# PXL – IT
# 42TIN1280 Software Analysis
# Model based Requirements Documentation

**Nathalie Fuchs**

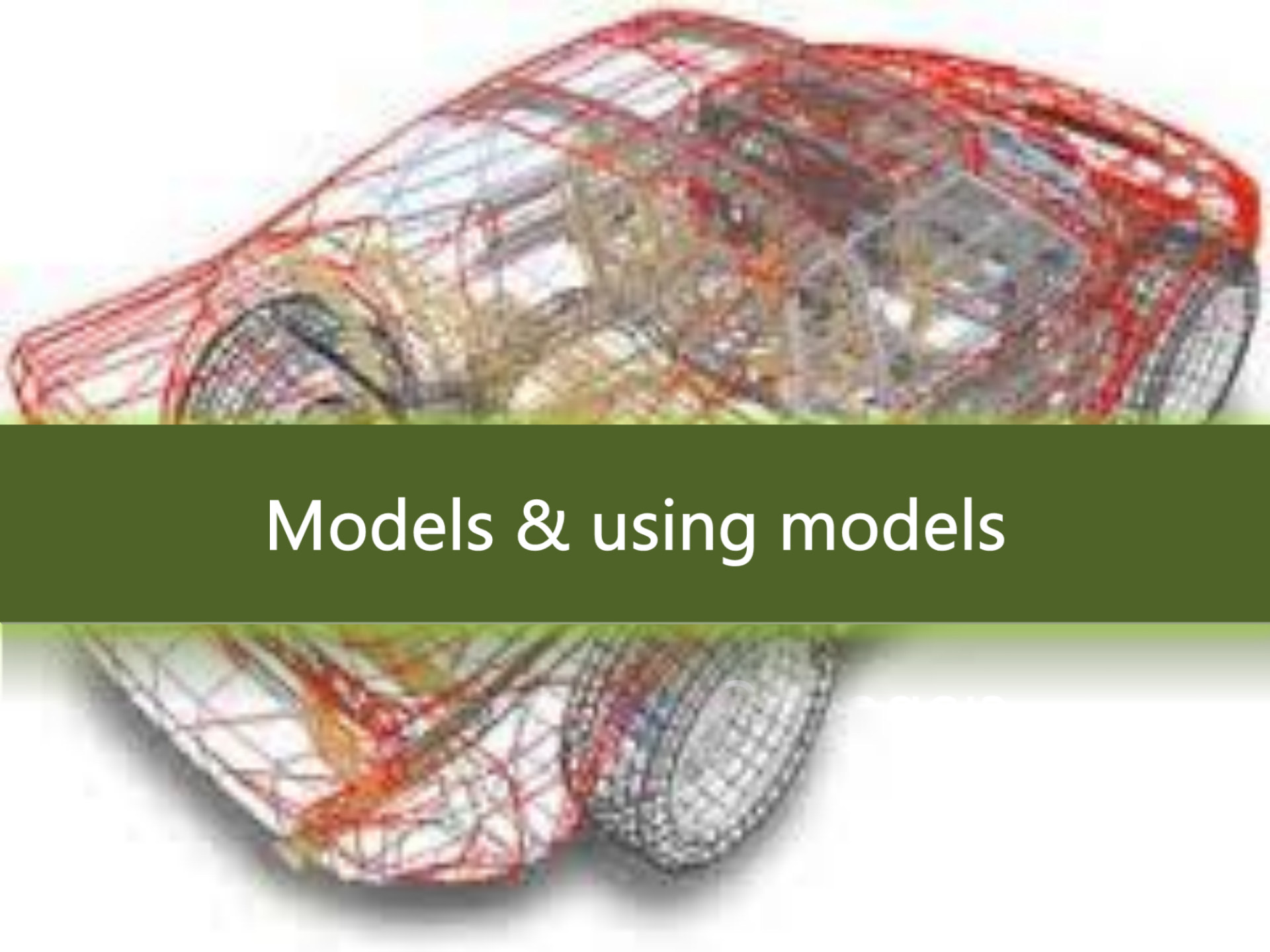
**Luc Doumen**

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

# 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)

- **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
  - Shows how processes operate with one another and in what order

Source: research by J. Hofmans, Improve Quality Services

# Topic overview

# Topic Overview

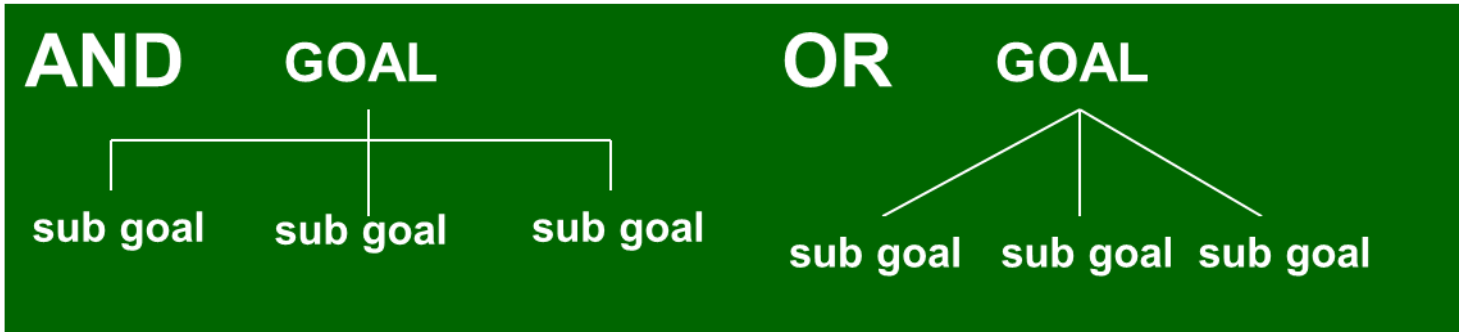
- Goal models
- System use case models (+ descriptions)
- Three perspectives on requirements
  - Data perspective
    - ERM, Class models (UML)
  - Functional perspective
    - Data flow diagrams, Activity diagrams (UML)
  - Behavioral perspective
    - State charts
- Sequence diagrams

# 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

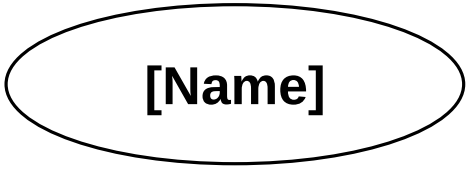
# System Use Case Models + Description

- 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)

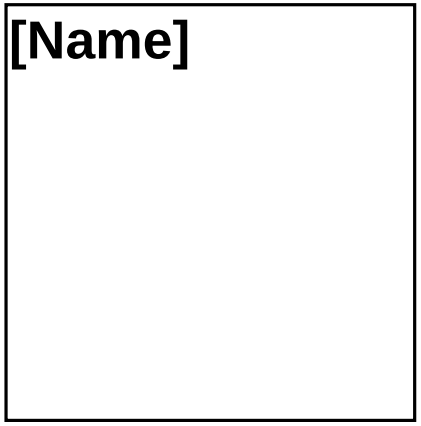
# System Use Case Diagrams

- 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)

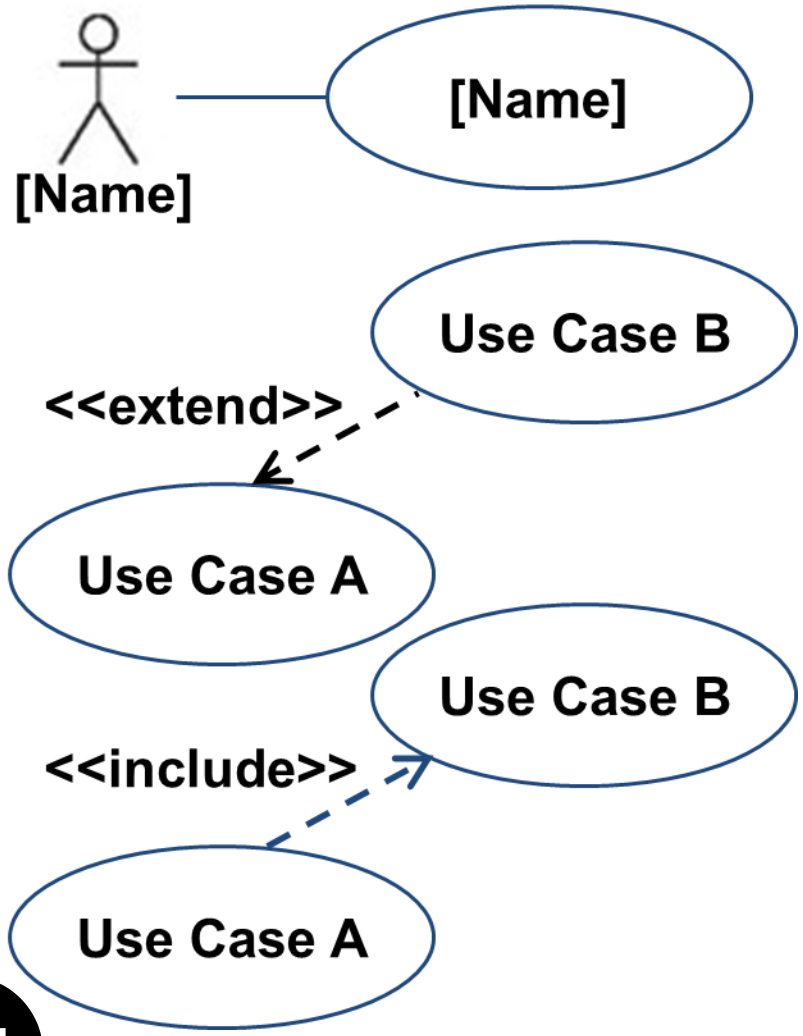







- ***Use Case*** (verbs)
- ***Actor***: persons and/or other systems that interact with the system
- ***System Boundary***: determines what's inside (the use cases) and outside (persons and other systems)

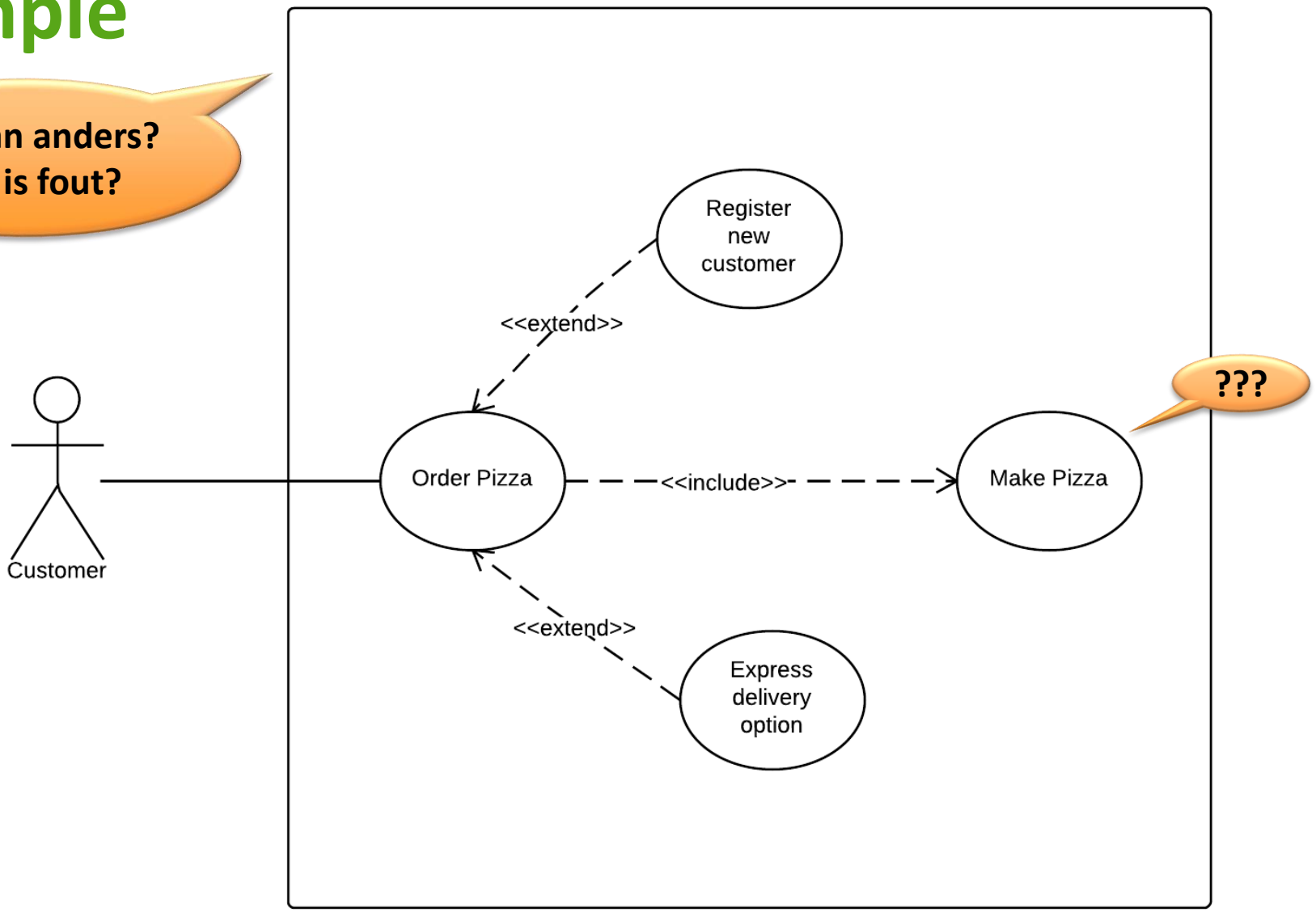
# SUC: Elements (2): Relations



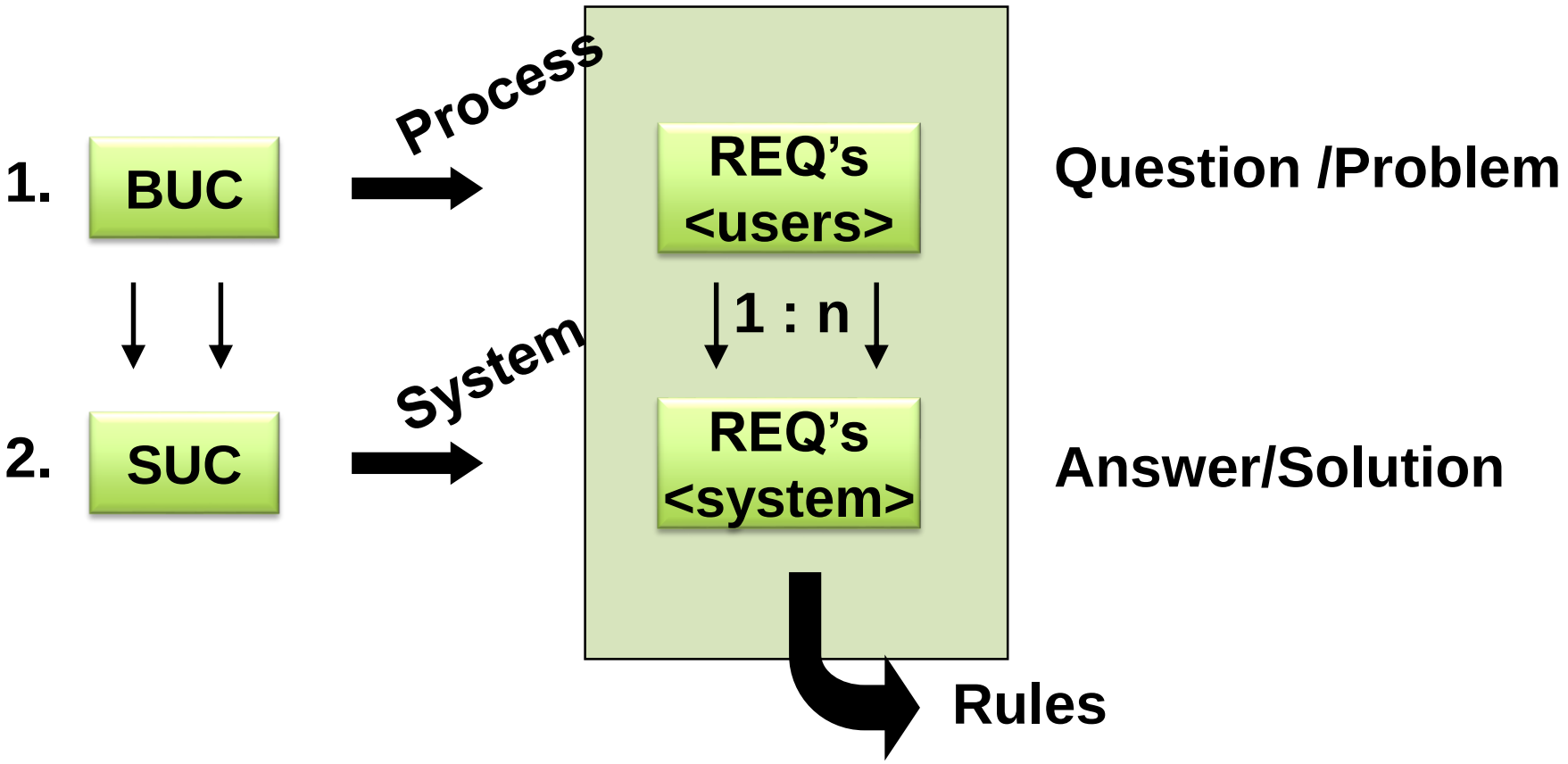

- ***Communication*** between actor & use case
- ***Extend***: indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions
- ***Include***: indicates that the use case to which the arrow points is included in the use case on the other side of the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

# Example

**Wat kan anders?**
**Wat is fout?**



# Use Case levels



# 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 req.
- Six step scenario’s are a great starting point
- You can add the requirements to each use case step. Used to discover missing requirements!
- Lots of templates available

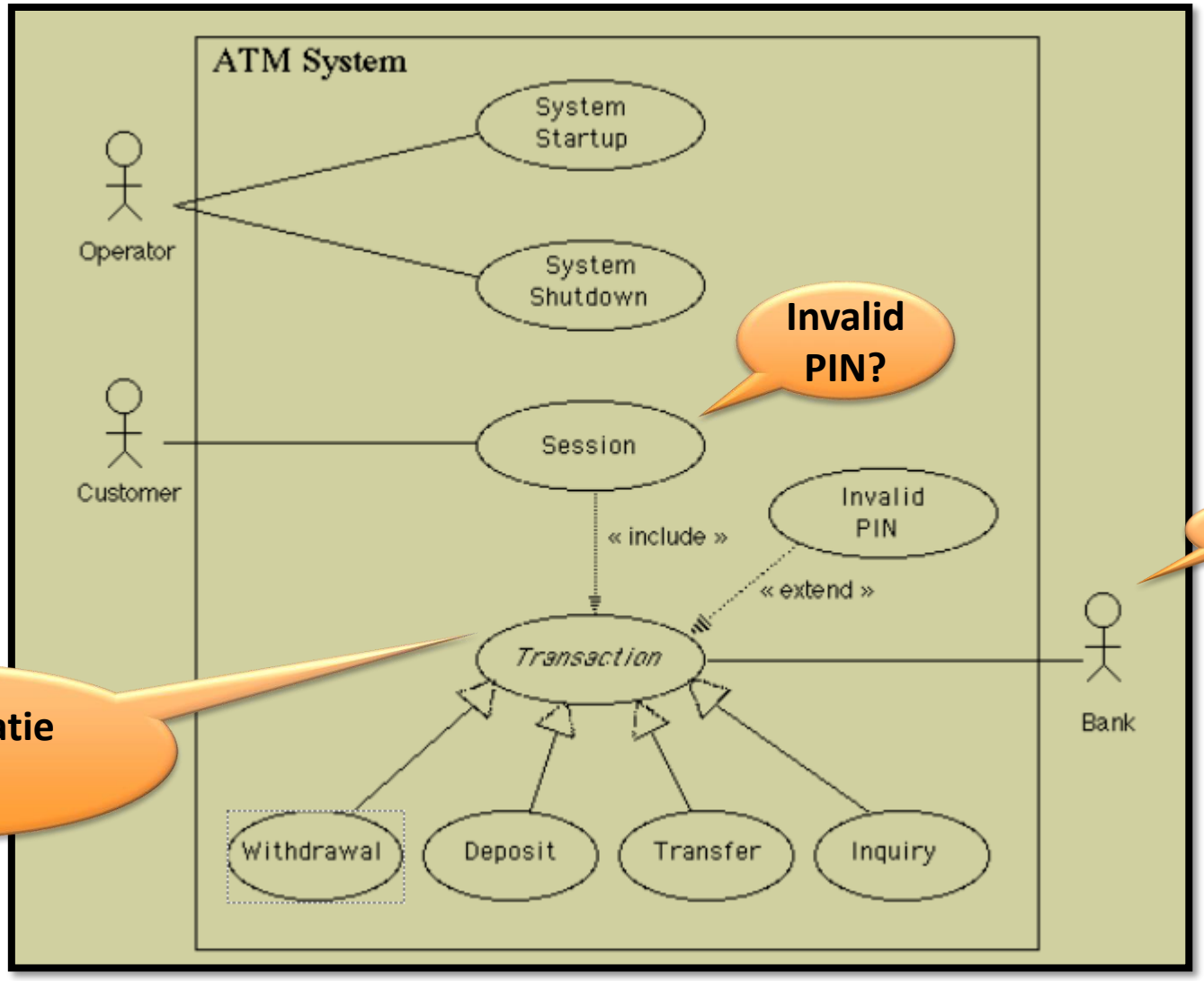
# System Use Case Specifications (2)

- 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

# SUC description: template elements

- Unique ID
- Title of the use case
- Description
- Pre and post conditions
- Actor
- Scenarios
  - Normal/Basic scenario (‘happy flow’)
  - Alternative scenarios
  - Exception scenarios
- …

# SUC – example – ATM system (1)



# SUC – example – ATM system (2)

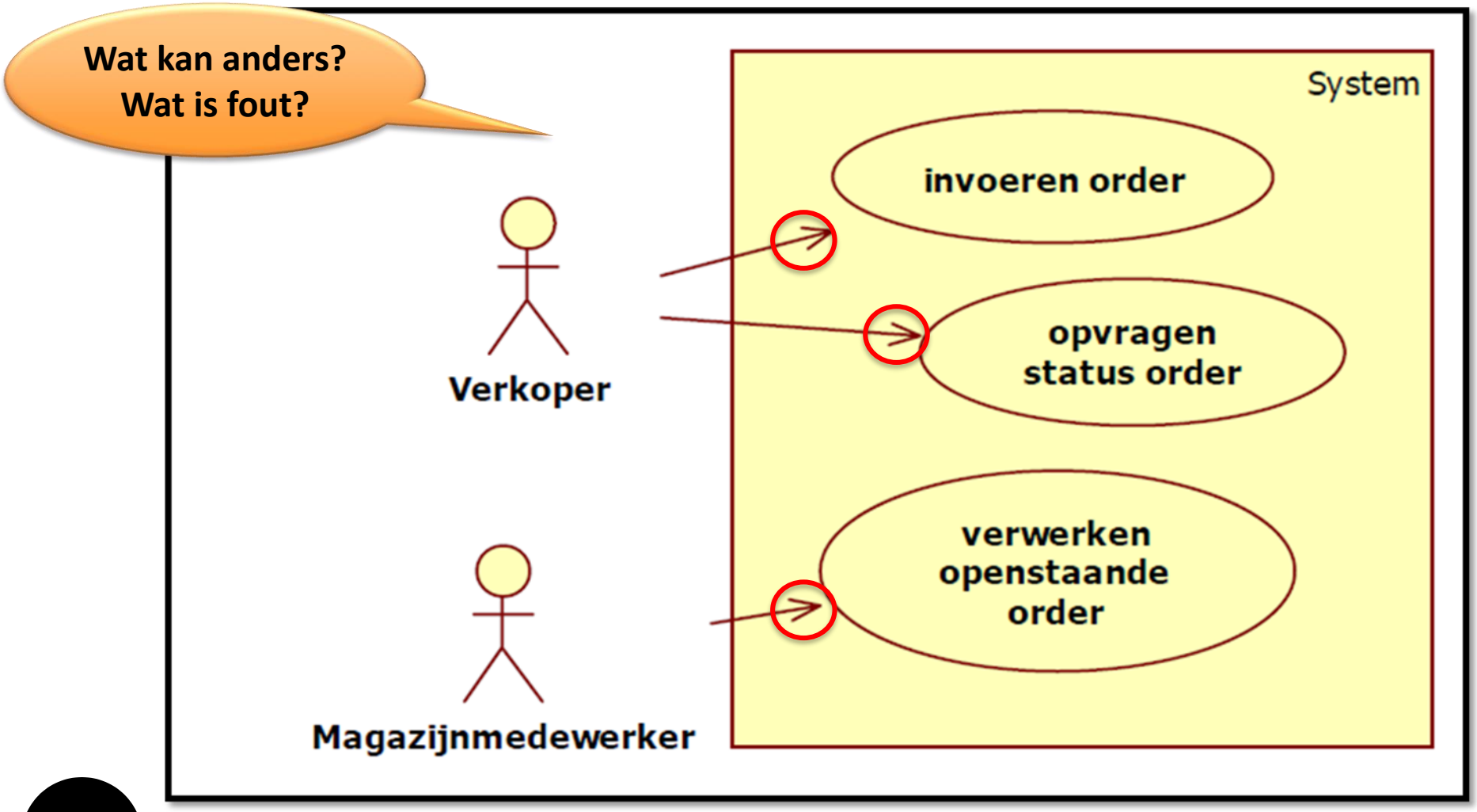
- Description
  - System Startup Use Case
  - System Shutdown Use Case
  - Session Use Case
  - Transaction Use Case
  - Withdrawal Transaction Use Case
  - Deposit Transaction Use Case
  - Transfer Transaction Use Case
  - Inquiry Transaction Use Case
  - Invalid PIN Extension

# 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?

# 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.

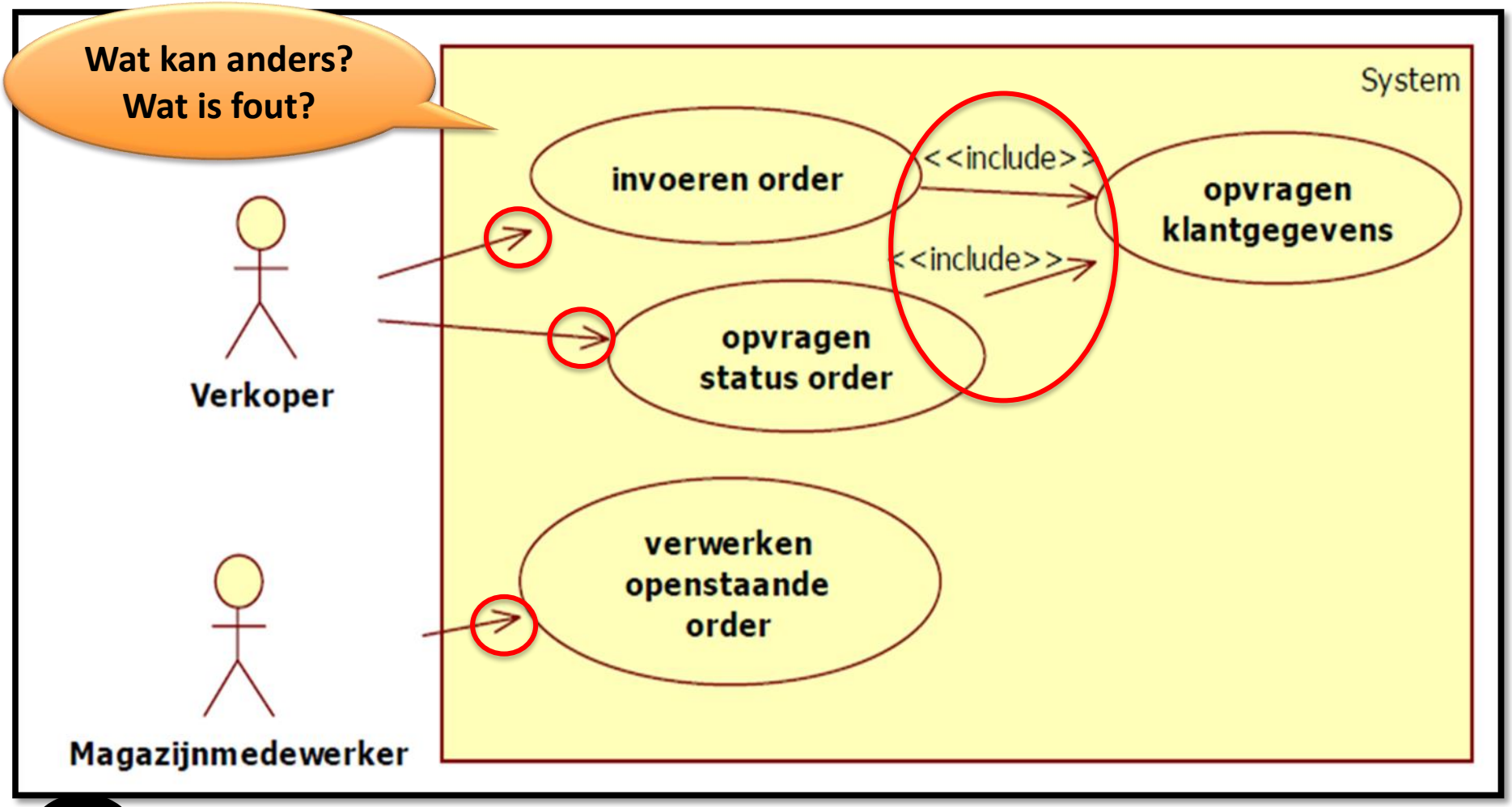
# SUC: exercise “Orders – bestellingen” (4)

- 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.

# 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.

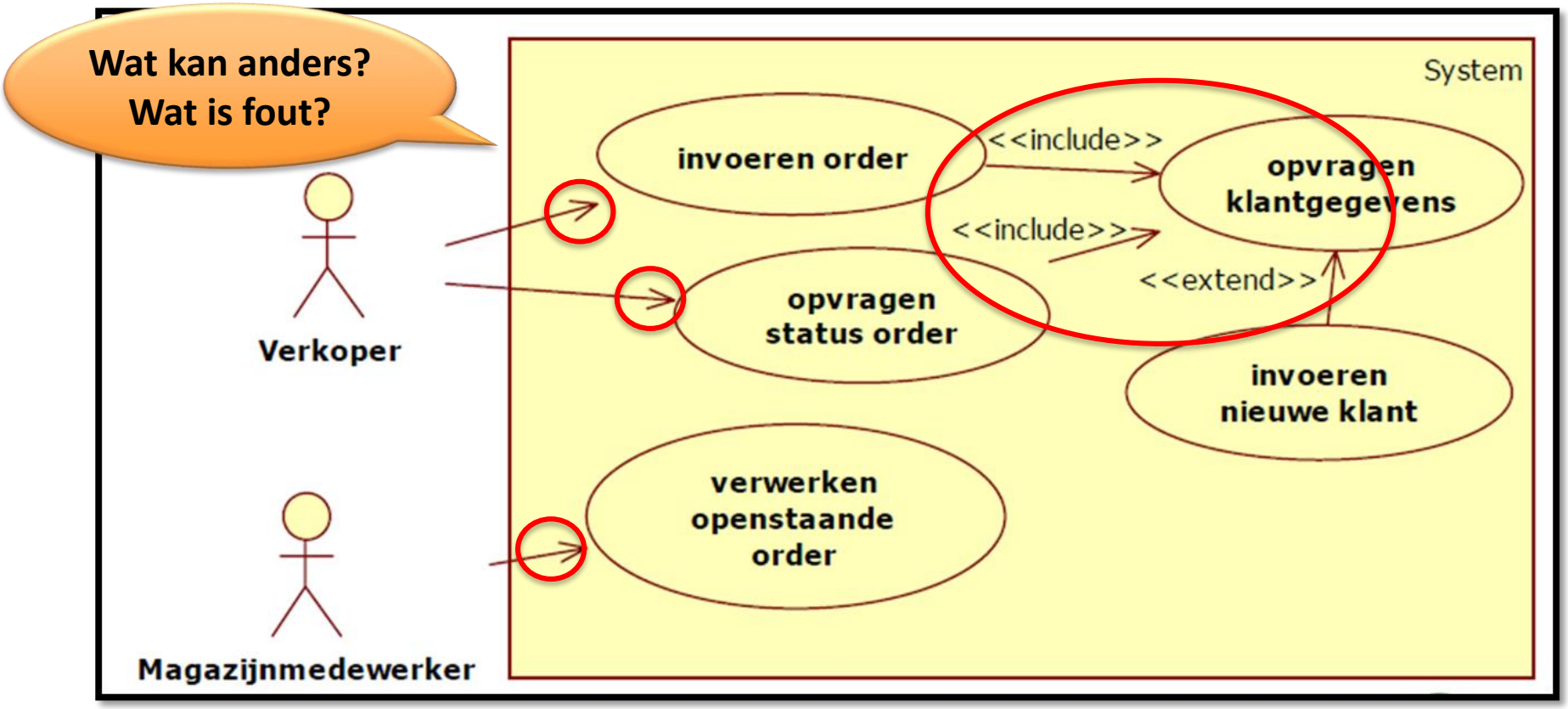
# SUC: exercise “Orders – bestellingen” (6)



# SUC: exercise “Orders – bestellingen” (7)

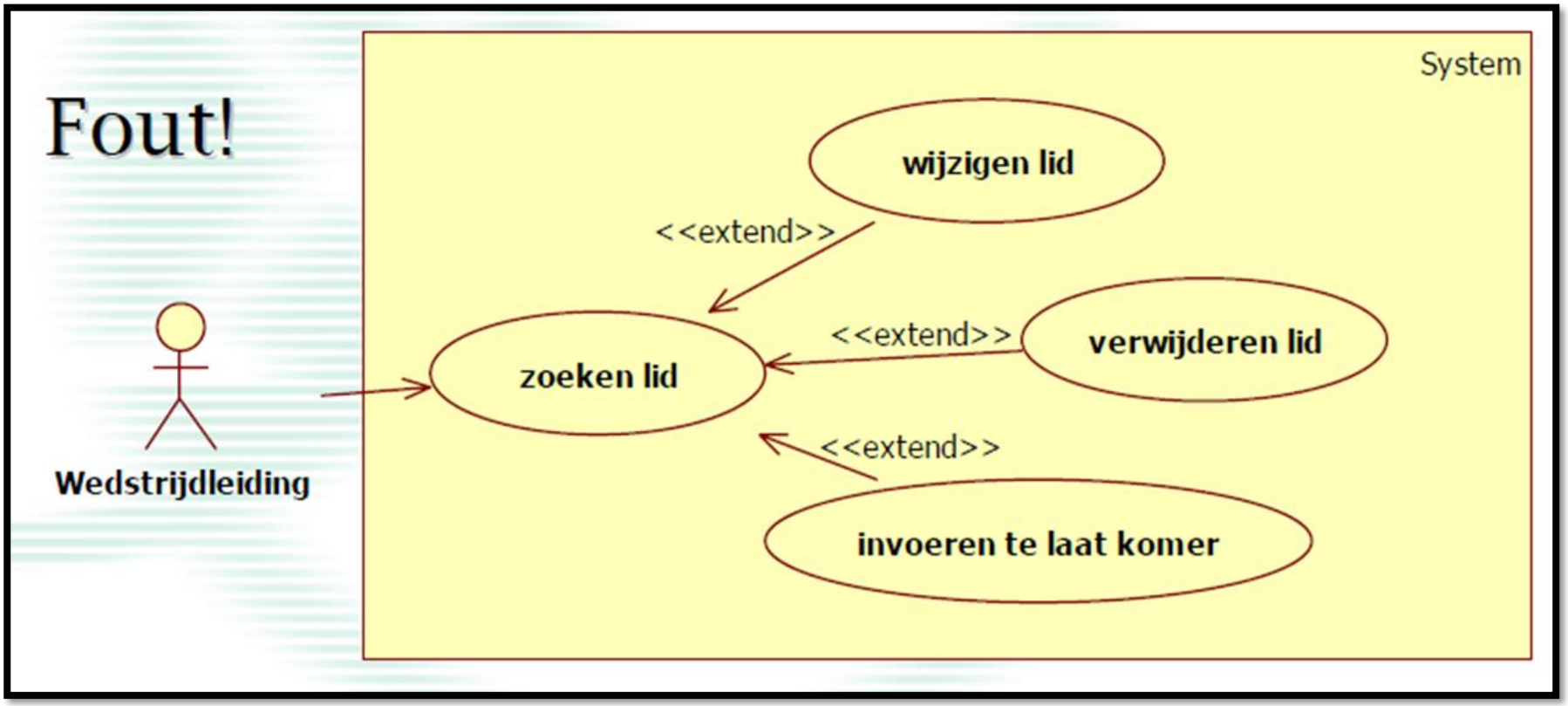
- Alternatieve flow - Invoeren nieuwe klant
  - De actor voert de achternaam, voorletters, het adres en de geboortedatum van de klant in.

# SUC: exercise “Orders – bestellingen” (8)



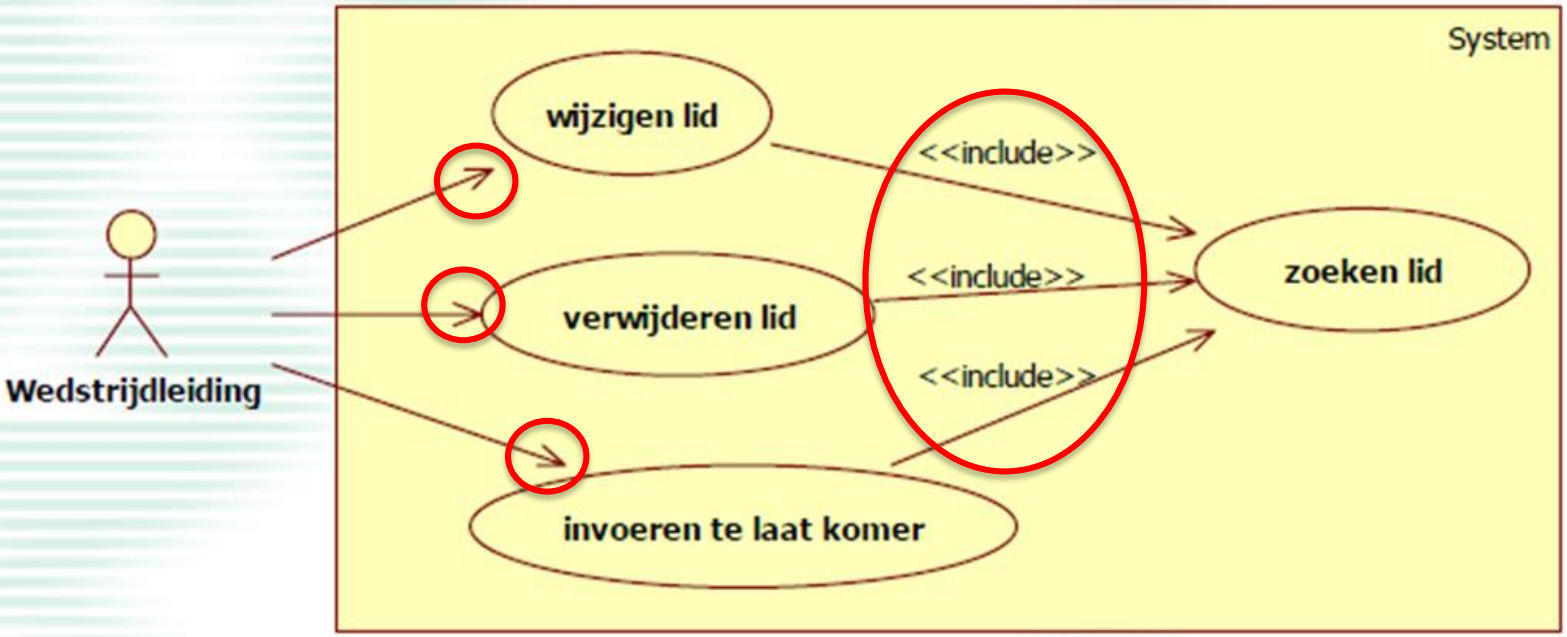
# SUC diagram: tips (1)

- GEEN schermflow-diagram!



# SUC diagram: tips (2)

Goed! **Met uitzondering van de syntax !!!**

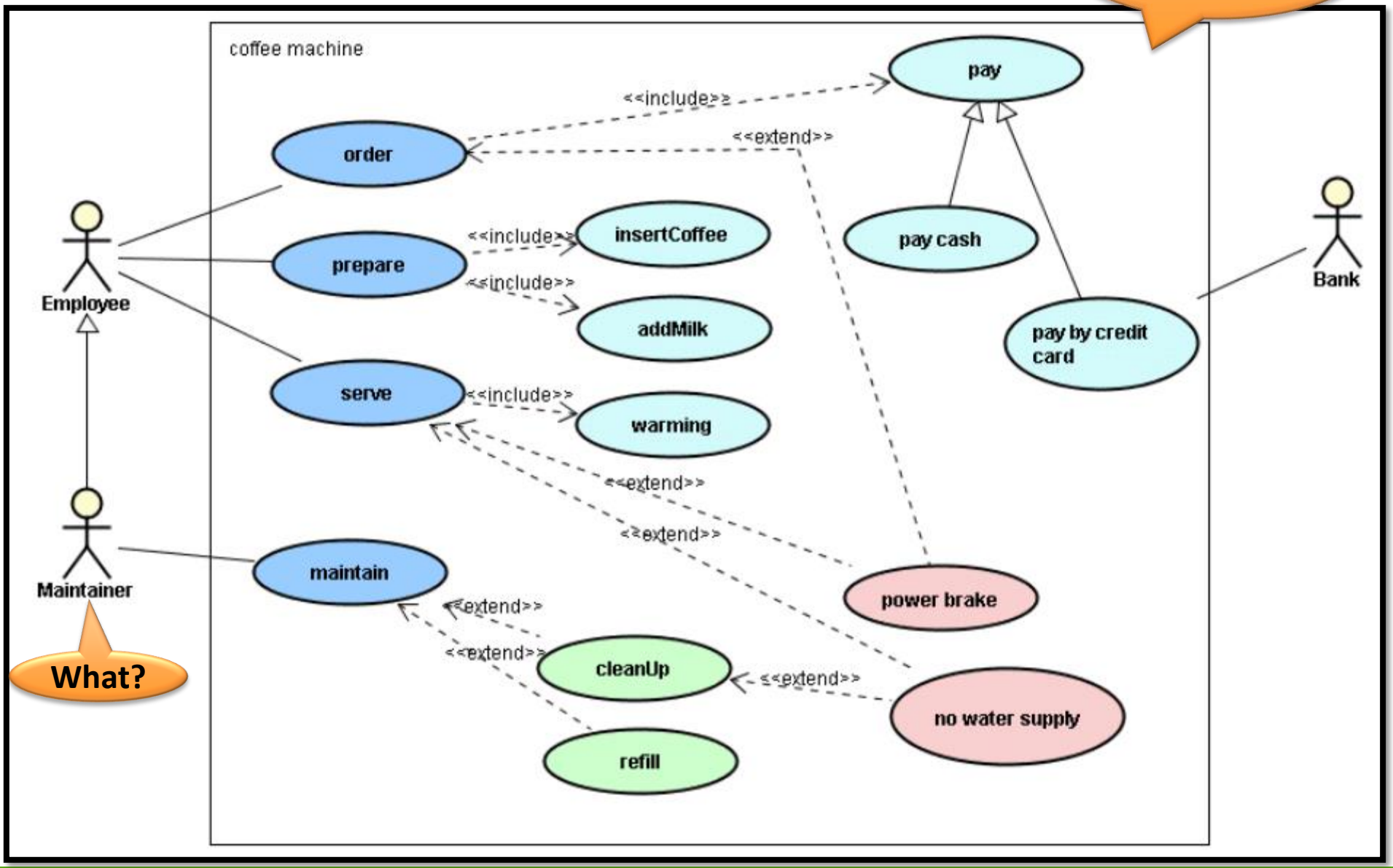


# SUC – exercise coffee machine

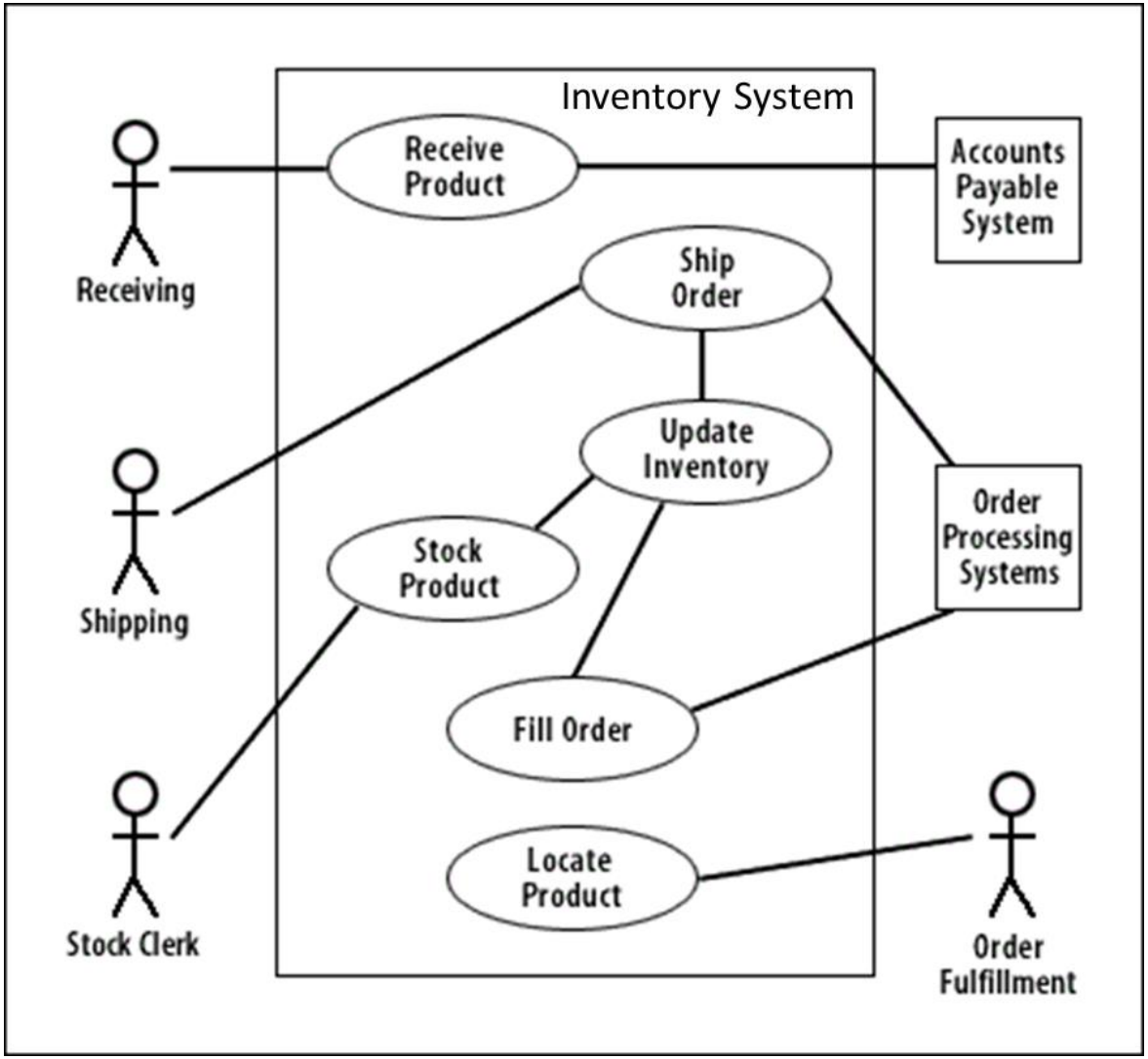
- Your customer is tired of his employees who keep coming late from coffee brakes because it takes them ages to prepare and drink their coffee and to clean their equipment afterwards.
- He asks you to design the prototype of a new revolutionary coffee machine that is both appealing to the eye and yet 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 coffee machine

Wat kan anders? Wat is fout?



# SUC – example SUC to SUC



# 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

# Data/Structural Requirements Models

- Documenting the structure of the systems' data as well as the usage and relationships inside the system context
- Traditionally: ERM ➔ Entity Relationship Models
- Today also substituted by Class models (UML)

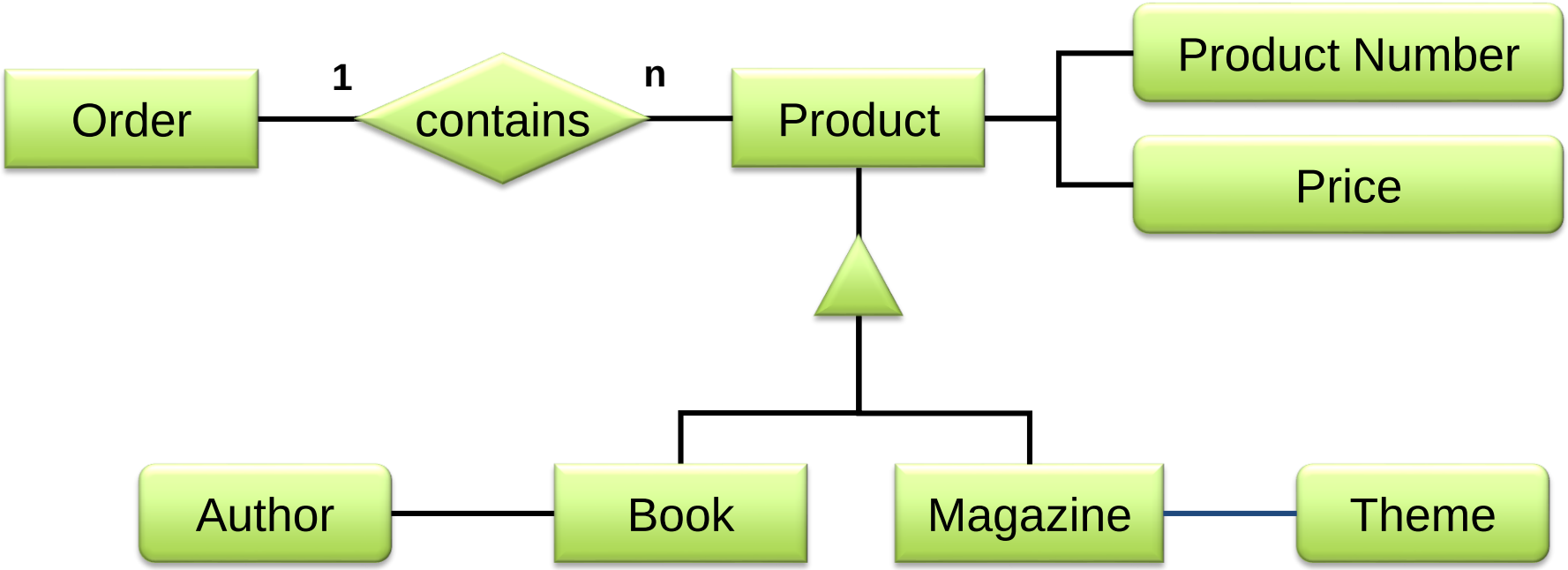
# Entity Relationship Model (ERM)

- Models the structure of data (static)
- Not for functions, behavior, control
- Widely used for (relational) databases
- Clarifies breakdown (structure, hierarchy)
- Understandable (user communication)
- Typical modelling elements:
  - Entity types
  - Relationship types
  - Attributes

# ERM - Elements

| Symbol | Term | Examples |
|---|---|---|
| Car | Entity | Order, Employee, Table |
| Colour | Attribute | Price, Address, Size |
|  | Generalization | Vehicle - {car, truck} |
|  | Relation | Order contains Article |
| (min,max) | Cardinality | (1,n) , 1 , (0,n) |

# ERM - Example


Order
1
contains
n
Product
Product Number
Price
Author
Book
Magazine
Theme

# ERM – Another example

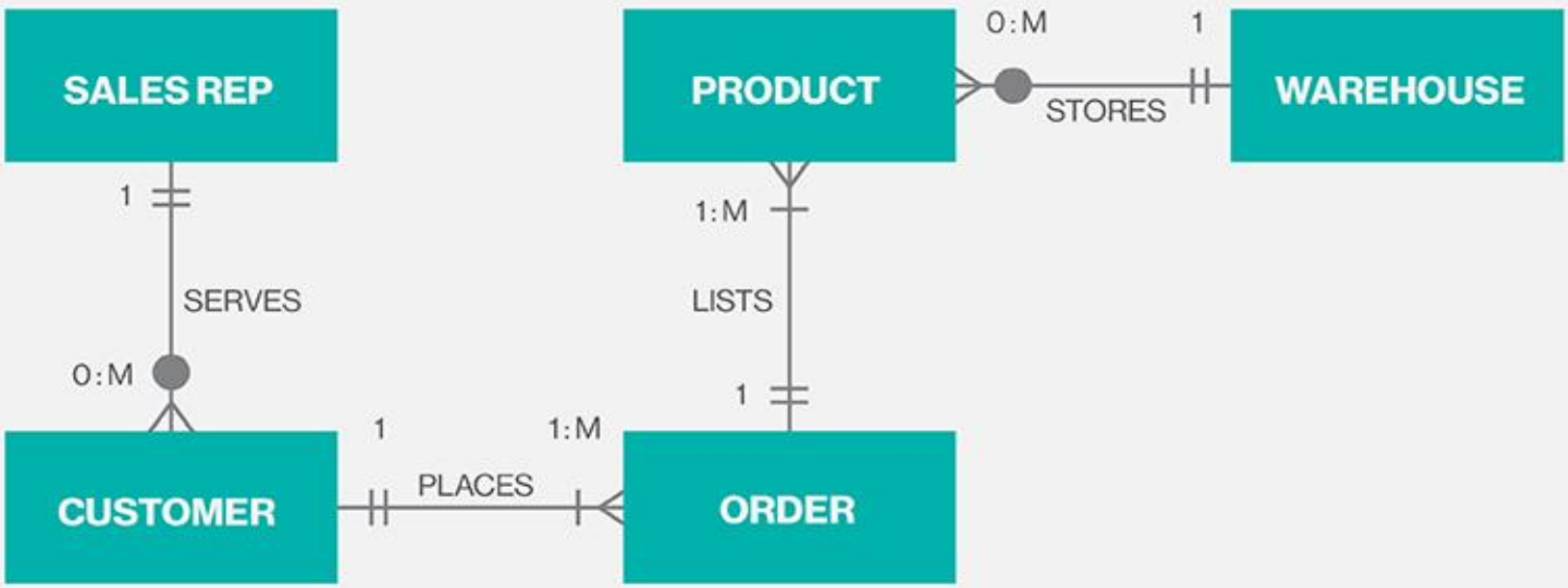
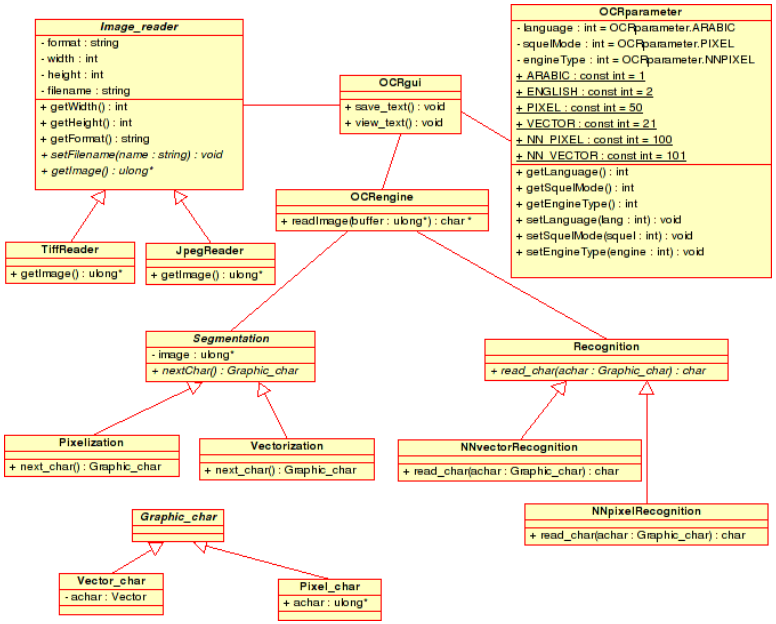



**Figure 1. Entity-Relationship Diagram**

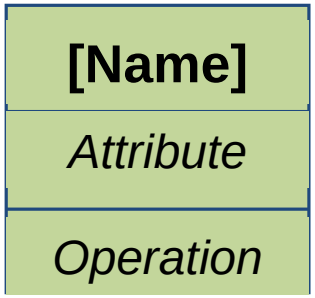
*1 INSTANCE OF A SALES REP SERVES 1 TO MANY CUSTOMERS
*1 INSTANCE OF A CUSTOMER PLACES 1 TO MANY ORDERS
*1 INSTANCE OF AN ORDER LISTS 1 TO MANY PRODUCTS
*1 INSTANCE OF A WAREHOUSE STORES 0 TO MANY PRODUCTS

# Class Models (UML)

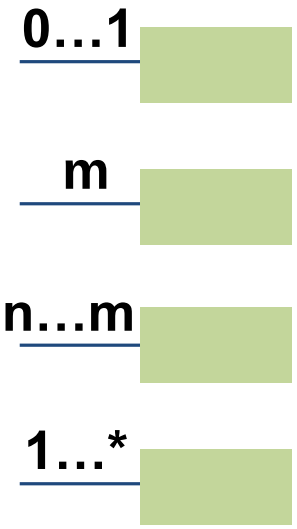



- To model static structures
  - Data, dependencies, relationships, constraints
- Domain specific knowledge
- Different levels
  - Conceptual, specification, implementation
- Tools: UML modelling tool (e.g. Rose)
- Frequently used modelling elements:
  - Classes
  - Associations (multiplicities and roles)
  - Aggregation, Composition, Generalization

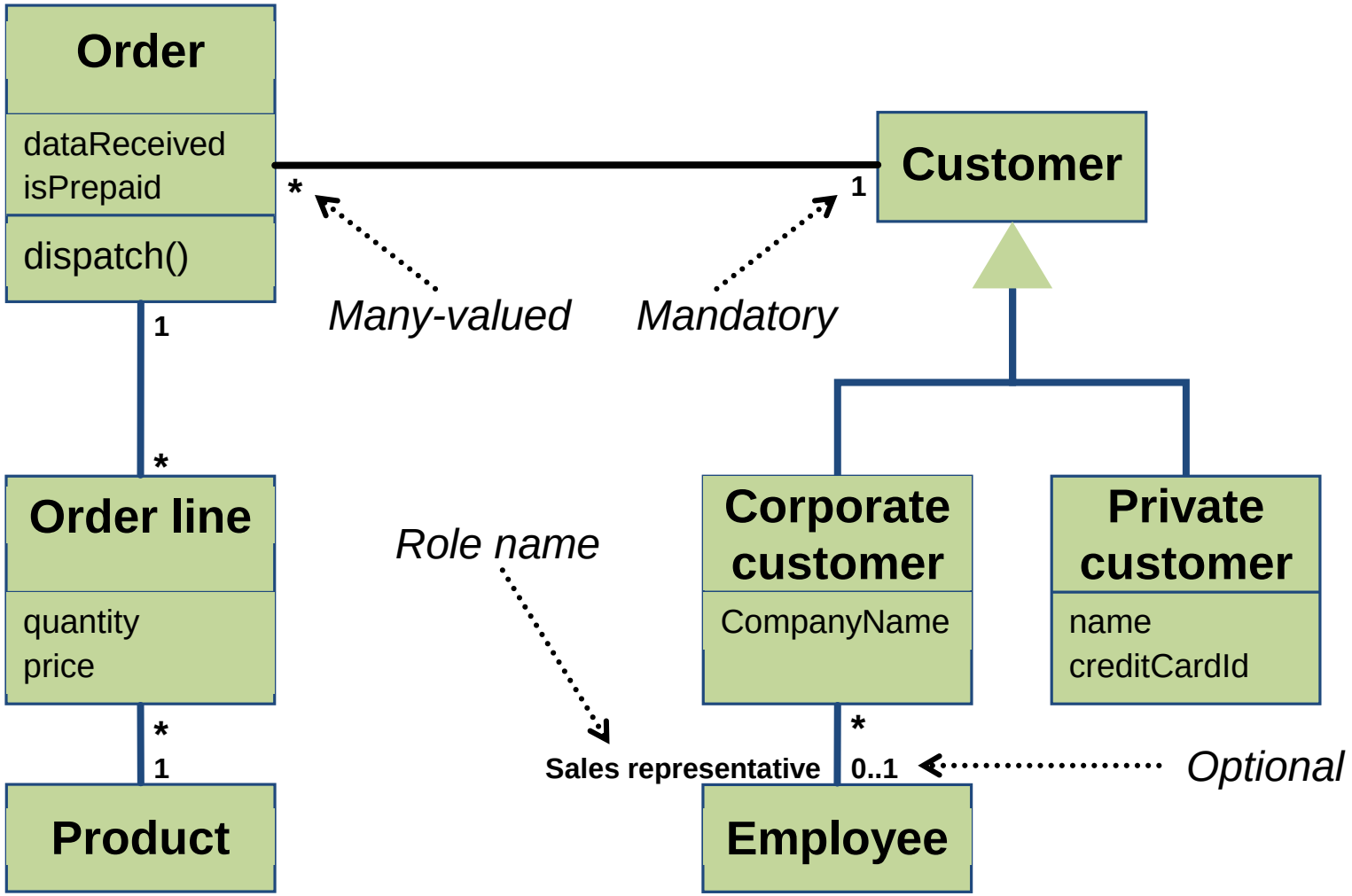
# Class Models: Elements




Classes




# Class Models: example

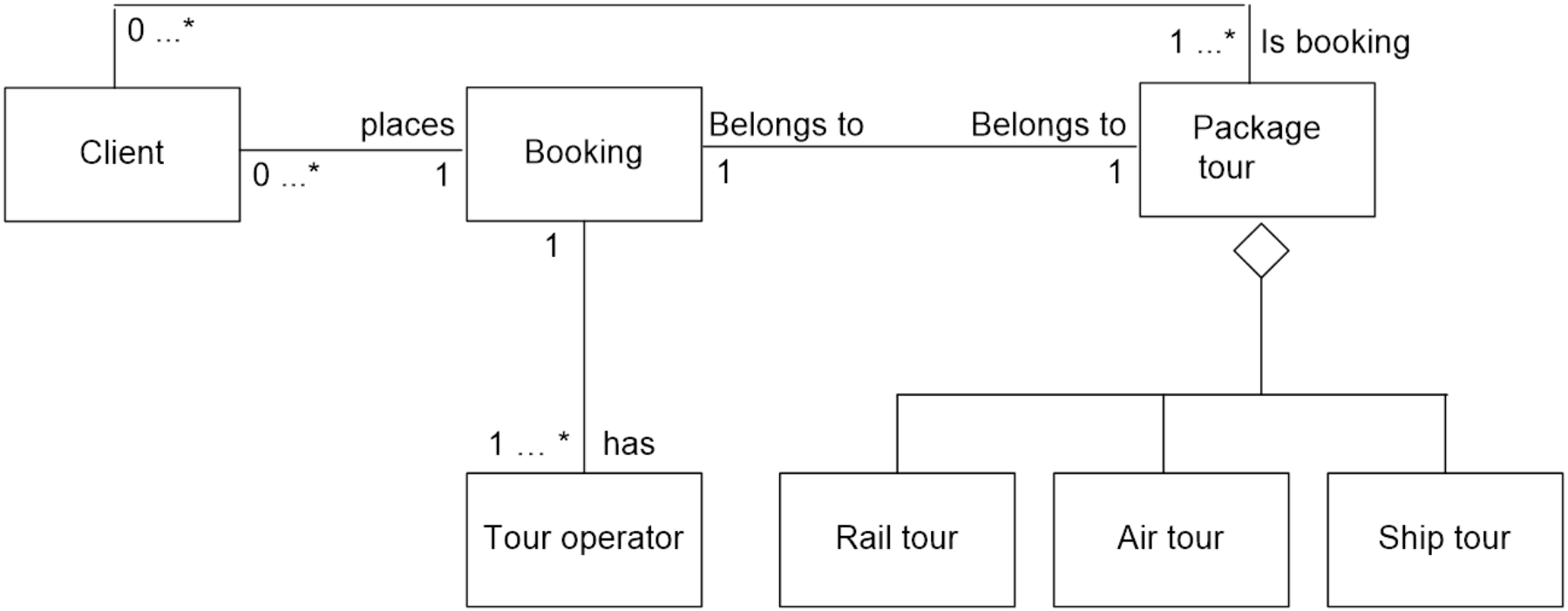


# Analysis: Identifying classes

| Type | Candidate | Examples |
|---|---|---|
| Noun | Class | Car, Motor, Tyre |
| Adjective | Attribute | Colour, Size, Address |
| Verbs | Relation | Submits, Buy, Has, Contains |

# Class diagram - Case Booking System (1)

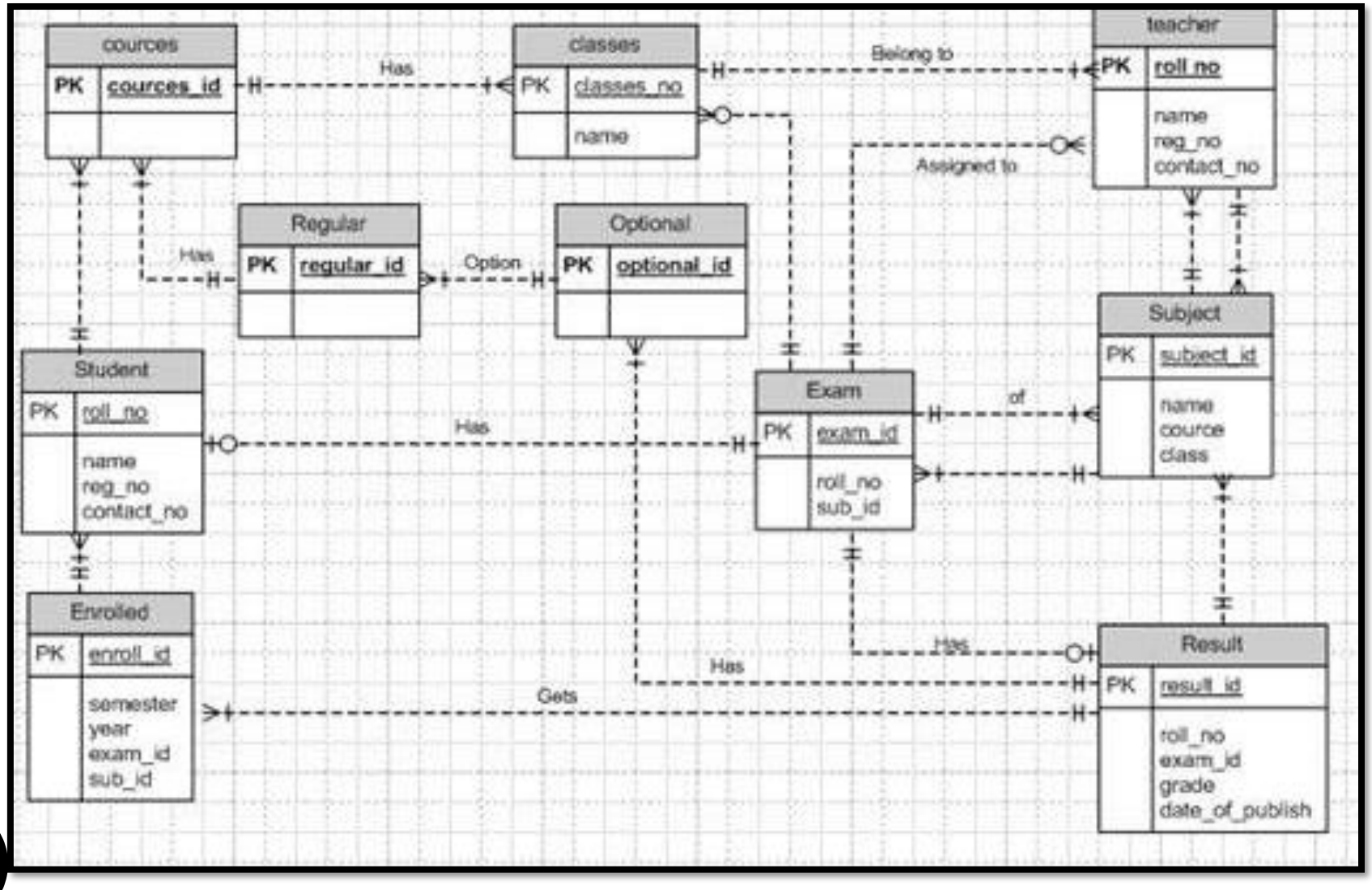
- Review the class diagram and identify the defects, if any.



# Class diagram - Case Booking System (2)

- True or False?
  - The association between client and package tour is redundant and also inconsistent; it should therefore be removed from the model.
  - The association between booking and tour operator makes no sense and should be replaced by a association between package tour and tour operator.
  - The client statement that the package tour sold can be either a rail, air or ship tour is correctly modelled.
  - The multiplicity of the association between client and booking adequately reflect reality.

# Class diagram - Explain



# 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

# Functional Requirements Models

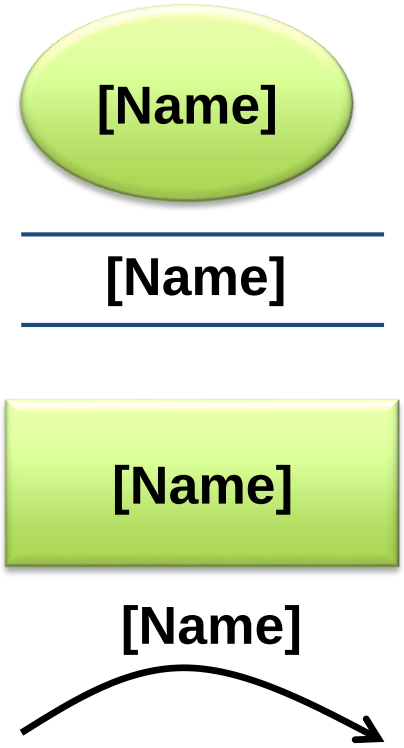
- Focus is on processing input data from the environment of the system to create output data of the system
- Most abstract form: System Use Case Model
- Others:
  - Data flow diagram
  - Activity diagram (UML)

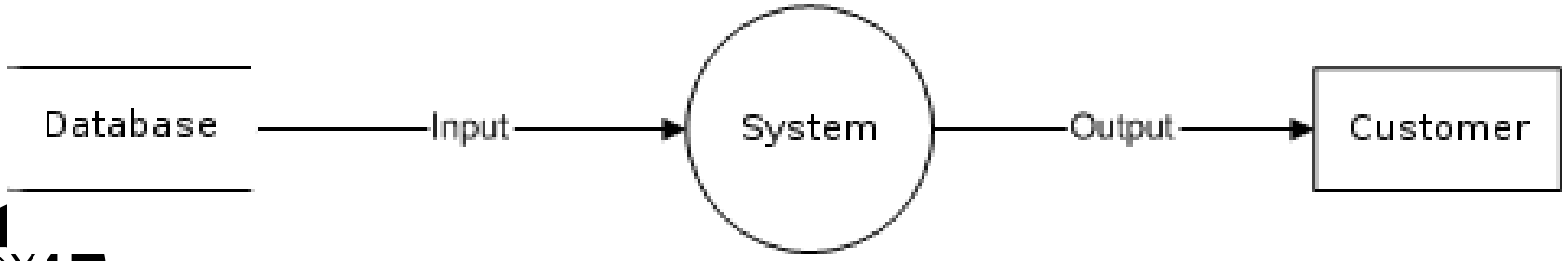
# Data Flow Diagram

- A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system
- Provides no information about the timing or ordering, sequences or parallel behavior
- Levels: high level diagram, low level diagram
- Modeling elements:
  - Processes
  - Data flows, Data stores
  - Sources / sinks

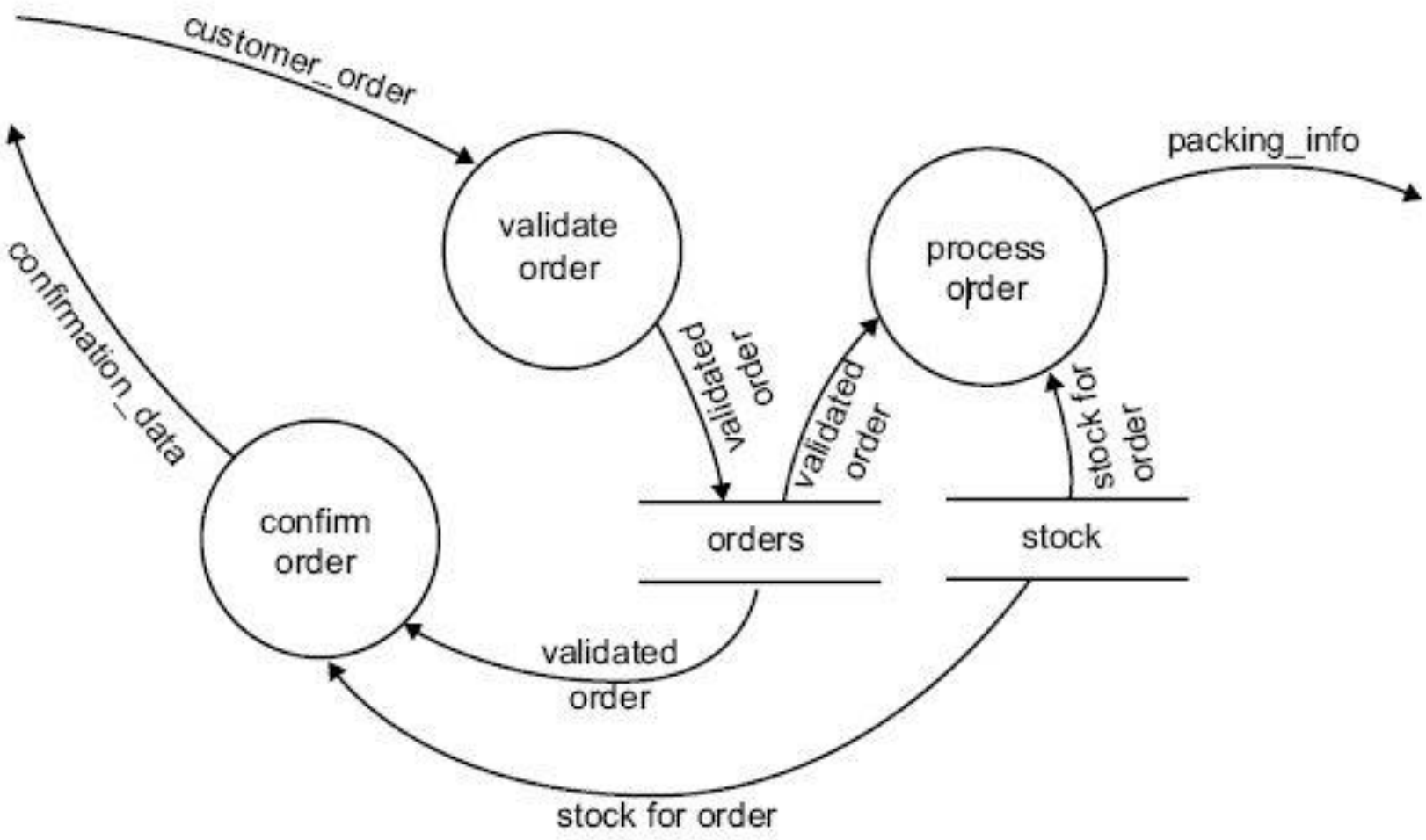
# Data Flow Diagram: elements

- Node: Function / Process
- Data store: File / Database
- Terminator: Input / Output
- Data flow



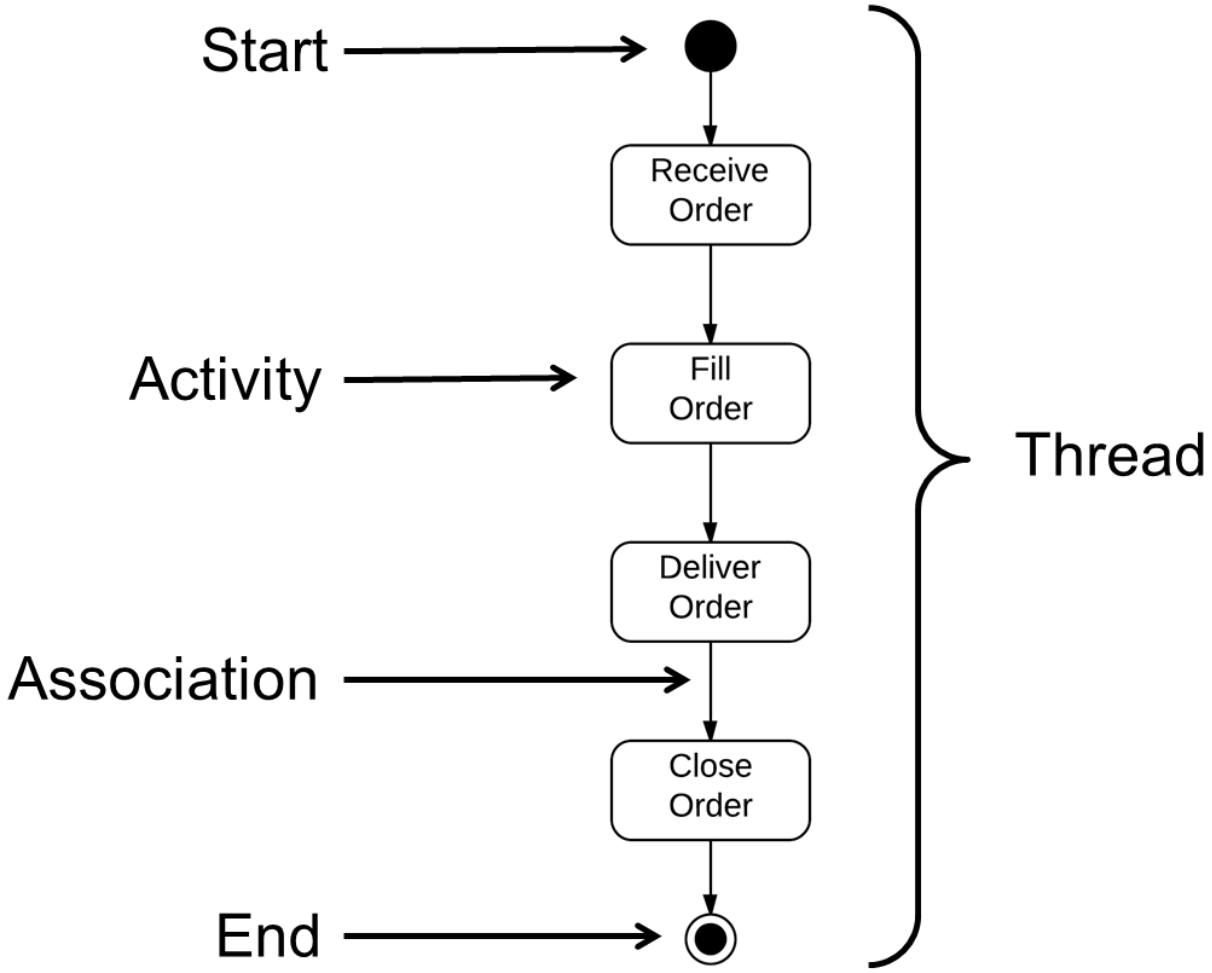
# Data Flow Diagram: example



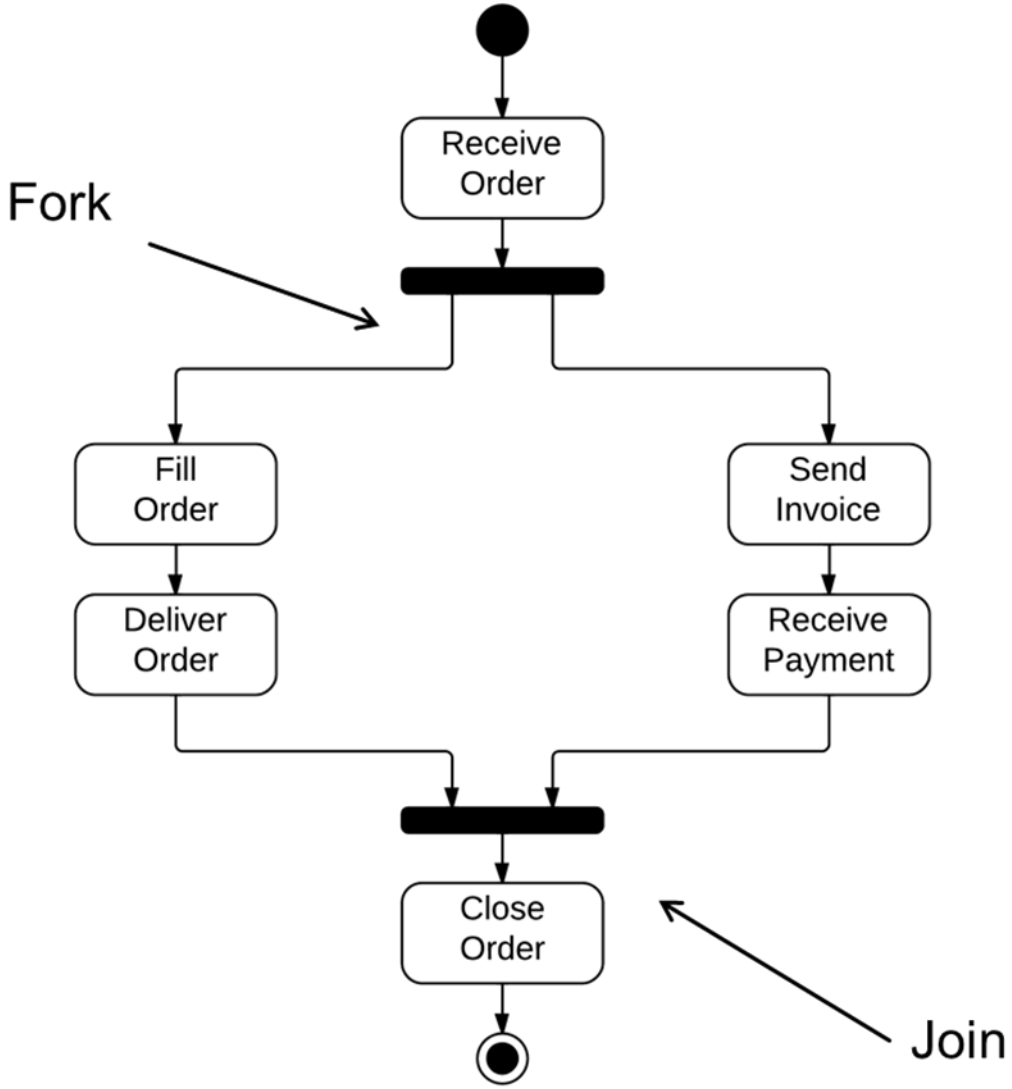
# Activity Diagram


- Focus on activities of a process and the control between them
- Describes the sequencing of activities
- Procedure resulting from individual actions
- Threads may be parallel or conditional
- Often used to clarify/detail the scenarios (normal, alternative and exception flows) of the use cases
- Essential modeling elements: actions, start- and end nodes, control flow, object flow, decision nodes, merge of alternative control flows, fork, join, hierarchic elements

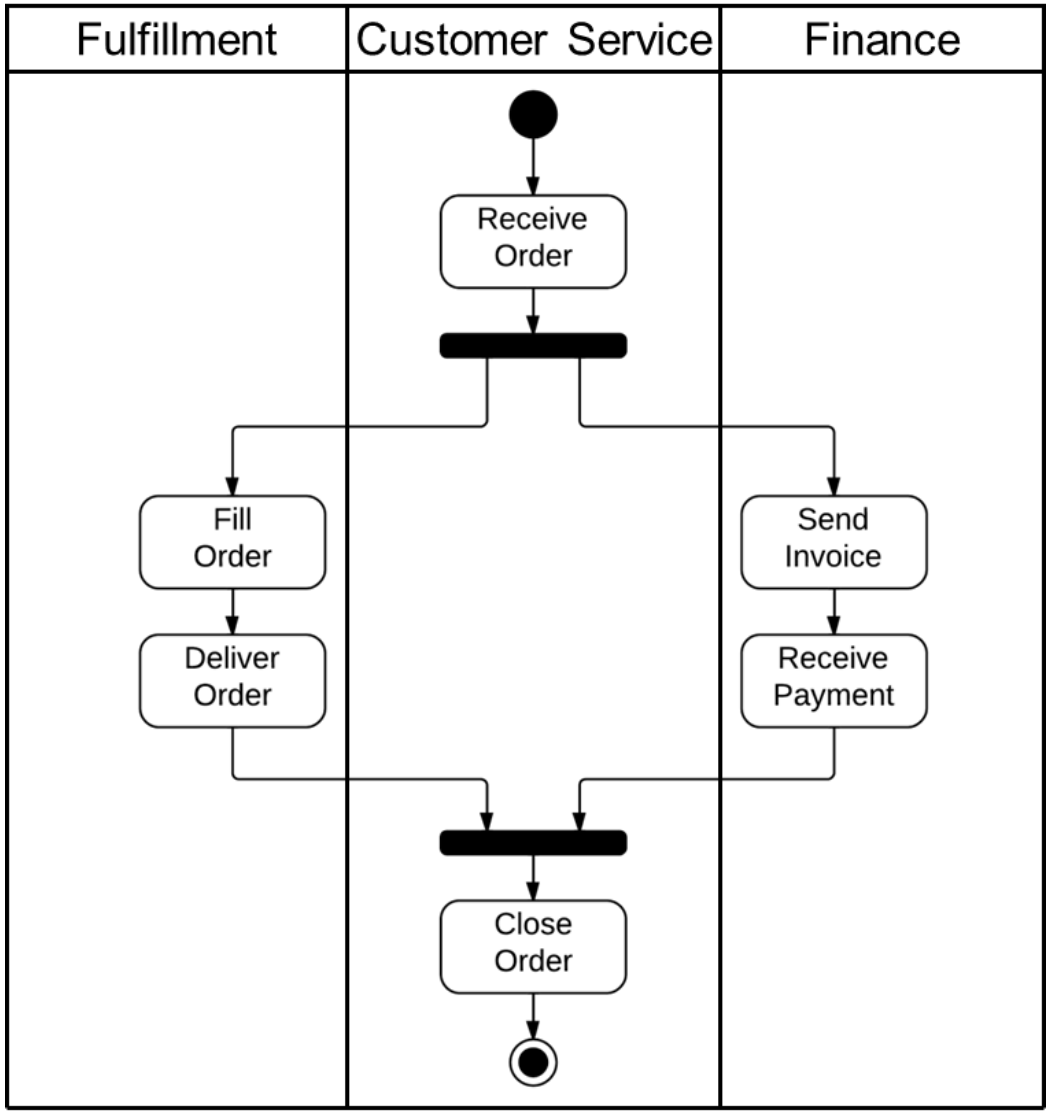
# Activity Diagram - model elements



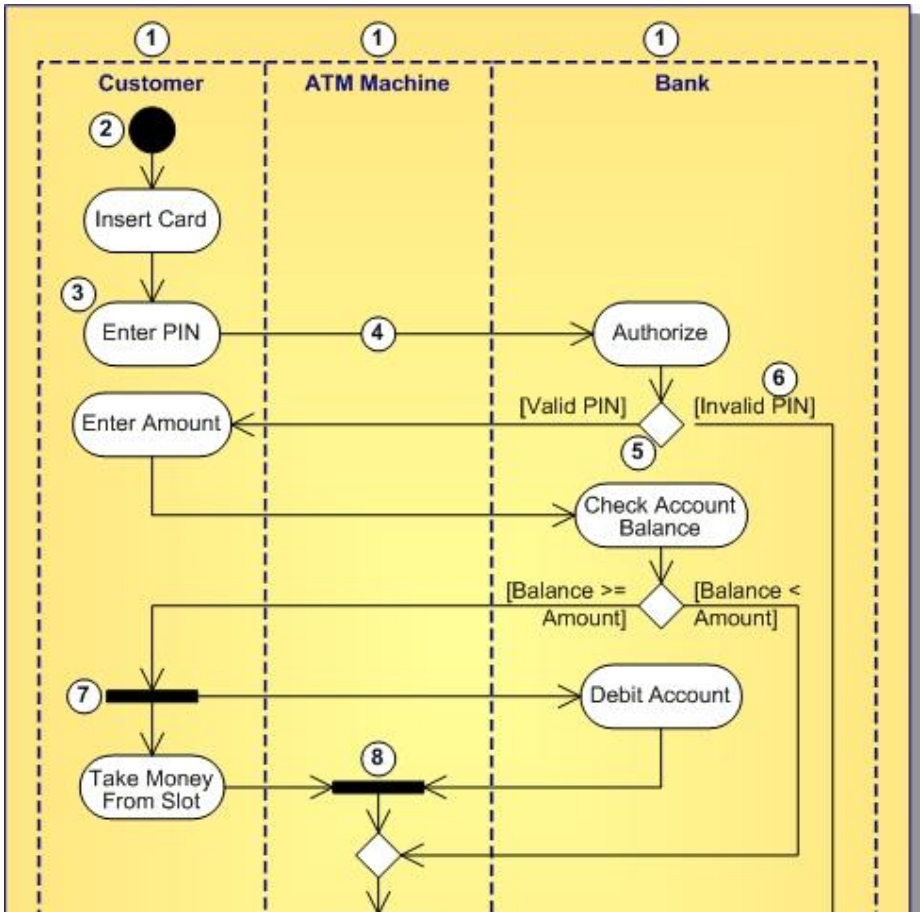
# Activity Diagram - parallel behavior

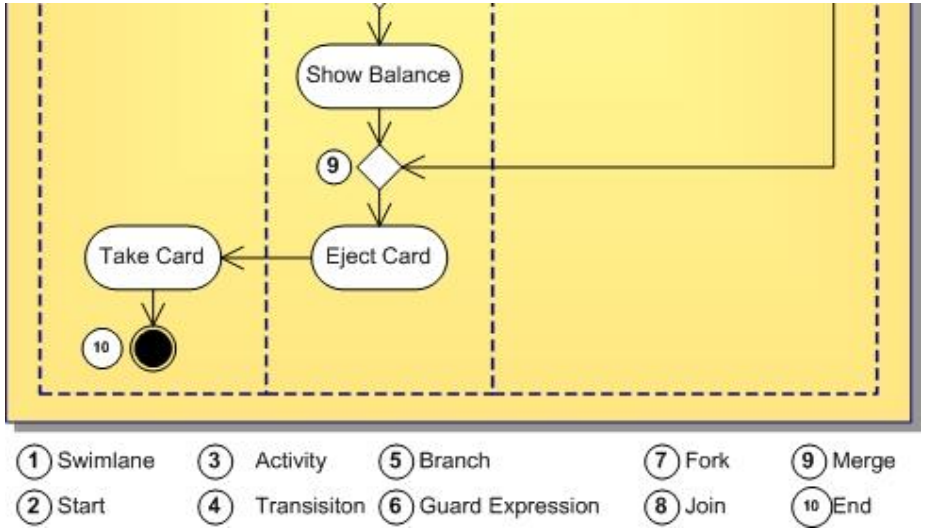


# Activity Diagram - swim lanes (1)

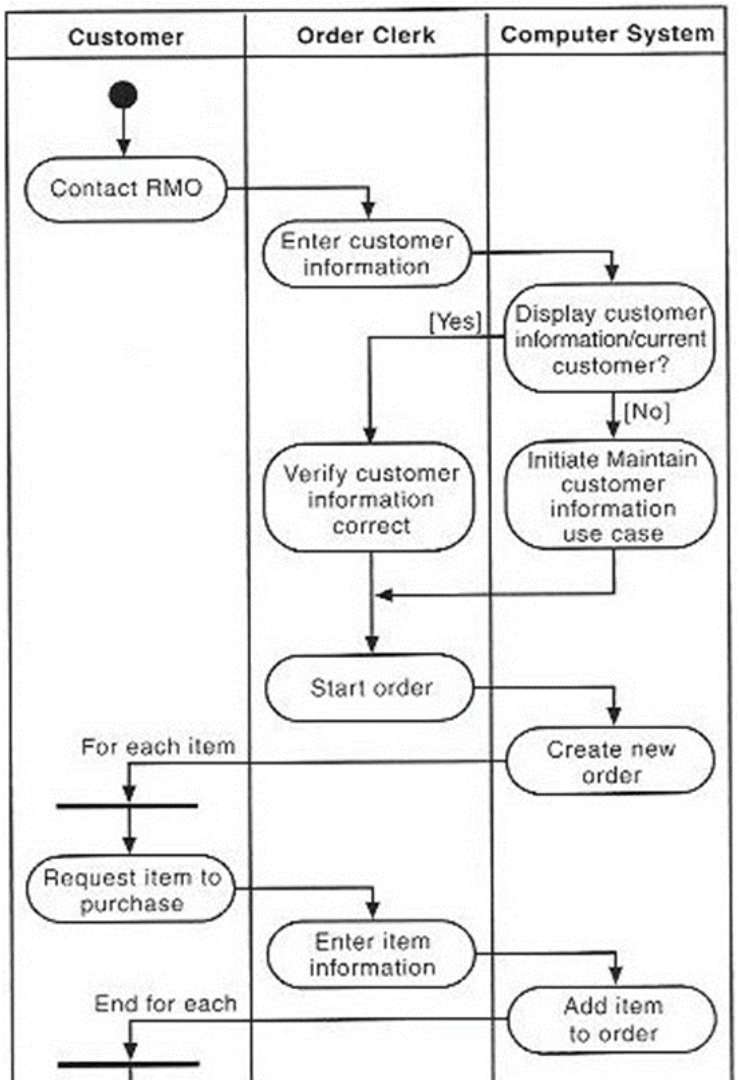


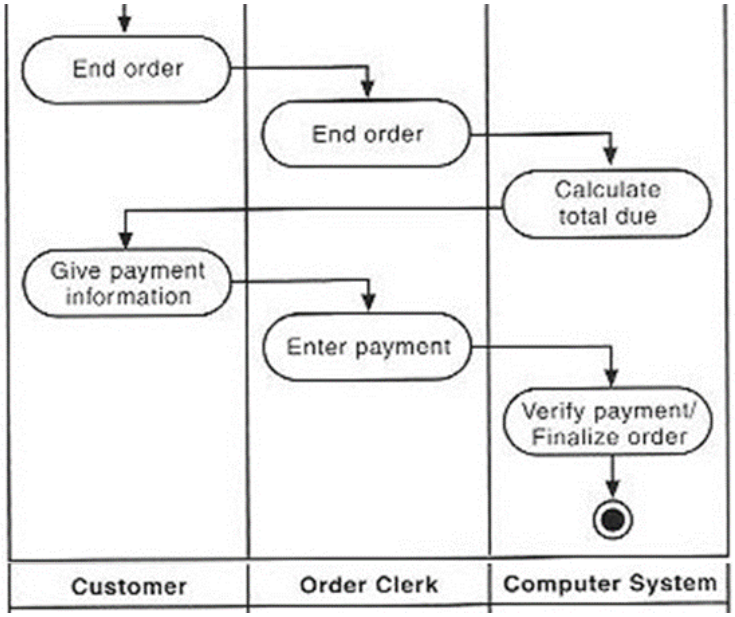
# Activity Diagram - swim lanes (2)



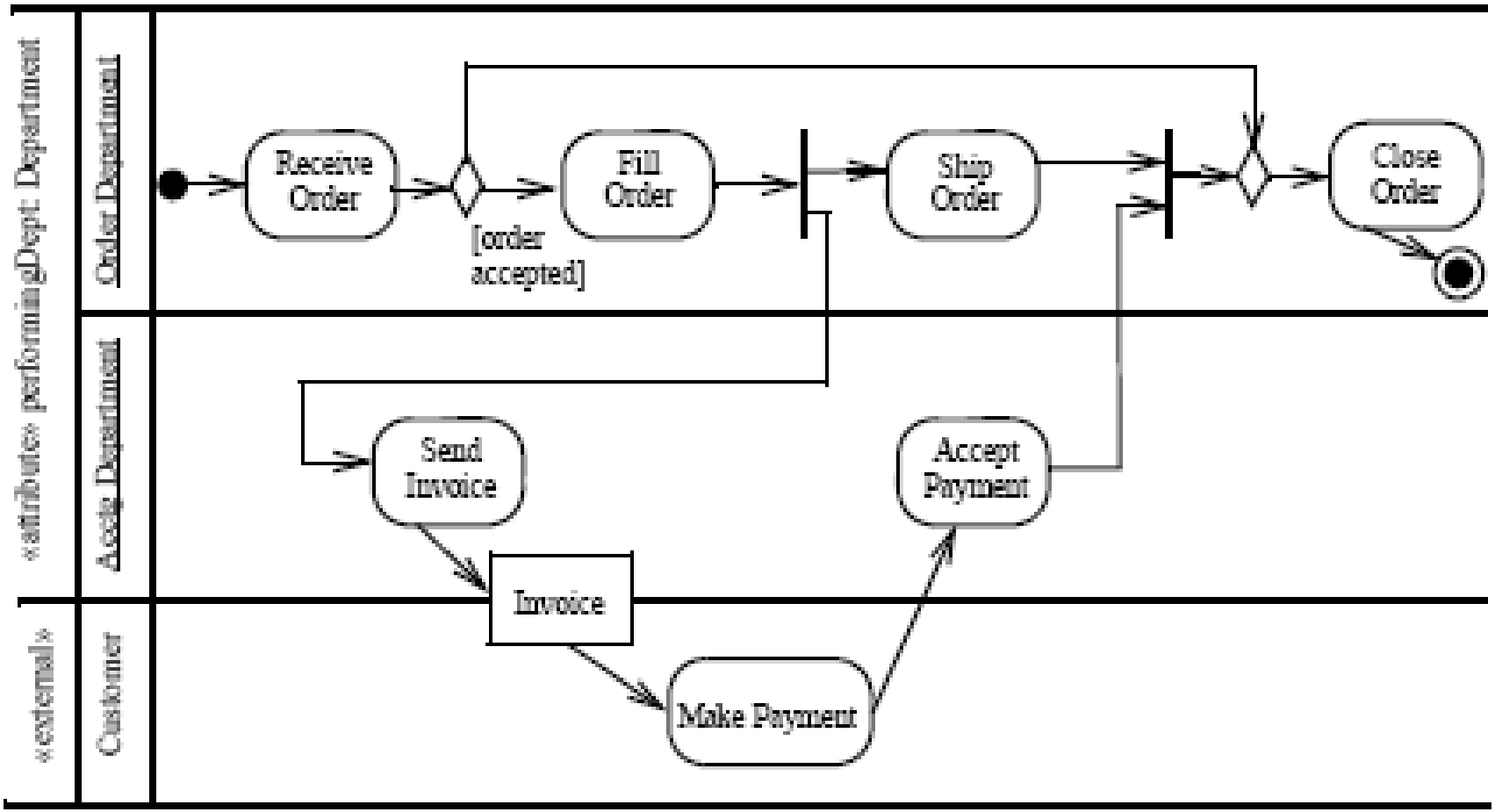



# Activity Diagram - swim lanes (3)

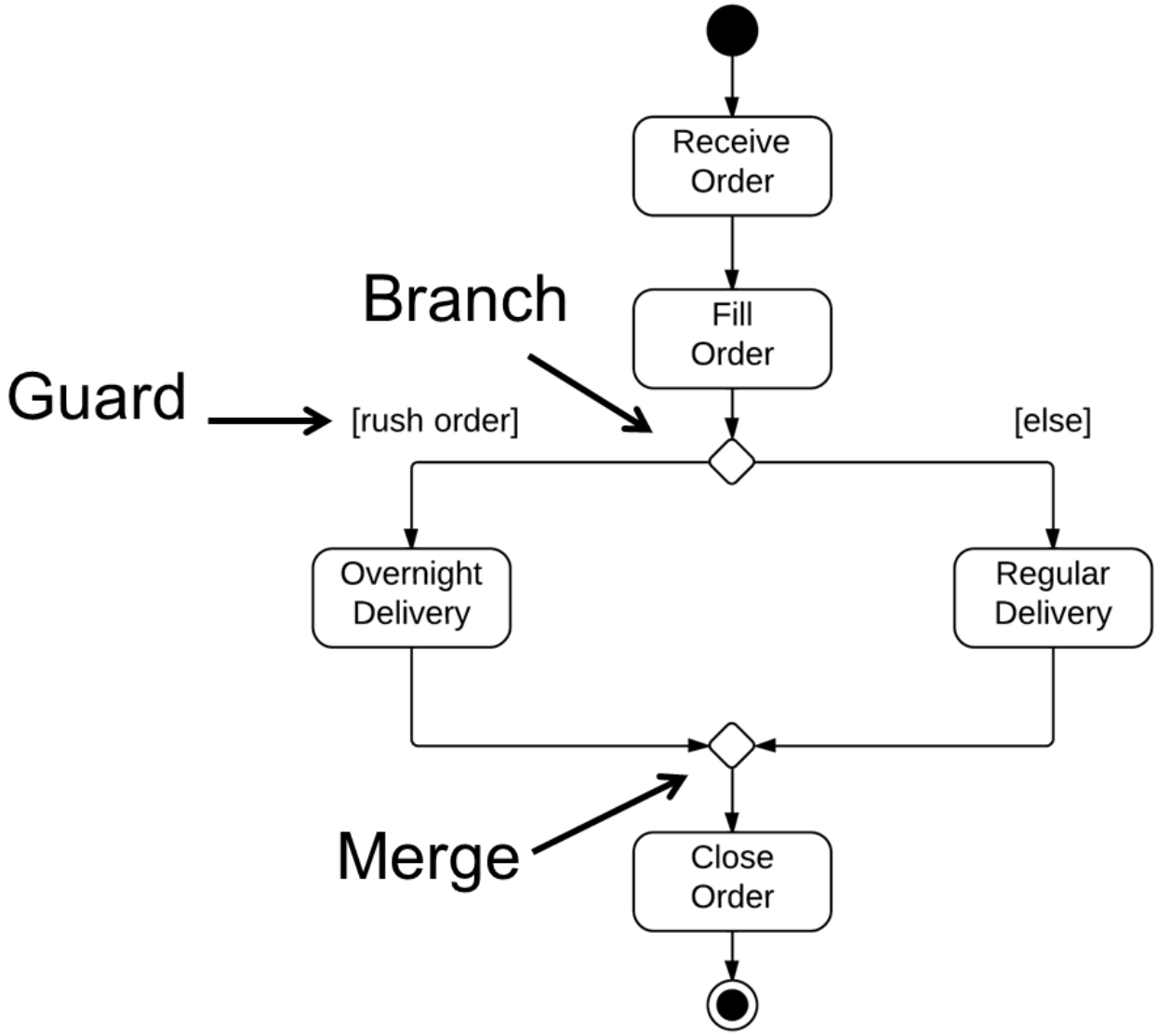


# Activity Diagram - swim lanes (4)



# Activity Diagram - conditional behavior



# Activity Diagram – how to draw (1)?

- Flow chart consisting of activities performed by system
- Activity diagram not exactly flow chart
  - Additional capabilities: branching, parallel flow, swim lane, etc.
- Before drawing activity diagram
  - Clear understanding about elements used
  - The main element = activity itself
    - Activity = function performed by system
- After identifying activities
  - Understand how activities associated with constraints and conditions

# Activity Diagram – how to draw (2)?

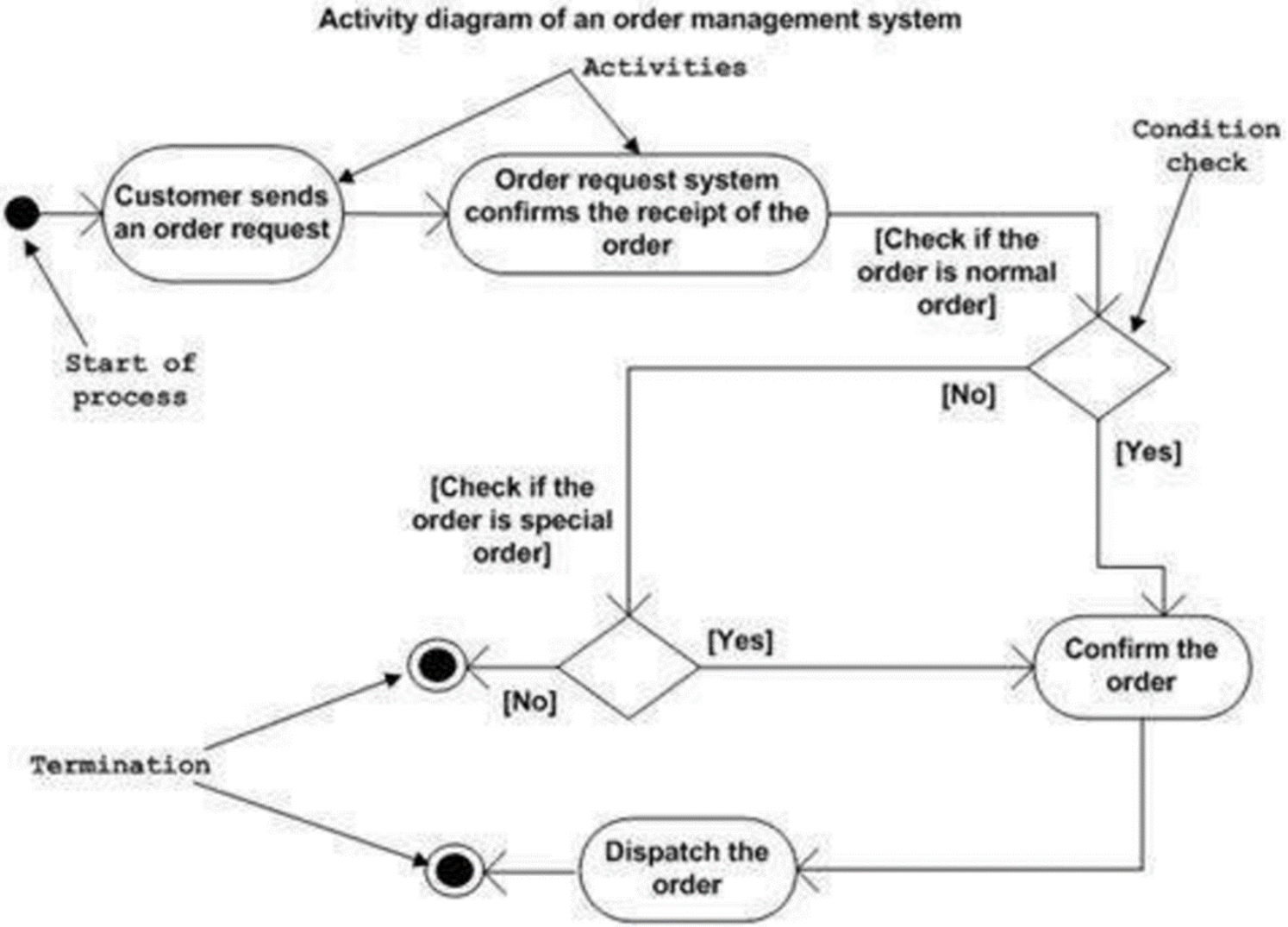
- So before drawing activity diagram, identify following elements:
  - Activities
  - Association
  - Conditions
  - Constraints

- Then draw diagram

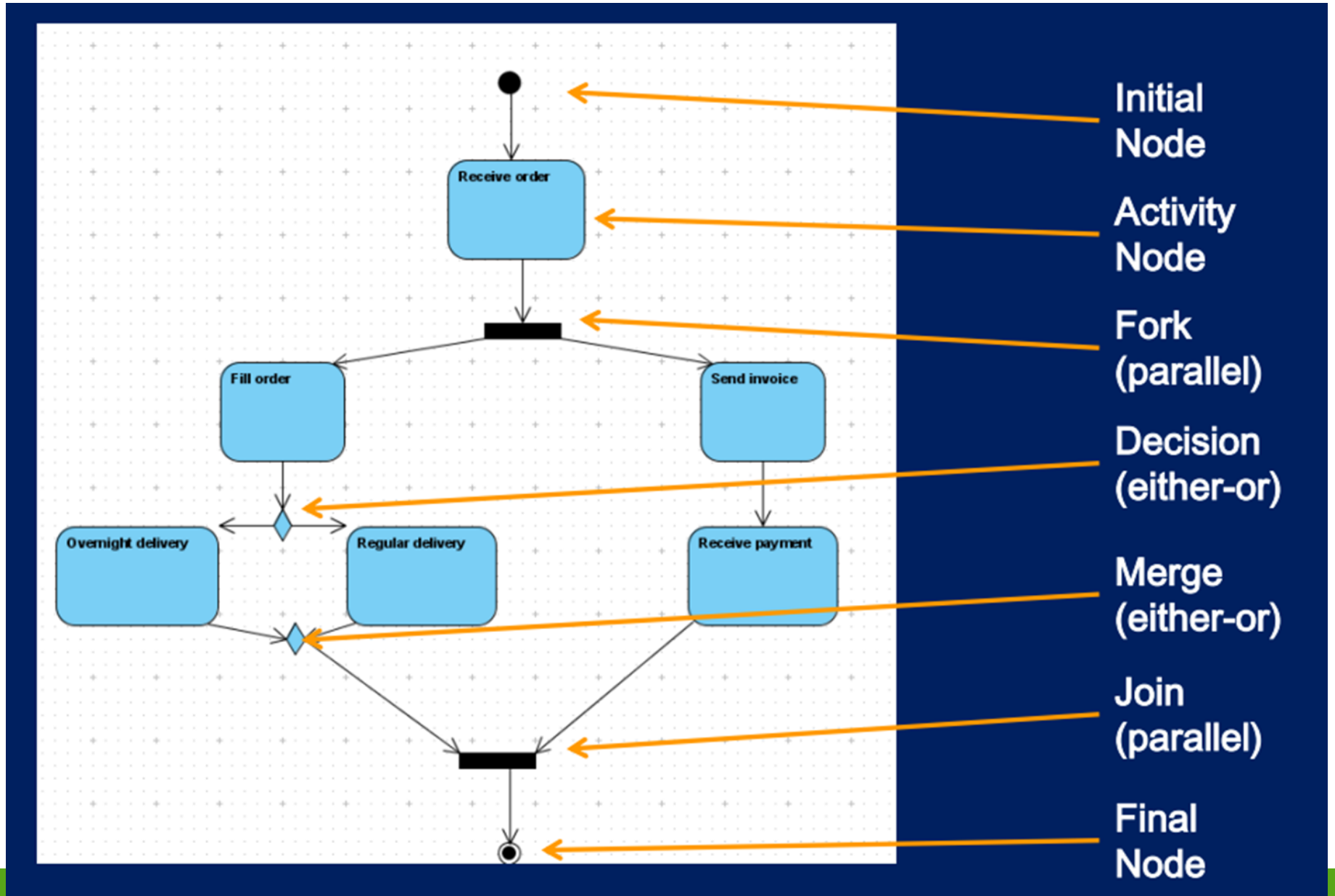
# Activity Diagram – example (1)

- Example of activity diagram for order management system.
- 4 activities identified which are associated with conditions
  - Send order by the customer
  - Receipt of the order
  - Confirm order
  - Dispatch order
- After receiving order request, condition checks are performed to check if “normal” or “special order”
- After that dispatch activity = termination of process.

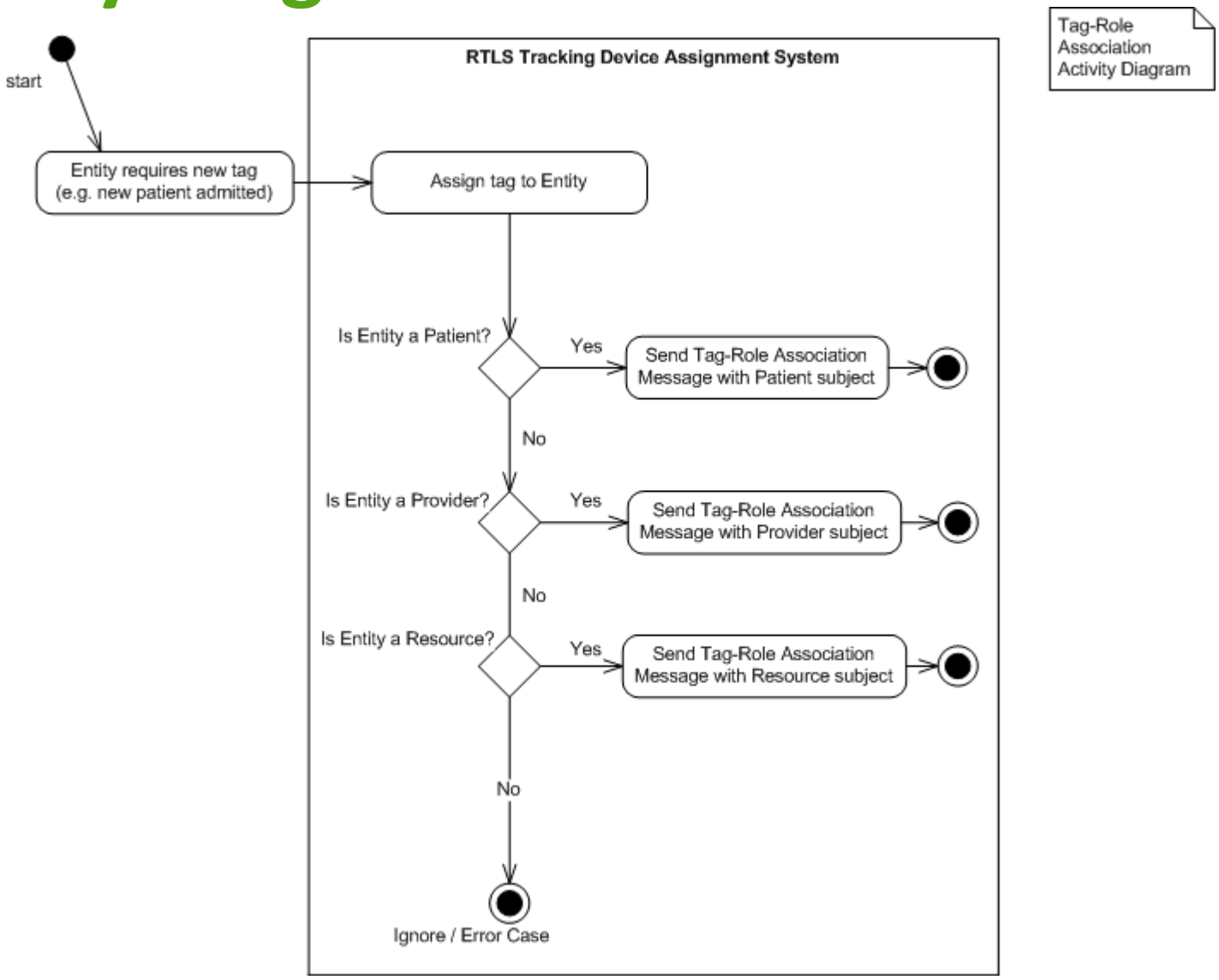
# Activity Diagram – example (2)



# Activity Diagram – example order



# Activity Diagram – switch case



# Activity Diagram - pro's and cons

- ☺ Easy to learn
- ☺ Widely applicable, e.g. use case details
- ☺ Can show complex, time-related procedures

- ☹ Pitfall: specifying too many details
- ☹ 50% of activities would (almost) be enough
- ☹ Little practical information in UML manual

# Activity Diagram – exercises “Aankoop”

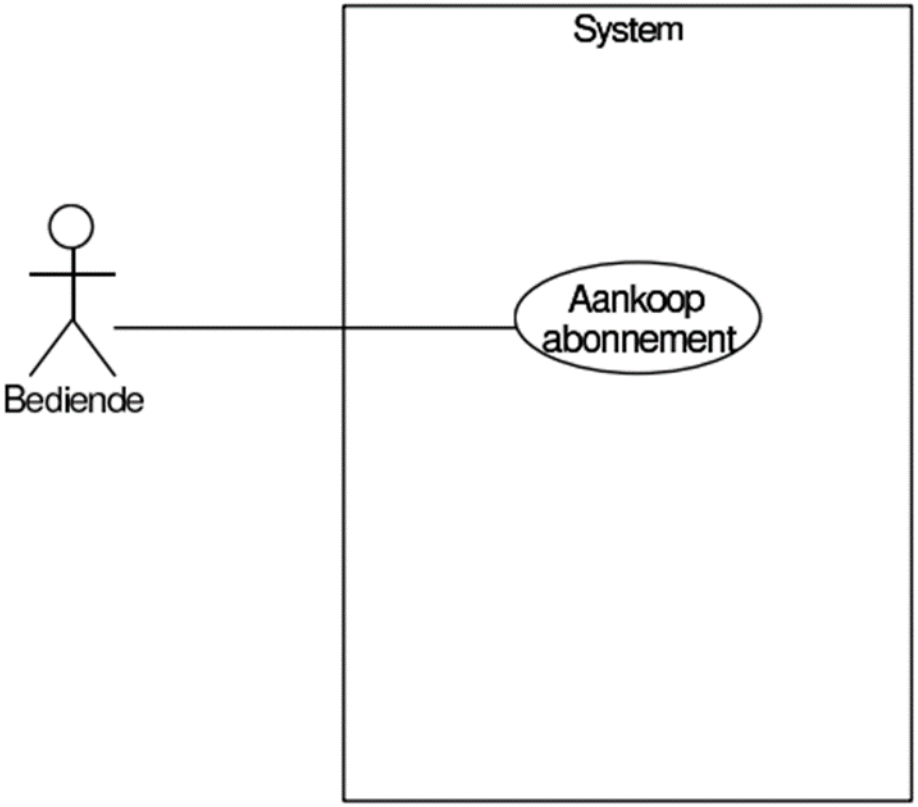
- De aankoopdienst verwerkt aankooporders van andere departementen in het bedrijf.
- Werknemers die het aankooporder indienen, worden de klanten van de aankoopdienst genoemd.
- Een assistent van de aankoopdienst ontvangt het order en volgt de verdere afhandeling op, tot het order is besteld en ontvangen.
- Indien de assistent een order ontvangt onder de 1.500 EUR, mag deze onmiddellijk een bestelbon opmaken en naar de vooraf goedgekeurde leverancier sturen.
- Bij orders boven de 1.500 EUR, moet er eerst een offerte worden gestuurd naar de leveranciers die het desbetreffende product leveren.
- Wanneer alle offertes terug aangekomen zijn, selecteert de assistent hieruit één leverancier, maakt de bestelbon op, en verstuurt deze naar de leverancier. Teken Activity Diagram

# Activity Diagram – exercises “De Lijn” (1)

- Een informatiesysteem van De Lijn schrijft bus- en tramabonnementen uit
- De klant komt aan het loket en geeft zijn persoonlijke informatie (o.a. naam, adres, telefoonnummer, geboortedatum) door aan de loketbediende die deze informatie ingeeft in het systeem.
- Het systeem zal zelf bepalen — op basis van de geboortedatum — welk type abonnement gepast is (jongeren, normaal of 60+) en wat de bijbehorende prijs is.
- Wanneer de betaling is geregistreerd, wordt het abonnement afgedrukt.
- Op het abonnement wordt de naam van de klant, zijn adres en geboortedatum, evenals het type abonnement afgedrukt
- Ga ervan uit dat het steeds gaat om een nieuwe klant, d.w.z. de persoonlijke informatie moet steeds worden ingegeven door de bediende

# Activity Diagram – exercises “De Lijn” (2)

- Stel het activity diagram op voor volgende use case



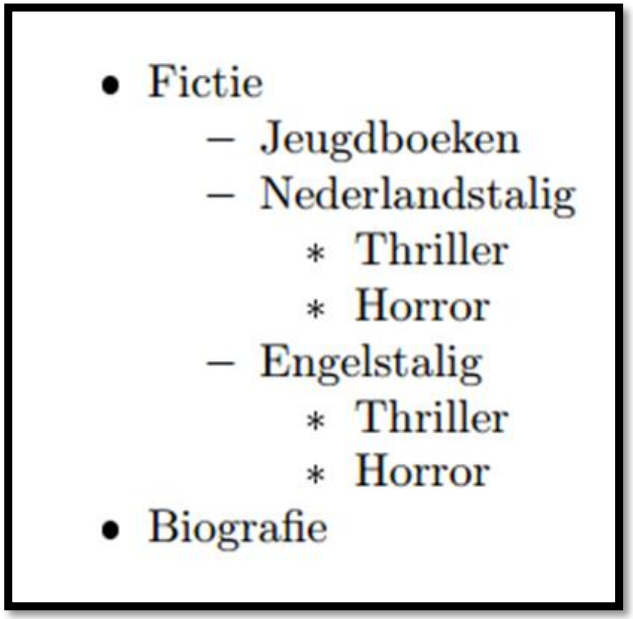

# Activity Diagram – exercises “De Lijn” (3)

- Stel het activity diagram op voor volgende use case

| **Use Case:** | Aankoop abonnement | |
|---|---|---|
| **Triggering event:** | Een klant koopt aan het loket een nieuw abonnement. | |
| **Beschrijving:** | Een klant arriveert aan een loket van De Lijn en koopt bij een bediende een nieuw abonnement aan. De gegevens van de klant en het abonnement moeten opgeslagen worden in het systeem. Het type abonnement dat wordt aangekocht, is afhankelijk van de leeftijd van de klant. | |
| **Workflow:** | *Actor* | *Systeem* |
| | 1. Ingeven van de persoonlijke gegevens van de klant.<br><br>2. Bezorg abonnement aan klant. | 1a. Bewaren van de persoonlijke gegevens in het systeem.<br>1b. Aanmaak nieuw abonnement (incl. bepalen van type en prijs).<br>1c. Afdrukken van het abonnement. |

# Activity Diagram – exercises “De Bib” (1)

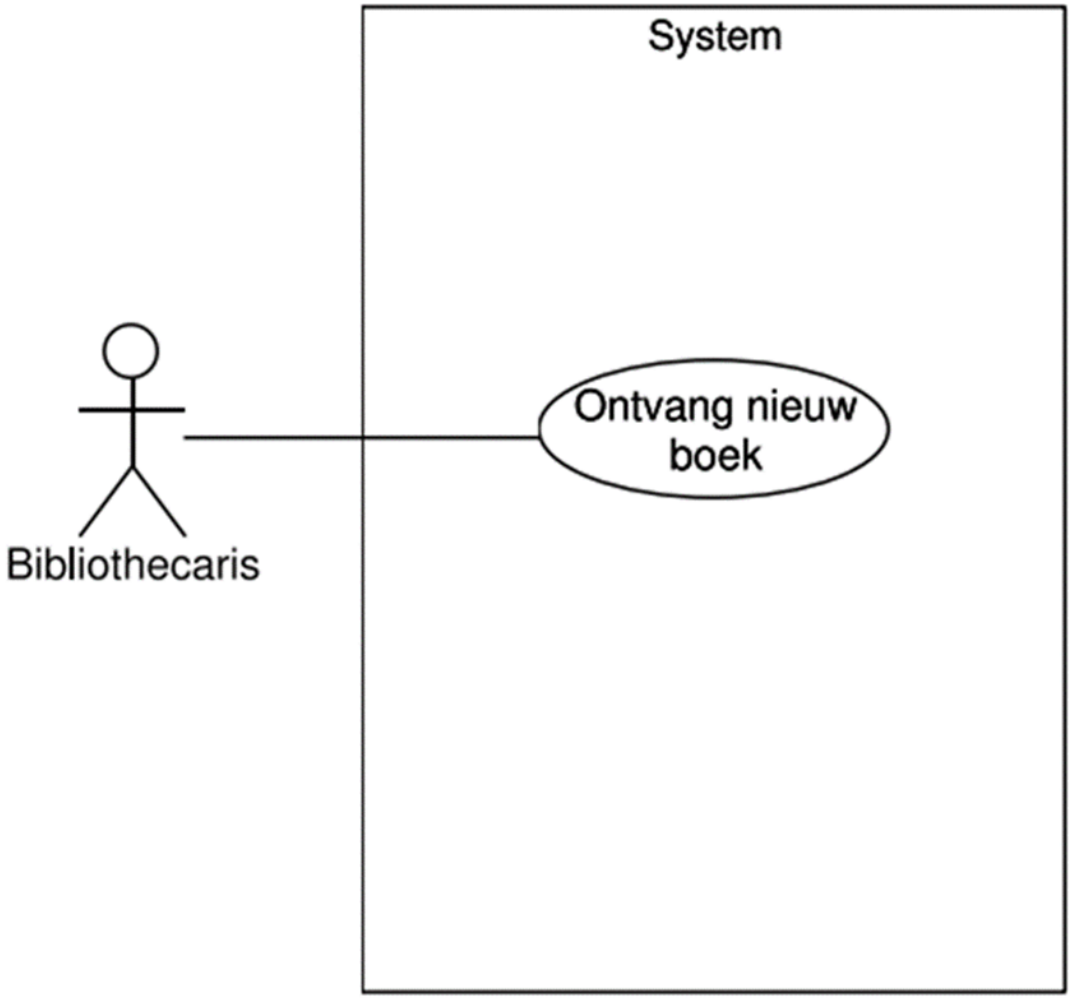
- Een bibliotheek wil een nieuw informatiesysteem introduceren. Binnen het informatiesysteem wordt het begrip “boek” gebruikt voor boeken met eenzelfde ISBN-nummer. Het is echter mogelijk dat de bibliotheek over meerdere fysieke exemplaren van een bepaald boek beschikt.
- Elk exemplaar krijgt een uniek plaatskenmerk toegekend. Per boek wordt de titel en het ISBN-nummer bijgehouden.
- Elk boek in de bibliotheek wordt toegekend aan één (en slechts één) categorie. Voorbeelden van categorieën zijn jeugdboeken, thriller en biografie. Elke categorie heeft een naam en een alfanumeriek ID. Elke categorie kan 0, 1 of meerdere subcategorieën bevatten zodat een boomstructuur ontstaat ➔➔➔



# Activity Diagram – exercises “De Bib” (2)

- De bibliothecaris kan bestellingen plaatsen om nieuwe boeken aan te kopen. Voor elke bestelling wordt het ordernummer, de datum van bestelling en de datum van levering bijgehouden. Er wordt momenteel niet bijgehouden op welk boek een bepaalde bestelling betrekking heeft.
- Op het moment dat er een bestelling wordt geplaatst, zal de bibliothecaris de gegevens van het boek ingeven in het systeem. Dit betekent dat er bij het ontvangen van een bestelling enkel moet geregistreerd worden dat er een nieuw exemplaar van het boek werd ontvangen. Het kan dus voorkomen dat er gegevens van boeken in het systeem voorkomen, waarvan de bibliotheek (nog) geen exemplaren bezit.
- Stel het activity diagram op voor aangegeven use case (volgende slide)

# Activity Diagram – exercises “De Bib” (3)



# Activity Diagram – exercises “De Bib” (4)

| **Use Case:** | Ontvang nieuw boek | |
|---|---|---|
| **Triggering event:** | Een nieuw aangekocht boek arriveert. | |
| **Beschrijving:** | De bibliothecaris moet registeren dat een bepaalde bestelling werd ontvangen, en dat er een nieuw examplaar van een bepaald boek is toegevoegd aan het aanbod van de bibliotheek. De gegevens van boeken die werden besteld, zijn reeds aanwezig in het systeem. | |
| **Workflow:** | *Actor* | *Systeem* |
| | 1. Ontvangst van boek, inclusief ontvangstbewijs met vermelding van bestelcode. | |
| | 2. Aanpassen van datum van levering van bestelling. | 2. Opzoeken van bestelling in database en updaten van leveringsdatum. |
| | 3. Registratie van een nieuw exemplaar van een bestaand boek a.d.h.v. het ISBN-nummer van het boek. Het nieuwe gekozen exemplaarnummer wordt eveneens ingegeven. | 3. Opzoeken van boekgegevens (ISBN-nummer) in het systeem, en toevoegen van nieuw exemplaar. |
| | 4. Fysiek boekexemplaar voorzien van sticker met plaatskenmerk. | |
| | 5. Plaatsen van exemplaar in rekken. | |

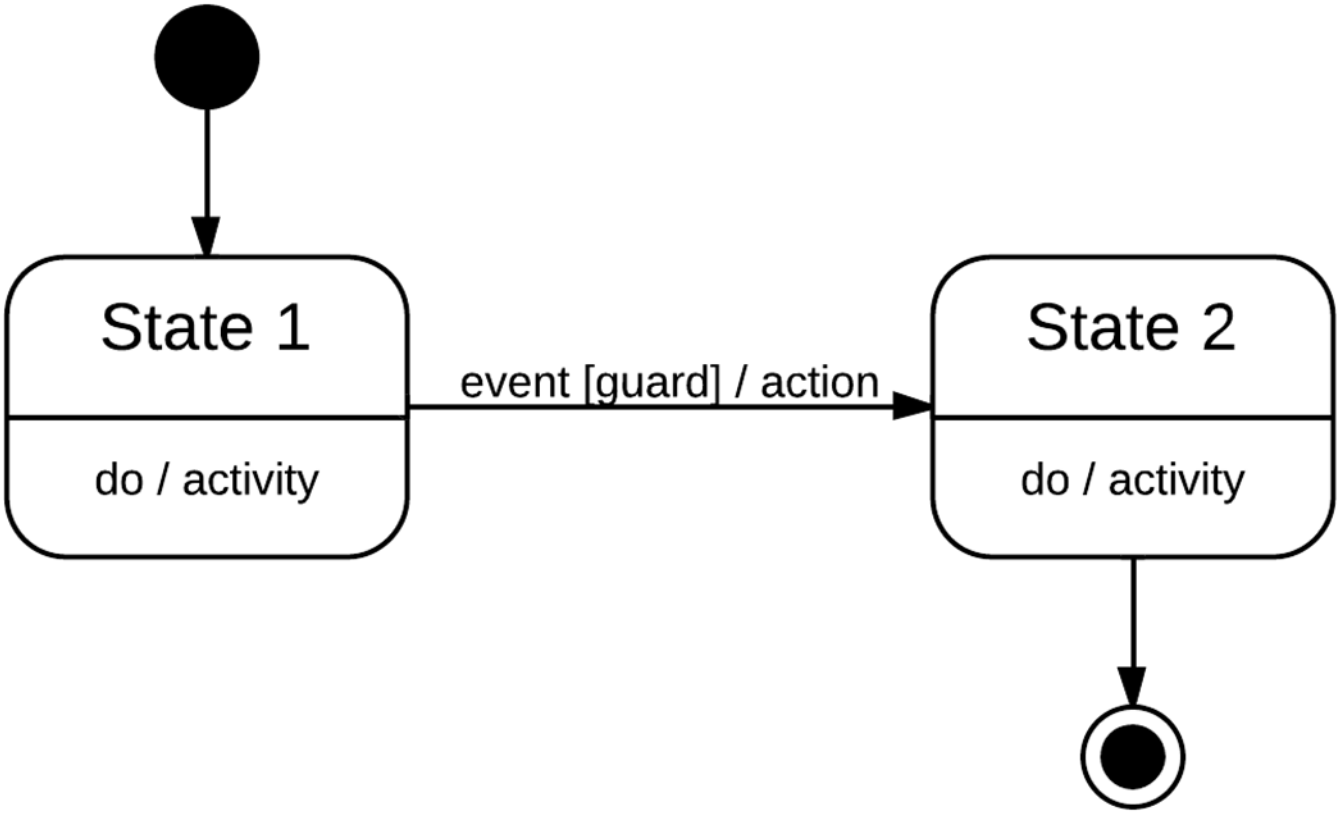
# Topic Overview

- 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

# Behavioral Requirements Models

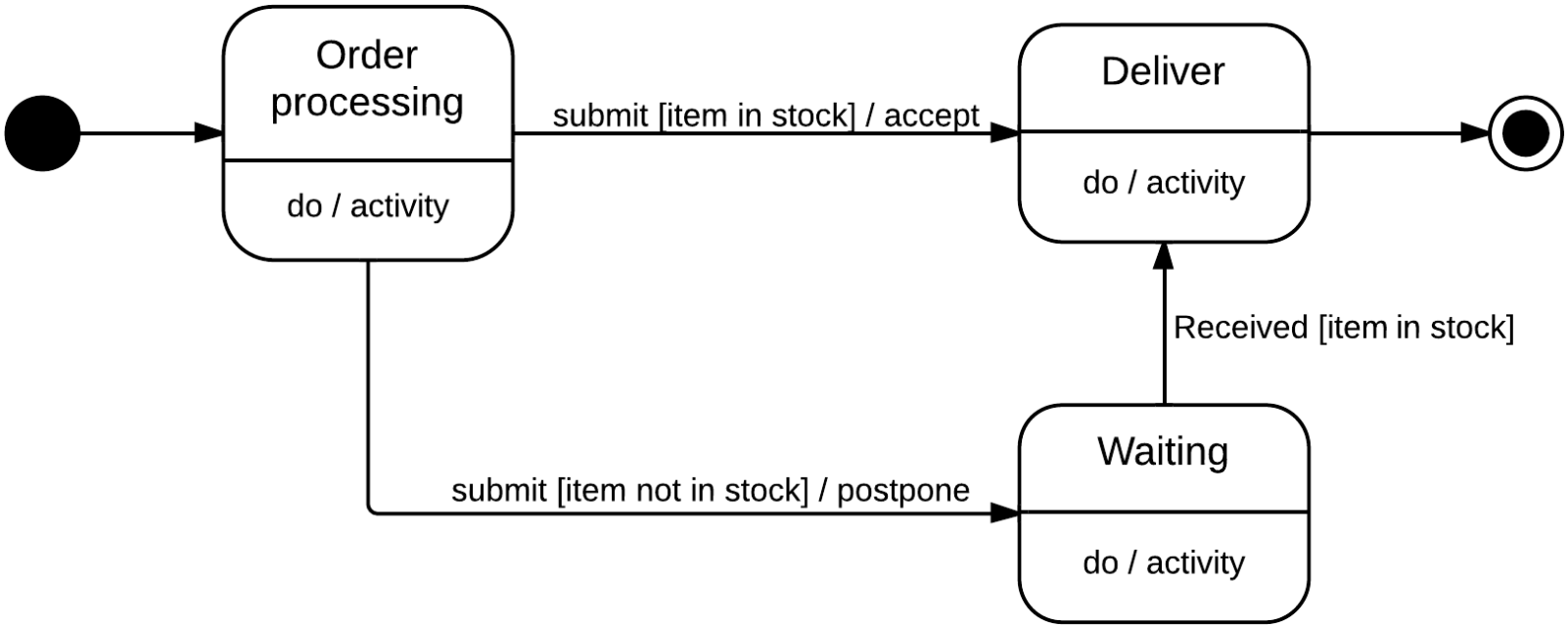
- For modeling the dynamic behavior of a system
- Focus is on system states and events that can change that state
- Behavioral models
- UML State diagrams
  - Based on principles of finite state machines
  - Modeling elements: state, start- and end states, state transition, concurrency

# State Transition Diagram (STD)

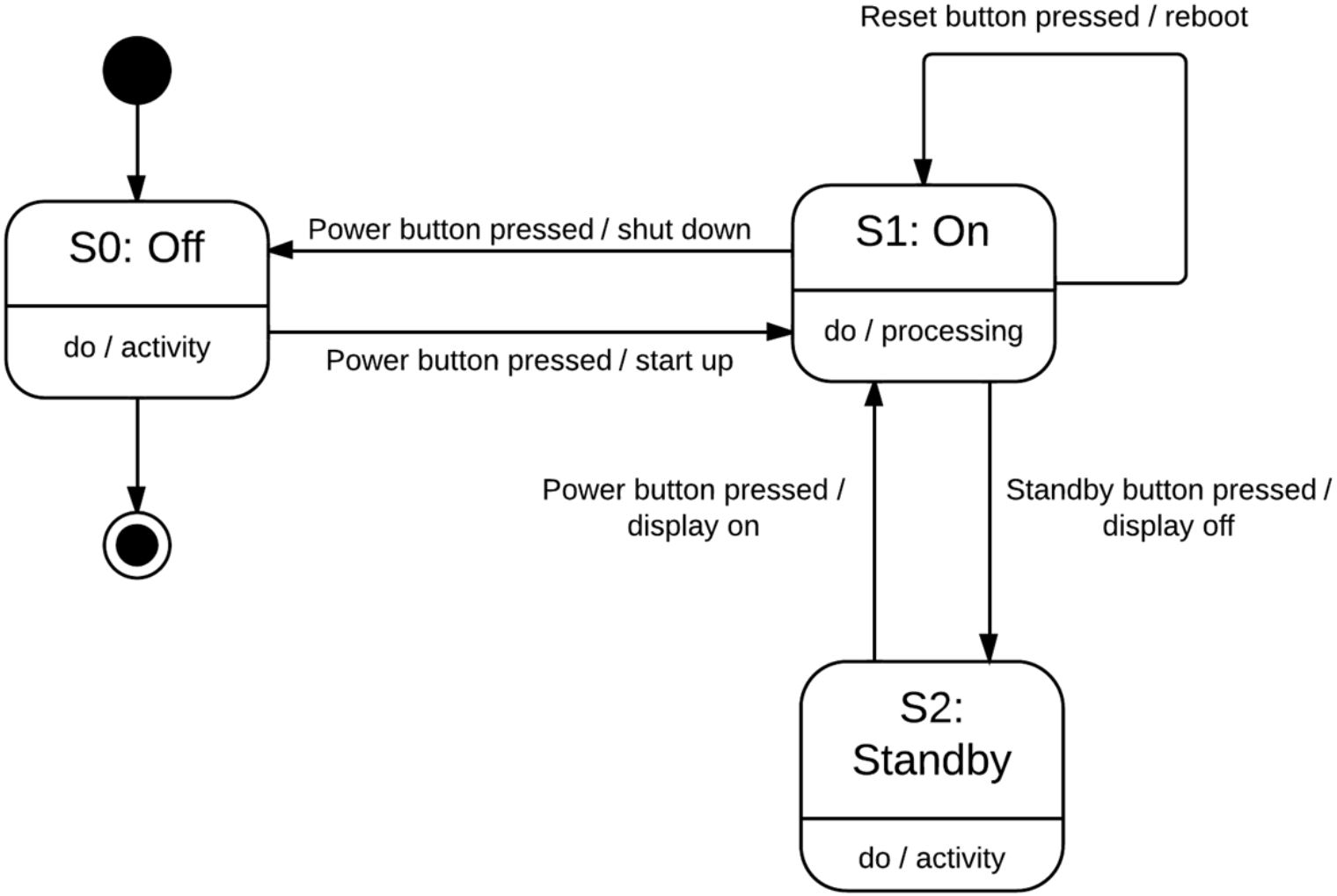


# STD - Guards

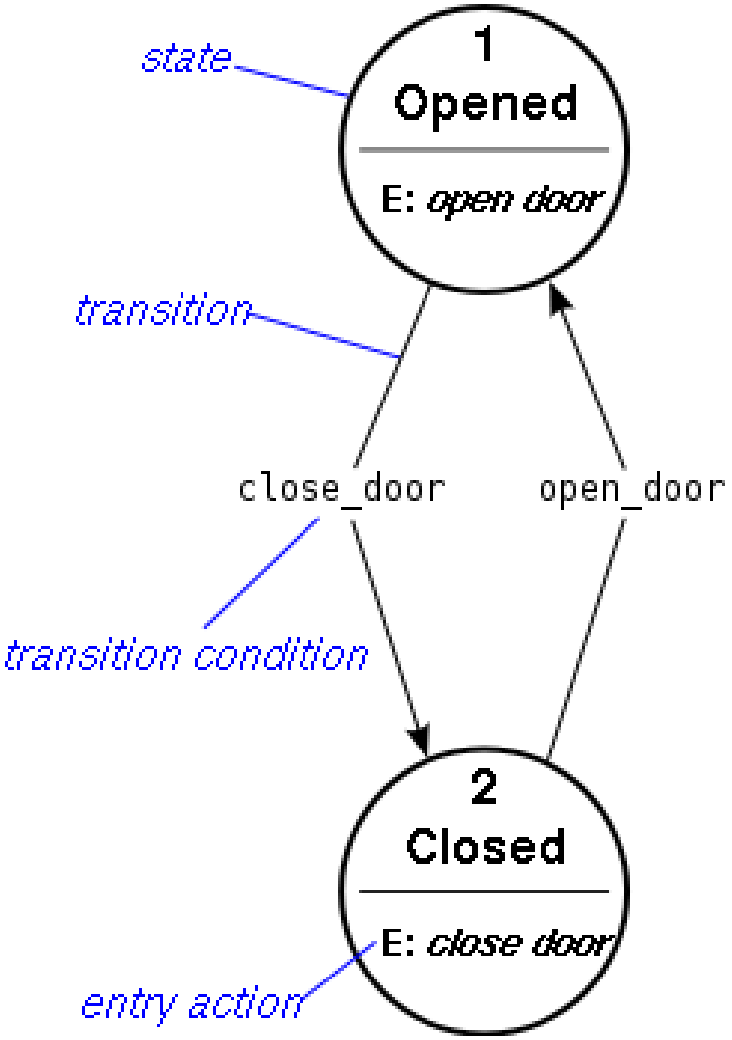
- A transition is only executed when its guard is true



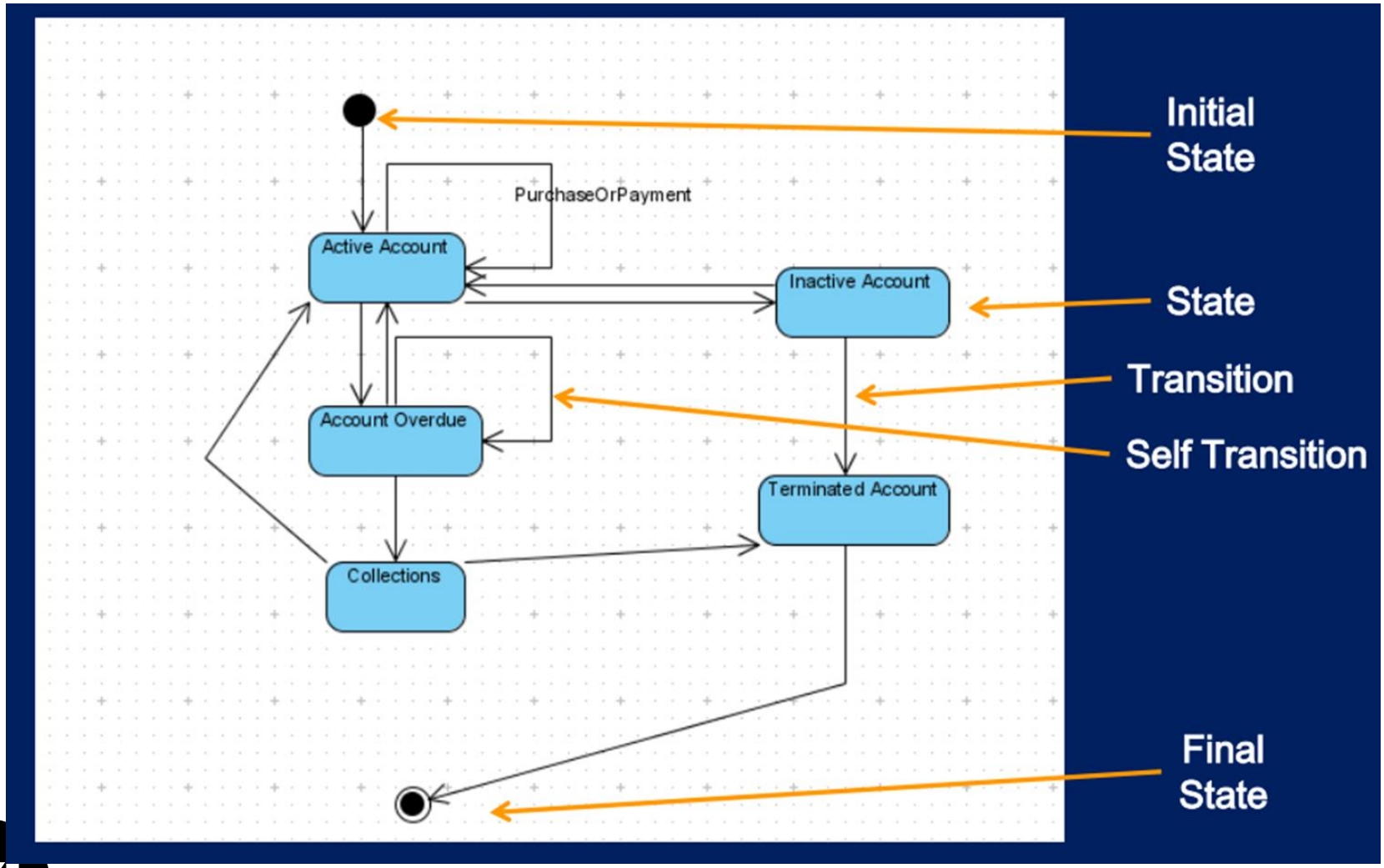
# STD – example laptop



# STD – example open/close door



# STD – example account



# STD - how to draw?

- In 5 steps
  - Define States
  - Describe States (for others to understand)
  - Draw Transitions
  - Define Transition Triggers
  - Define Guard Conditions

# STD - pro's and cons

- Describes the behaviour of a system
- All possible states a system can get into
- State changes as result of events
- Actions are short effects of transitions

☺ Suitable for modelling interfaces

☺ Suitable for simulation and testing

☹ May be difficult to read for outsiders

# STD - Case - web site

- A web site has a homepage, by choosing the button contact ‘ contact information’ is provided. By choosing the button ‘services’ an overview is provided of the services. The ‘services’ page also has a hyperlink to the contact information page. The home button will bring you back to the homepage.

- Draw the state transition diagram
- You are allowed to ask questions!

# STD – exercises – digital pet program

- You're creating a digital pet program. What happens to the pet when he receives different stimuli is determined by the state he's in, so you decide to model the digital pet with a state diagram.
- The behavior of the digital pet program is as follows:
  - When the pet is turned on, it starts out happy
  - If the pet is happy and receives punishment, then he becomes sad
  - If the pet is sad and receives praise, it becomes happy
  - If the pet is sad and receives punishment, it is heart-broken

# STD – exercises – fuel pump

- Model the behavior of a fuel pump controller.
  - User can buy fuel after inserting a credit card, which is read and validated by the controller.
  - Then the user takes the hose out of the holster, and pushes the nozzle trigger, to fuel his car.
  - When the nozzle is off, the fuel flow is stopped and the price is charged on the credit card
  - If invalid card or timeout the system returns to the initial waiting state.

# 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**

# Sequence diagrams

- Tutorials pluralsight and youtube

# Key Learning Points (1)

- Knowing models and their properties
- Advantages of using requirements models
- Knowing and understanding 2 types of goal decomposition
- Drawing, knowing and understanding system use case diagrams and use case specifications/descriptions
- 3 Perspectives on requirements
  - Next slide

# Key Learning Points (2)

- 3 Perspectives on requirements (continued)
  - Data perspective
    - Knowing and understanding entity relationship diagrams
    - Knowing and understanding UML class diagrams
  - Functional perspective
    - Knowing and understanding data flow diagrams
    - Drawing, knowing and understanding UML activity diagrams
  - Behavioral perspective
    - Drawing, knowing and understanding UML state charts

# Key Learning Points (3)

- Knowing and understanding sequence diagrams
- Be able to compare the models
- Knowing and understanding the pro’s and con’s of the models

# Questions & answers