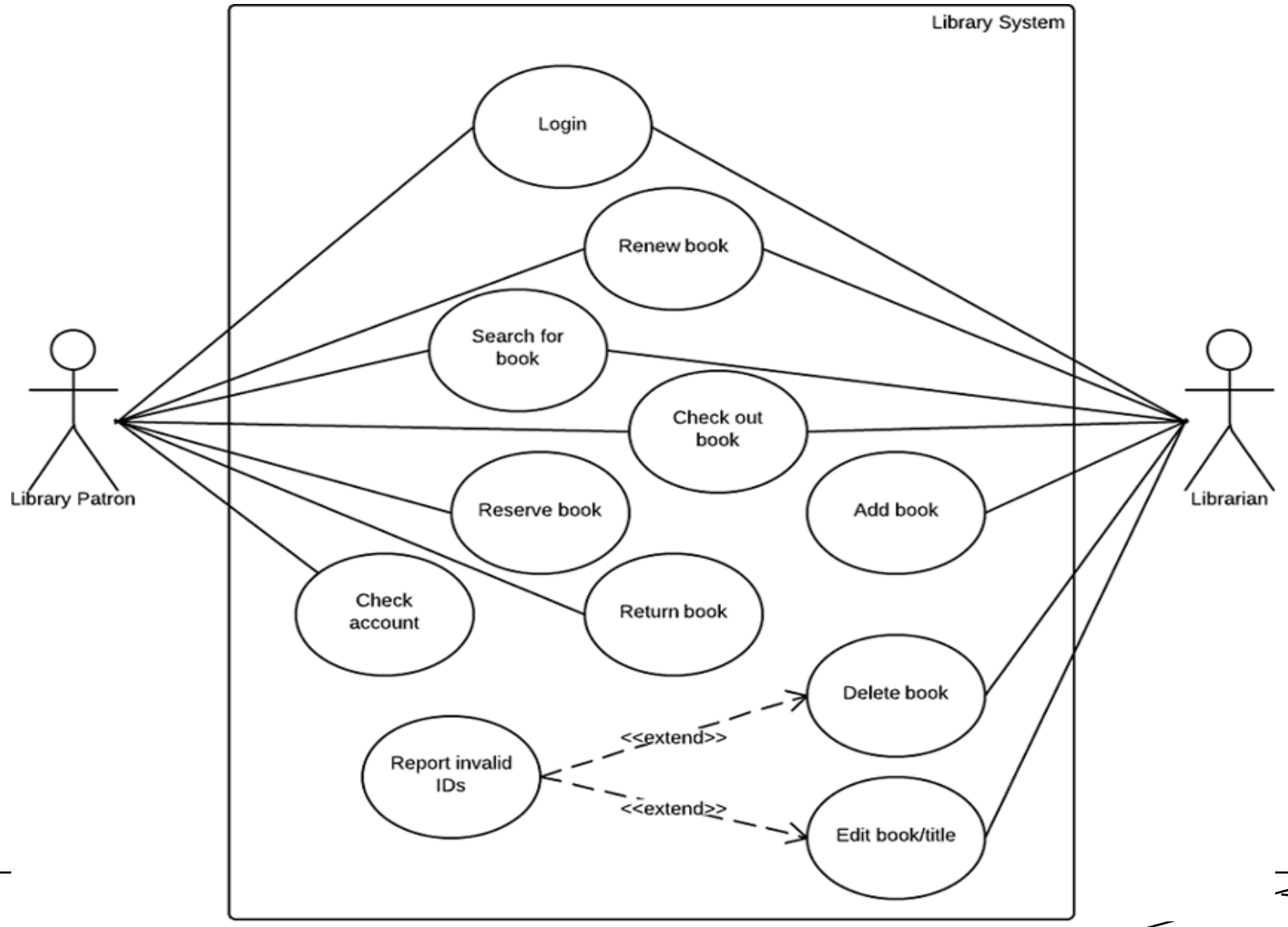
- ***Communication*** between actor & use case
- ***Extend***: indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions
- ***Include***: indicates that the use case to which the arrow points is included in the use case on the other side of the arrow

SUC: Elements (3): Relations

- An ***association*** is a relationship between classifiers

- A ***generalization*** relationship is a relationship in which one model element (the child) is based on another model element (the parent). The child receives all of the attributes, operations, and relationships that are defined in the parent.

SUC: Elements (4): Example



How to Find Use Cases (1)?

- Find actors first
- Identify boundaries of the system
- Define use cases for every actor
- Define for every use case
  - Preconditions
  - The interaction
  - The possible exceptions
- Describe every use case and draw the use case diagram

How to Find Use Cases (2)?

- Choose the **system boundary**
  - What you are building?
  - Who will be using the system?
  - What else will be used that you are not building?
- Find **primary actors** and their goals
  - Brainstorm the primary actors first
  - Who starts and stops the system?
  - Who gets notified when there are errors or failures?
- Define **use cases that satisfy user goals**
  - Prepare an actor-goal list (and not actor-task list)
  - In general, one use case for each user goal
  - Name the use case similar to the user goal

System Use Case Specifications (1)

- Think of **“what if scenarios”**
- One or more use cases per business event
  - Also consider **“misuse cases”**, e.g. for security requirement
- **Six step scenarios** are a great starting point
- You can add the requirements to each use case step used to discover missing requirements!
- Lots of templates available

System Use Case Specifications (2)

- Expanding the use case diagrams by more precisely describing the important characteristic properties in natural language
- A sequence of transactions in a dialogue between a user and the product with a specified result
- A scenario might be the generic “normal case” story, or it would tell a specific story

- TIP: NOT a screen flow diagram !!!

SUC description: elements & explanation

- Cf. WK09 - 01 - SWA1718 - Documentation of Requirements Model Based - 01 Template SUC DescriptionExplanation.docx

- WK09 - 01 - SWA1718 - Documentation of Requirements Model Based - 02 Template SUC DescriptionBlanc.docx

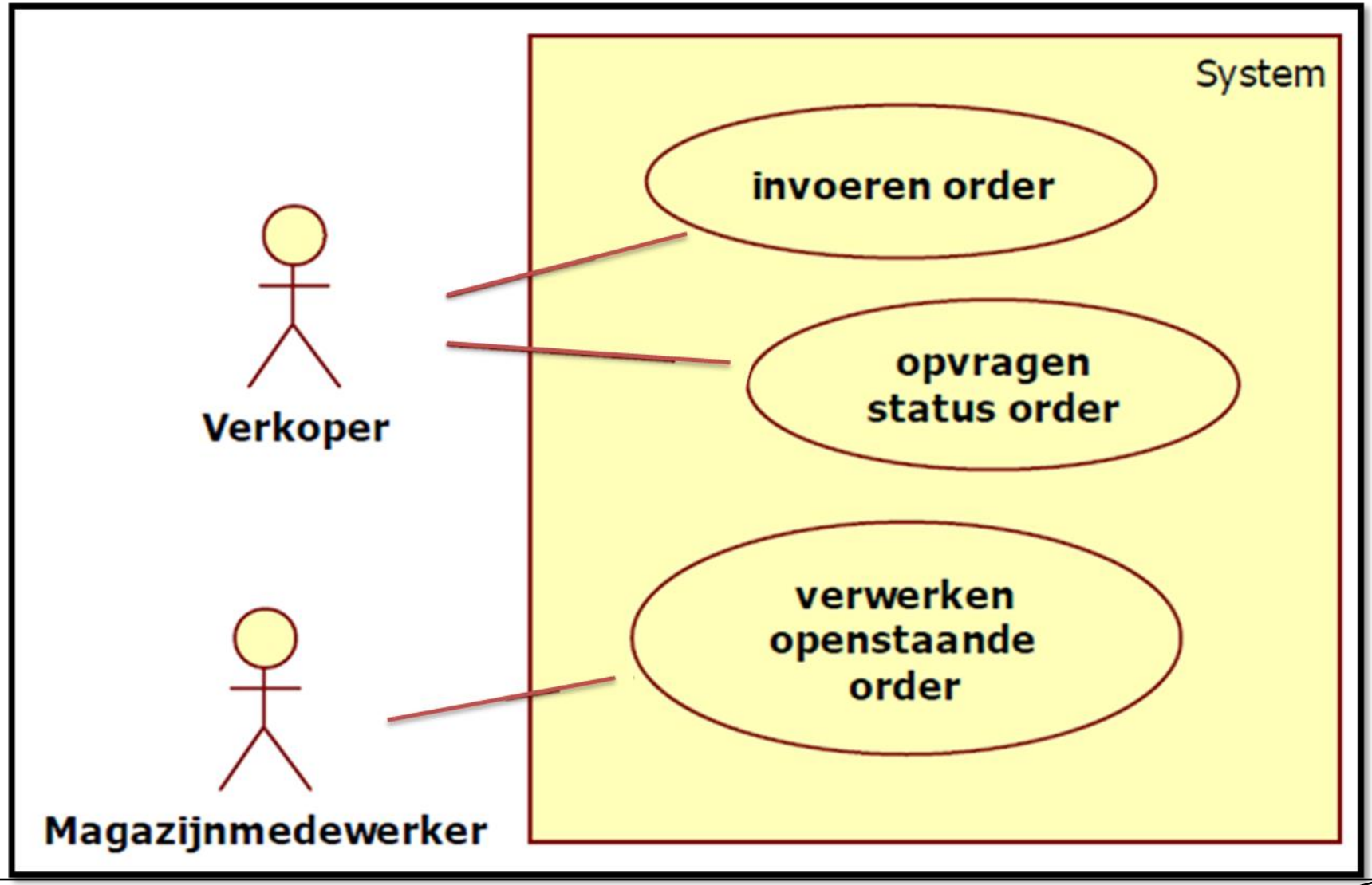
- WK09 - 01 - SWA1718 - Documentation of Requirements Model Based - 03 Example SUC diagram & description ATM.docx

SUC: exercise “Orders – bestellingen” (1)

- Een verkoper belt een klant met een aanbieding of de klant belt zelf voor een bestelling. De verkoper moet in het systeem de gewenste order kunnen vastleggen. In het magazijn moet een medewerker alle orders kunnen opvragen. Voor de goederen die op voorraad zijn maakt hij de bestelling in orde en laat een adressticker afdrukken.
  Soms belt een klant om te vragen naar de status van de order.

- Vraag:
  - Actoren?
  - Use cases?
  - Andere shapes niet vergeten!

SUC: exercise “Orders – bestellingen” (2)



SUC: exercise “Orders – bestellingen” (3)

- Invoeren order – opgave
  - De actor voert de achternaam van de klant in.
  - Het systeem toont alle klanten met die achternaam.
  - De actor kiest de juiste klant.
  - Het systeem laat alle details van de klant zien zodat de actor kan zien of hij de juiste klant voor zich heeft. De actor voert alle gewenste producten en de aantallen in en bevestigt het order.

Dit is de NORMAL FLOW van de SUC description

SUC: exercise “Orders – bestellingen” (4)

- Opvragen status – opgave
  - De actor voert de achternaam van de klant in.
  - Het systeem toont alle klanten met die achternaam.
  - De actor kiest de juiste klant. Het systeem laat alle details van de klant zien.
  - Het systeem toont een lijst met orders van de klant.
  - Voor ieder order wordt aangegeven wanneer het order verwerkt is en of alle producten in één keer geleverd konden worden.
  - De actor kan een order selecteren om te zien welke producten in welke hoeveelheden geleverd zijn.

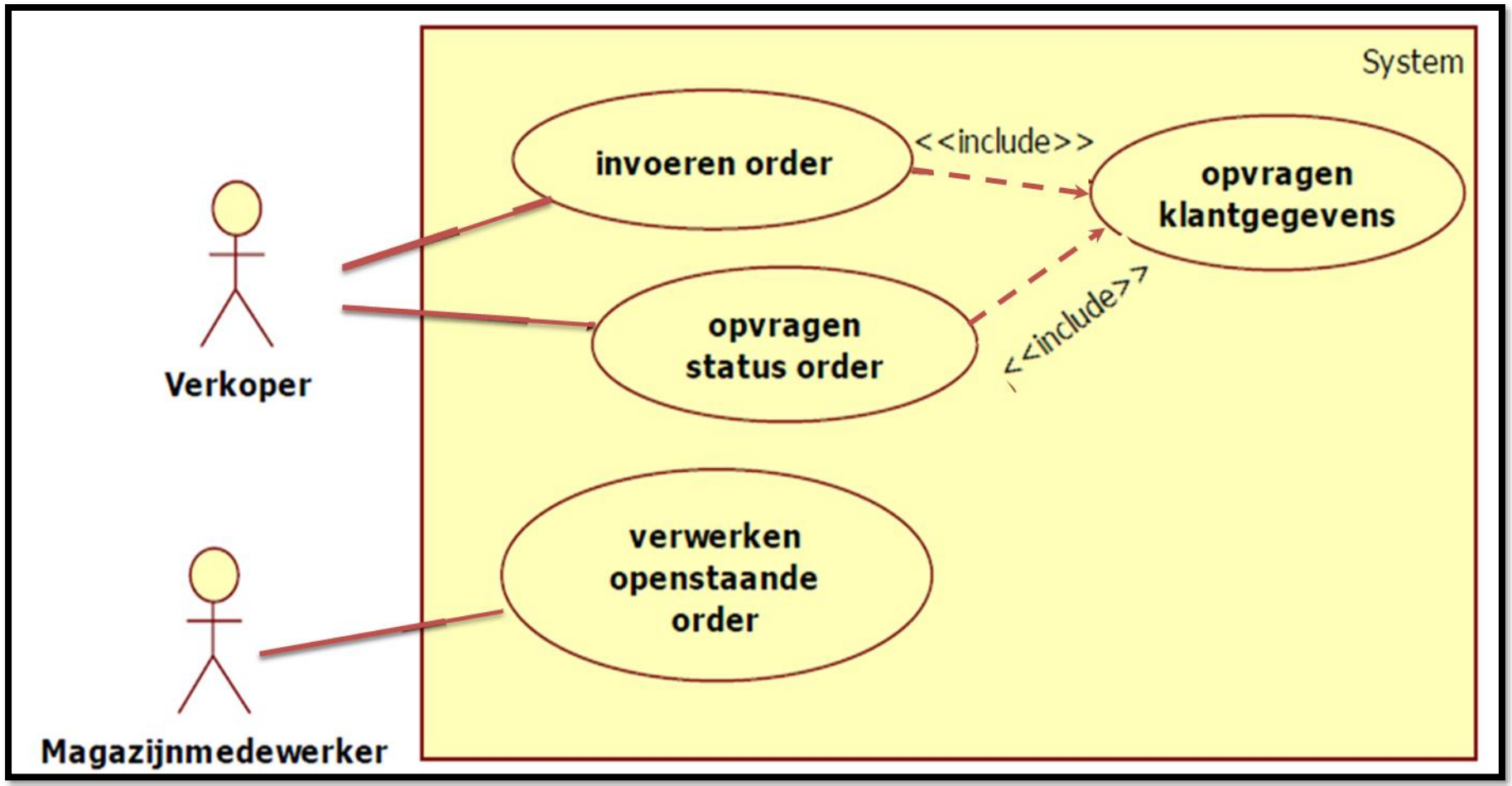
Dit is de NORMAL FLOW van de SUC description

SUC: exercise “Orders – bestellingen” (5)

- Verwerken openstaand order
  - De actor voert de achternaam van de klant in.
  - Het systeem toont alle klanten met die achternaam.
  - De actor kiest de juiste klant. Het systeem laat alle details van de klant zien.
  - Het systeem toont een lijst met orders van de klant.
  - Voor ieder order wordt aangegeven wanneer het order verwerkt is en of alle producten in één keer geleverd konden worden.
  - De actor kan een order selecteren om te zien welke producten in welke hoeveelheden geleverd zijn.

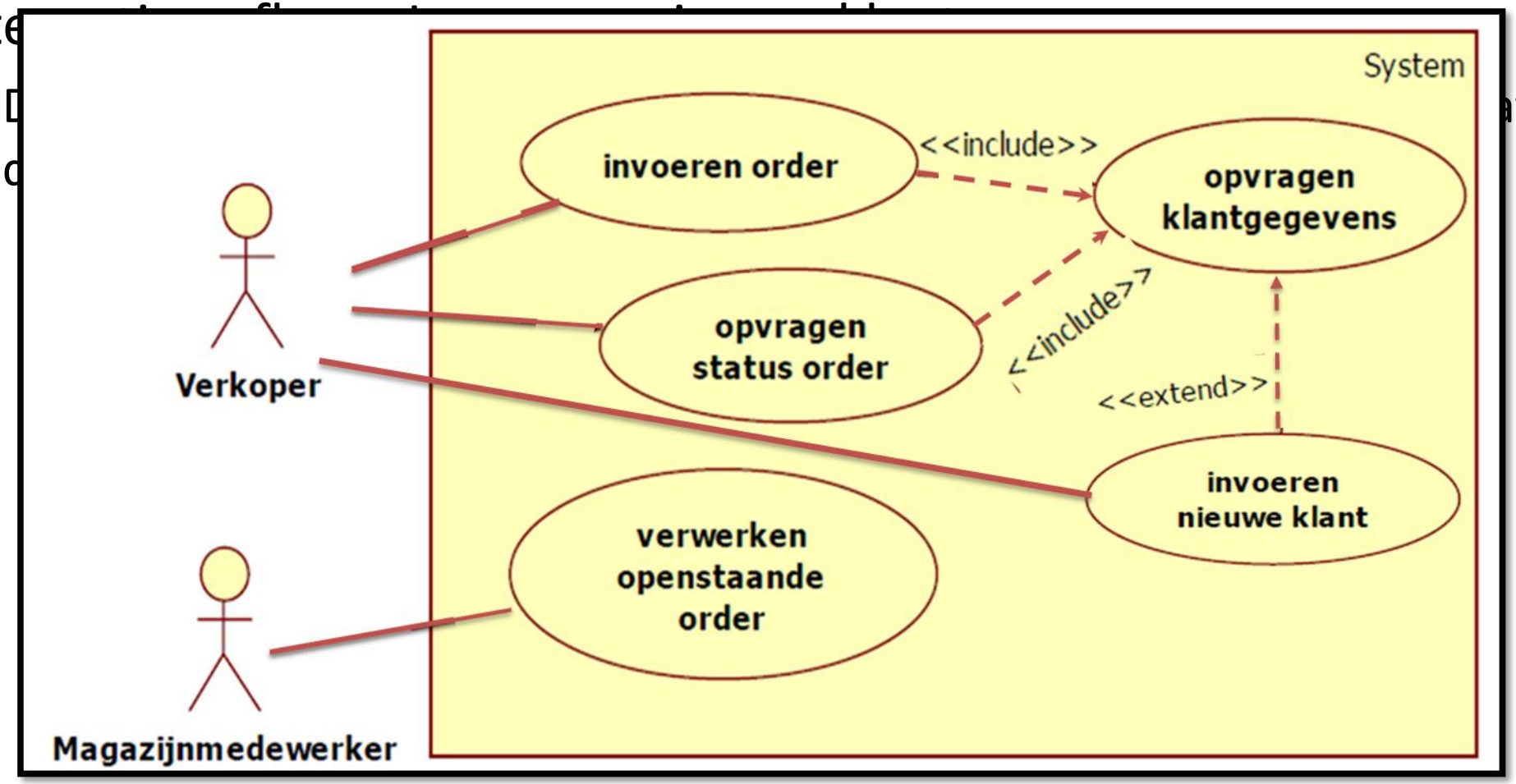
Dit is de NORMAL FLOW van de SUC description

SUC: exercise “Orders – bestellingen” (6)



SUC: exercise “Orders – bestellingen” (7)

- Alte
  - D atum van
    o

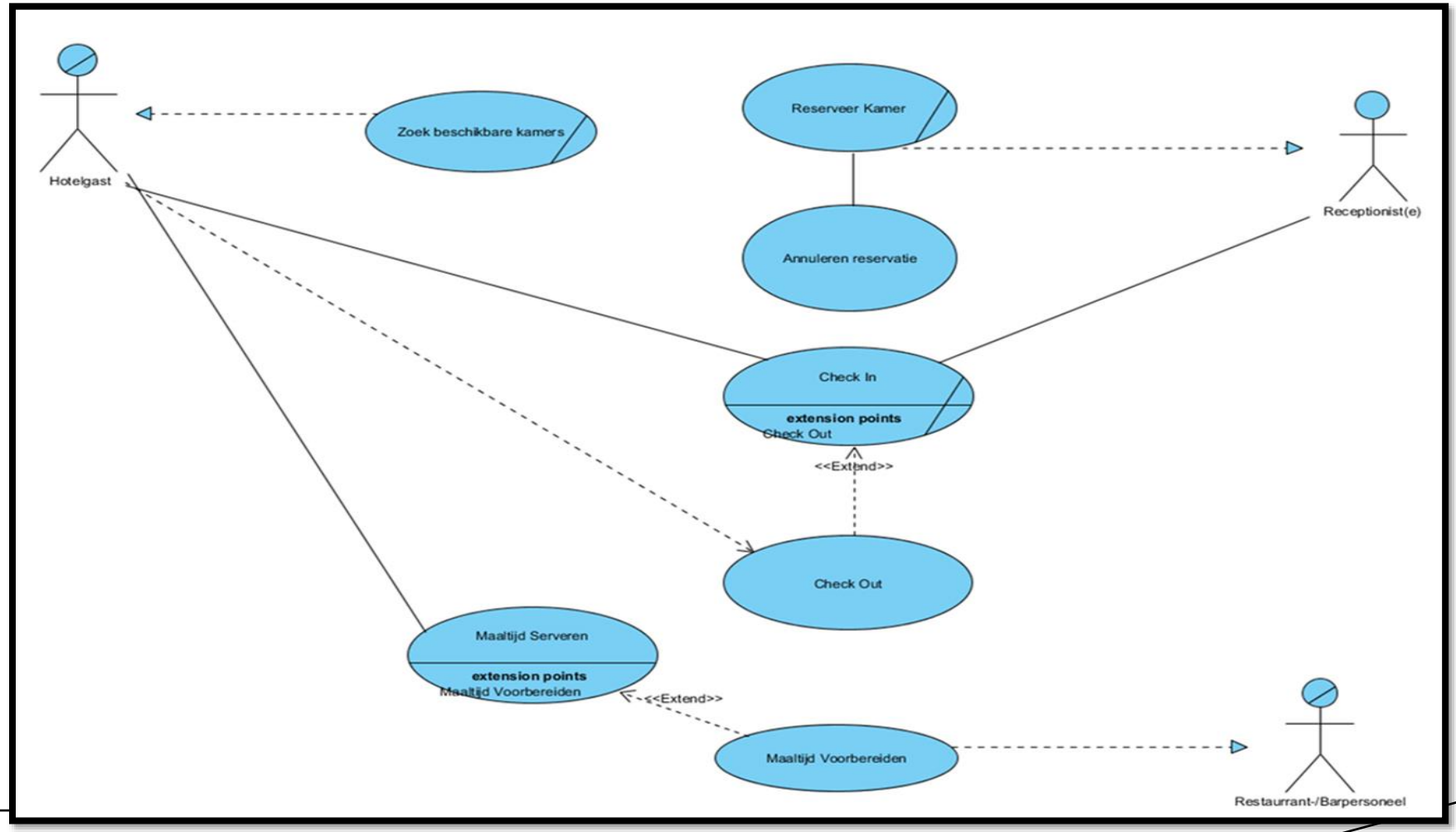


SUC: exercise “Coffee Machine” (8)

- Your customer is tired of his employees who keep coming late from coffee breaks because it takes them ages to prepare and drink their coffee and to clean their equipment afterwards.
- He asks you to design the prototype of a new revolutionary coffee machine that is both appealing to the eye and yet easy enough to handle even for an IT-specialist. During a brainstorming session on functional requirements, among the first use cases that are brought up there are things like prepare, serve, and cleanUp. While appearing simple at first glance, it soon turns out that the design must pay attention to a lot of details, so for each of the above-mentioned use cases there will probably be several included use cases like insertCoffee, or addMilk. Also bear in mind that the coffee machine needs to be refilled, maintained. There might be power brakes or problems with the water supply. Try to cover as many boundary conditions as possible to ensure the office keeps running.
- Do not spend to much time on the individual use cases. The emphasis of this task lies on the syntax and semantics of the UML use case diagram. Your diagram should contain examples of <<includes>> and <<extends>> relations. An example of generalization would be nice.

SUC: exercise (exam) “Room Booking” (9)

- Redesign ➔ SUC diagram



SUC: exercise (exam) “Answering machine” (10)

- Gegeven is het volgende antwoordapparaat. Als de telefoon na vier keer overgaan nog niet is opgenomen zal dit apparaat de telefoon automatisch beantwoorden.

- Het welkombericht is als volgt:

  “Hallo, met Monique. Ik ben er momenteel even niet, maar spreek een boodschap in na de biep.”
- Welke 2 actoren kan je onderscheiden in dit systeem?
- Teken het SUC diagram
- Werk de SUC description uit voor het “Opnemen van een welkombericht”

SUC: exercise (exam) “Vending machine” (11)

- A vending machine sells small, packaged, ready to eat items (chocolate bars, cookies, candies, etc.).
- Each item has a price and a name. A customer can buy an item, using a smart card (issued by the vending machine company) to pay for it. No other payment forms (i.e. cash, credit card) are allowed. The smart card records on it the amount of money available.
- The functions supported by the system are:
  - Sell an item (choose from a list of items, pay item, distribute item)
  - Recharge the machine
  - Set up the machine (define items sold and price of items)
  - Monitor the machine (number of items sold, number of items sold per type, total revenue)

Questions & answers