Requirements Engineering: System Models

CECS 590

Assignment

- Your first individual assignment is a literature review, due on Feb 15.

Requirements Engineering – Outline

- WHY do we need Requirements Engineering and what is it?
- Principles: Definitions, process, roles, problem/solution view, artifact orientation
- **System Models: Decomposition and abstraction, system views**
- Frameworks: What reference structures can I use for requirements?
- Business Case Analysis: Why are we building this system?
- Stakeholders: Who are the people to talk to about requirements?
- Goals and Constraints: What are the major objectives for the system?
- System Vision: What exactly do we want to achieve?
- Domain Models: What are the surrounding systems ours interacts with?
- Usage Models: How will the system interact with the user?
- Software quality models: How to determine the quality characteristics?
- Quality requirements: How to specify which qualities need to be met?
- Process requirements: How to specify constraints for development?
- Towards a system specification: How to hand over to design?
- Quality assurance: How to ensure that RE is done in a good way?
- Change management: How to evolve requirements?

Recap time!

- Which roles do we have in an RE team?
- Which interfaces to other teams do we have?
- Which elements do we need to define for a process model?
- What is problem-orientation vs. solution-orientation?
- What is activity-orientation vs. artifact-orientation?

Outline: System Models

- Why and what? Definition
- Refinement and Decomposition
- Abstraction Levels
- Modeling Views / View types
- Rationale
- Traceability

K Rayker, stock.xchng

System Models: Why?

- Requirements for a system demand characteristics of the system under development
- Defining characteristics requires a clear concept and understanding of „system“
- If characteristics shall be formalized (logically) then a mathematical model is necessary

→ Basis for structuring and formalising reqs.

System models: General idea

System models

- Conceptual models for describing systems by means of modeling views.
- Determine what requirements describe:
  - What is a system? What are the essential characteristics, views and structures?
  - Which parts/ what structure does a system have?
  - What aspects do we have to consider?

Delimitation: Artifact models

- Built (ideally) on the basis of a system model:

→ Concept model (Content Model)

→ Structure of the concept model (Structure Model)

Meaning of system models

- Give solid standard for terminology and concepts
- Prescribe modeling concepts
- Allow for precise specification of usage behavior
- Allow for specification of the context behavior
- Detect inconsistencies in requirements and design
- Allow for a discussion of the completeness of a requirements specification
- Support the classification of requirements
- Are the interface to design
- Are the basis for code generation

Examples for system models and modeling languages

- Plethora of system models and modeling languages available and widely spread in practice

→ Software: programming models (example: object orientation, OO models, UML), software architectures

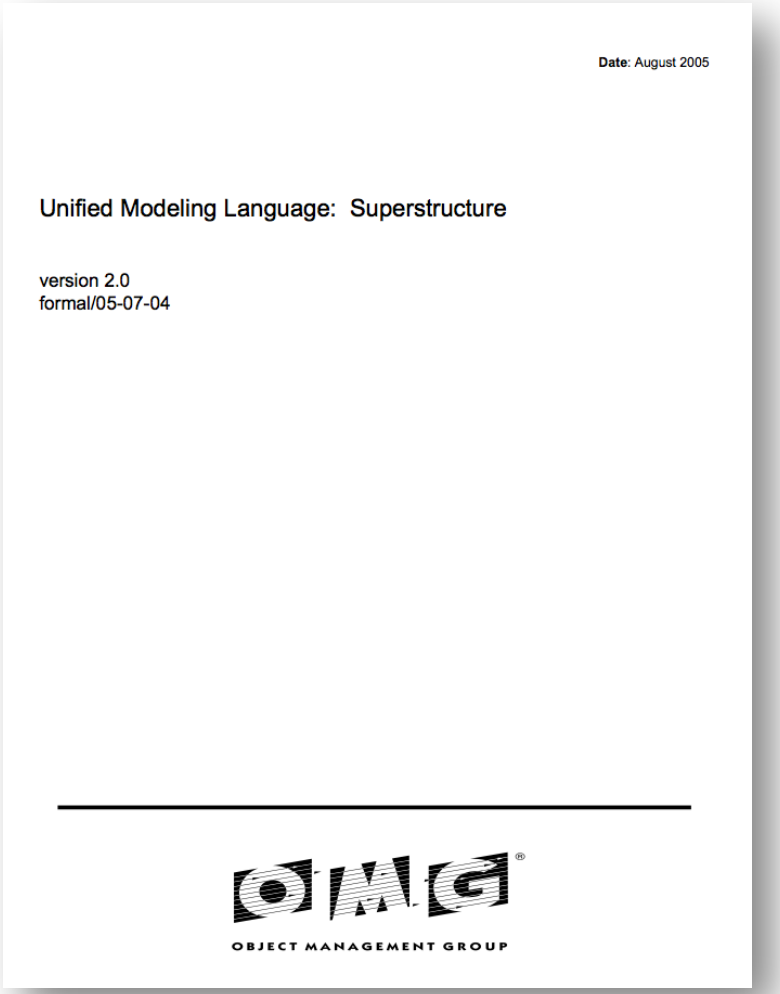
→ Embedded systems: control engineering (MATLAB simulink models), discrete event systems

→ Business information systems: Service orientiented architectures – SOA

Problem: Modeling languages often lack semantics!

Example: Unified Modeling Language (UML)

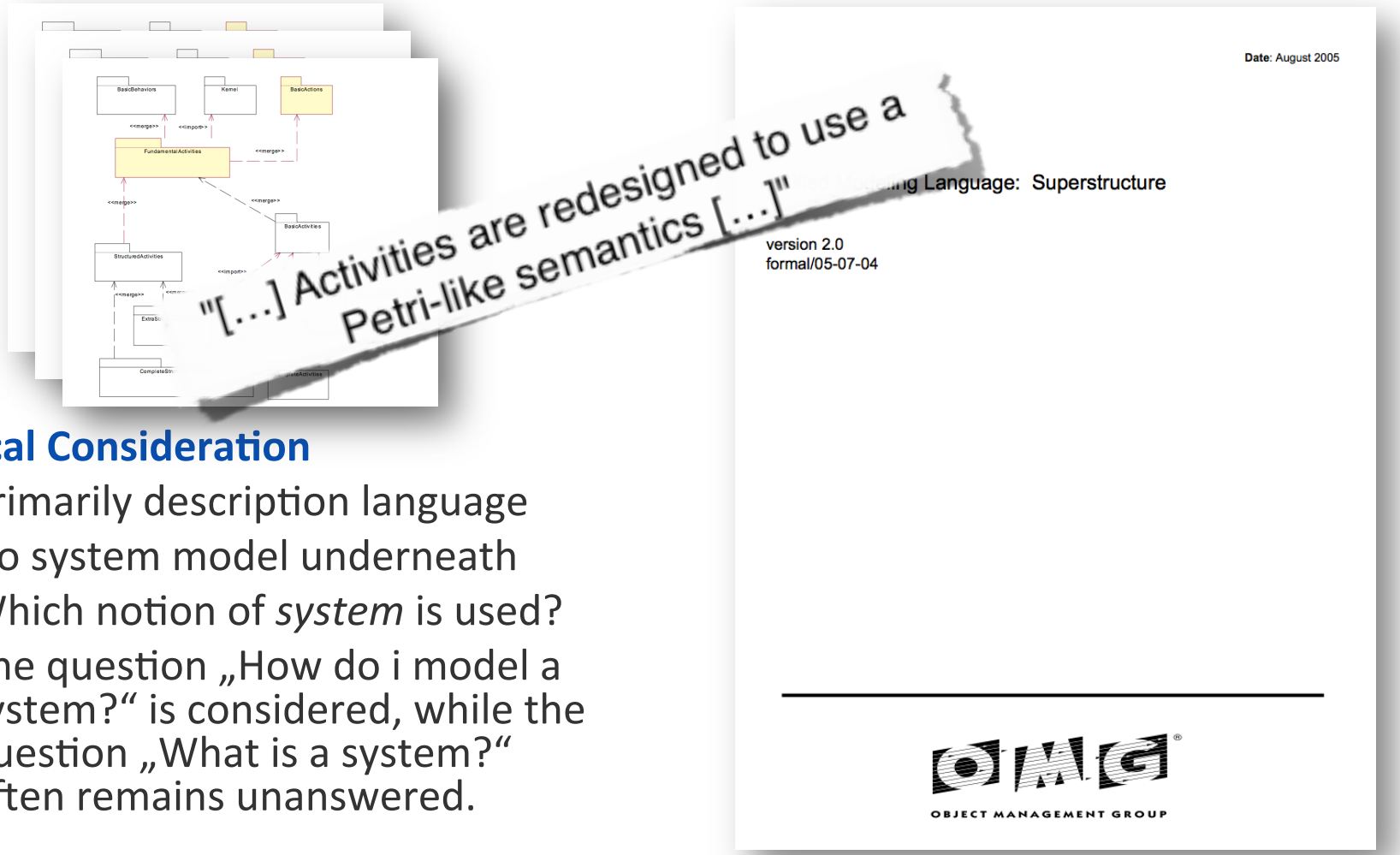
- Definition of abstract syntax as well as number of description languages



Meaning and usage

- Used for modeling system behavior, system structure, ...
- Primarily: Pragmatic and practical applicability of description languages

Example: Unified Modeling Language (UML)



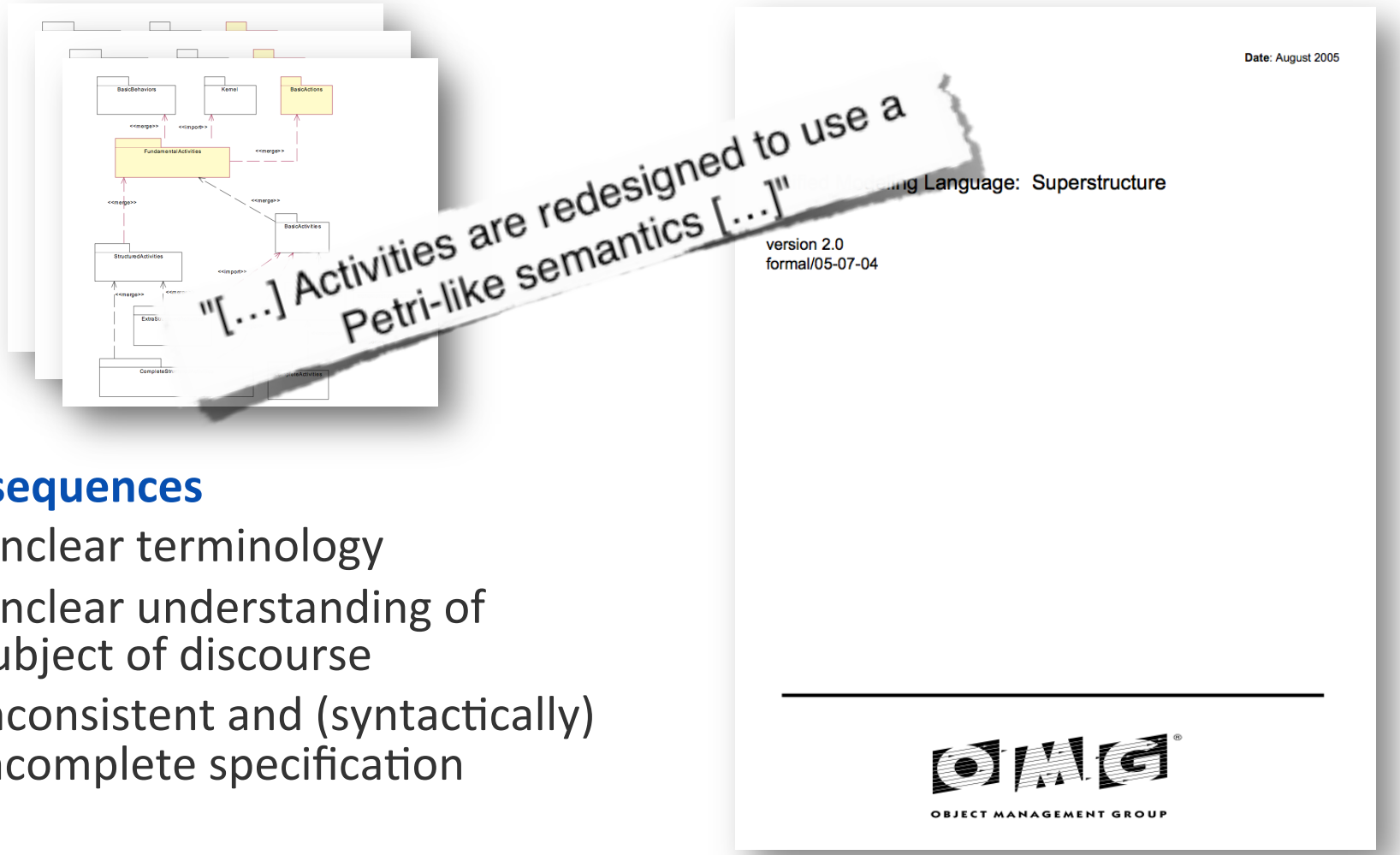
Critical Consideration

- Primarily description language
- No system model underneath

→ Which notion of *system* is used?

→ The question „How do i model a system?“ is considered, while the question „What is a system?“ often remains unanswered.

Example: Unified Modeling Language (UML)



Consequences

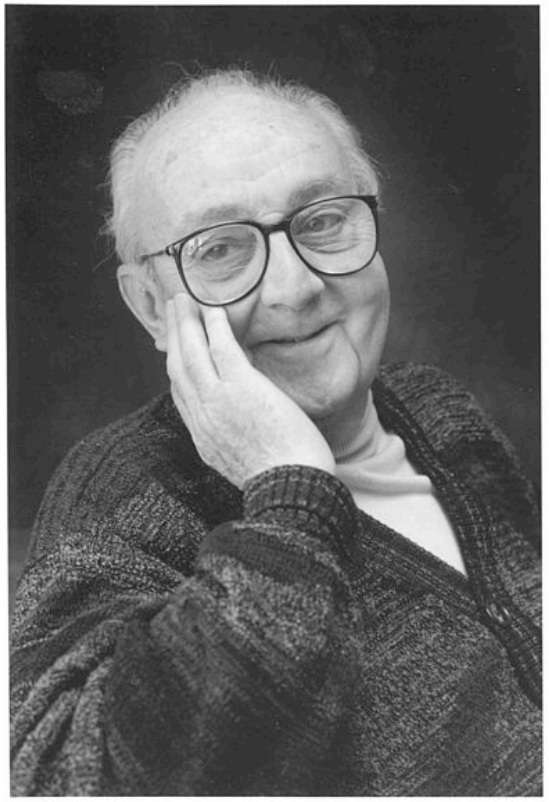
- Unclear terminology
- Unclear understanding of subject of discourse
- Inconsistent and (syntactically) incomplete specification

Discussion: Models

"Essentially,
all models are wrong,
but some are useful."

- George E. P. Box

Why is that?



Meaning of formal system models

Modeling theories and system models

- Clearly define notion and meaning of *system*...
- ... on the basis of precise models, e.g. mathematical models

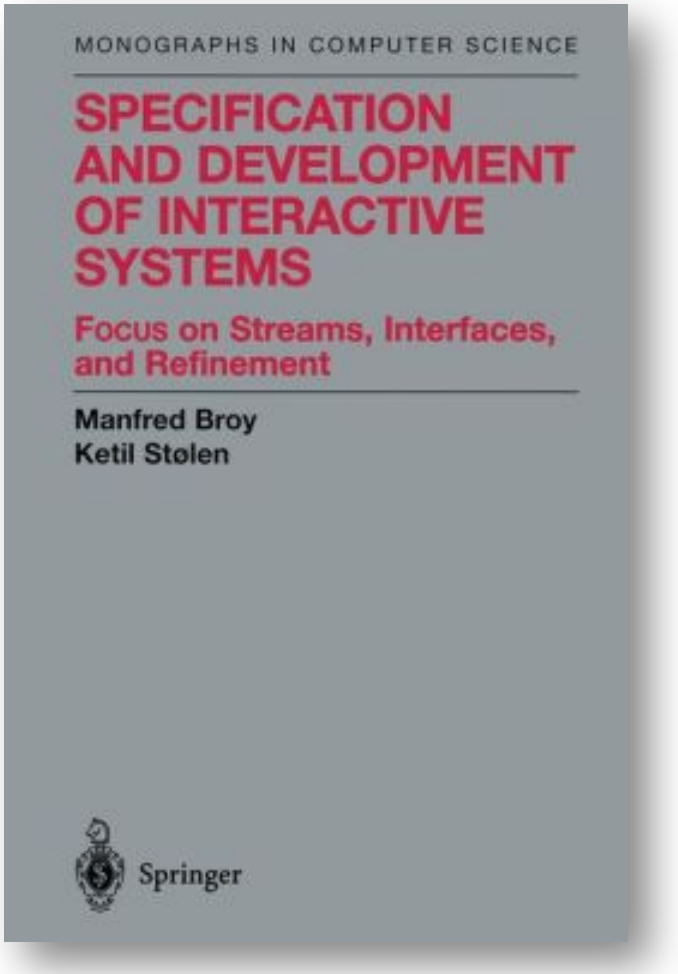
Meaning for RE

- Requirements can demand system characteristics, e.g. via logical predicates
- For the phrasing of system characteristics we need a clear notion of a system (founded on a mathematical model)

Building on that

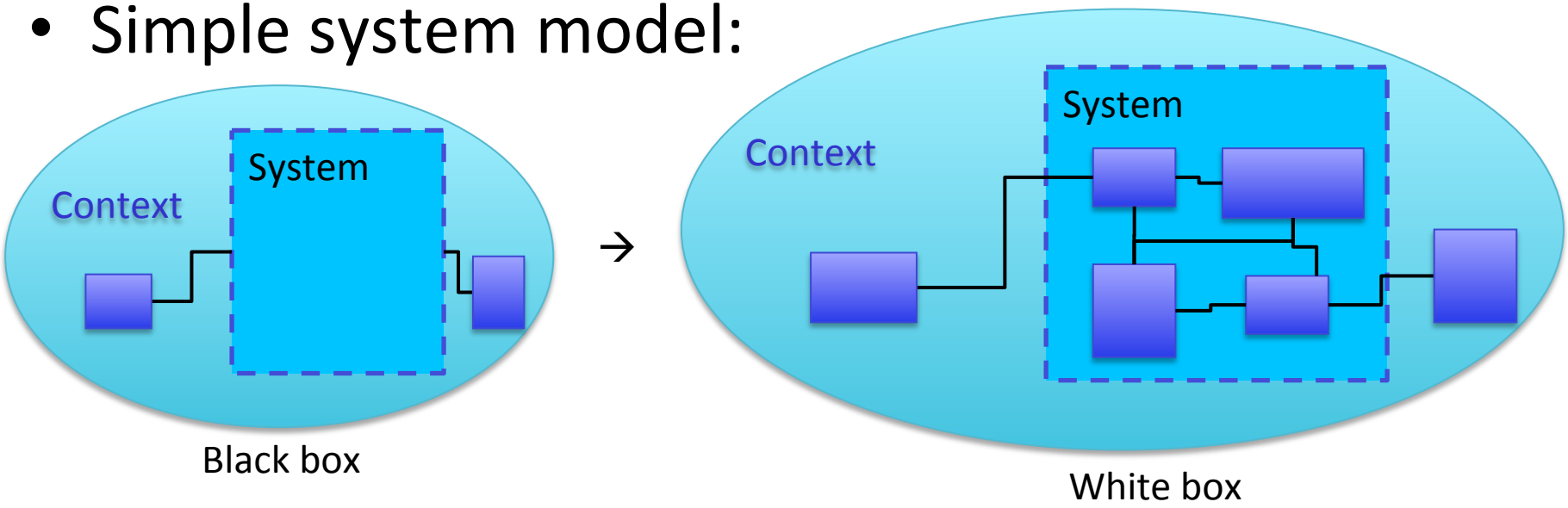
- Development of encompassing concept models (and structuring via artifact models)

→ Seamless, systematic specification of system characteristics („seamless modelling“)



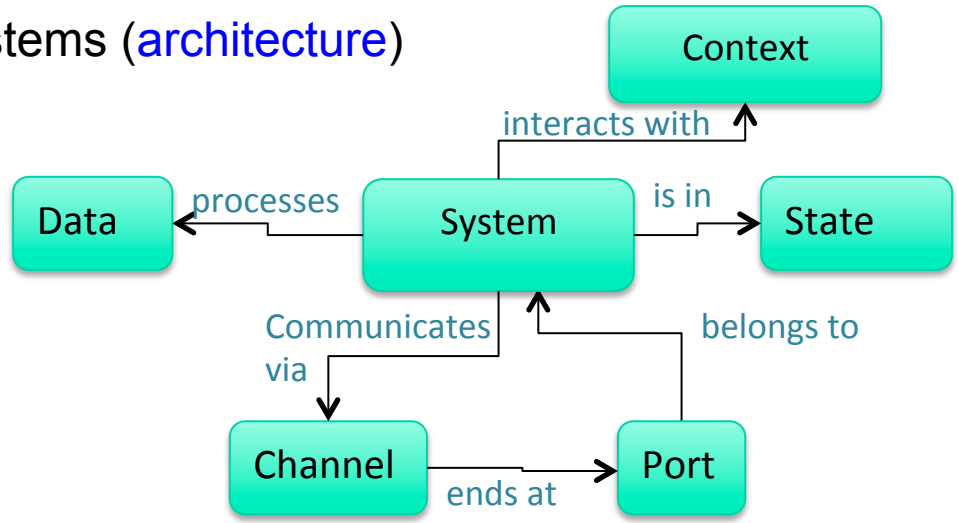
A (discrete) System Model

- Def.: A system model is a conceptual (generic) model to describe systems. It describes
  - The result of a conceptualisation
  - The essential characteristics, views, and structure
- Simple system model:



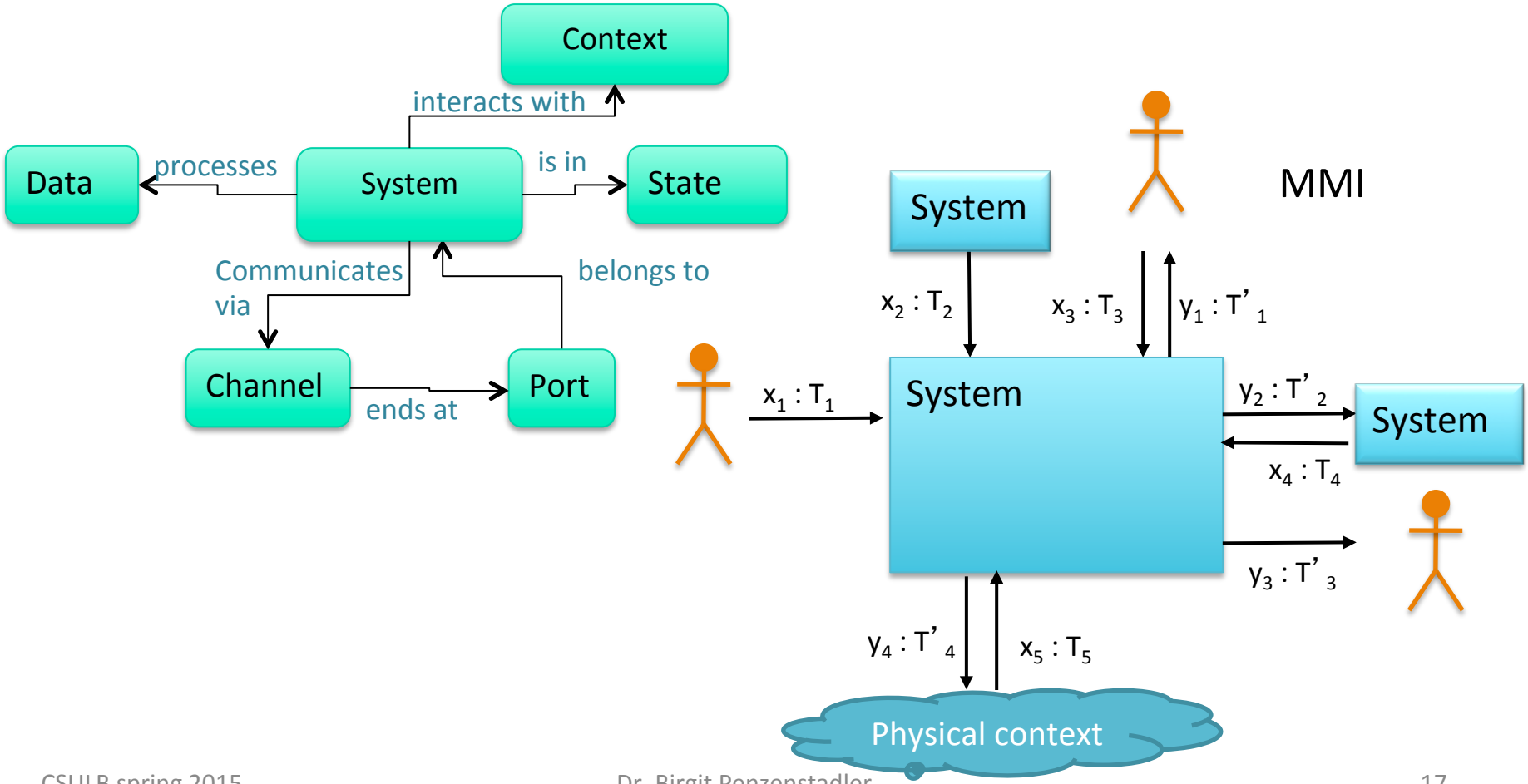
A (discrete) System Model

- A System has
  - A system border, that determines, what is part of the system and what is outside of the system (the context)
  - An interface (determined by the system border), that determines,
    - Which forms of interaction between a system and ist environment are possible (static/syntactic interface)
    - Which behavior the system displays from the view of the interface (interface behavior, dynamic interface, interaction view)
- An inner structure, given by
  - The decomposition into subsystems (architecture)
  - It's states and state transitions (state view)
- The state view and interaction view are based on a data model
- The views can be documented via suitable models

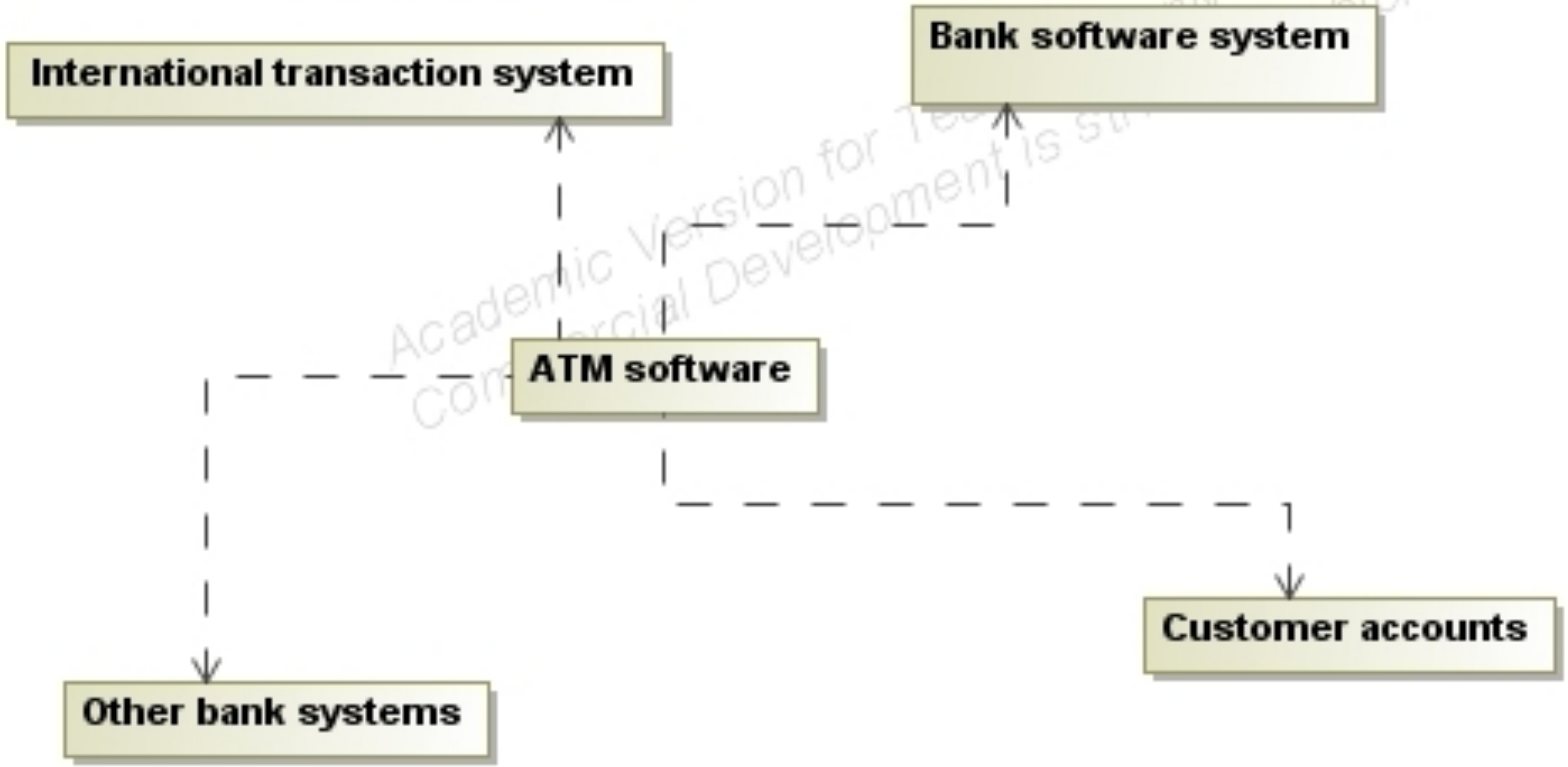


System Models: Modeling Views

- Context view

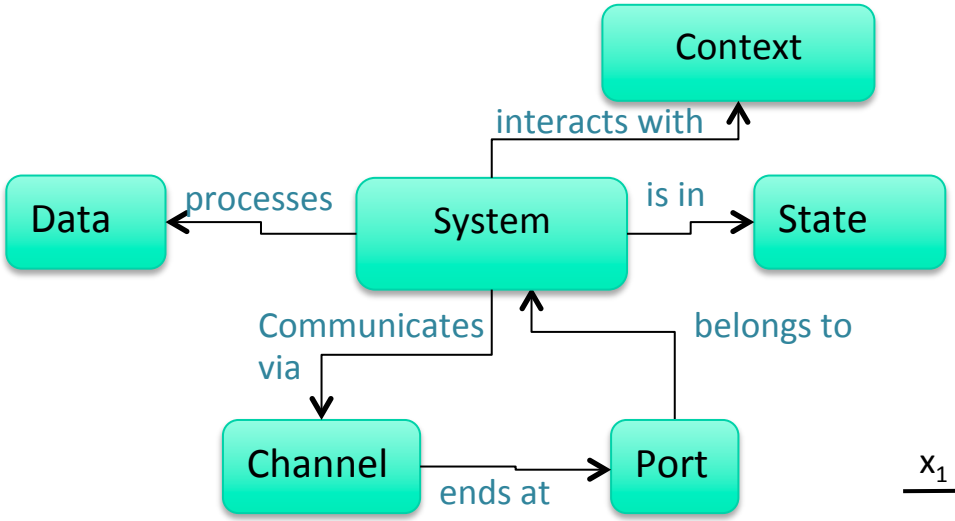


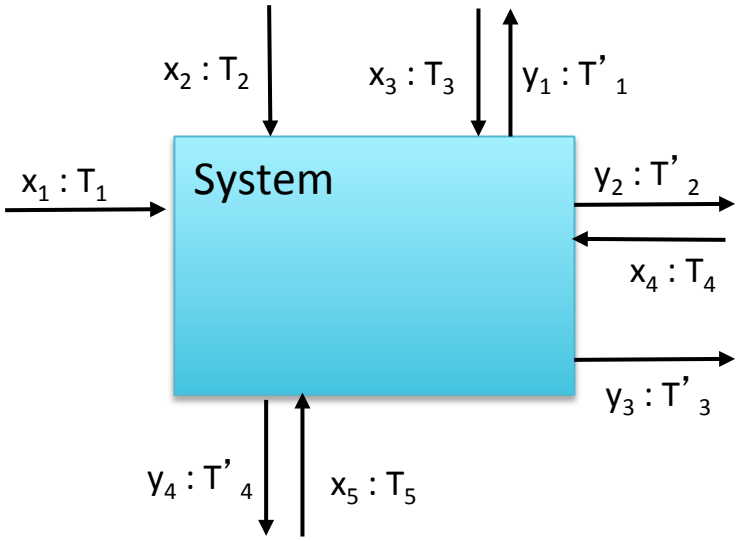
Example: Context view



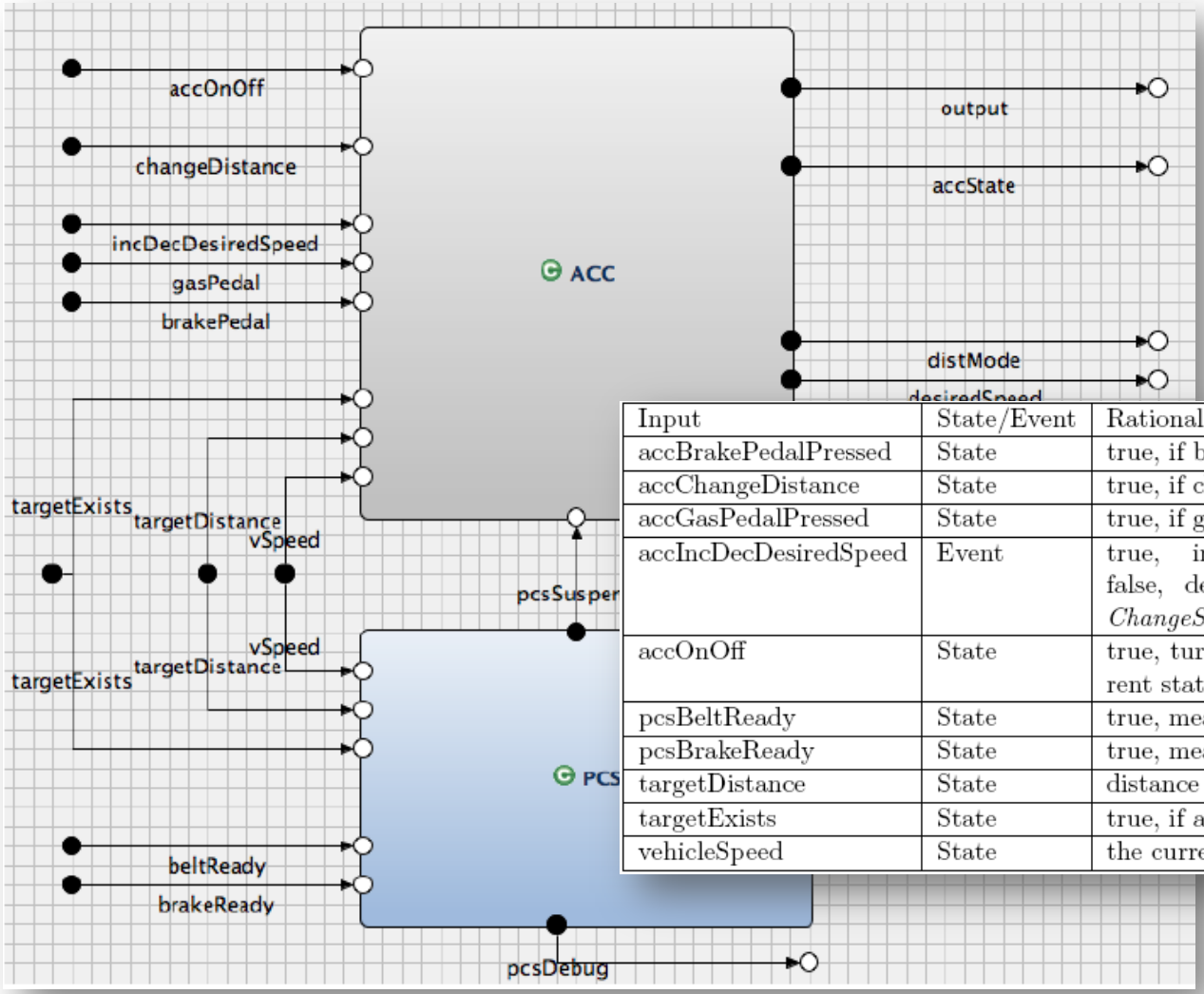
System Models: Modeling Views

- Interface view with syntactic interface





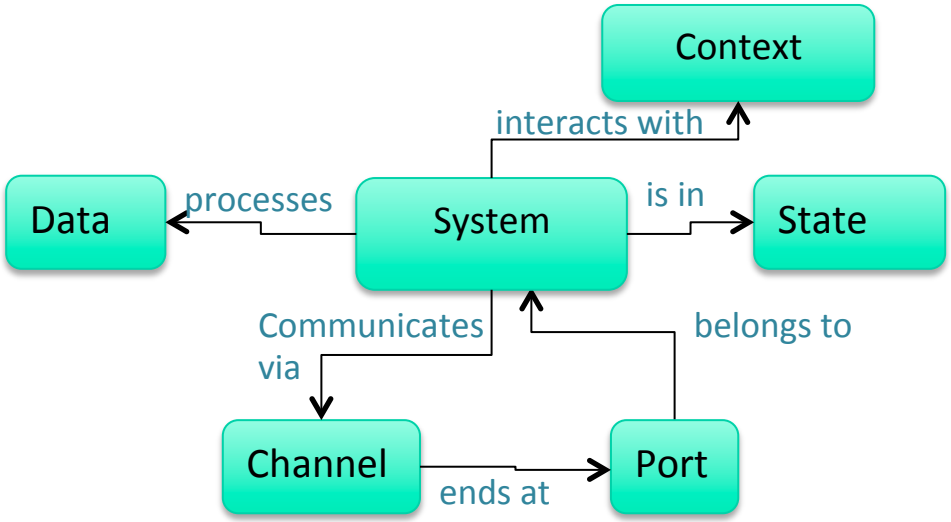
Example: Syntactic interface view

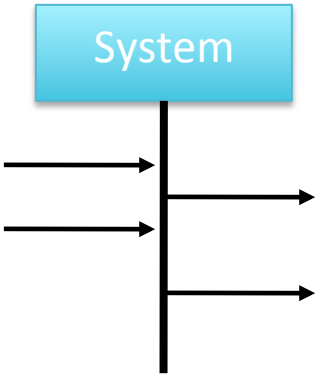


Input	State/Event	Rationale
accBrakePedalPressed	State	true, if brake pedal is pressed
accChangeDistance	State	true, if change distance button is pressed
accGasPedalPressed	State	true, if gas pedal is pressed
accIncDecDesiredSpeed	Event	true, increases the desired speed setting; false, decreases the desired speed setting by $ChangeSpeed_{const}$km/h.
accOnOff	State	true, turns the ACC on and off; false, holds current state
pcsBeltReady	State	true, means belt pre-load device is ready
pcsBrakeReady	State	true, means brake system is ready
targetDistance	State	distance to the followed target vehicle
targetExists	State	true, if a vehicle is detected in front
vehicleSpeed	State	the current vehicle speed

System Models: Modeling Views

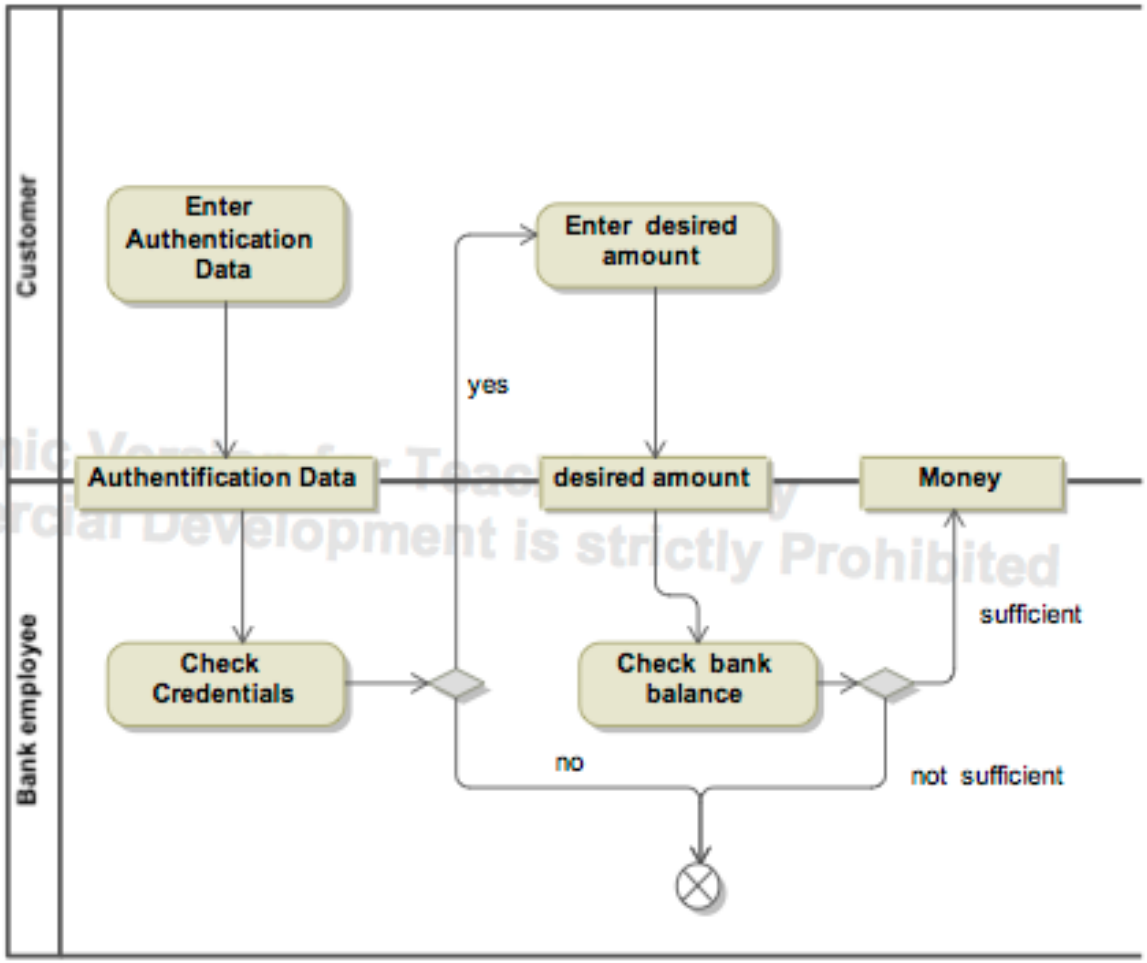
- Behavioral / interaction view





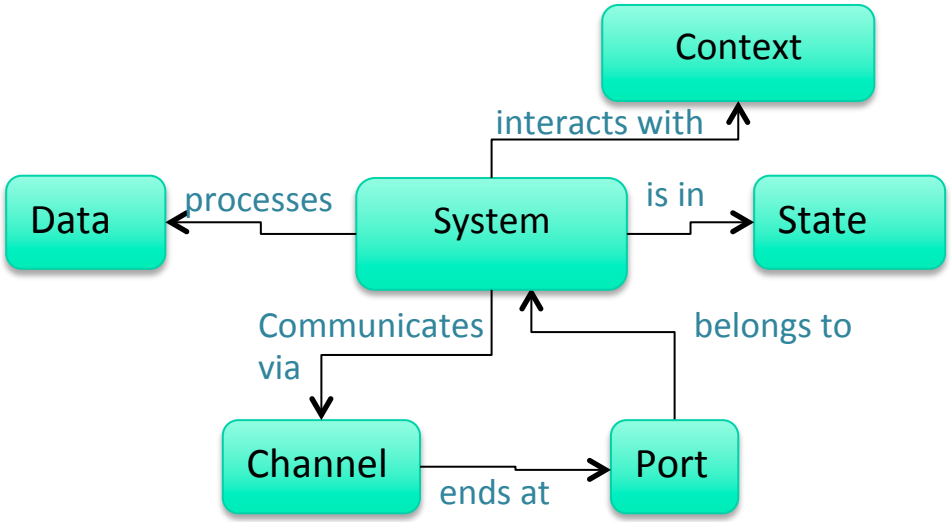
Example

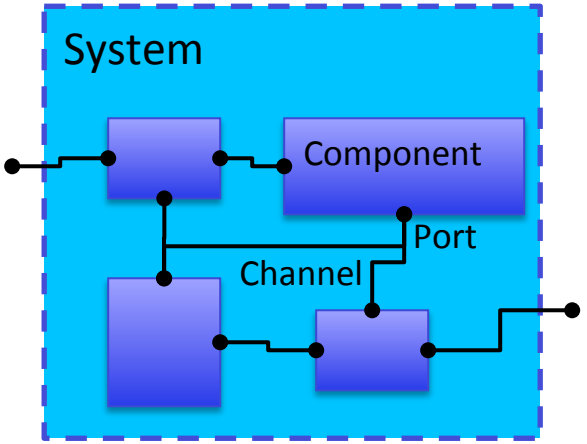
- Behavioural / interaction view



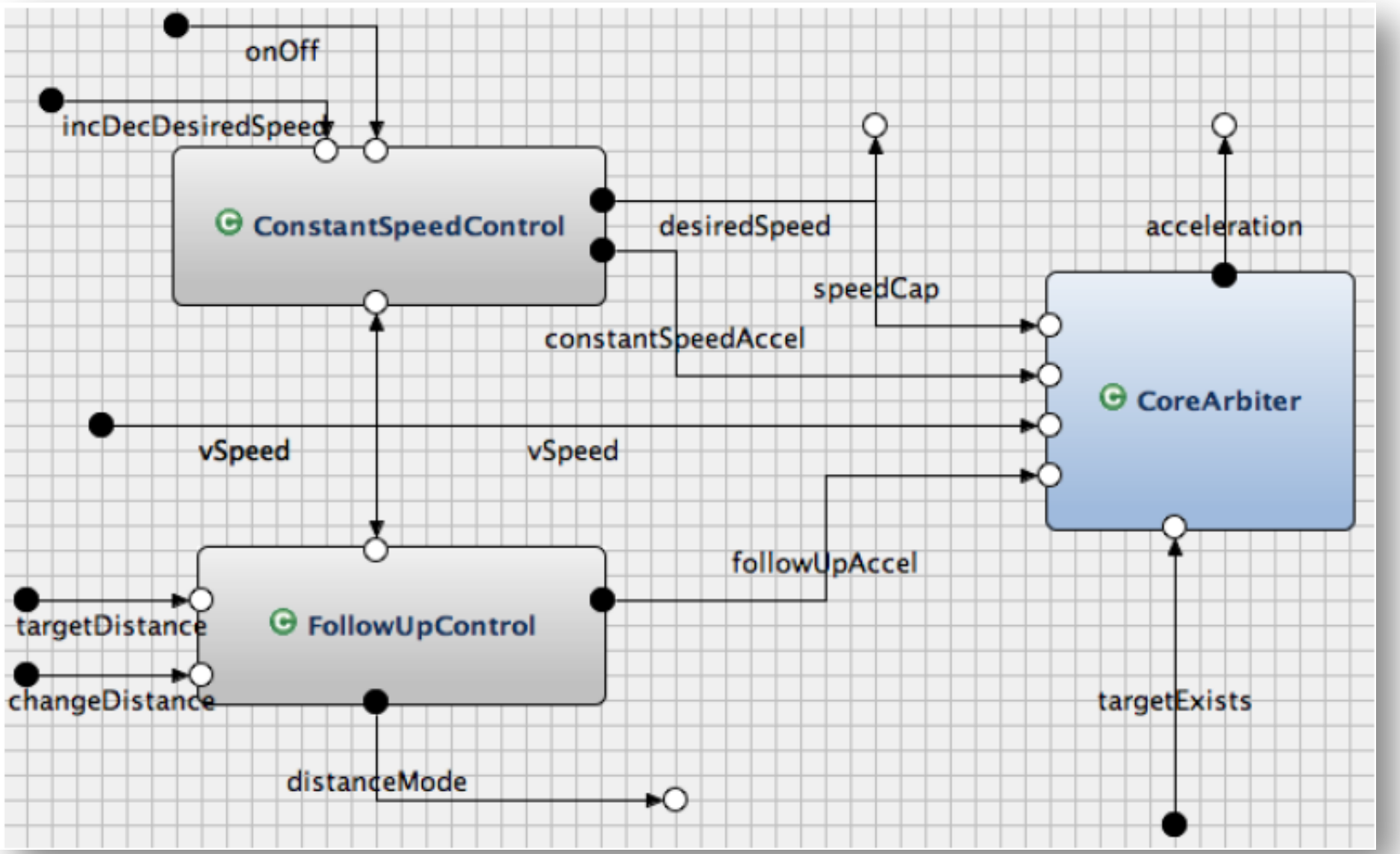
System Models: Modeling Views

- Structural view



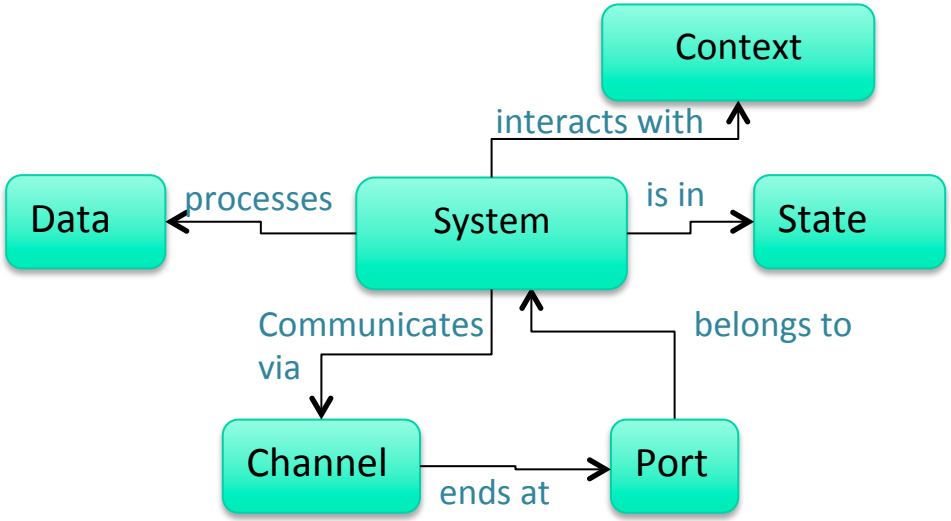


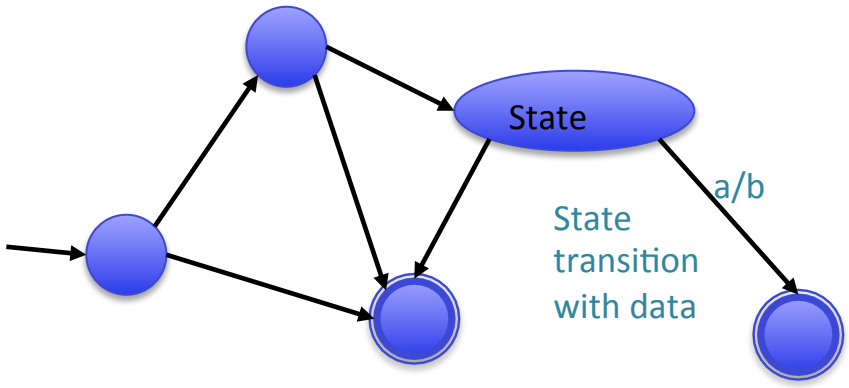
Example: Structural view



System Models: Modeling Views

- State view





Example Behaviour Model

Content and purpose

- Component-wise specification of semantic interfaces / of (internal system) behavior

Example

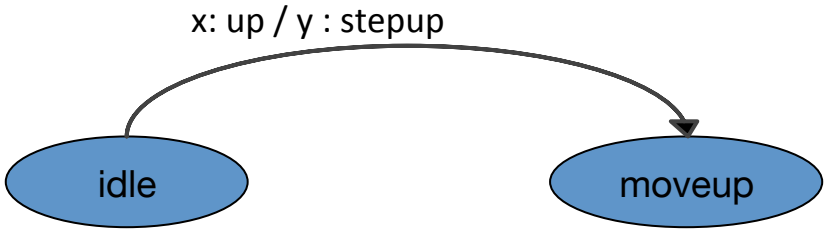
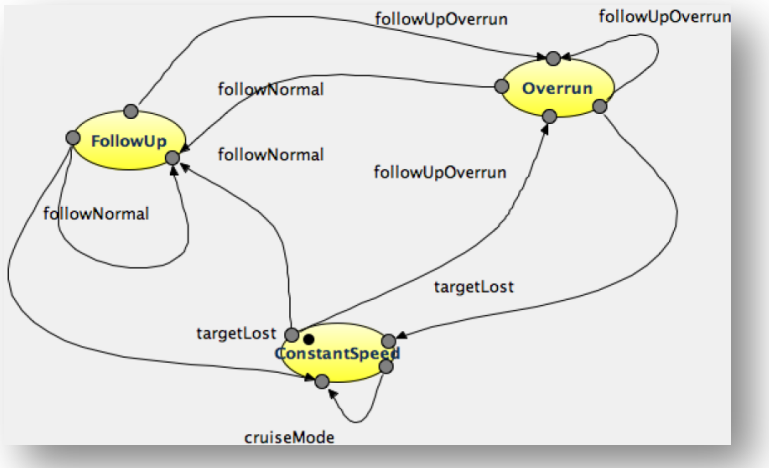


Table Notation

State space: Type State = {idle, movup, ...}

state	x	z	next state	y
idle	up	-	movup	stepup
...	...	-	...	...
...	...	...	...	
...				

Modeling state transition diagram



Exercise: System Model

Envision an online shop and model the following views:

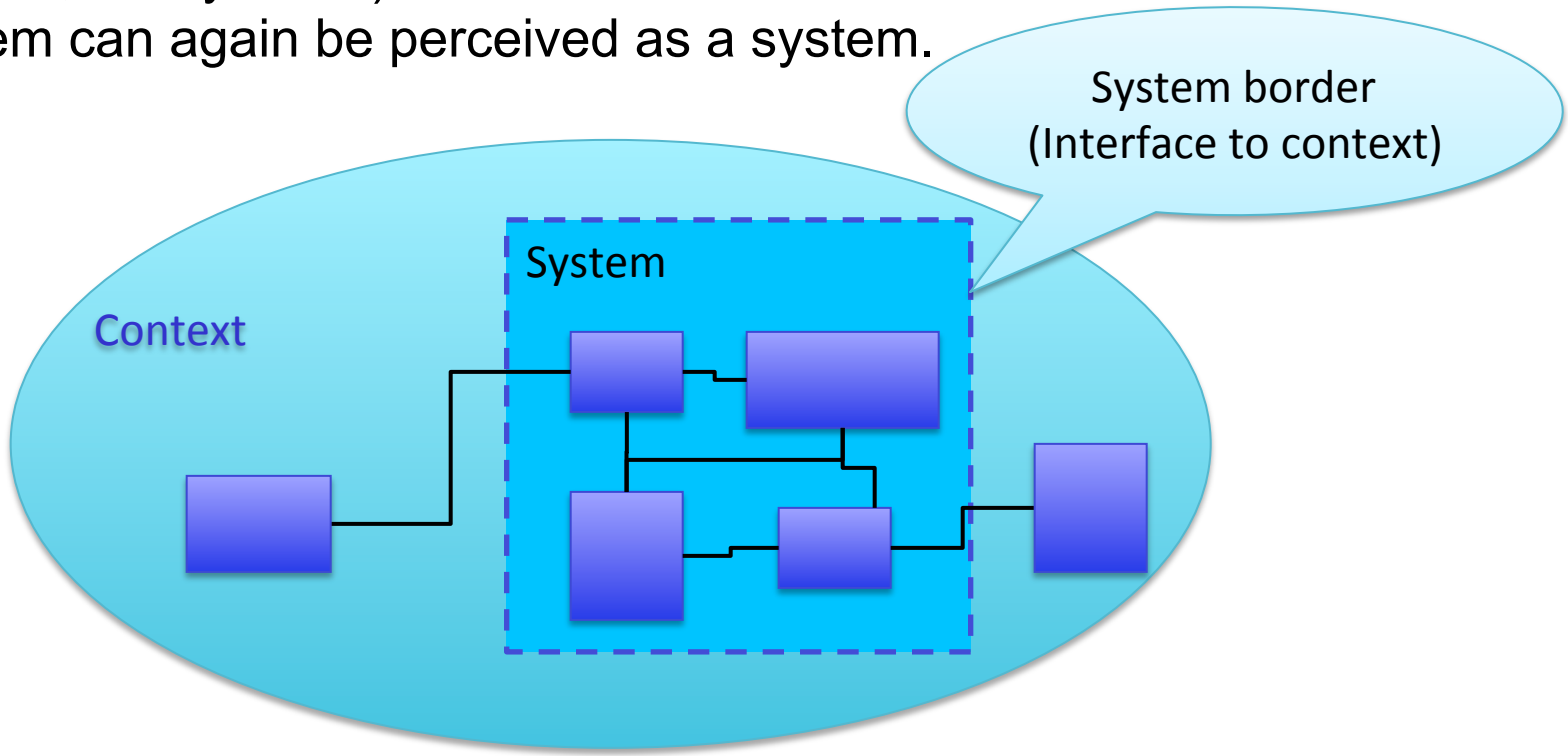
- Context
- Interface
- Behavior
- Structure
- State

System Models: Rationale

- Why is a specific concept modeled as is?
- Rationale (the justification / motivation) for a design decision is often not documented, e. g., technical constraints, stakeholder goals, ...
- Consequences: wondering, redesigning (plus potential conflict/mismatch later on)
- Crucial for rationale: Traceability

System Models: Refinement and Decomposition

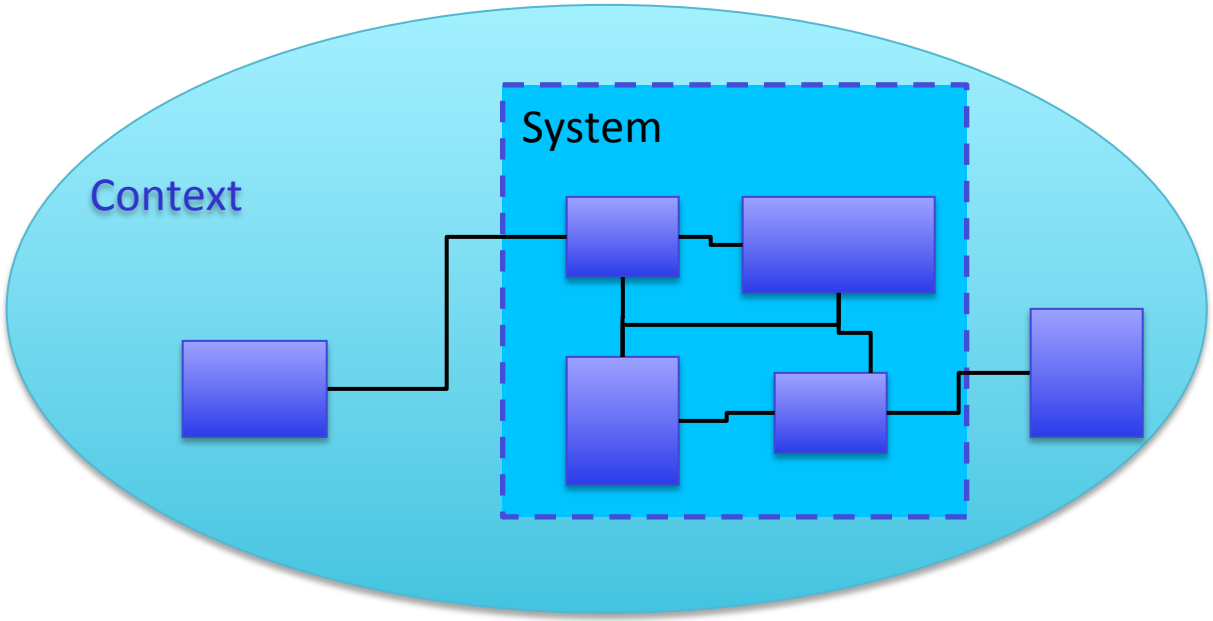
System: A system is delimited from ist context by the system border. A system has an interface. A system is composed of elements (components, subsystems) that are interrelated.
A subsystem can again be perceived as a system.



System Models: Refinement and Decomposition

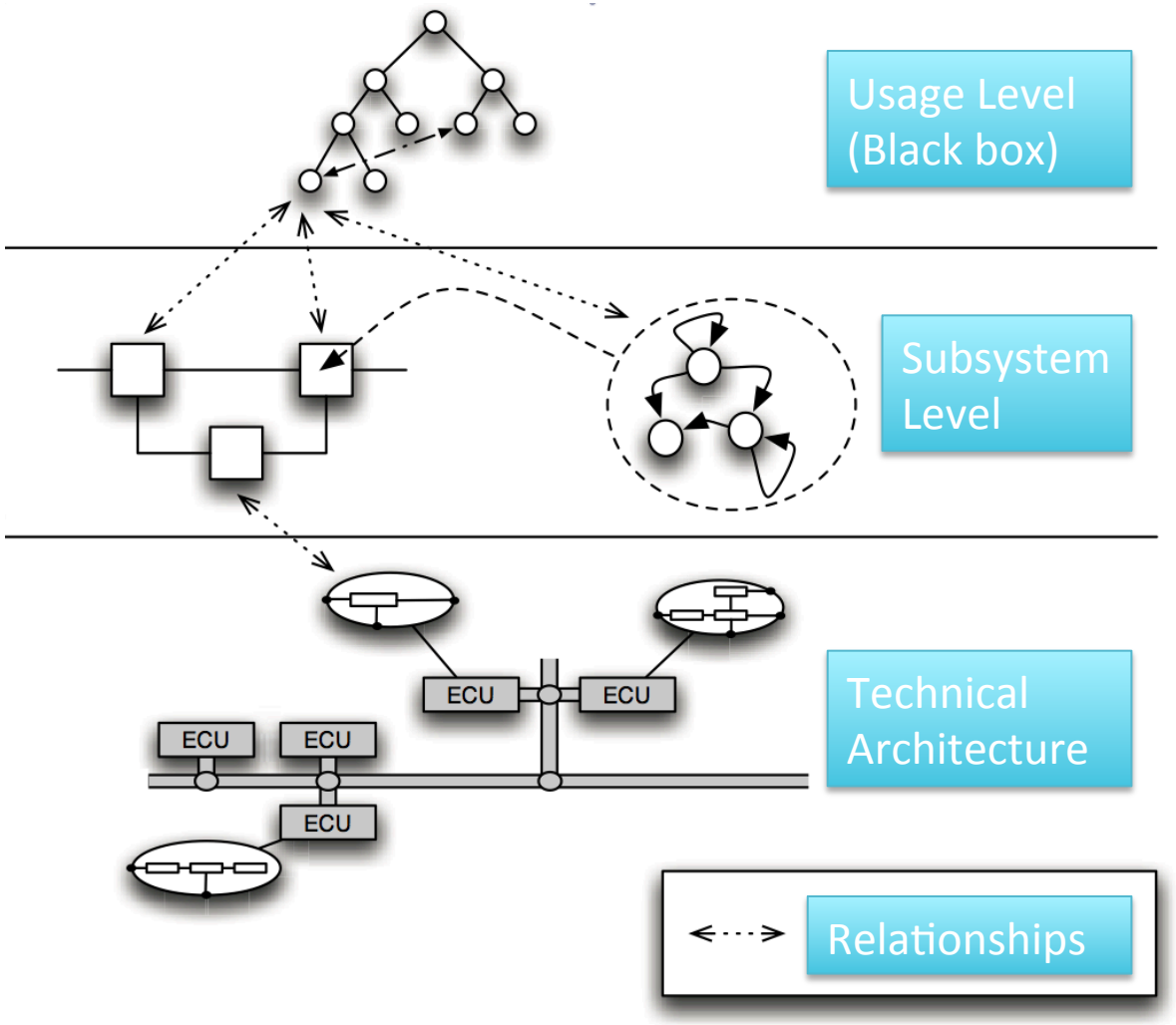
Refinement: Enriching information with more detail

Decomposition: Separating information into several parts



System Models: Abstraction Levels

An abstraction level describes the whole system on a certain level of abstraction. The degree is chosen such that each level reflects a specific aspect important for software development.



Exercise: Abstraction Levels

Envision the online shop:

- What are the different abstraction levels?
- What can we find on the usage level / subsystem level / technical architecture level?

Discussion

Code is the best view of the system level (technical architecture level) because

1) the code doesn't age

2) there are no additional costs for documentation

Do you agree? Why (not)?

Please undermine your arguments with examples.

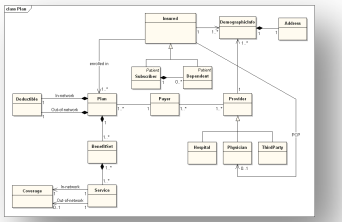
System Models: Traceability

- Use case to spec:

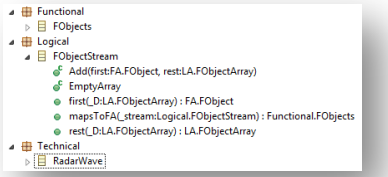
Use Case

Use Case	Wunschgeschwindigkeit halten
Actors	Fahrer
Intent	Fahrzeuggeschwindigkeit automatisch auf Wunschgeschwindigkeit des Fahrers halten
Assumptions	Fahrzeug fährt auf Autobahn, Landstraße (→ präzisieren) Kein Fahrzeug/Hindernis voraus (→ präzisieren)
Precondition	ACC ist bereit; Fahrzeuggeschwindigkeit > 30 km/h und < 250 km/h
Flow of events	1. Fahrer möchte die aktuelle Geschwindigkeit (ak.Gw.) beibehalten und drückt die ACC-Taste am Lenkstockhebel, um das ACC zu aktivieren und die ak.Gw. als Wunschgeschwindigkeit (W.g.) einzustellen. 2. Das System ermittelt die ak.Gw. und hält diese W.g. durch entsprechende Ansteuerung/Regelung des Motors und der Bremsen. 3. Das Kombiinstrument zeigt die (gehaltene) W.g. an. 4. Der Fahrer drückt/zieht den Lenkstockhebel, um die W.g. zu erhöhen/erniedrigen. 5. …. [Exception: Fahrer tritt auf Gaspedal/Bremse] [Exception: W.g. <= 30 km/h oder W.g. >= 250 km/h] [Exception: Fahrzeug voraus]
Exeptions	Tritt der Fahrer auf Gaspedal/Bremse wird das ACC ausgeschalten. Ist die neue W.g. < 30 km/h erfolgt eine Benachrichtigung des Fahrers …
Quality constraints	Unmittelbare, eindeutige und zuverlässige Anzeige/Benachrichtigung des Fahrer Zuverlässigkeit W.g. Regelung
Monitored variables	Fahrzeuggeschwindigkeit, Bremsen, Gaspedal, Wunschgeschwindigkeit
Controlled variables	Beschleunigung, Bremsenfunktion
Postcondition	Fahrer hat Kontrolle über Beschleunigung/Bremsen und steuert Fahrzeug

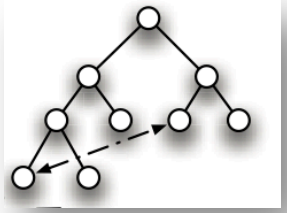
Domain model



Data model



Function / service model



All views refer to same system model!
→ autom. consistency

Discussion: System Views

- What are the relations between the different system model views?
- What consistency conditions arise from that?
- How can these conditions be checked or verified?

Usage of models in RE

- Characteristic-oriented formalisation of functional requirements/behavioral requirements

→Example temporal logic: Basis for formal verification and mathematical checking of demanded system characteristics

- Usage of prototypes
  Human-Maschine-Interface, Demonstration, Simulation, Proof of concept
- Deducing test cases
  Testing of requirements, Acceptance test, System/Integration test
- Executable models
  - Help for generating automata
  - Validation of system characteristics (e.g. via simulation)

Challenge: as complex as necessary, as simple as possible.