Model-Based Systems Requirements

Jean-Michel Bruel
João Araújo

Summary

- Introduction
- System Engineering
- Systems Requirements
- Requirements elicitation process
- KAOS overview
- SysML overview
- Requirements in SysML
- Mapping KAOS models into SysML models
- Practical case study

Introduction

- This tutorial aims at presenting an integrated approach for systems requirements elicitation and modeling
- The elicitation phase is based on a goal based approach → the KAOS Approach
- The modeling phase uses SysML, an OMG modeling language for systems
  - getting more and more popularity (used in Airbus, Thales, Continental)
  - start to be a pivot language for many others (e.g., Modelica, Simulink)

Summary

- Introduction
- System Engineering
- Systems Requirements
- Requirements elicitation process
- KAOS overview
- SysML overview
- Requirements in SysML
- Mapping KAOS models into SysML models
- Practical case study

Systems engineering

- Not Software Engineering…
- …Before Software Engineering!
  - In the development process
- Involves specifying, designing, implementing, validating, deploying and maintaining systems
- Concerned with the system's services, constraints and operation

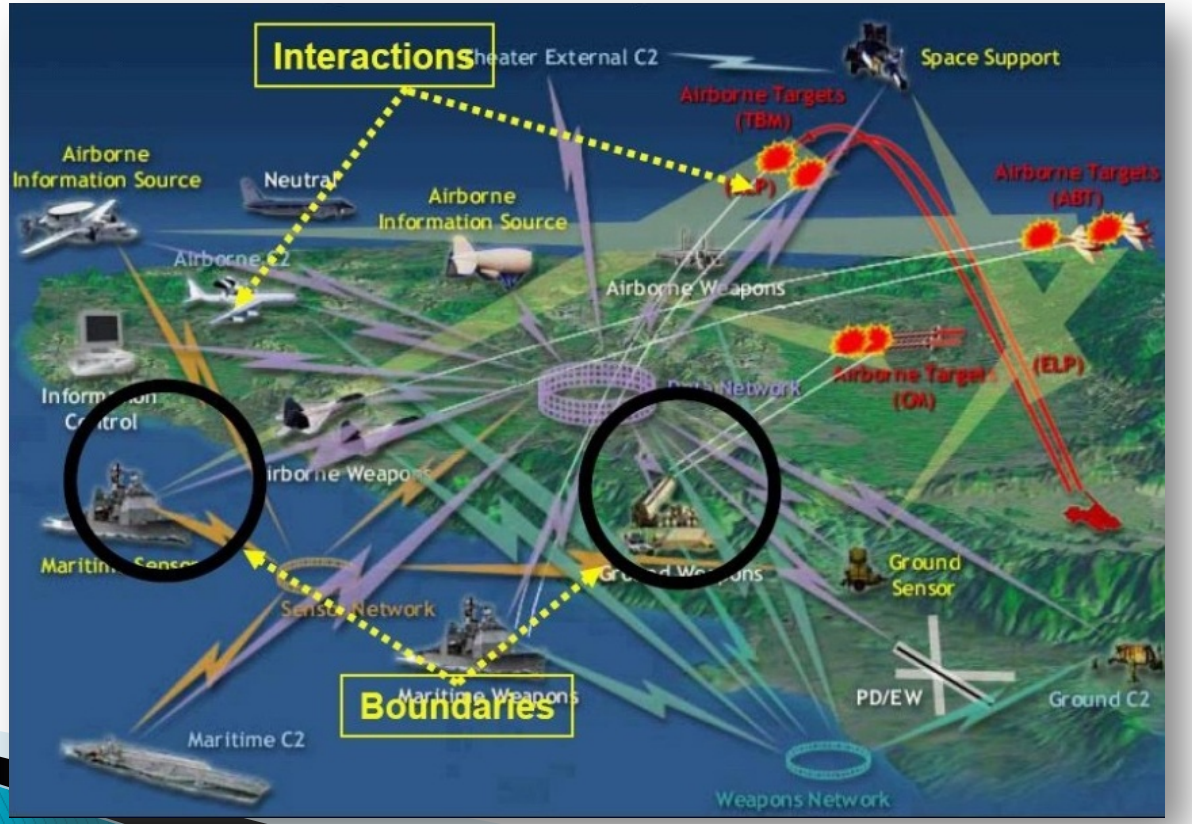
A Complex System

- Set of human and material elements composed of various technologies
  - Computer, Hydraulic, Electronic,…
- Integrated to provide services to its environment corresponding to the system finality
- Interacting between themselves and the environment

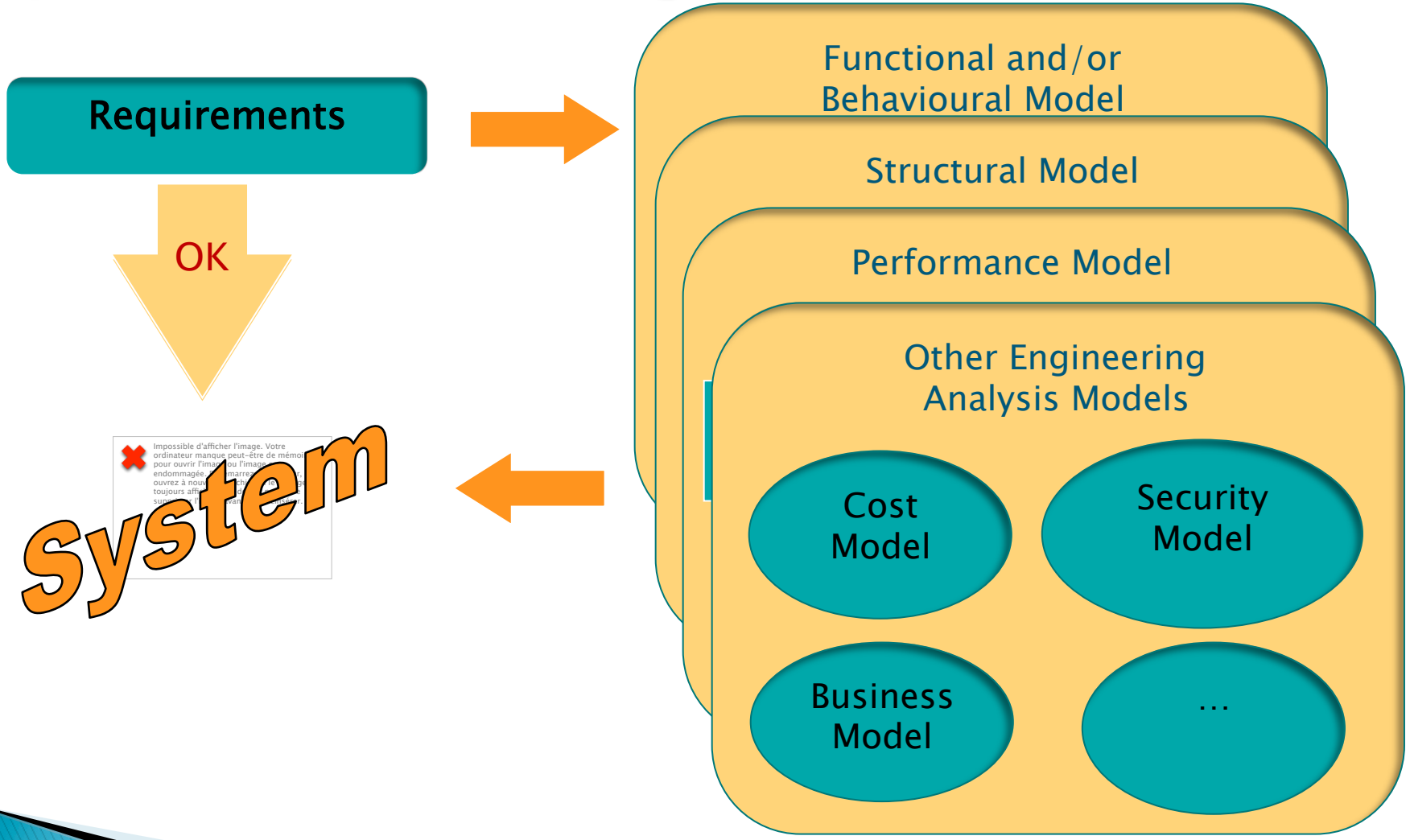
A complex system is very different from a simple software system

Systems of Systems

- A system
  - Should manage interactions between parts
  - Support expected behavior
  - Handle unexpected ones



System Modeling



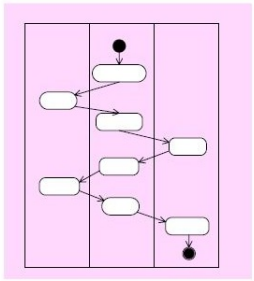
SE practices for modeling systems

Generate lot of writing work

Not adapted to discuss within a multi-domain team

finiti

Before



After

Moving from **Document centric**
To **Model centric**

Types of requirement

- User requirements
  - Statements of the services the system must provide and its operational constraints.
  - Target: customers.
- System requirements
  - A structured document with **detailed** descriptions of the system's services and operational constraints.
  - Part of a contract between client and contractor.

System requirements

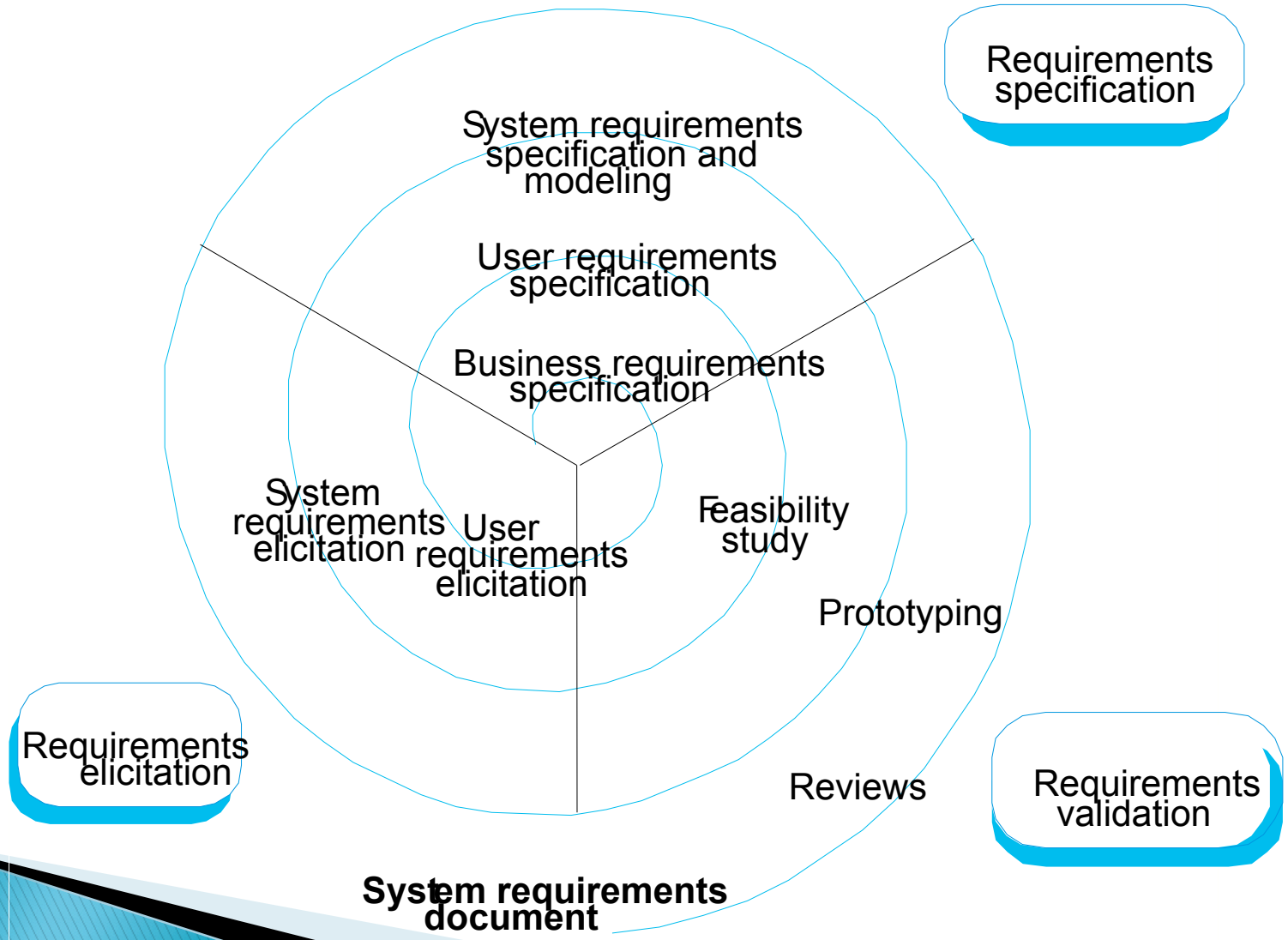
- More detailed specifications of system services (F) and constraints (NF) than user requirements
  - Functional: descriptions of system's services, and how the system should react in particular situations.
  - Non-functional: may be more critical than functional requirements. If these are not met, the system is useless. E.g. availability, timing constraints, reliability, security, safety
- They are intended to be a basis for designing the system
- They may be incorporated into the system contract
- System requirements may be specified using models

System requirements problems

- Complex systems must be concerned with
  - Changing as the system is being specified.
  - Must anticipate hardware and communications evolution
  - Hard to define non-functional requirements and analyse their interactions

System requirements in RE

(Sommerville, 2010)



Elicitation and analysis

- Also called requirements elicitation or requirements discovery.
  - Involves interaction with stakeholders to discover their requirements.
  - Domain requirements are also discovered at this stage.
- Developers should work with customers to learn about the application domain, the services that the system should provide and the operational constraints of the system.
- May involve different *stakeholders:* end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc.

Requirements and design

- In principle, requirements should state what the system should do and the design should describe how it does this.
- In practice, requirements and design are difficult to separate
  - A system architecture may be designed to structure the requirements;
  - The system may inter-operate with other systems that generate design requirements;
  - The use of a specific design may be a domain requirement.

What are *Goals*?

- *Goals* are desired system properties that have been expressed by some stakeholders
- Goals can be specified in different levels of abstraction, covering at a higher level strategic concerns and at a lower level technical issues
- They can be:
  - functional – related to the services provided
  - non-functional – related to quality attributes (e.g. Security, performance, availability)
- Examples of goals (TV system):
  - "The system should provide access to search and choose channels programs"
  - "The system must be available 24h/7days a week"

Goals and NF requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
  - Close to a goal → or a softgoal!
  - Goals are helpful to developers as they convey the intentions of the system users.

- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.

Examples

- **A system goal**
  - The TV system should be easy to use by ordinary users and should be organised in such a way that user errors are minimised.
- **A verifiable non-functional requirement**
  - Ordinary users shall be able to use all the system functions after reading the manual for an hour.

GORE

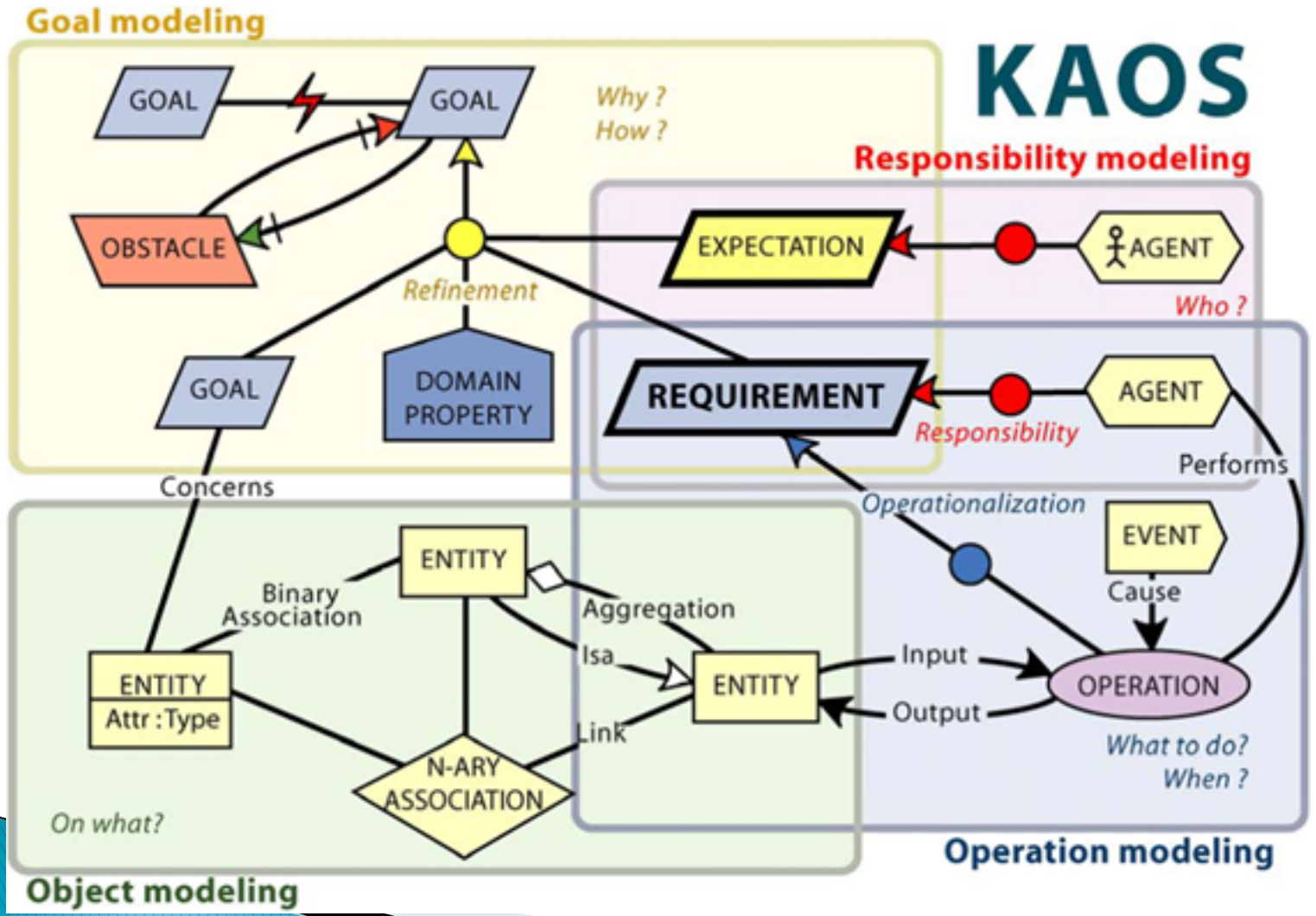
- GORE is considered an established paradigm in RE to handle elicitation, specification, analysis, negotiation and evolution of requirements by using goals
- GORE approaches were developed to support the development of large-scale systems where the goal model is the central one.
  - KAOS, i*, GRL, NFR Framework, GQM
- Eliciting requirements for large-scale models is performed in a stepwise manner.
  - The higher-level goals are decomposed into less abstract goals.

KAOS

- KAOS is one of the most well-known GORE approaches.
- It is a methodology based on the decomposition and refinement of goals to support the entire requirements development and acquisition process.

- These models allow:
  - Goals development
  - Objects identification
  - Operations identification
  - Goals operationalization
  - Responsibilities assignment

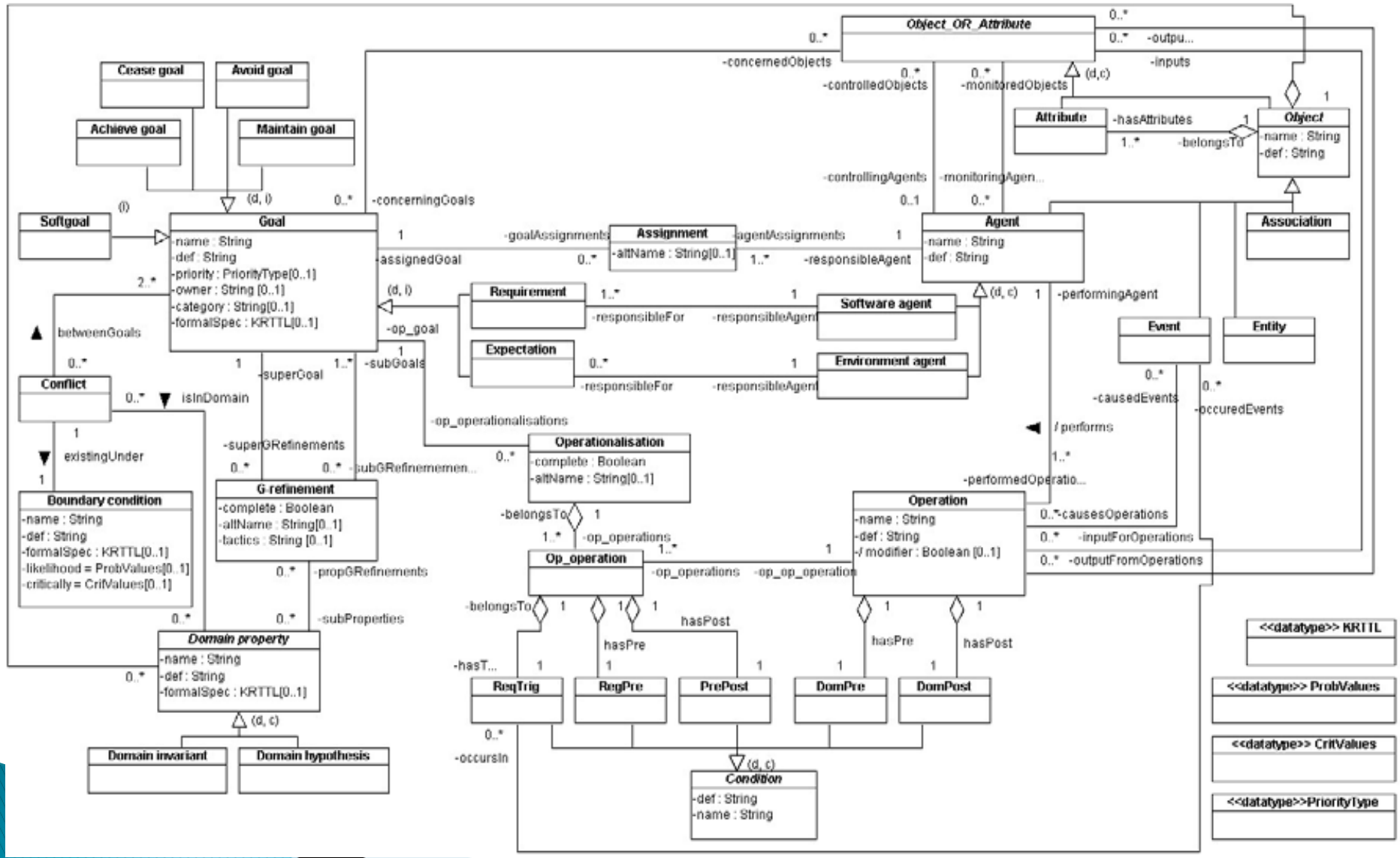
KAOS main model elements

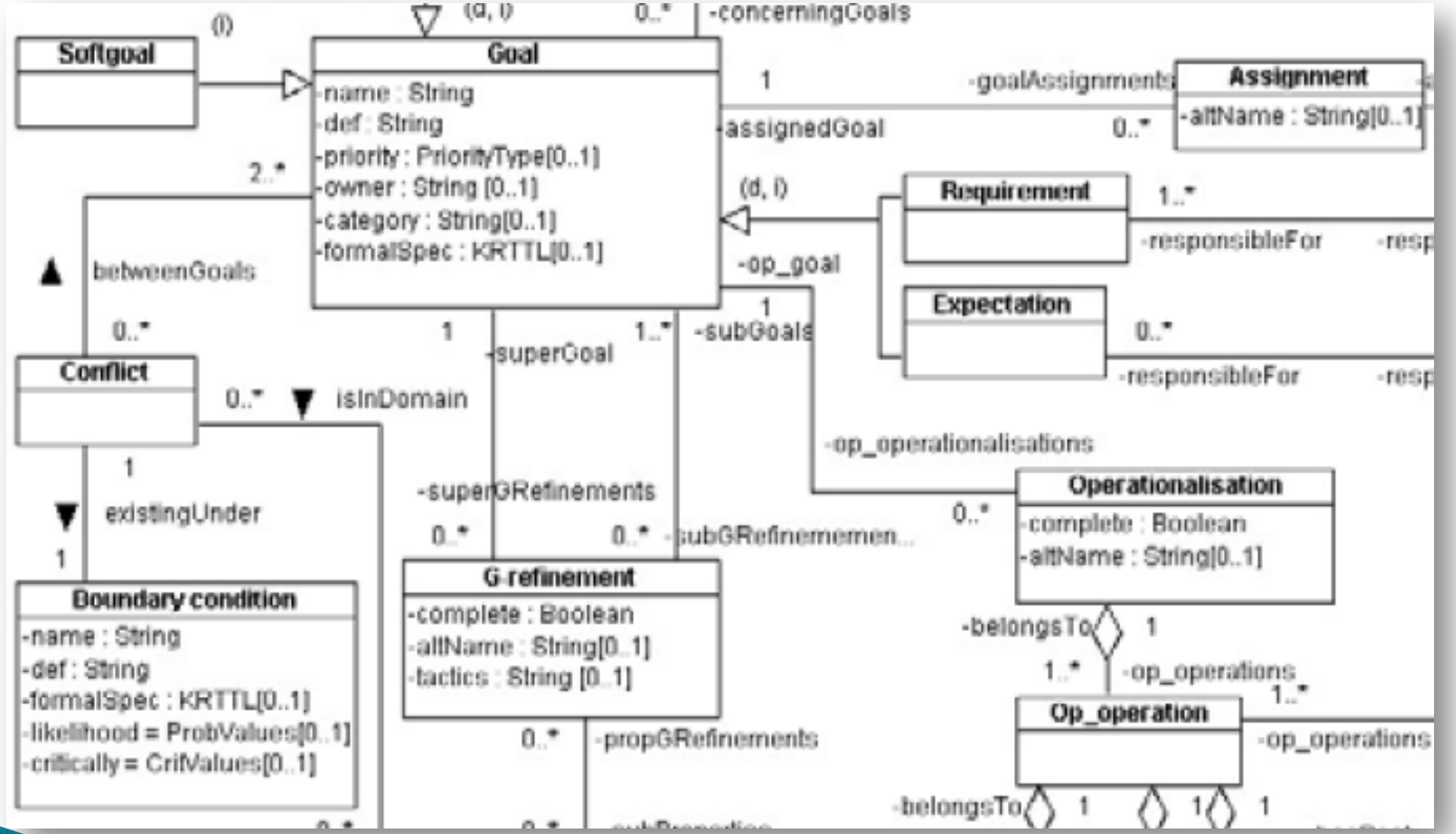


KAOS (cont.)

- It is a systematic approach to discover and structure requirements while:
  - Avoiding ambiguous or irrelevant requirements
  - Allowing efficient and easy communication between stakeholders
  - Clarifying stakeholders responsibilities
- It also provides mechanisms to:
  - Choose between different alternatives
  - Manage conflicts
  - Refine goals to structure complex requirements
  - Do domain analysis for reuse purposes

KAOS metamodel




Softgoal
Goal
-name : String
-def : String
-priority : PriorityType[0..1]
-owner : String [0..1]
-category : String[0..1]
-formalSpec : KRTTL[0..1]
-concerningGoals
-goalAssignments
Assignment
-altName : String[0..1]
-assignedGoal
Requirement
-responsibleFor
Expectation
-op_goal
betweenGoals
-superGoal
-subGoals
Conflict
isInDomain
existingUnder
-superGRefinements
-subGRefinemen...
-op_operationalisations
Operationalisation
-complete : Boolean
-altName : String[0..1]
-belongsTo
-op_operations
Op_operation
G-refinement
-complete : Boolean
-altName : String[0..1]
-tactics : String [0..1]
-propGRefinements
Boundary condition
-name : String
-def : String
-formalSpec : KRTTL[0..1]
-likelihood = ProbValues[0..1]
-criticality = CritValues[0..1]

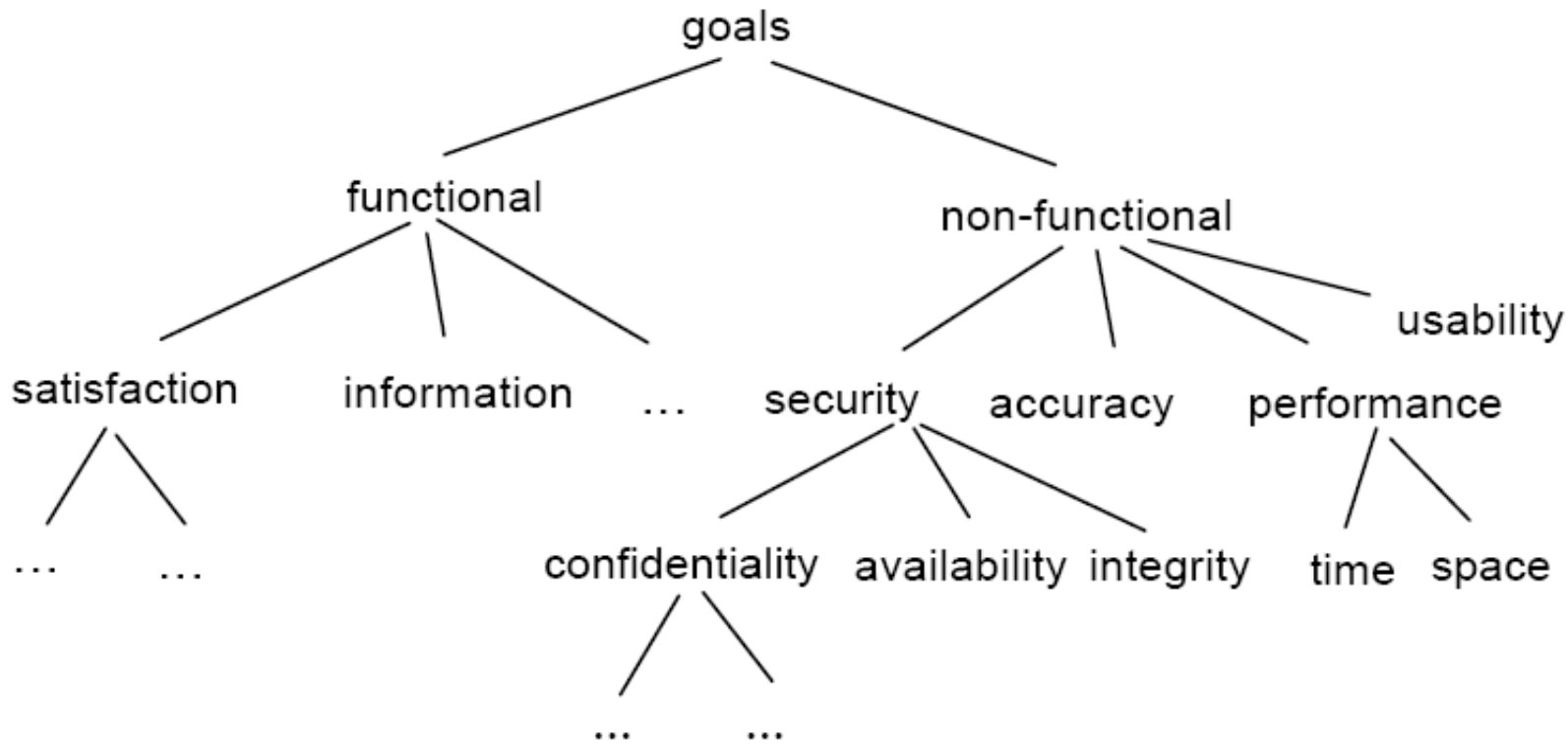
KAOS models

- Goal model
- Responsibility model
- Object model
- Operation model

Identifying Goals

- Discover system goals through interviews, technical documents, etc.
- Each goal in the model (except the roots) is justified by at least another goal explaining **why** the goal was introduced in the model
- Each goal in the model (except the leaves, bottom goals) is refined by a collection of subgoals describing **how** the refined goal can be reached
- The identification is both top-down and bottom-up
  - In summary, refining and abstracting goals (WHY & HOW)

Goals : Types


goals
functional
non-functional
satisfaction
information
…
security
accuracy
performance
usability
…
…
confidentiality
availability
integrity
time
space
…
…

Specification of goals

- Informal (text in natural language)
- Semi-formal (diagrams)
- Formal (use of temporal logic formulas)
  - Not addressed in this tutorial

Goals relationships

- AND Refinements
- OR Refinements
- Conflicts
- Obstruction and resolution links
- Responsibilities links

Responsibility model

- **Agents** are humans or automated components that are responsible for achieving requirements expectations
- **Expectations** are requirements on agents interacting with the system
  - They are introduced to show how the SW system and its environment have to cooperate to achieve the goals
  - It is type of goal to be achieved by an agent part of the environment of the system
- A **requirement** is a low level type of goal to be achieved by a software agent
  - The software agent is responsible for it

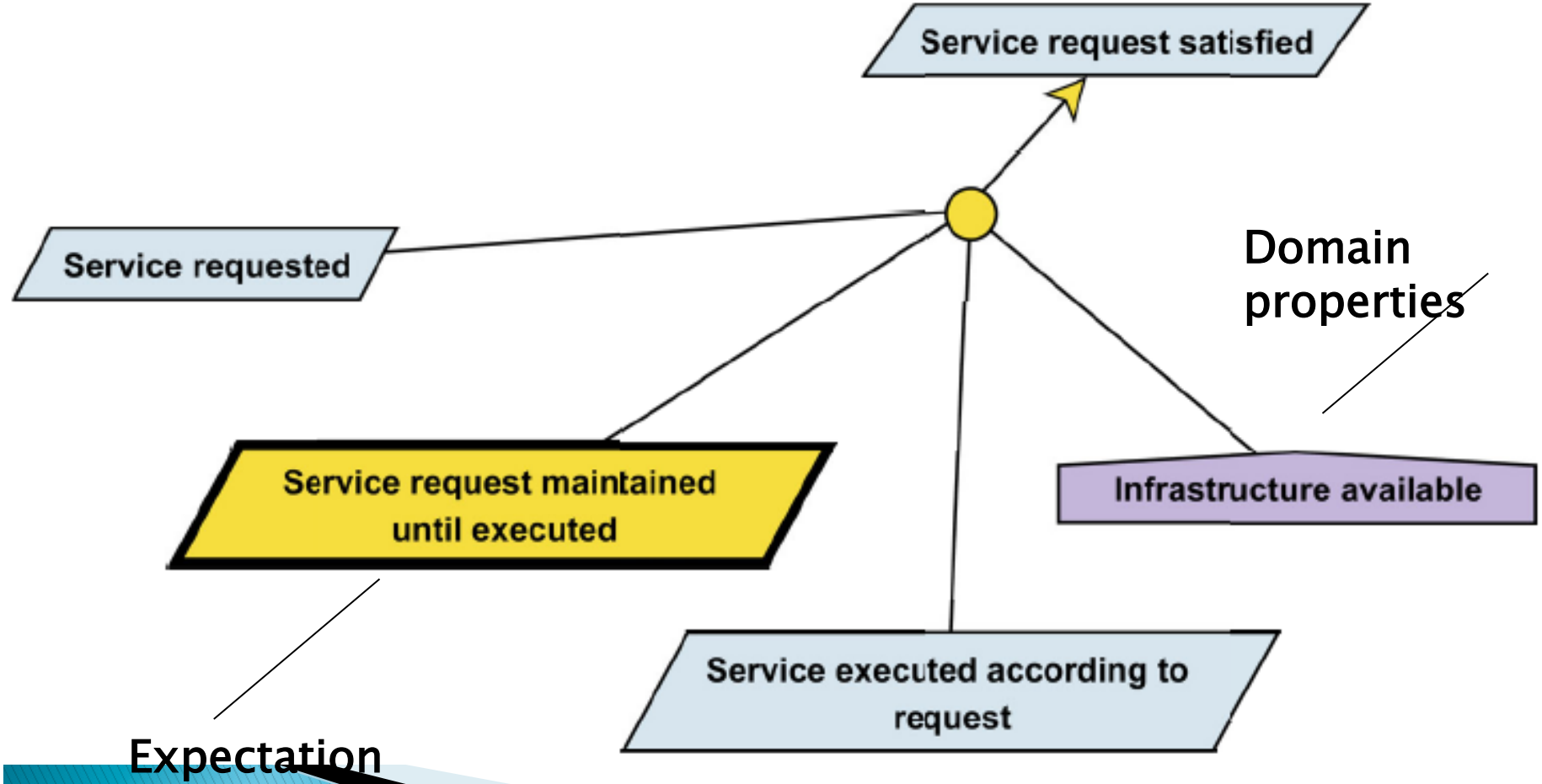
Completeness criteria

- Criterion 1:
  - A goal model is said to be complete with respect to the refinement relationship if and only if every leaf goal is either an expectation, a domain property or a requirement
- Criterion 2:
  - A goal model is said to be complete with respect to the responsibility relationship if and only if every requirement is placed under the responsibility of one and only one agent or implicitly if the requirement refines another one which has been placed under the responsibility of some agent

Requirements patterns
Goal Model

- Available patterns can be instantiated and reused
- KAOS consists of modelling generic patterns of requirements
  - They are progressively built

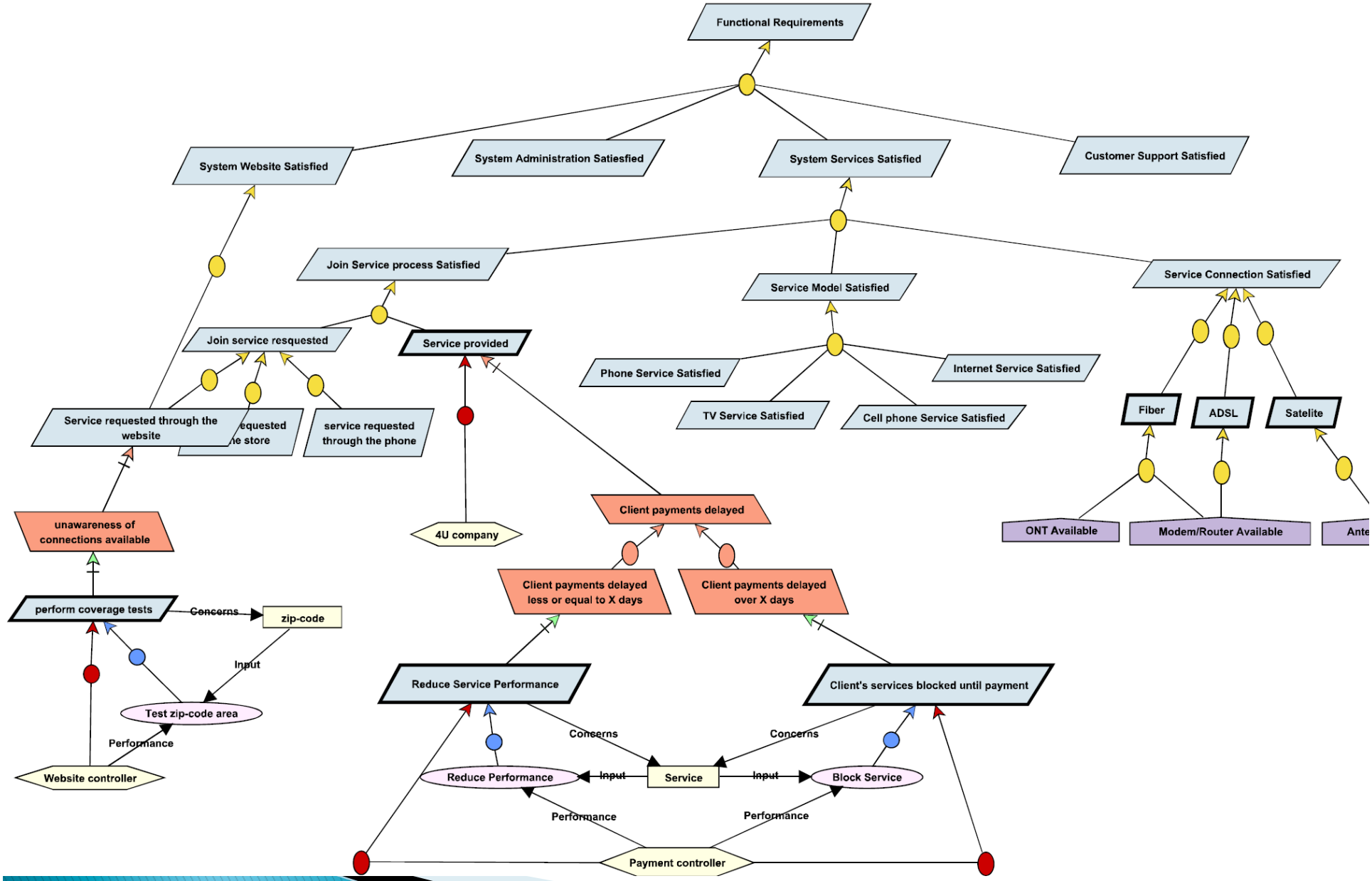
Generic Pattern: Expectations and Domain Properties



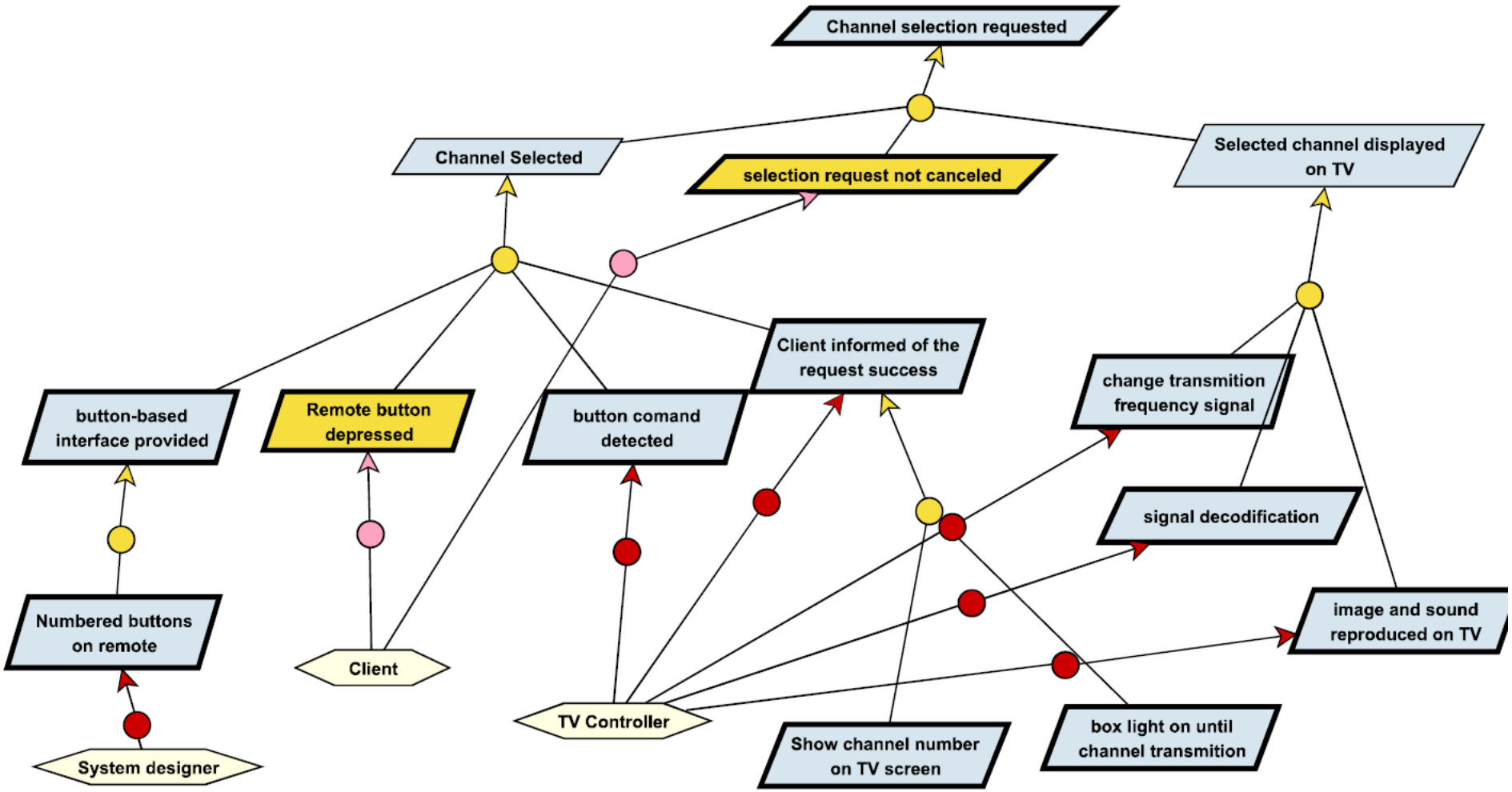
Example:Tv+Internet+Mobile+Phone package

Ceci n'est pas un smartphone

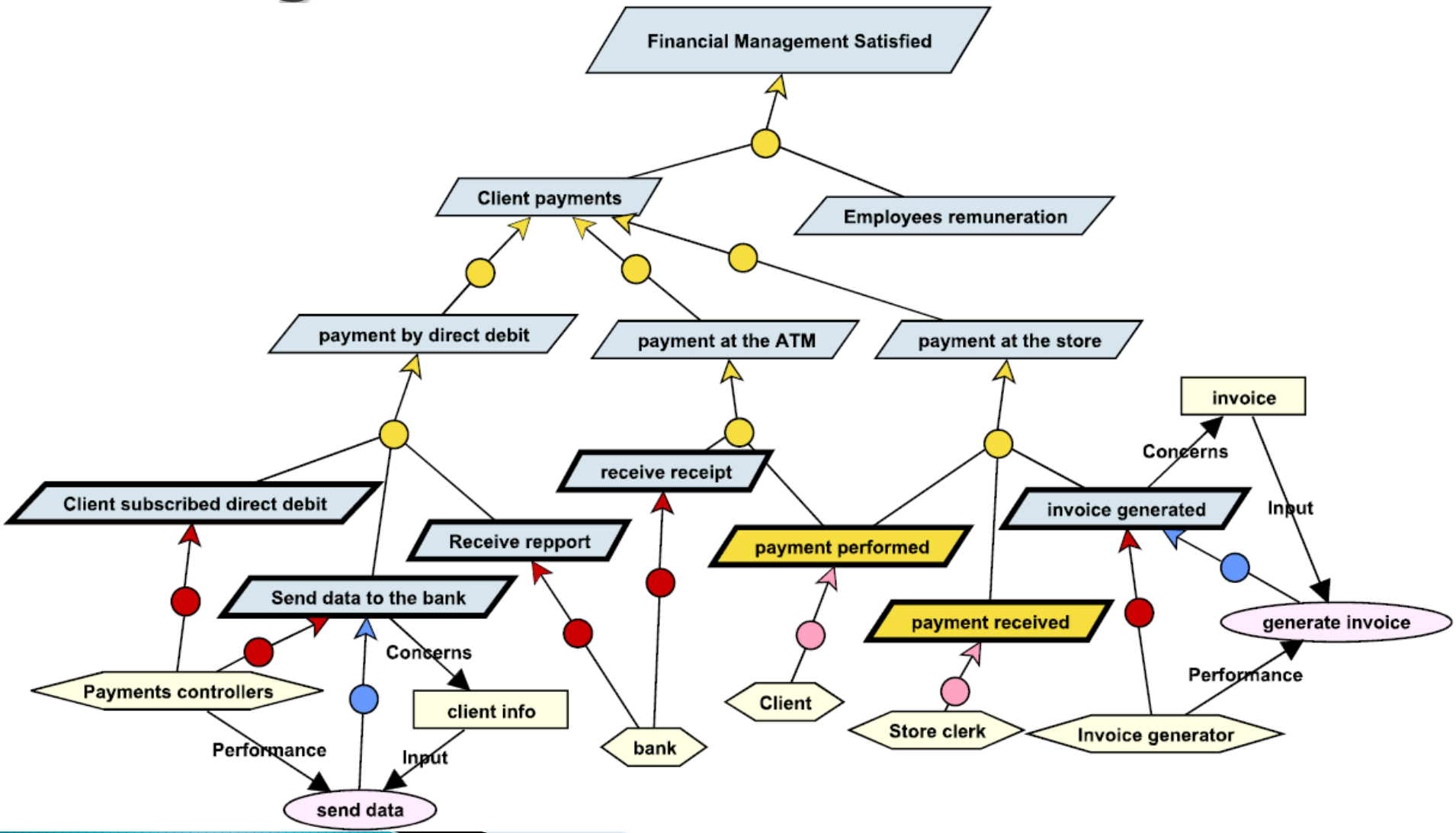
Functional requirements



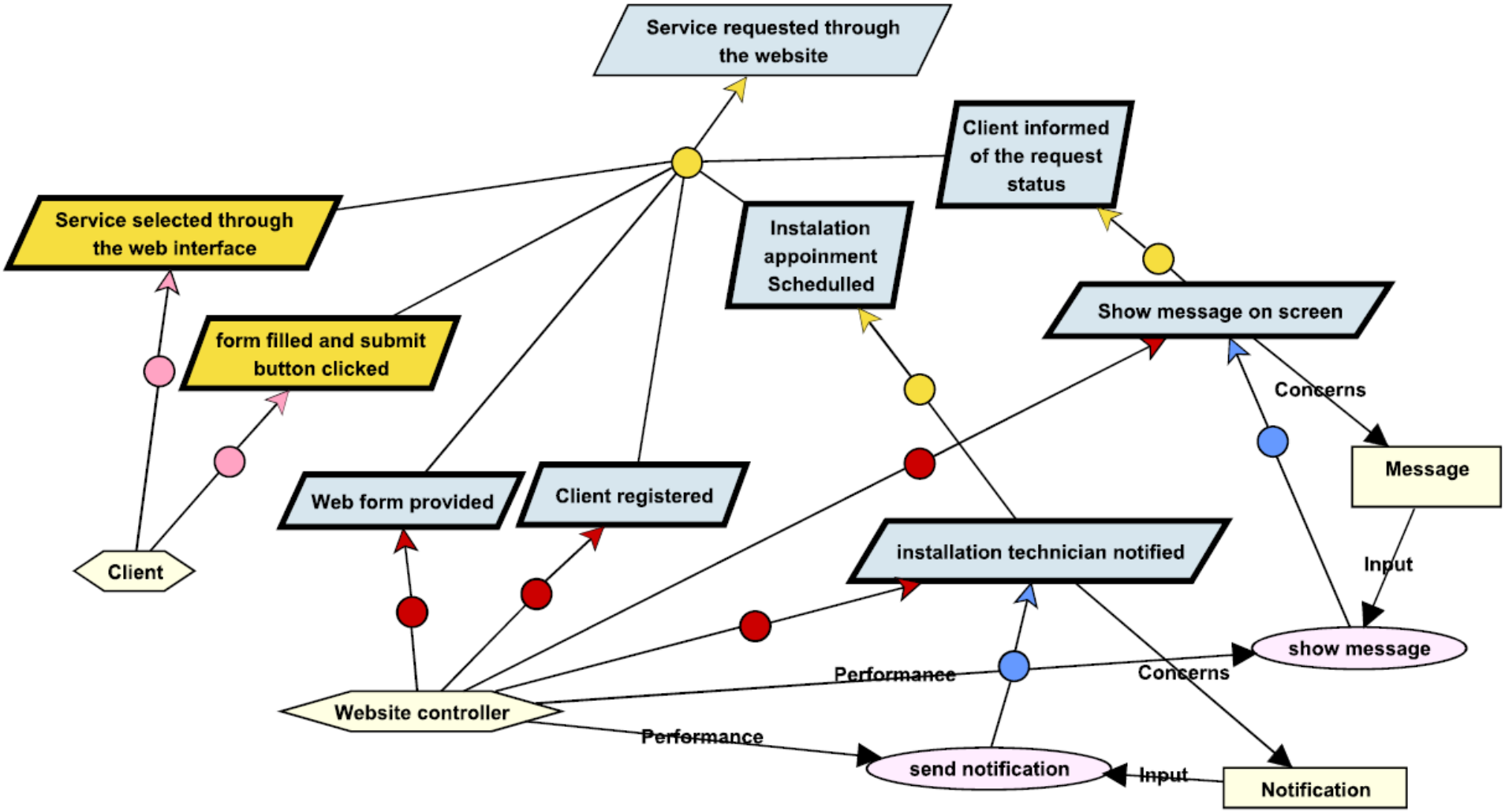
Functional reqs: Channel selection



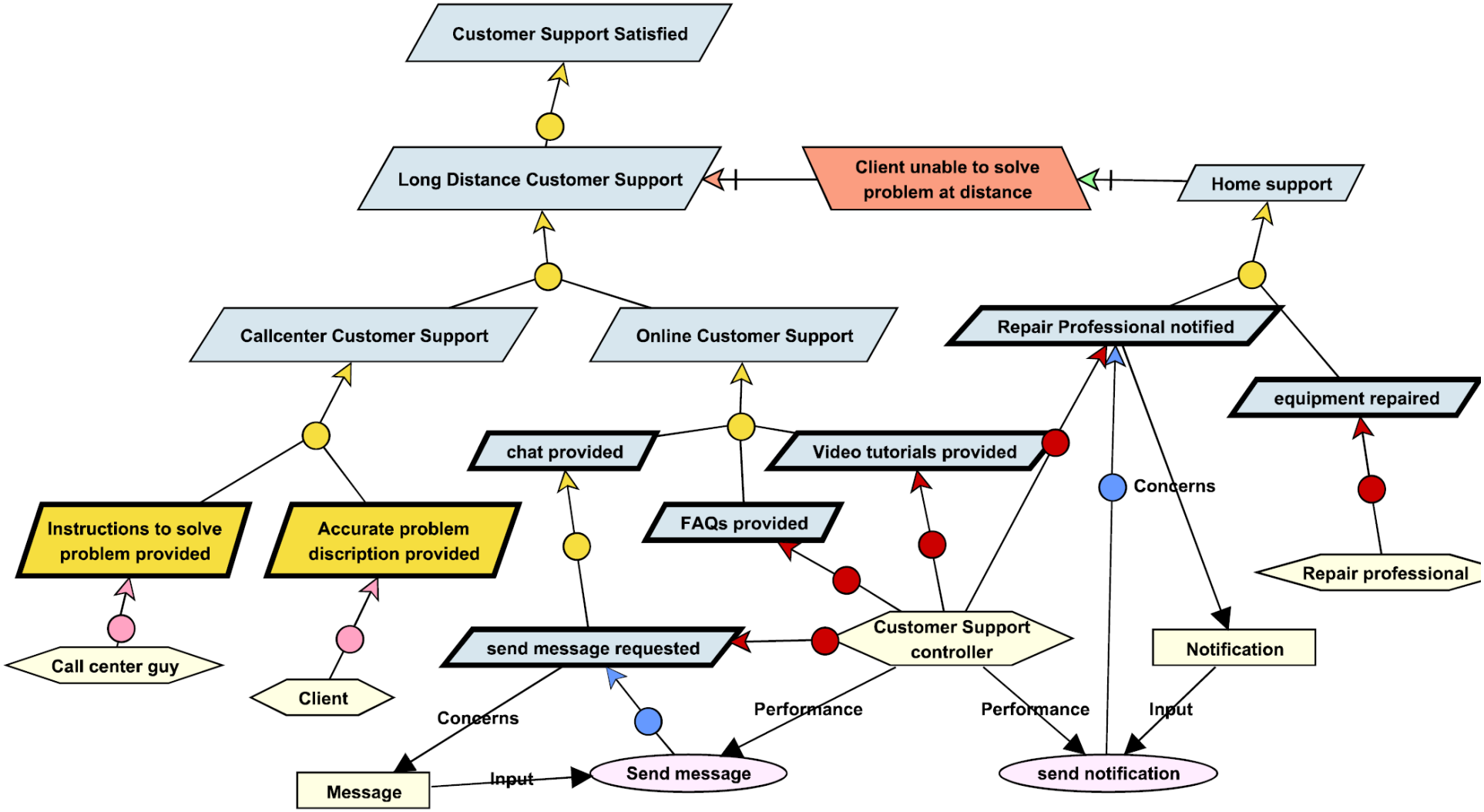
Functional reqs: Financial Management Satisfied



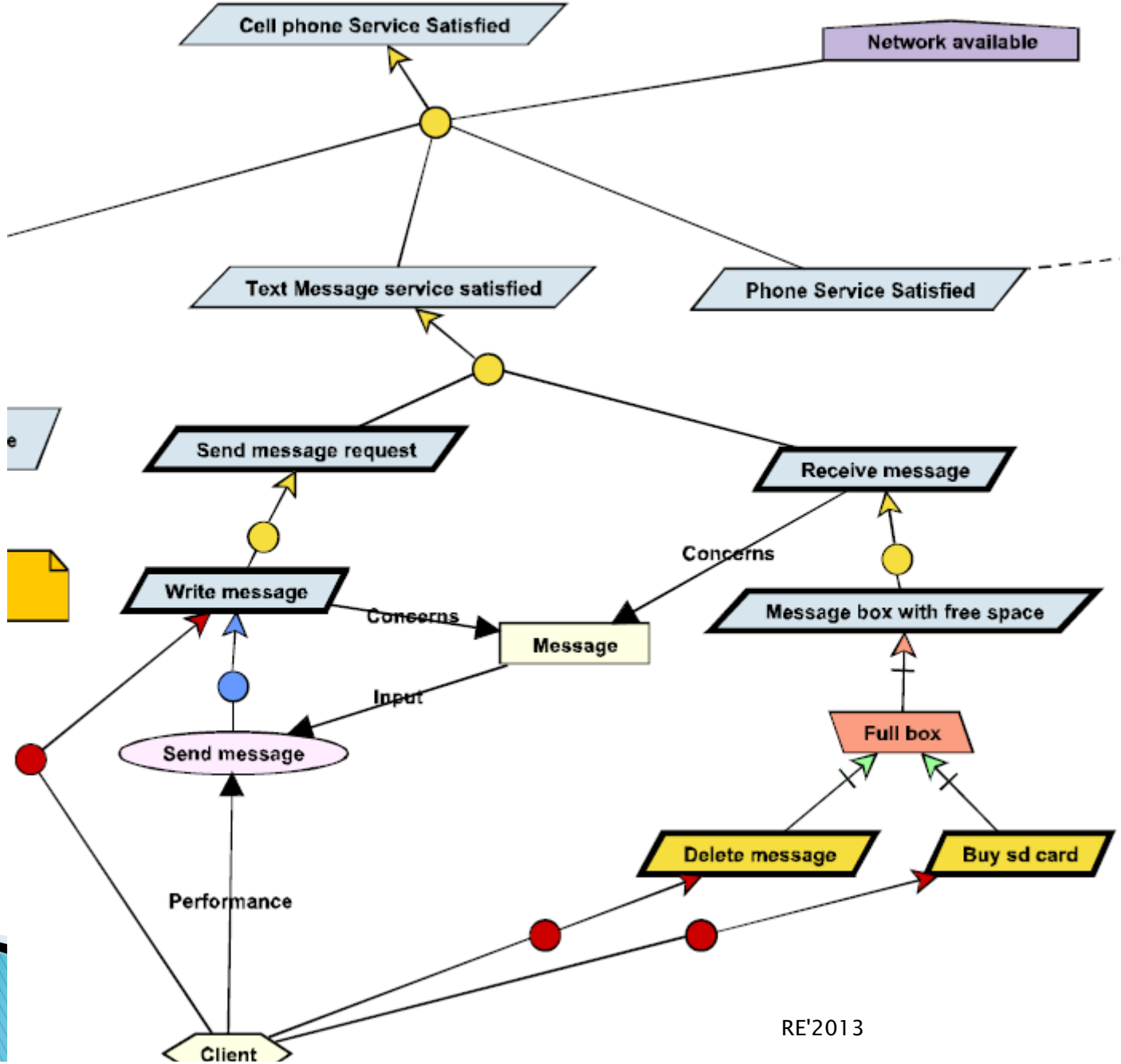
Entities and operations



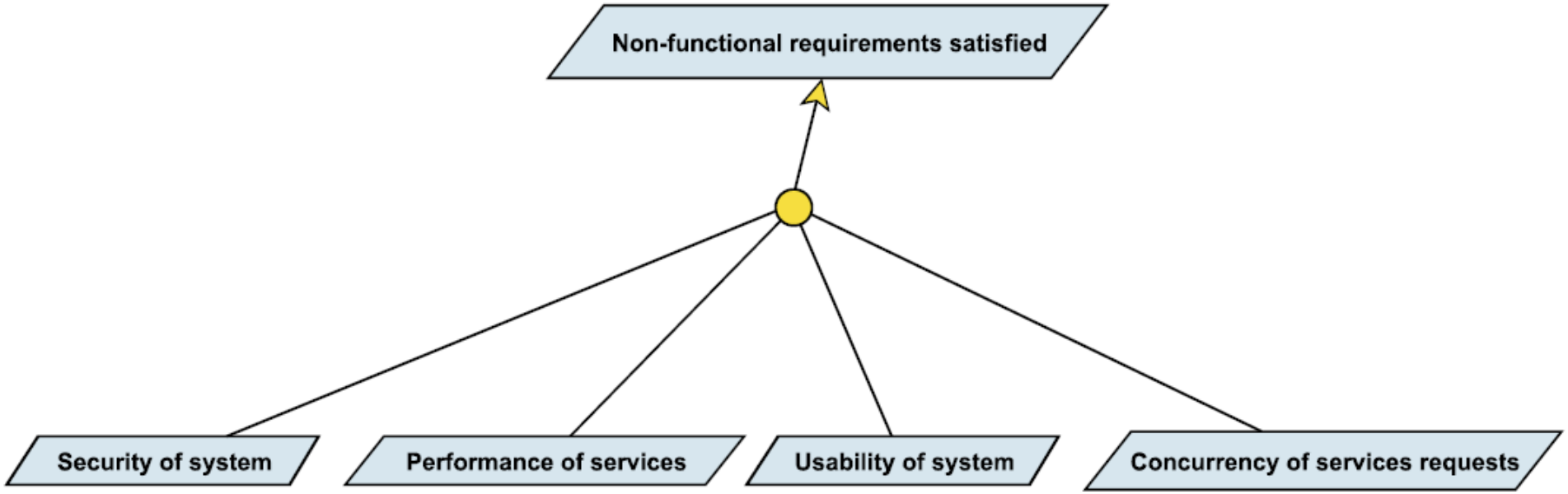
Obstacles and resolutions (1)



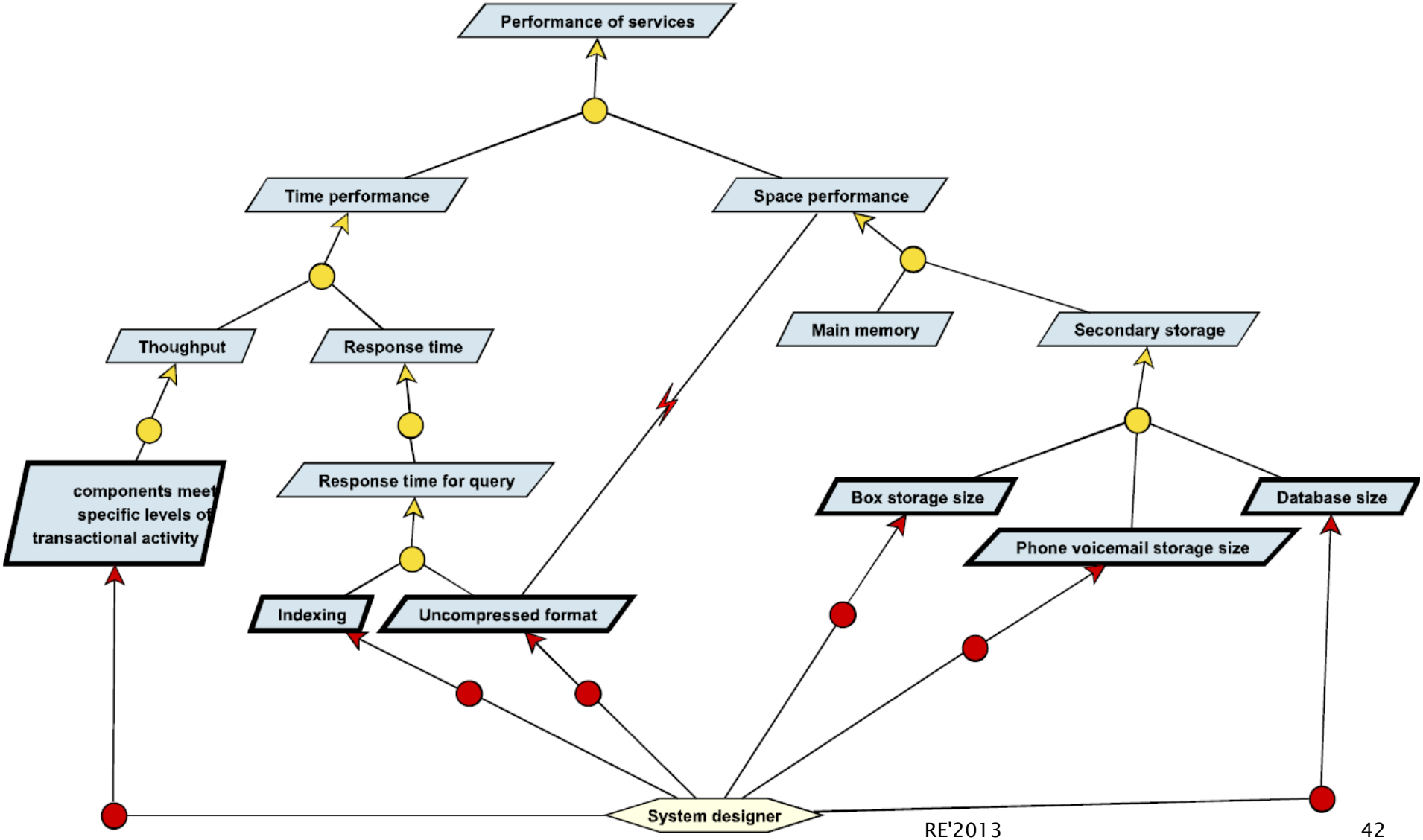
Cell phone service goal



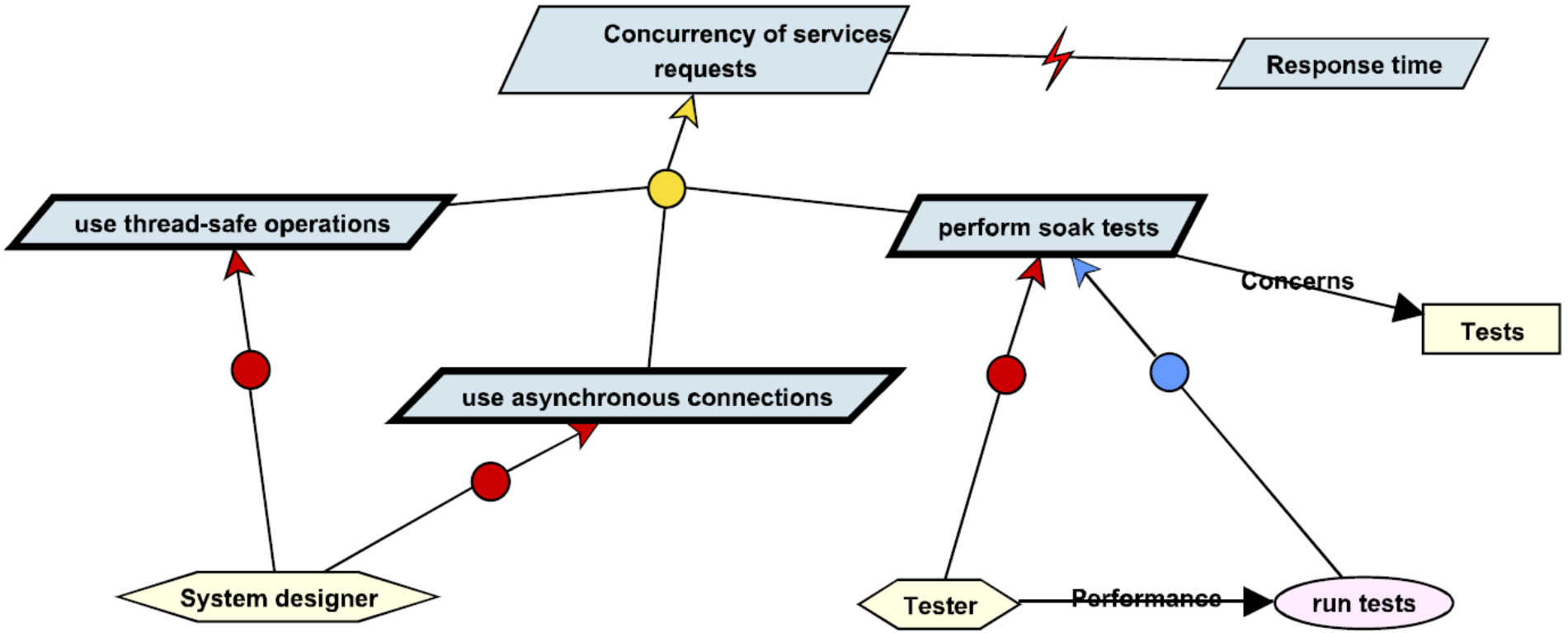
Non-functional requirements



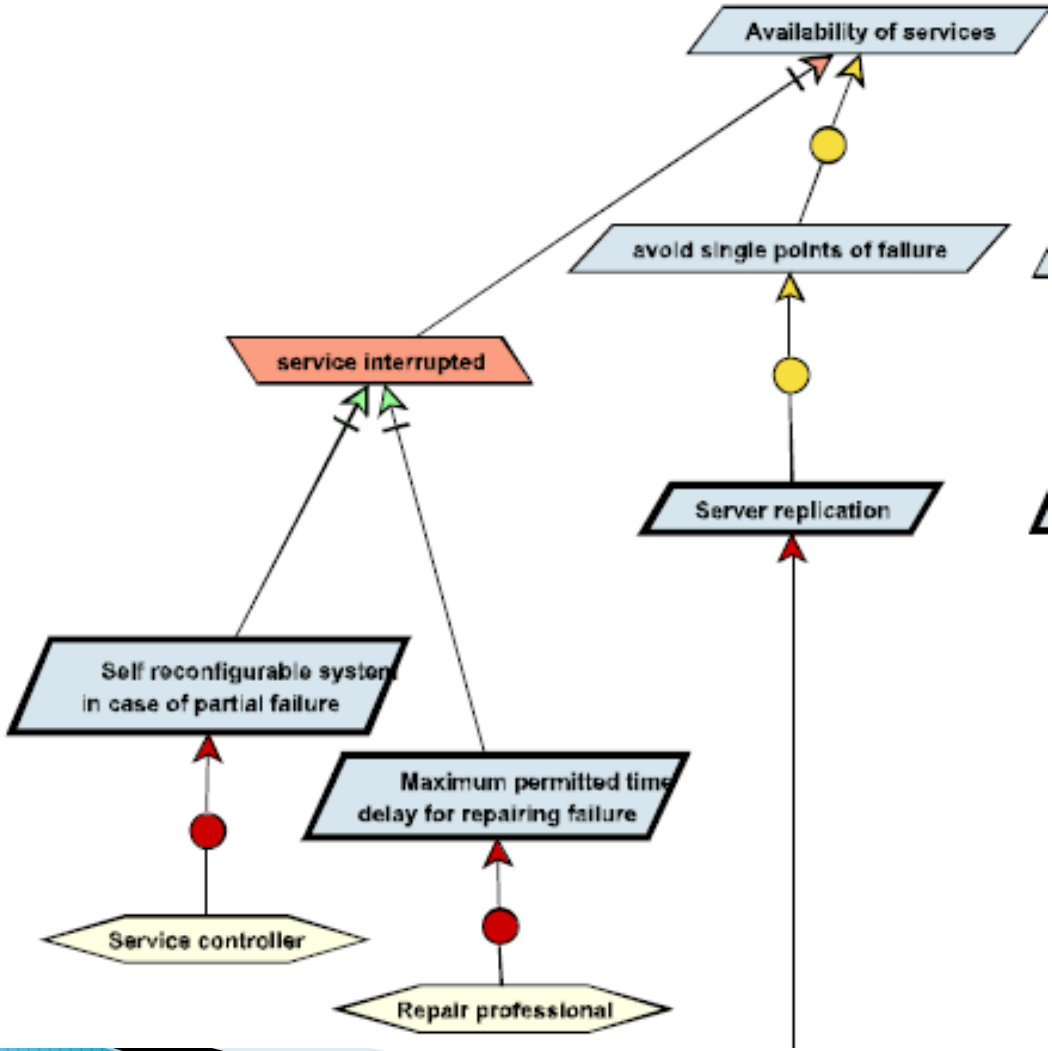
Performance



Concurrency



Availability

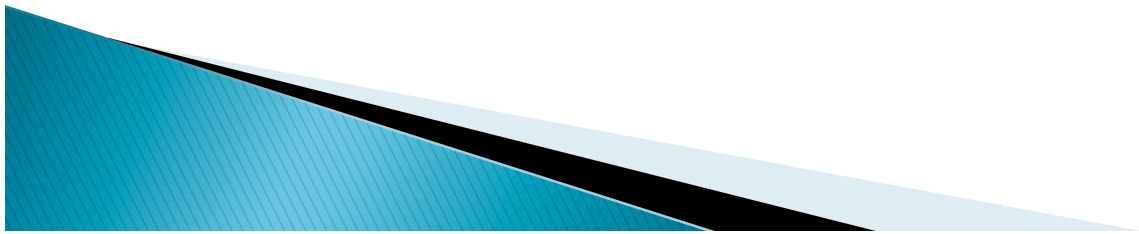


Conflicting goals

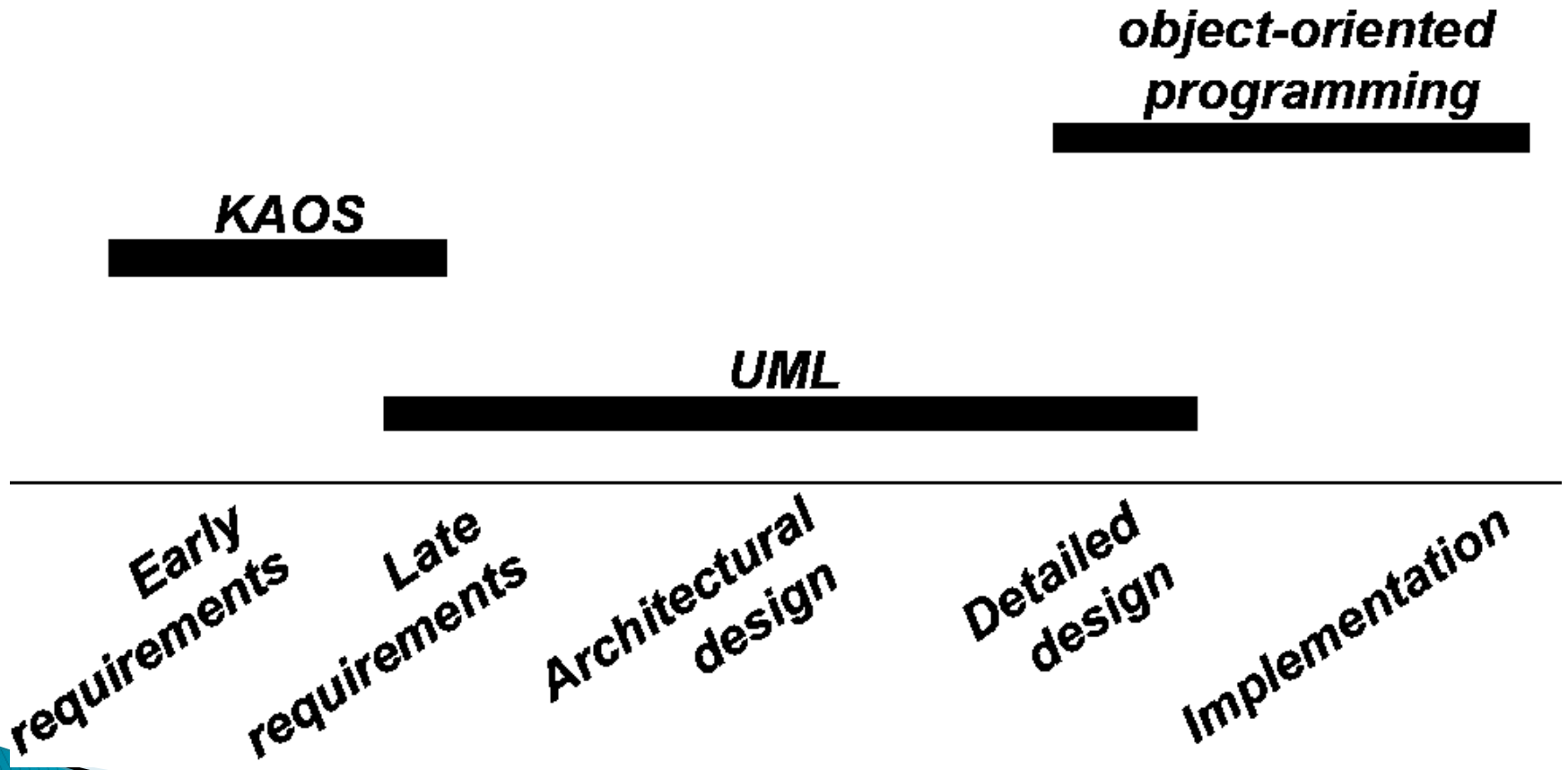
- When it is not possible to satisfy two goals simultaneously
  - Performance goals may conflict with safety goals
  - Information goals may conflict with security and privacy goals
- Obstacles prevent goals from being achieved
  - Defensive approach
- Dealing with conflicts (or more generally, with obstacles) allows
  - to build a more complete requirements document and
  - To build a more robust system

Conflict management

- When conflicts are detected:
  - Negotiation to conflict resolution
    - Select alternatives or
    - Re-evaluate the priorities or
    - Revise requirements



Goals and UML



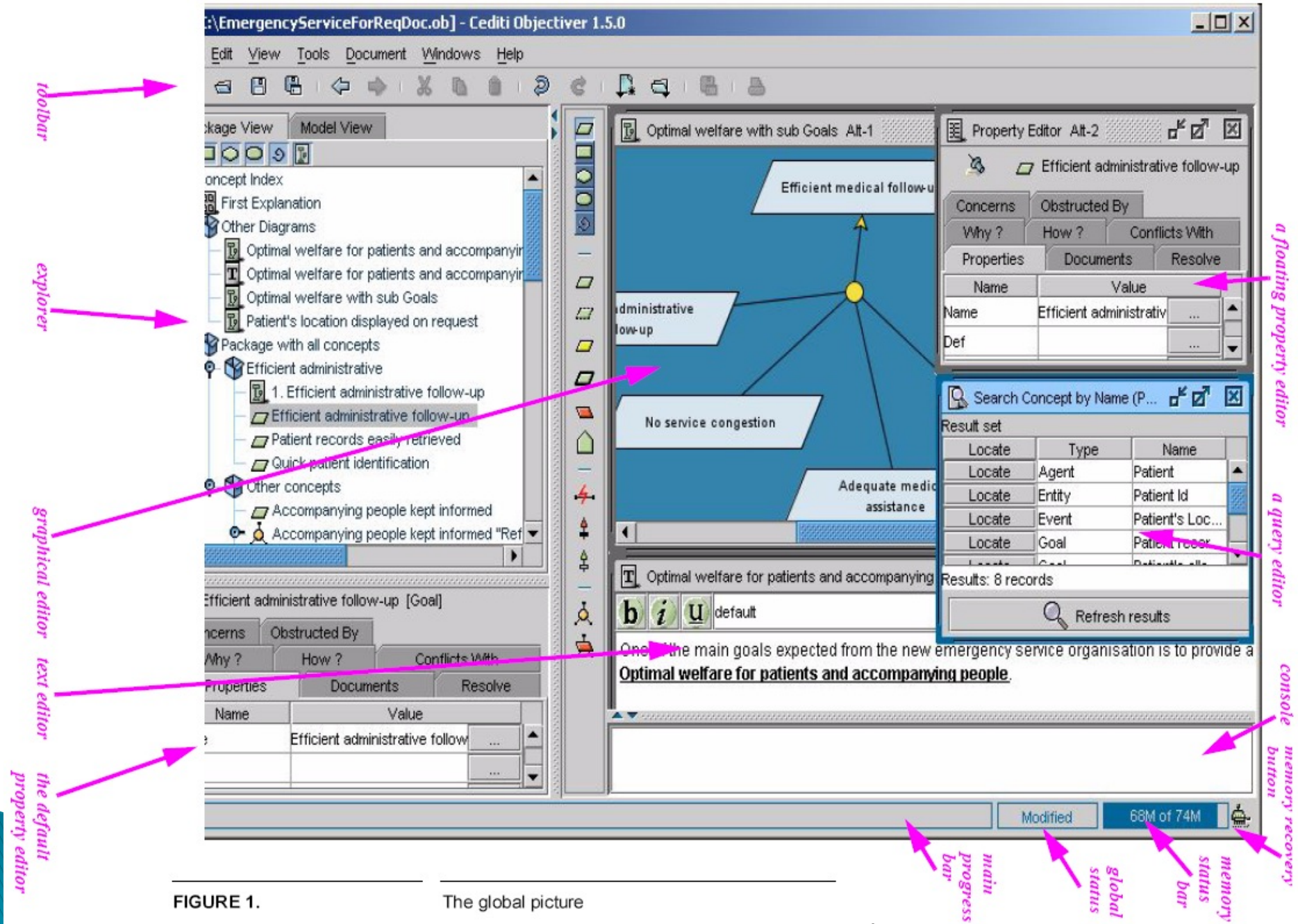


FIGURE 1. The global picture

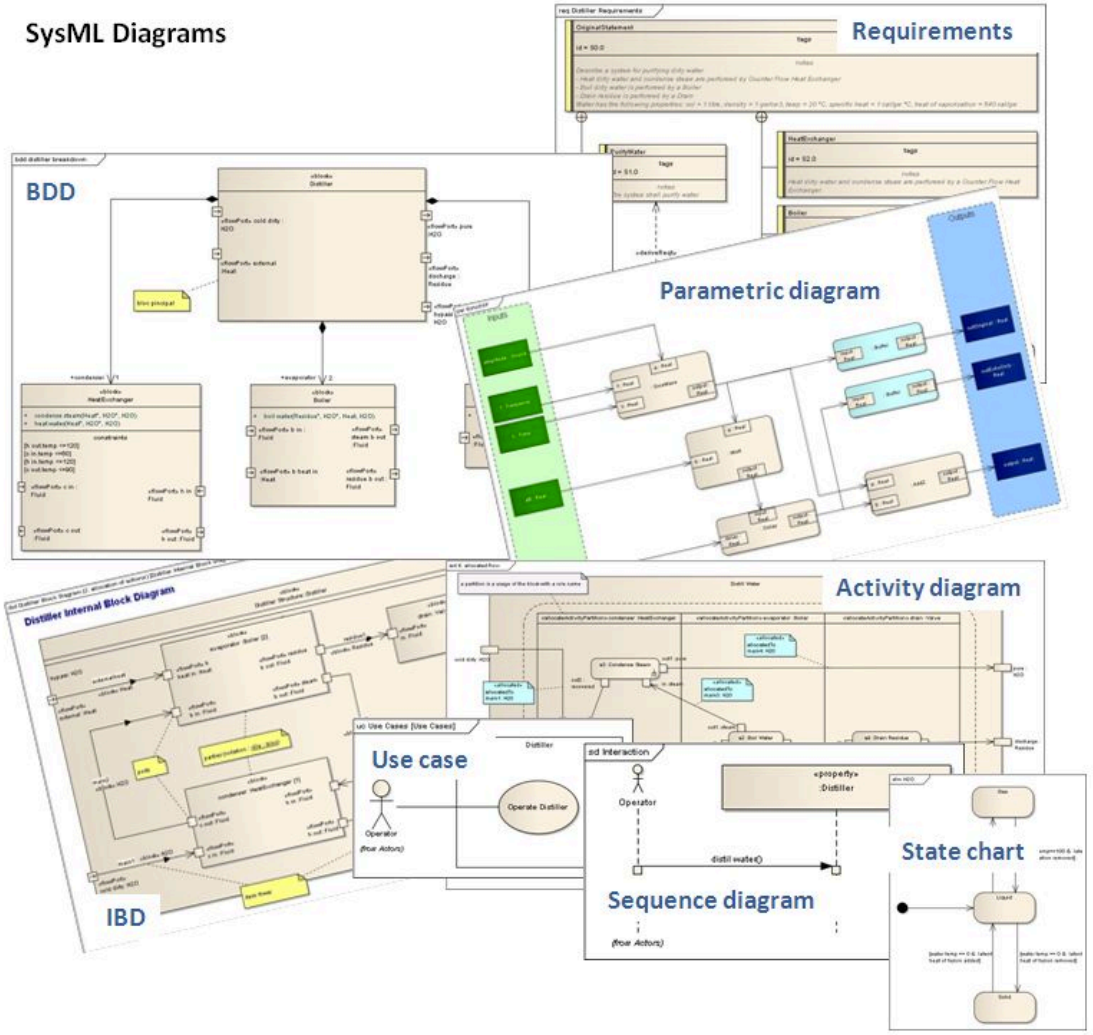
Bibliography

- Lamsweerde, Axel van; "*Goal-Oriented Requirements Engineering: A Guided Tour*"; *Université Catholique de Louvain; Louvain-la-Neuve; Belgium; 2001*
- "A KAOS Tutorial"; Objectiver; 2003
- Lamsweerde, Axel van; "*Goal-Oriented Requirements Engineering: From System Objectives to UML Models to Precise Software Specification*"; *Université Catholique de Louvain; Louvain-la-Neuve; Belgium; May 2003*
- Lamsweerde, Axel van; "*Requirements Engineering - - from system goals to UML models to software specifications*"; *Wiley, 2009*

Summary

- Introduction
- System Engineering
- Systems Requirements
- Requirements elicitation process
- KAOS overview
- SysML overview
- Requirements in SysML
- Mapping KAOS models into SysML models
- Practical case study

SysML diagrams

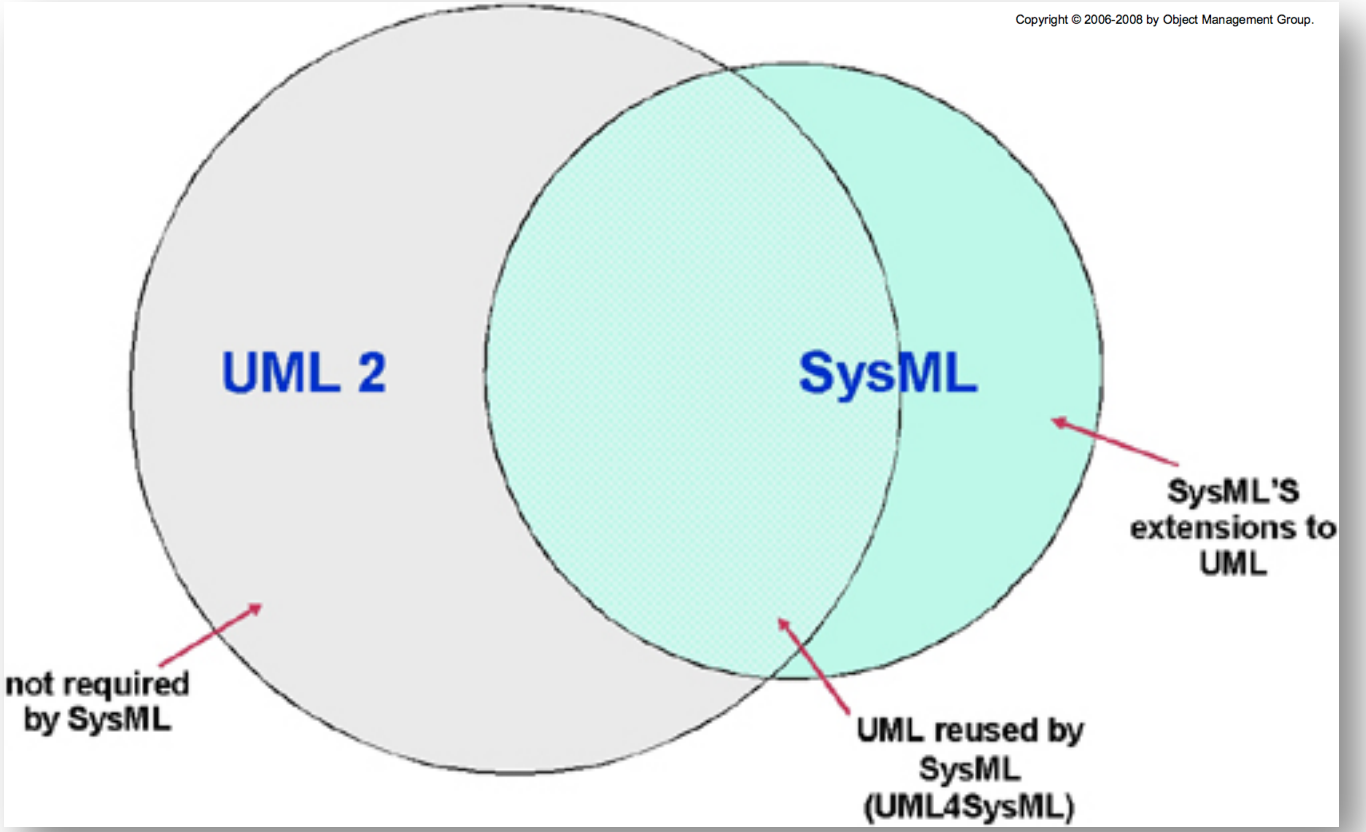


SysML: identity card

- Date of birth: 2001!
- Current version: 1.3 (June 2012)
- Father: OMG/UML + INCOSE
- Leading authors
  - Conrad Bock
  - CrisKobryn
  - Sanford Friedenthal

SysML / UML

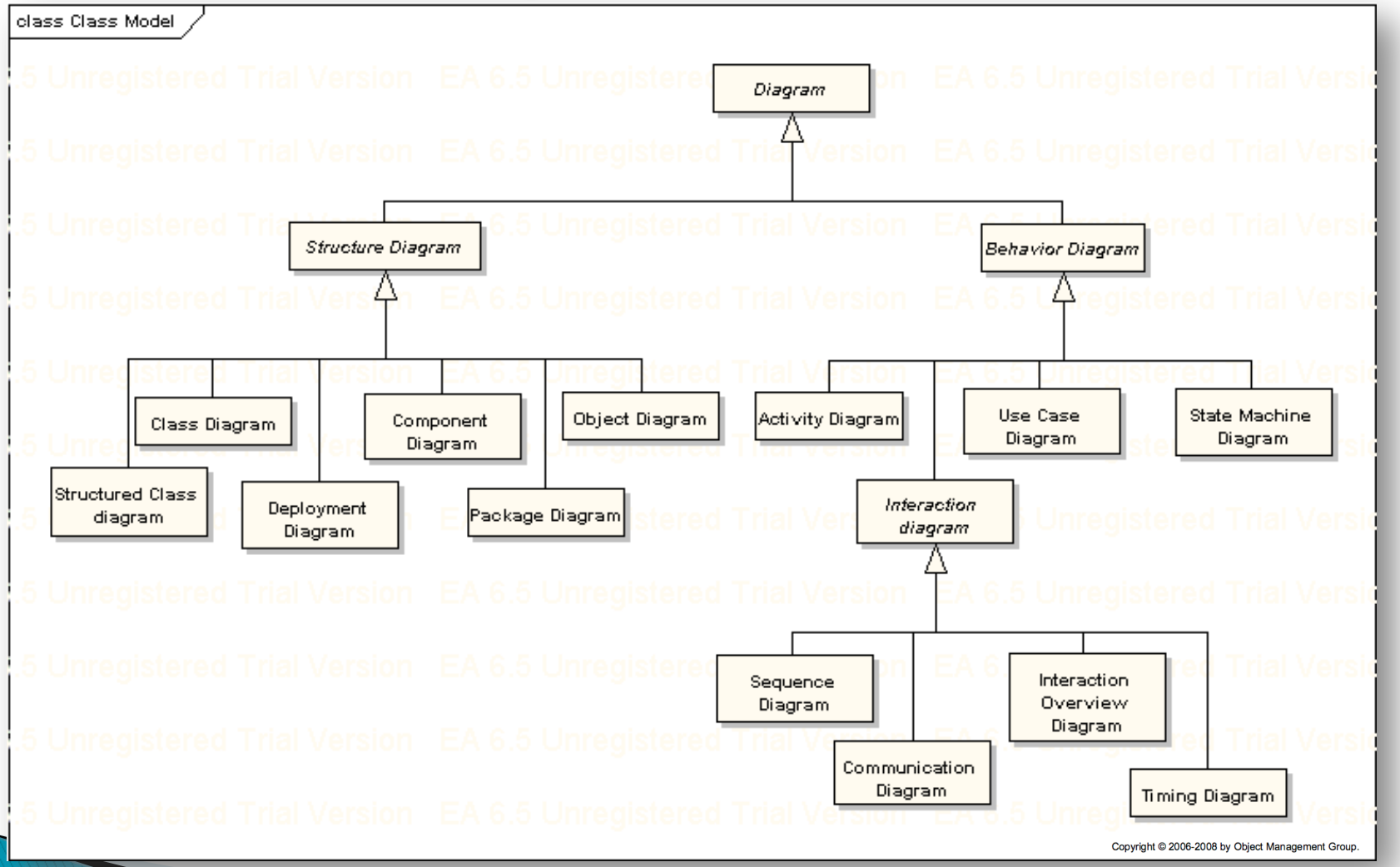
- Relationship between the two



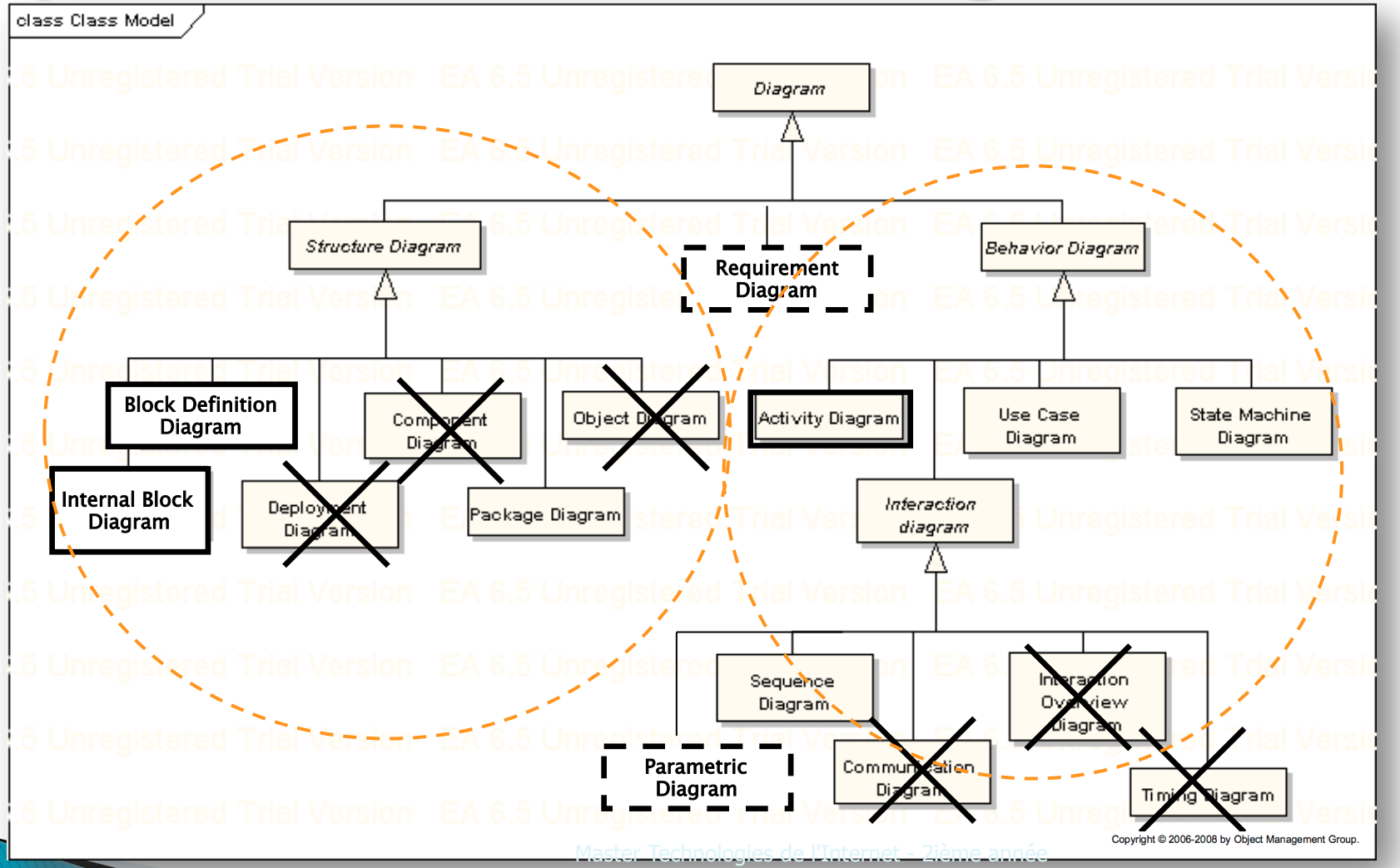
SysML: who's behind

- Industry
  - American Systems, BAE Systems, Boeing, Deere & Company, EADS Astrium, Eurostep, Israel Aircraft Industries, Lockheed Martin, Motorola, NIST, Northrop Grumman, oose.de, Raytheon, Thales, …
- Tool vendors
  - Artisan, EmbeddedPlus, Gentleware, IBM, Mentor Graphics, PivotPoint Technology, Sparx Systems, vitech, …
- Other organisations
  - AP-233, INCOSE, Georgia Institute of Technology, AFIS, …

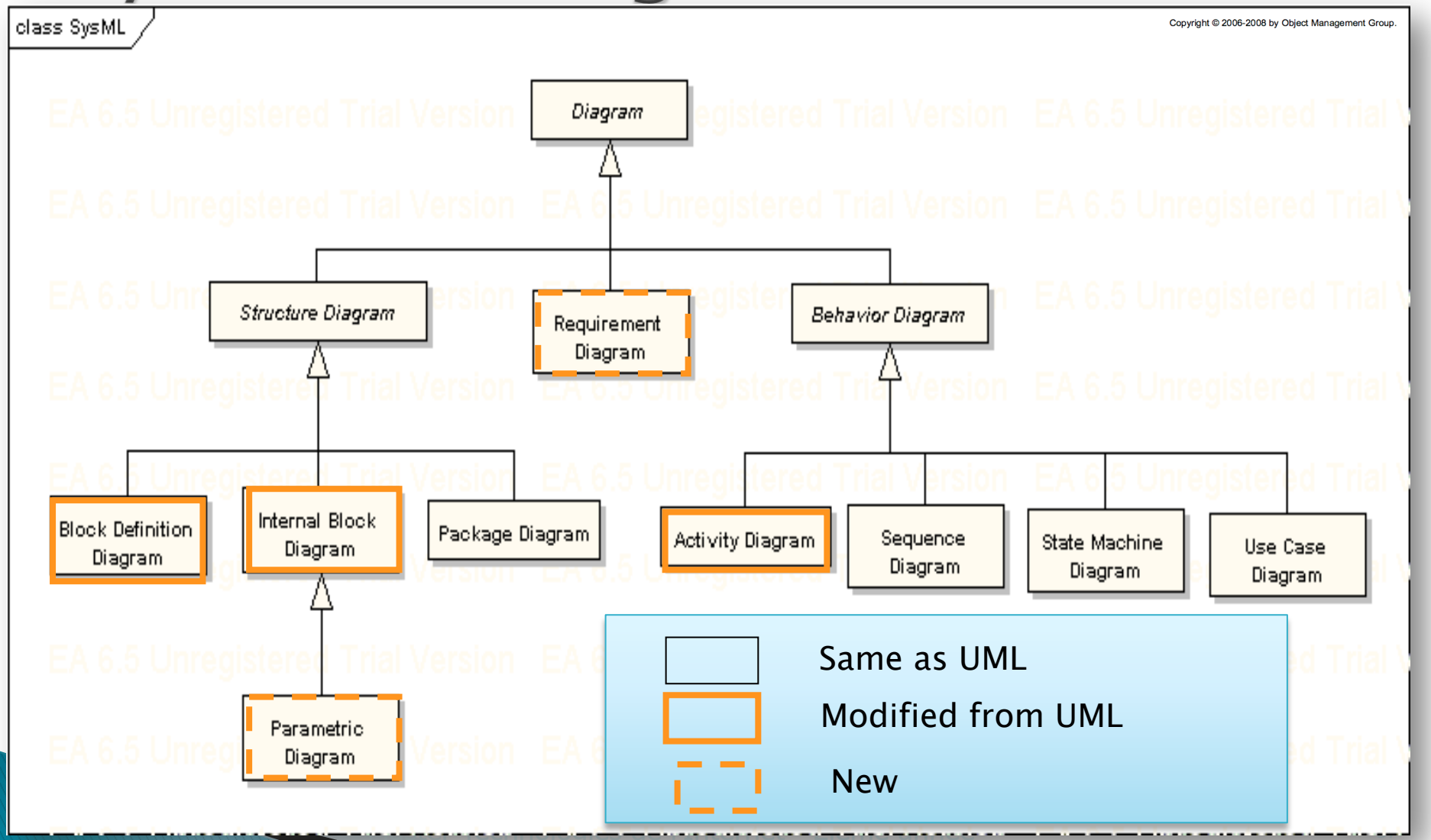
UML: 13 diagrams (in 2001)



SysML: ~~13~~-7+2=9 diagrams

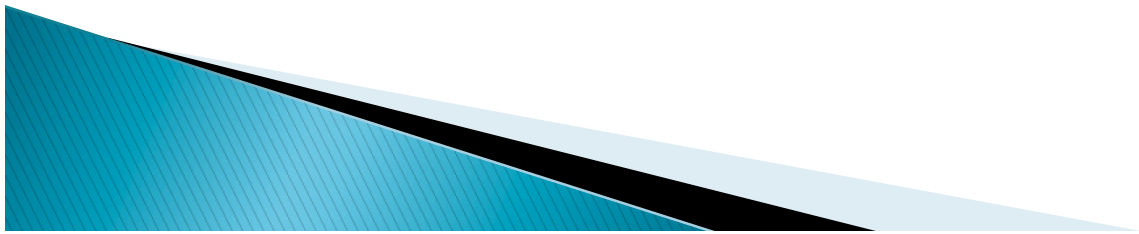


SysML 1.3 diagrams



SysML overview

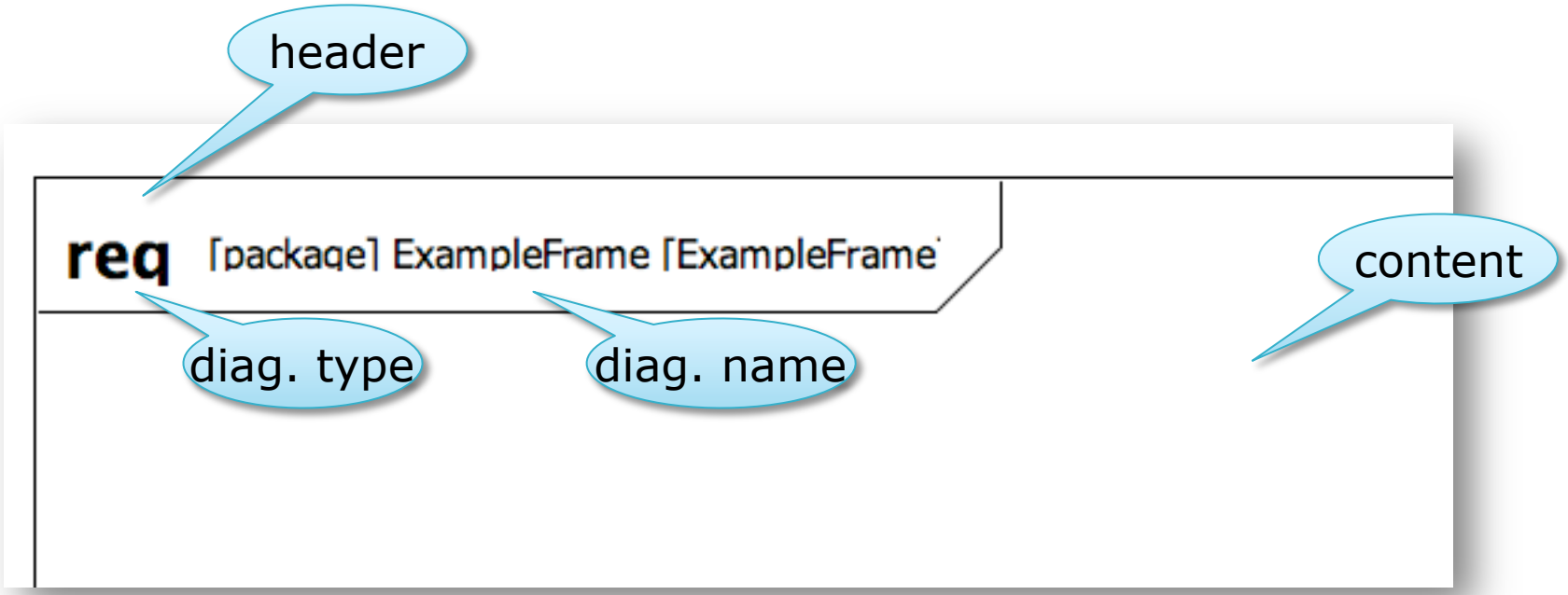
- Diagram notation
- Structure diagrams
- Behavioral diagrams
- Cross-cutting constructs



SysML diagram frames

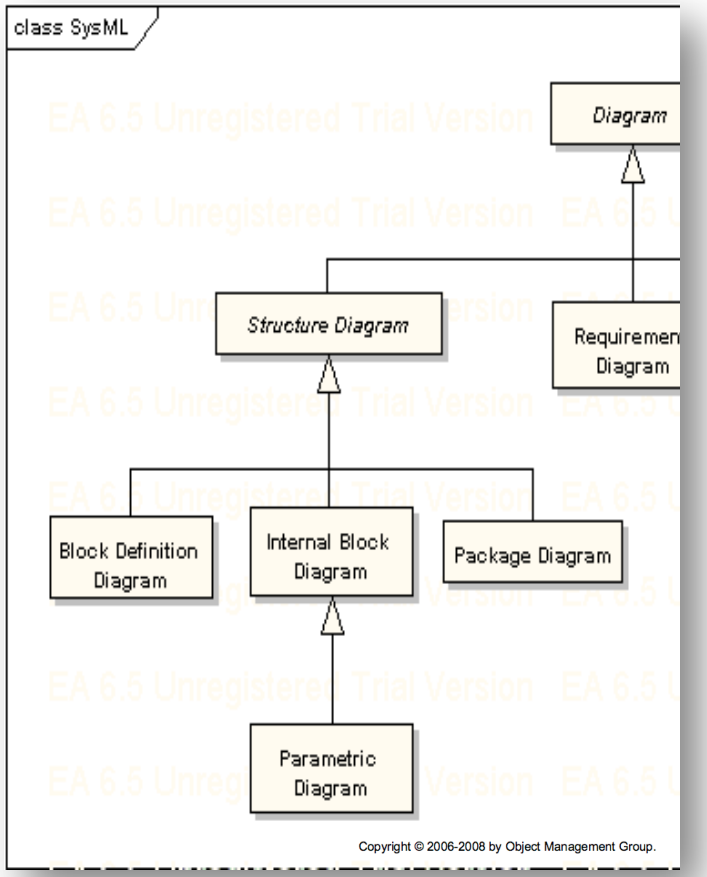
- Each SysML diag. represents a model element
- Each SysML diag. must have a Diagram Frame
- Diagram context is indicated in the header:
  - Diagram kind (req, act, bdd, ibd, sd, etc.)
  - Model element type (package, block, activity, etc.)
  - Model element name
  - User defined diagram name or view name
- A separate diagram description block is used to indicate if the diagram is complete, or has elements elided

SysML diagram frames (e.g.)



SysML structure diagrams

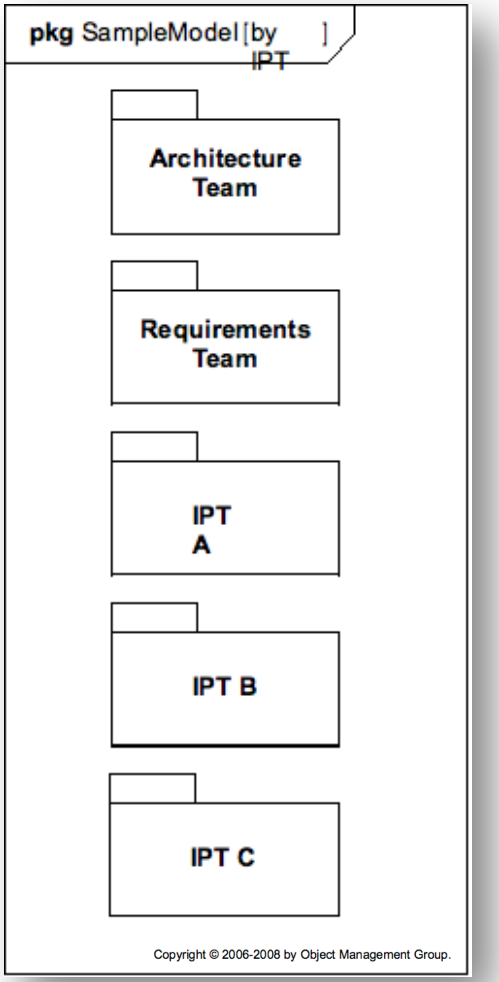
- Package
- Block Definition
- Internal Block
- Parametric



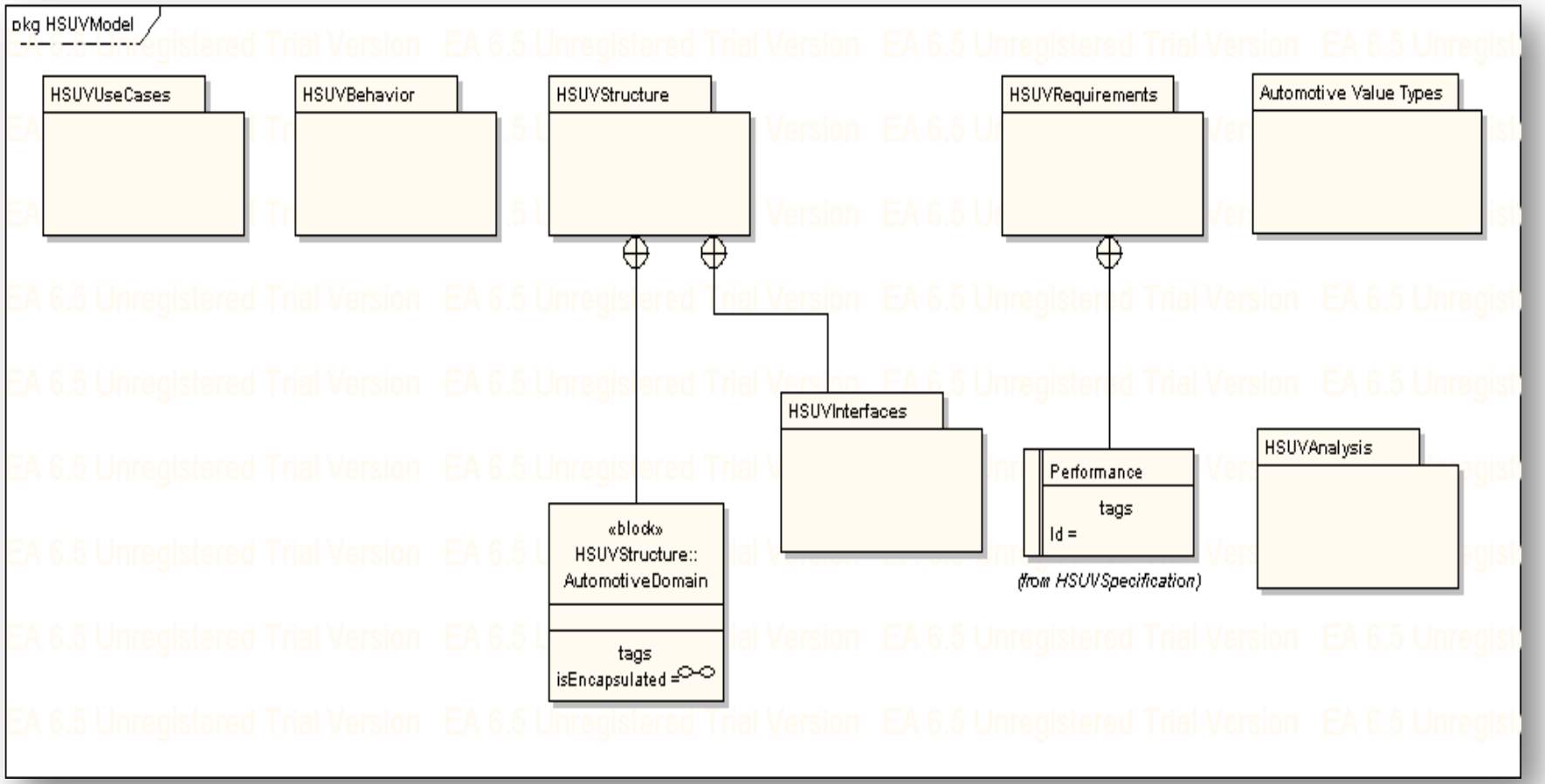
Package Diagrams (pkg)

- Same as UML
  - to organize the model
  - name space
- Model can be organized in multiple ways:
  - System hierarchy
    - e.g., enterprise, system, component
  - Diagram kind
    - e.g., requirements, use cases, behavior
  - Use viewpoints to augment model organization

Package Diagrams (e.g.)

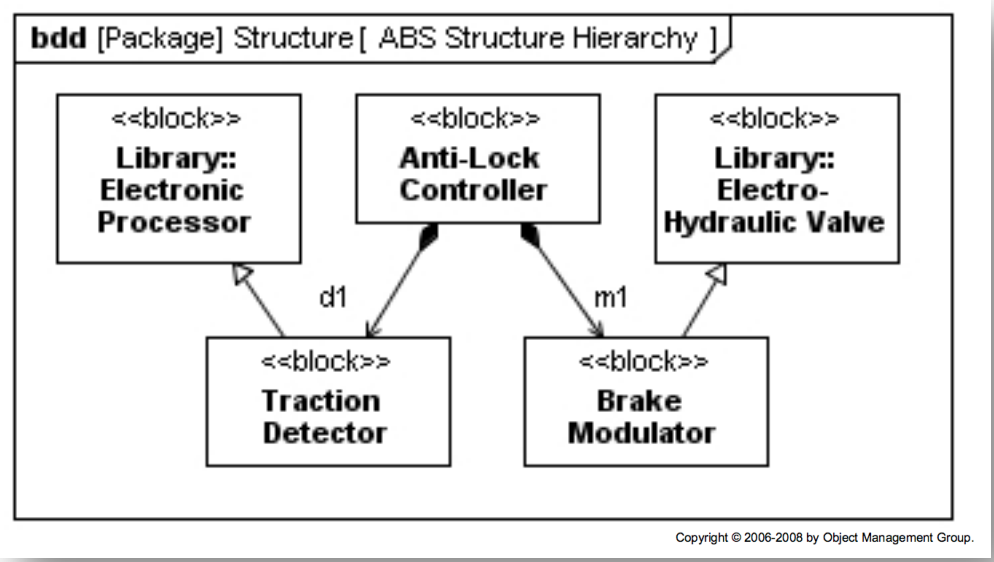


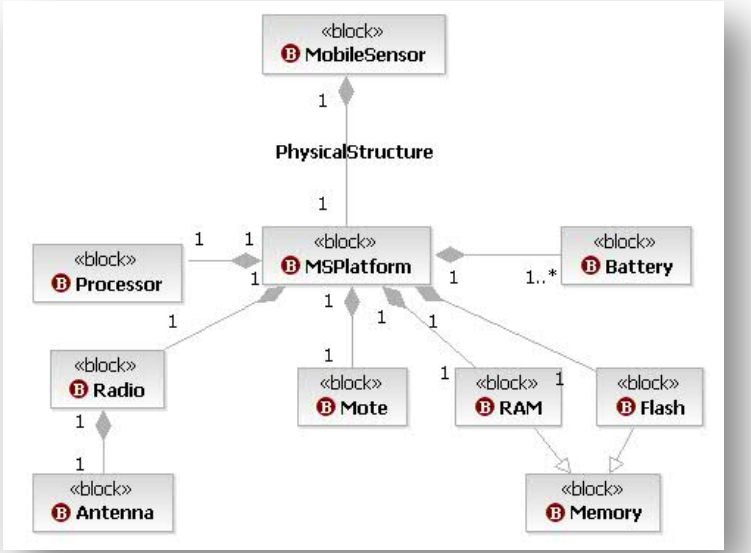
Package Diagrams (links)



Block Definition Diagrams (bdd)

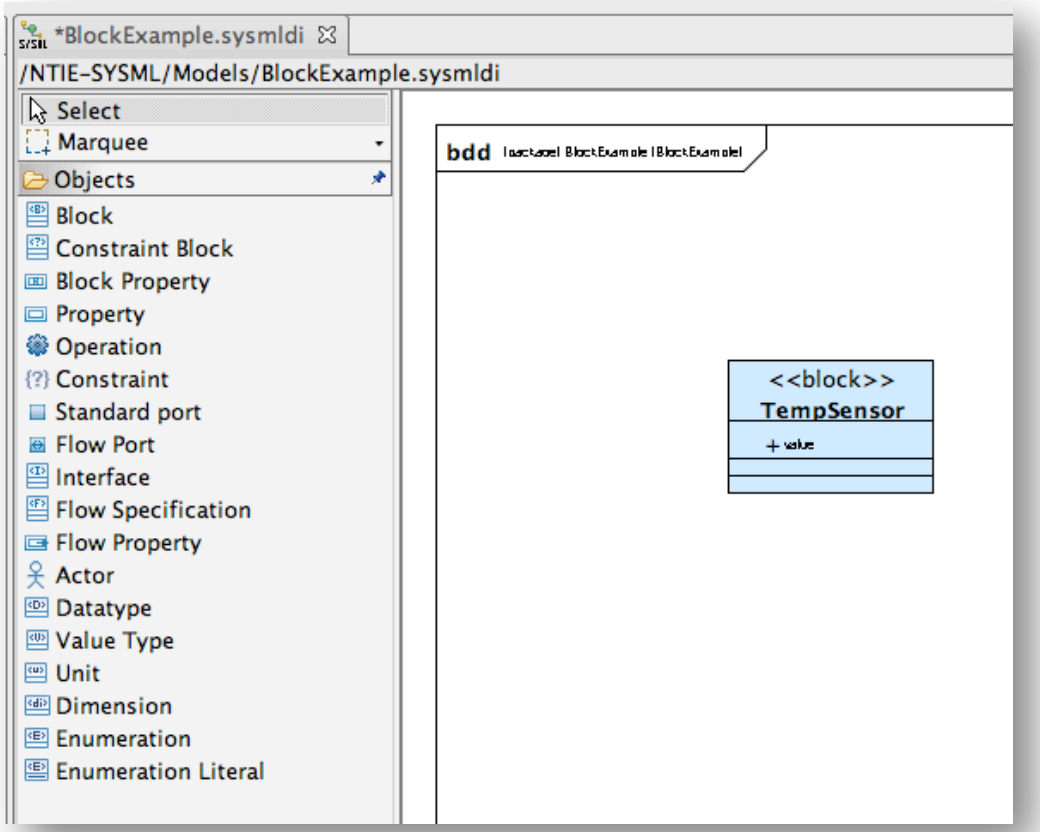
- Classes are dead… welcome to blocks!
  - Can be anything (System, Hardware, Software, Data, Procedure, Facility, Person)
  - Satisfy Systems Engineers





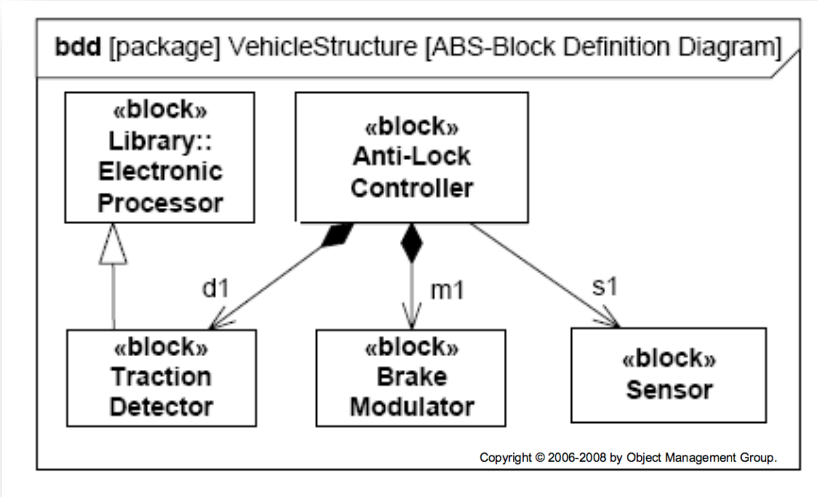
Block Definition Diagrams (bdd)

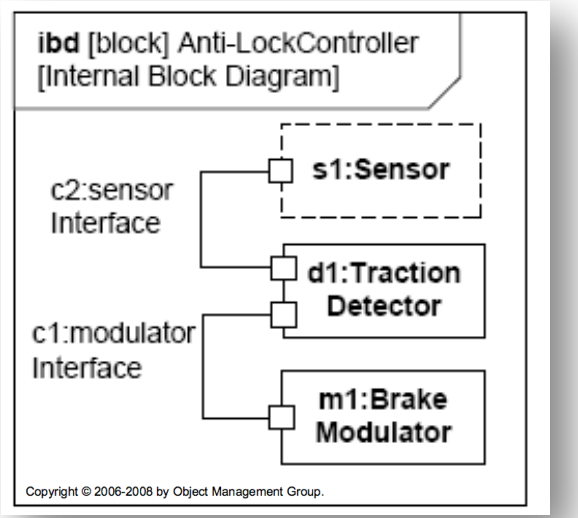
- Compartments
  - Properties
  - Operations
  - Constraints
  - Allocations
  - Requirements
  - User defined!



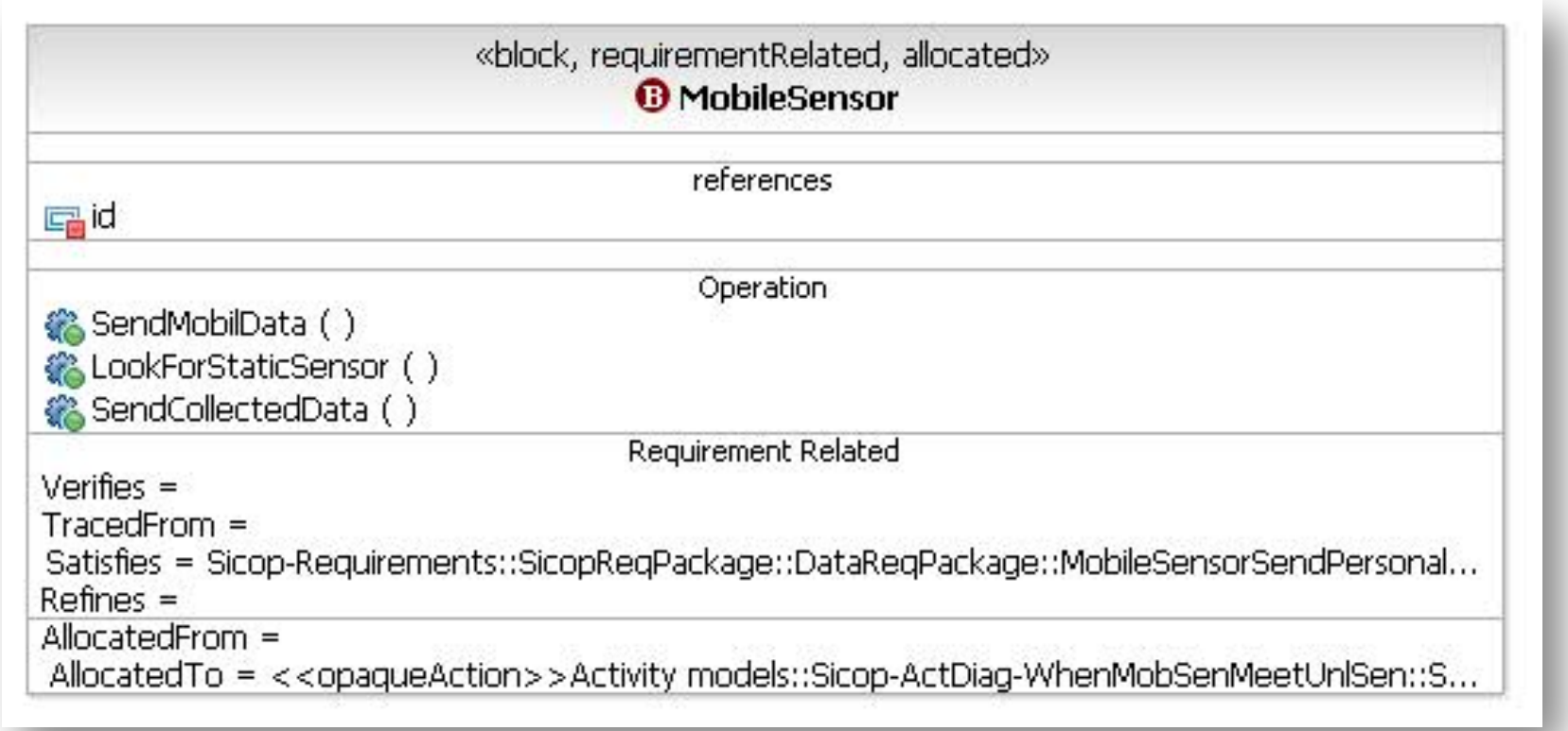
Block Definition vs. Usage

- Diagramme de Définition de blocs (BDD)
  - Décrit les relations entre les blocks (composition, généralisations…)
- Diagramme Interne de bloc (IBD)
  - Décrit la structure interne d'un bloc sous forme de *parts*, *ports* et *connecteurs*.

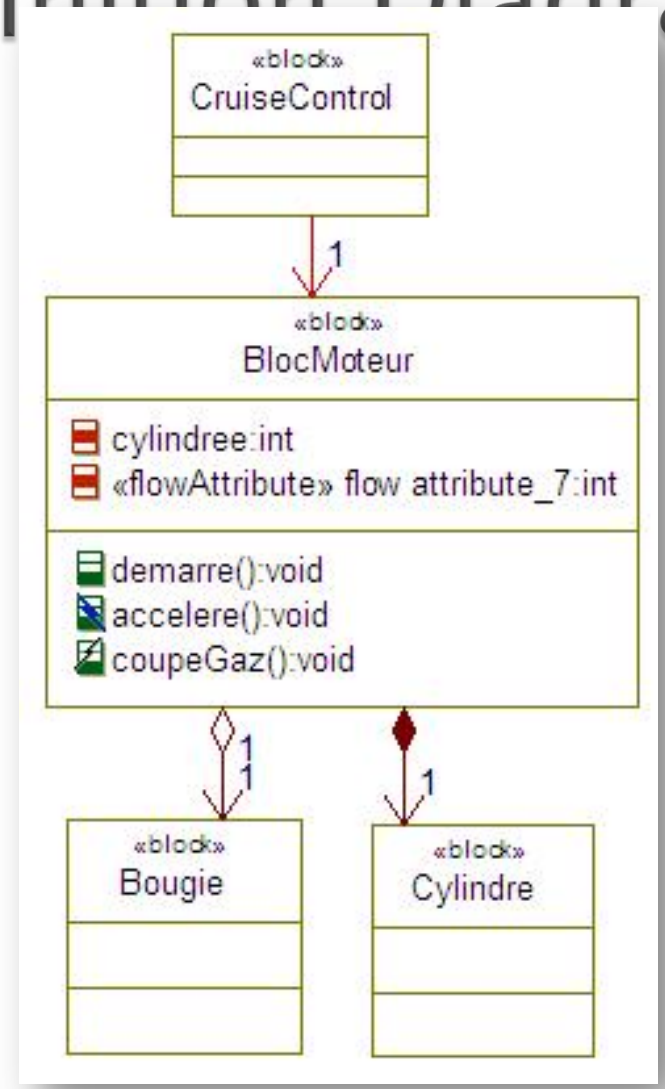




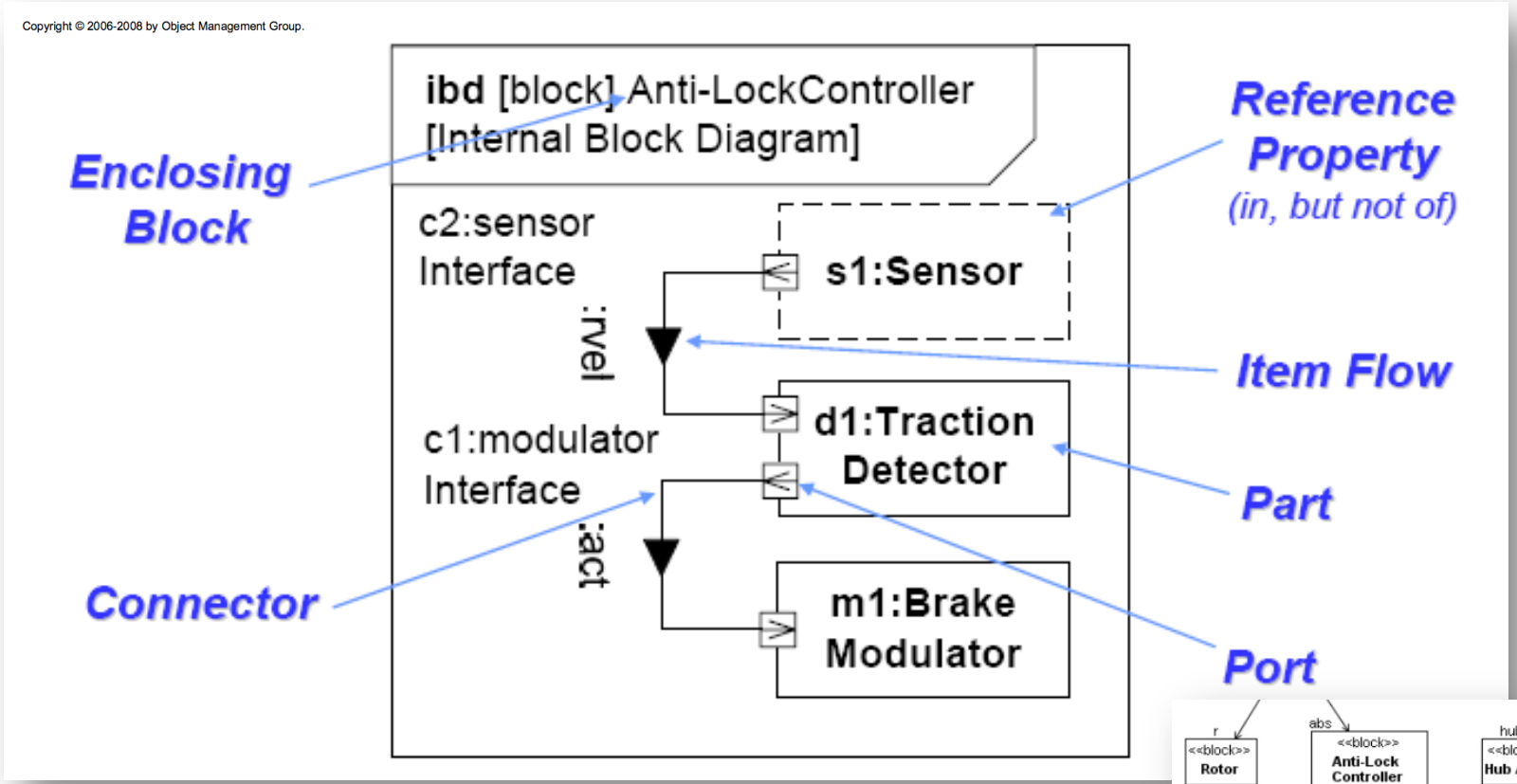
Block example



Block Definition Diagram (e.g.)

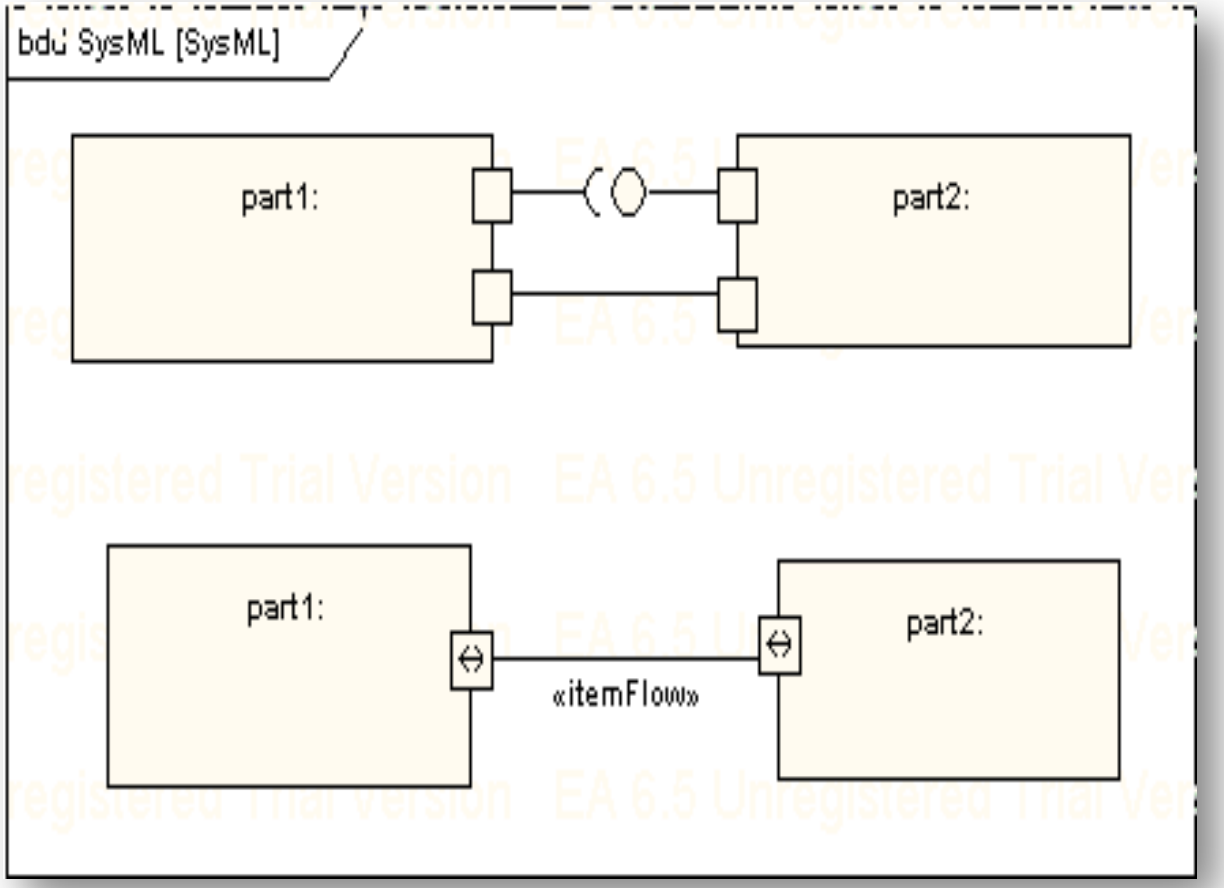


Internal Block Diagram (ibd)



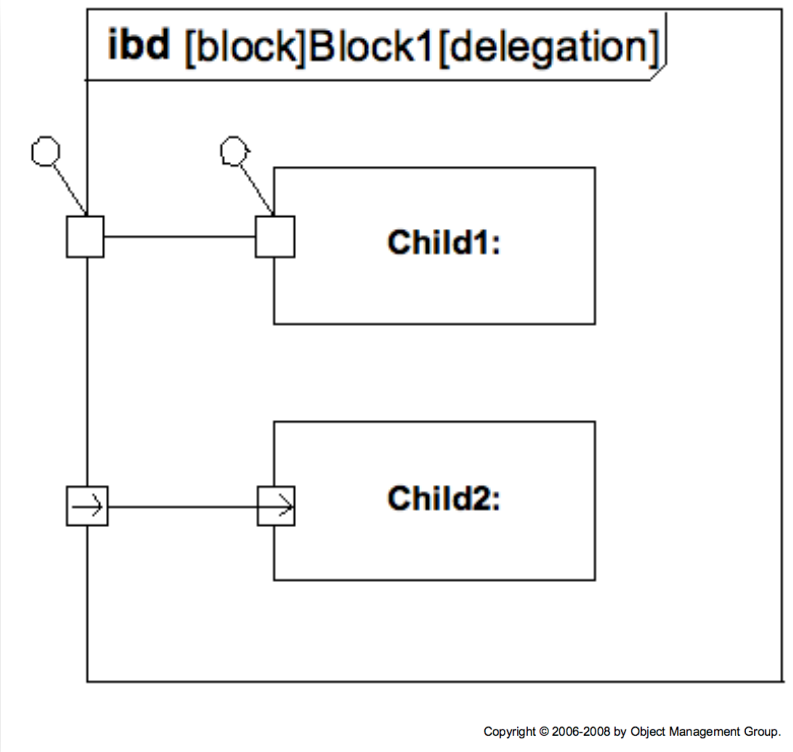
SysML Ports (cont.)

- Standard
- Flow

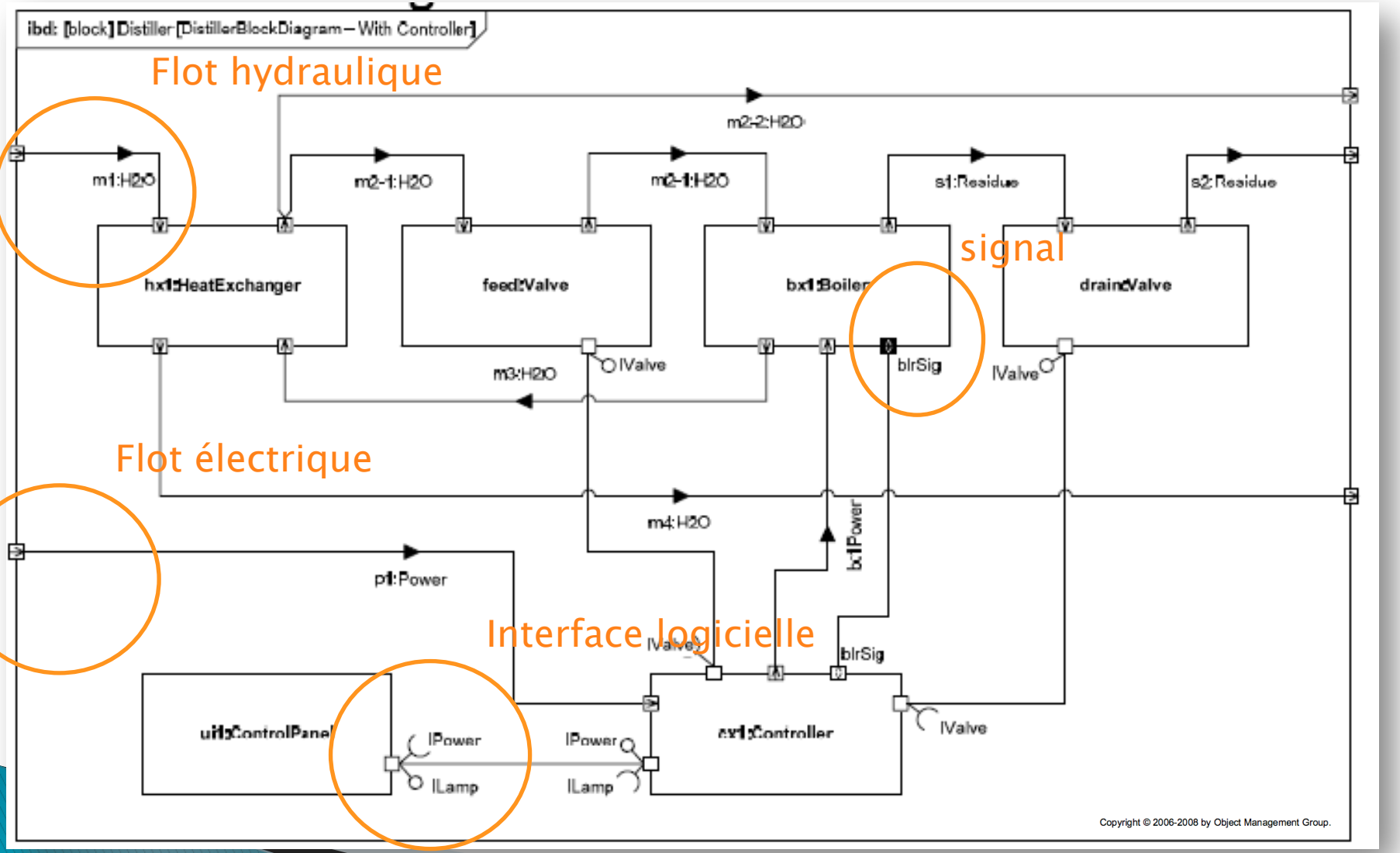


SysML Ports (delegation)

- to preserve encapsulation of block
- interactions at outer ports are delegated to ports of child parts
- ports must match
  - same kind, type, direction, etc.
- connectors can cross boundary without requiring ports at each level of nested hierarchy



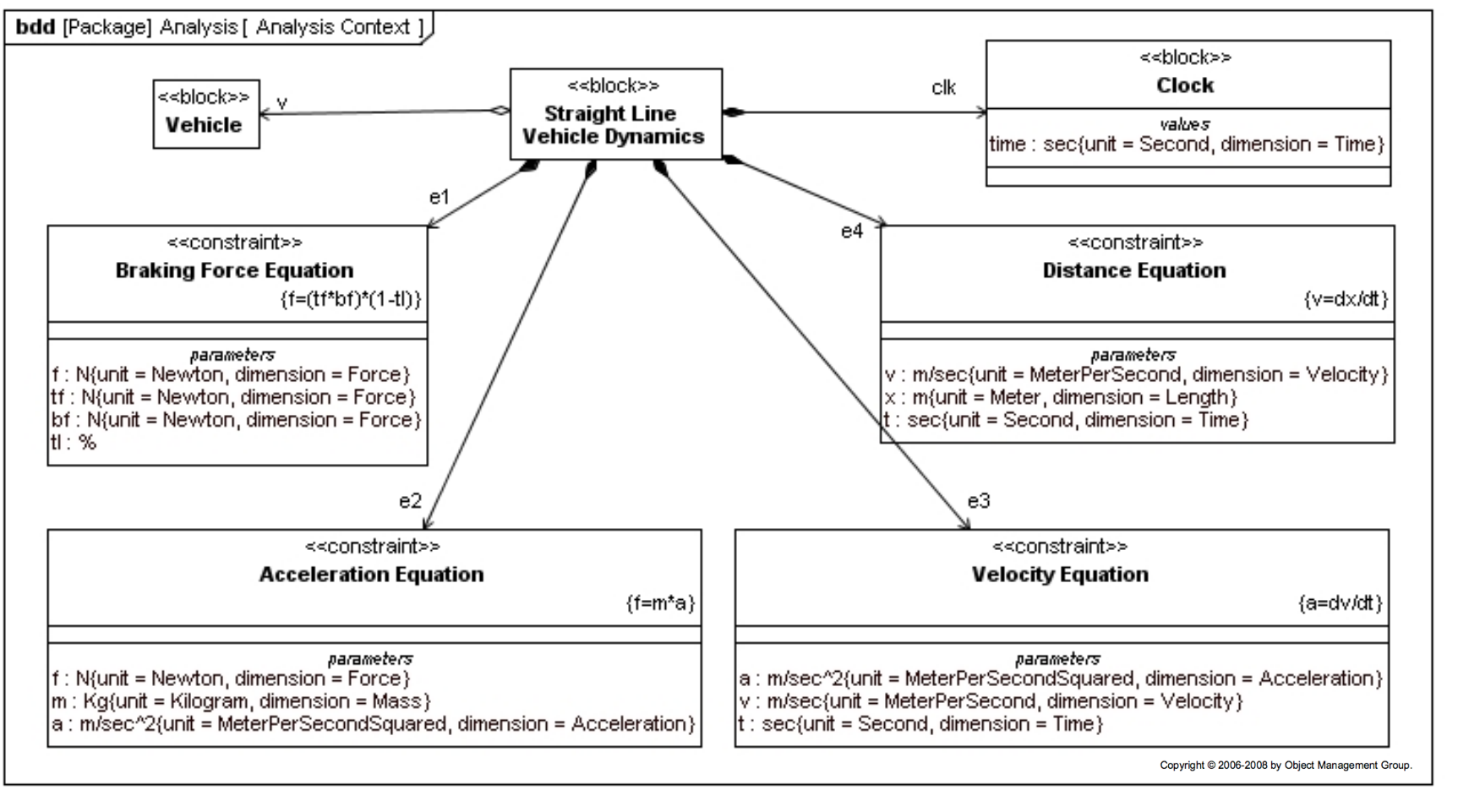
SysML Ports (e.g.)



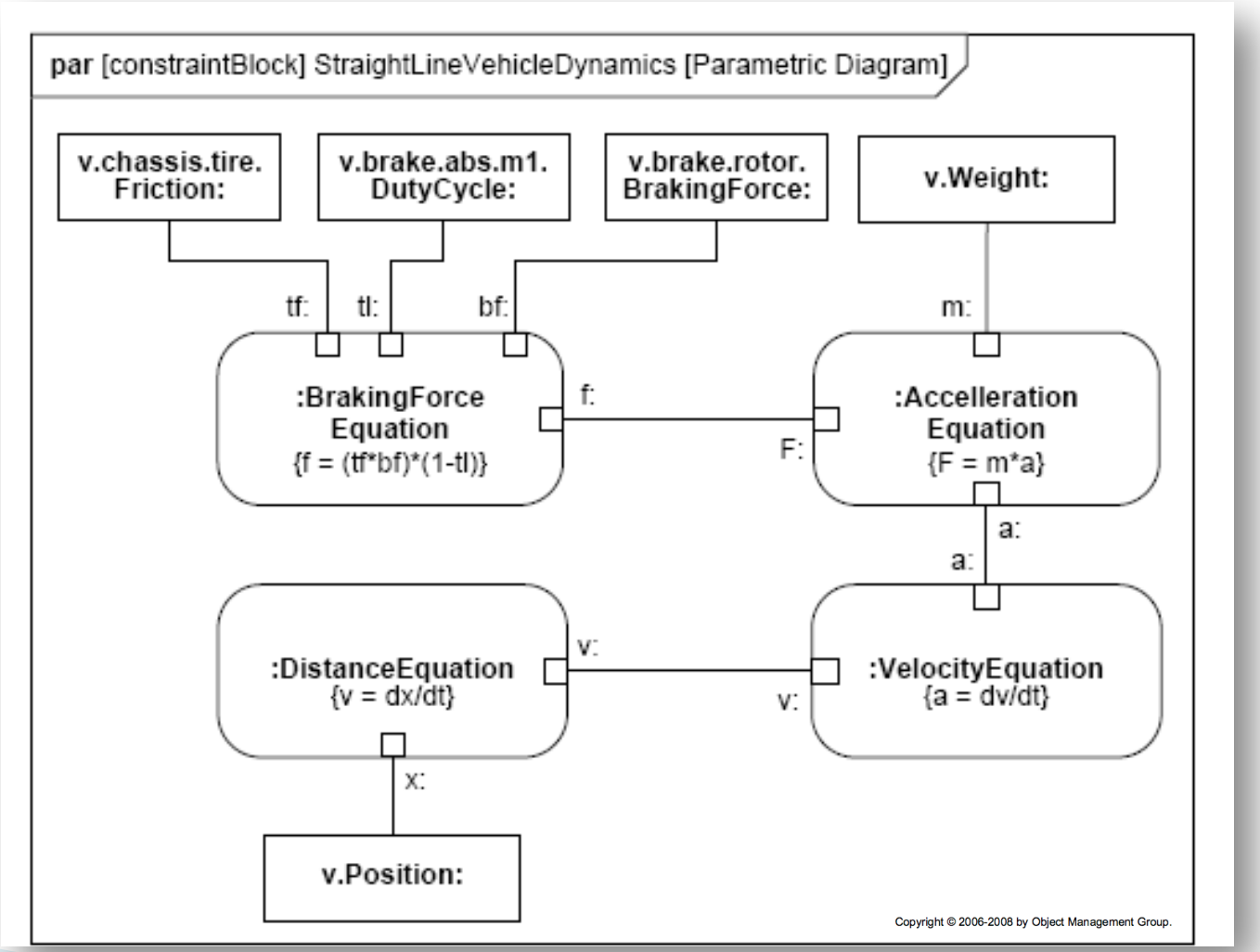
Parametric Diagrams (par)

- To express constraints between value properties
  - equations
  - support for engineering analysis (e.g., performance)
  - identification of critical performance properties
- Constraint block captures equations
  - Expression language can be formal (e.g., MathML, OCL)
  - Computational engine is not provided by SysML
- Parametric diagram
  - usage of the constraints in an analysis context

Block Definition Diagram

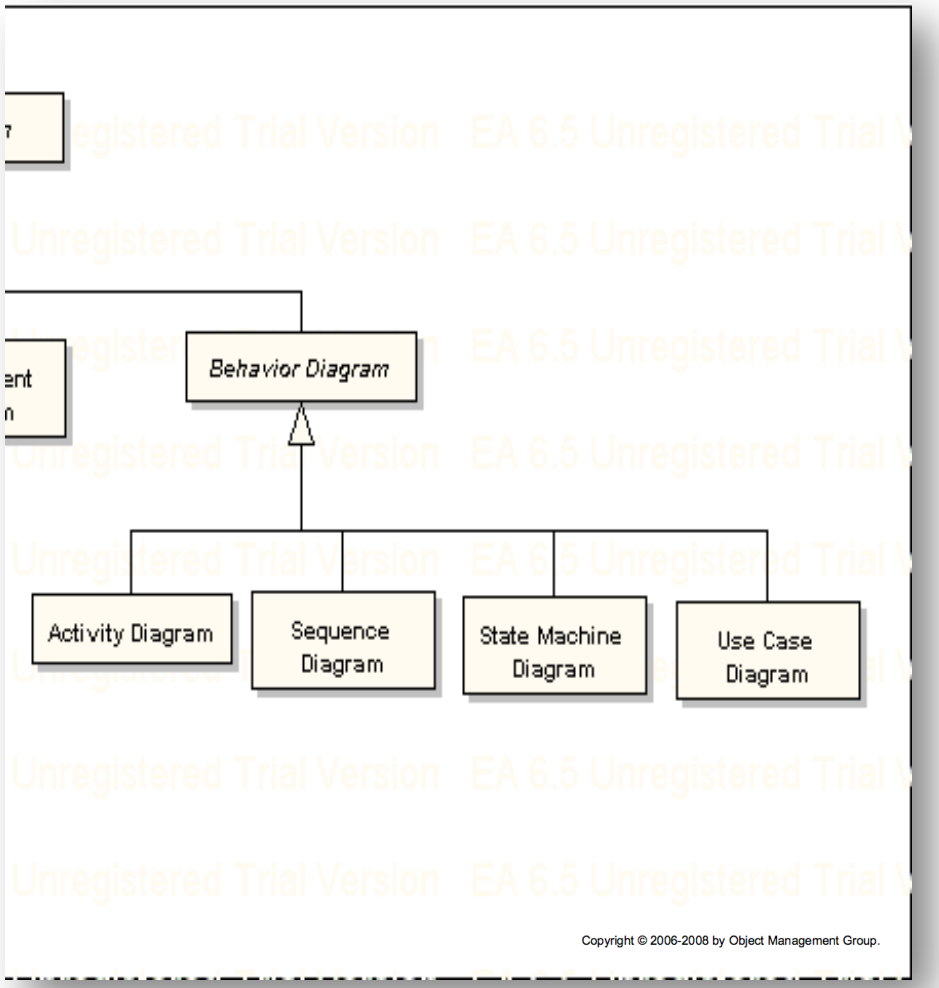


Parametrics (e.g. 1)



SysML behavioraldiagrams

- Activity
- Sequence
- State Machine
- Use Case



ActivityDiagrams (act)

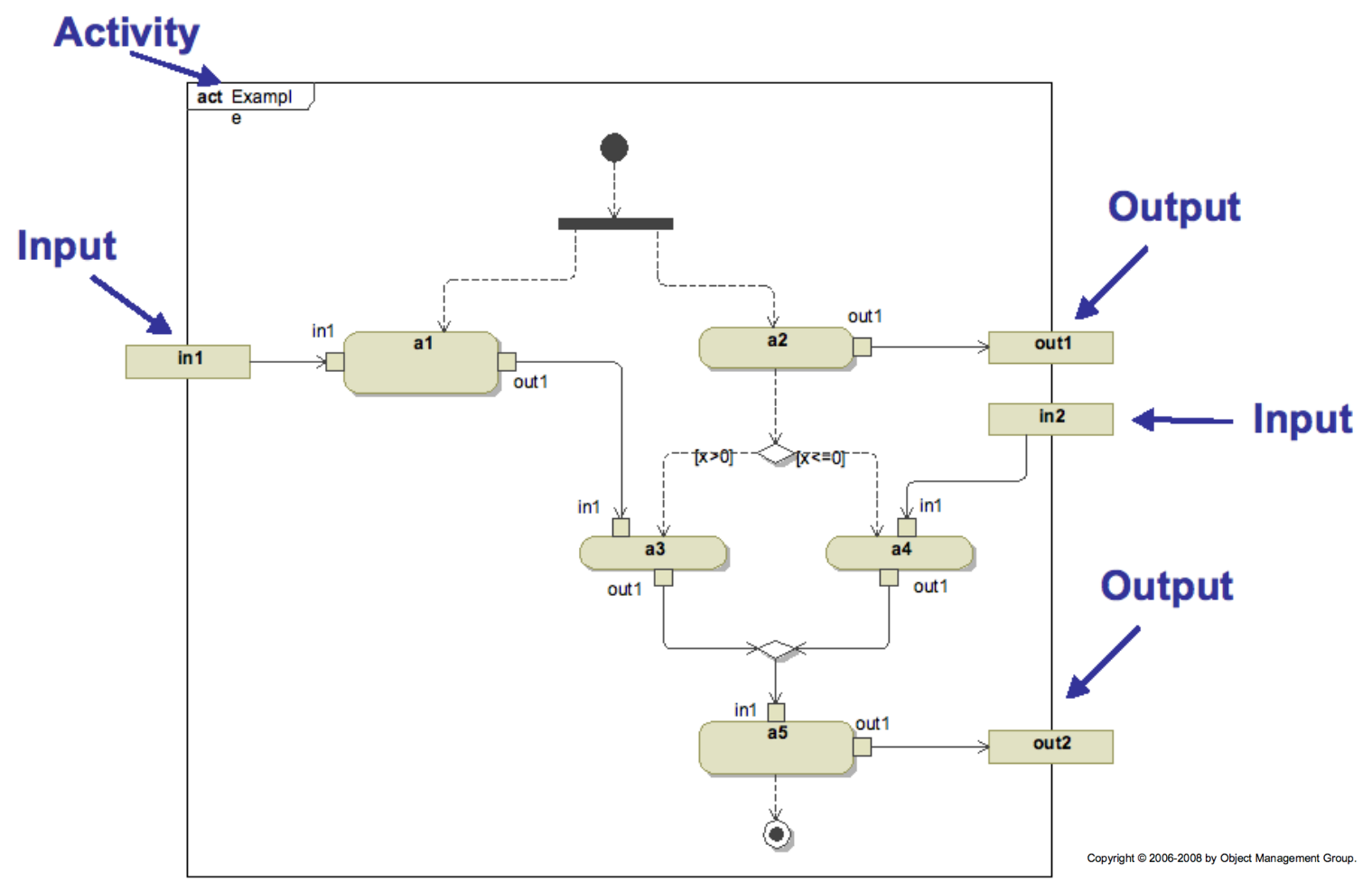
- to specify
  - controlled sequence of actions
  - the flow of inputs/outputs
  - control, including sequence and conditions for coordinate activities

- Swimlanes
  - to show responsibility of the activity

ActivityDiagrams (cont.)

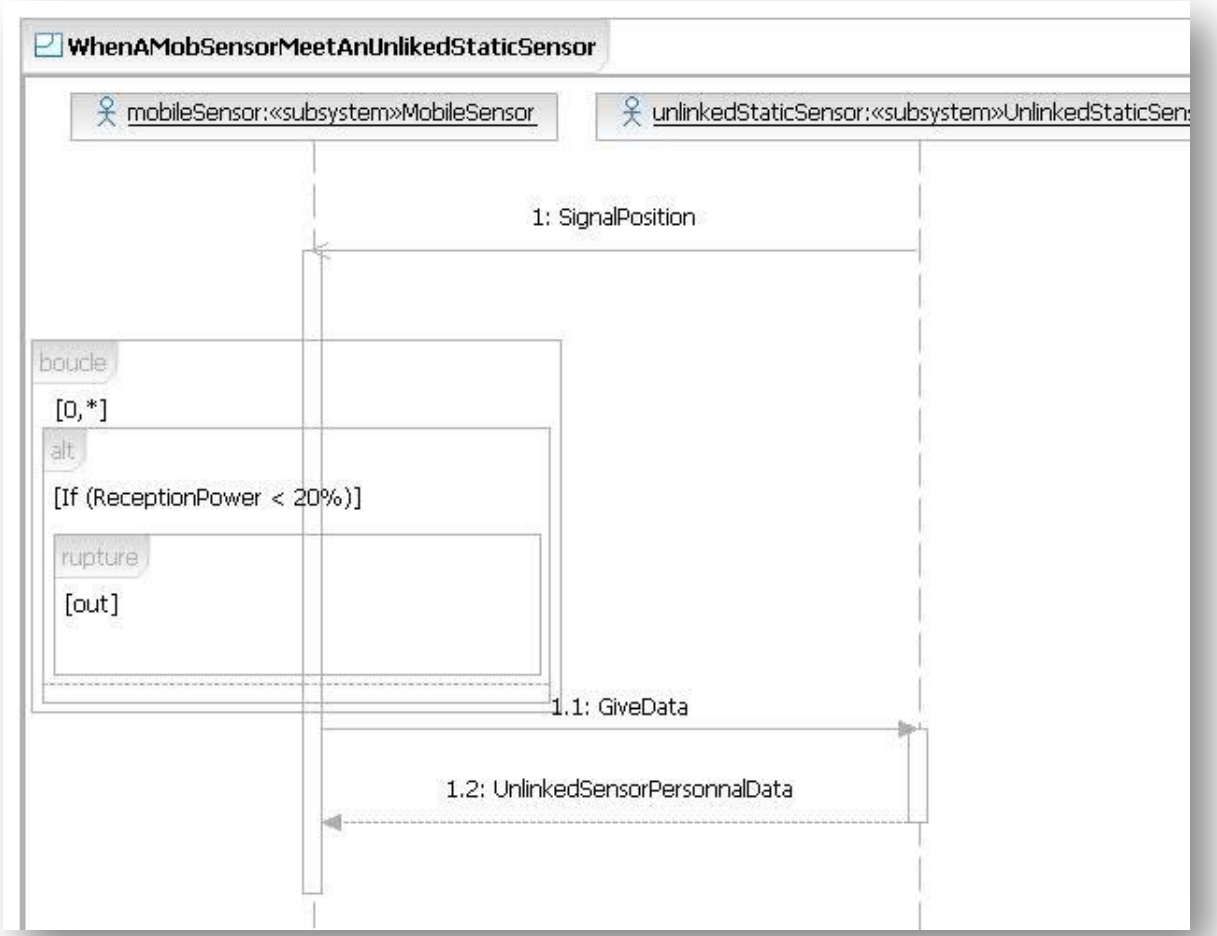
- Improvements from UML:
  - continuous or discrete flow
  - control operators
    - to start/stop other actions
  - Overwrite and NoBufferports
    - for continuous flows
  - probabilities on transitions or parameters

ActivityDiagrams (e.g.)



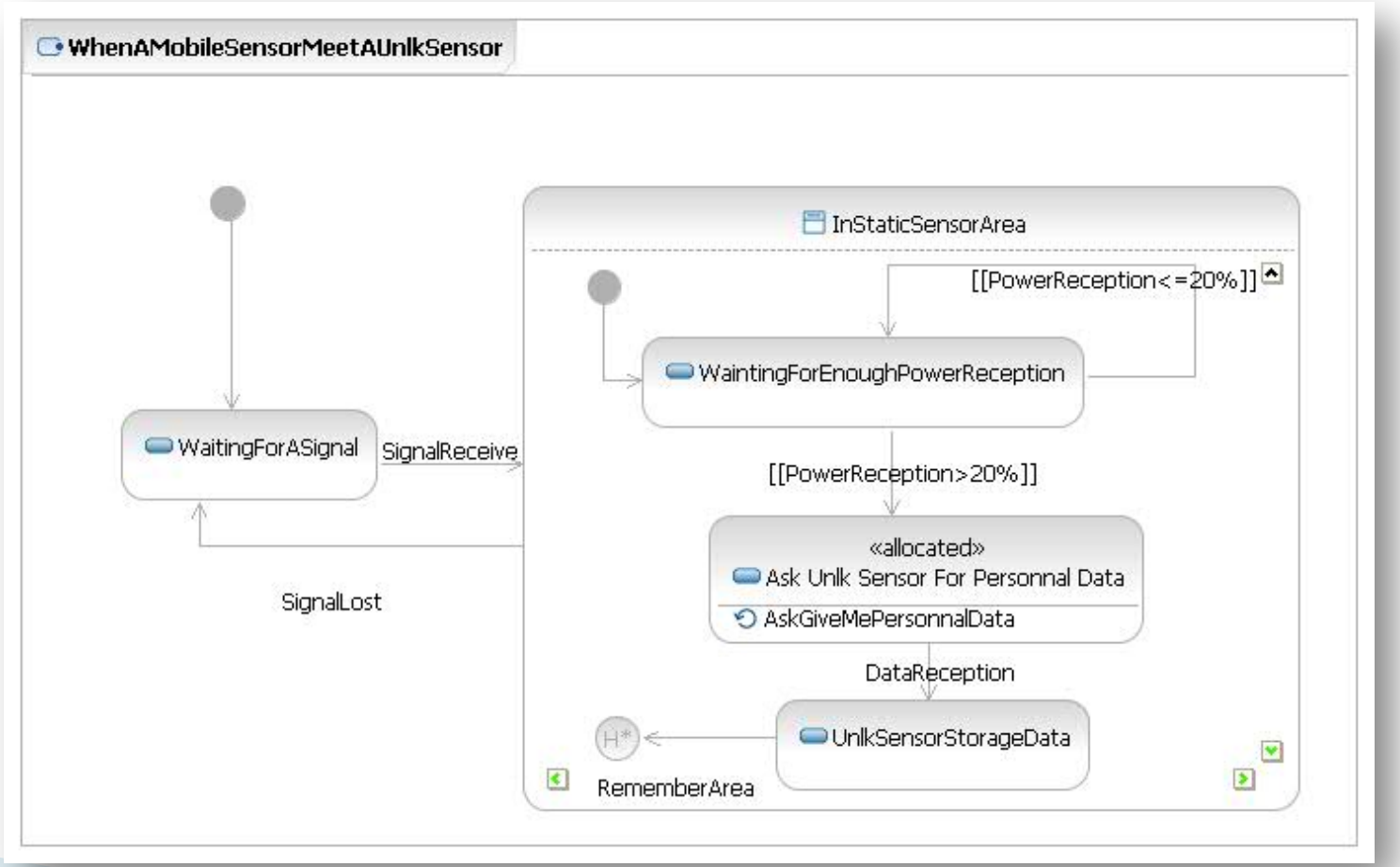
SequenceDiagrams (sd)

- Like UML



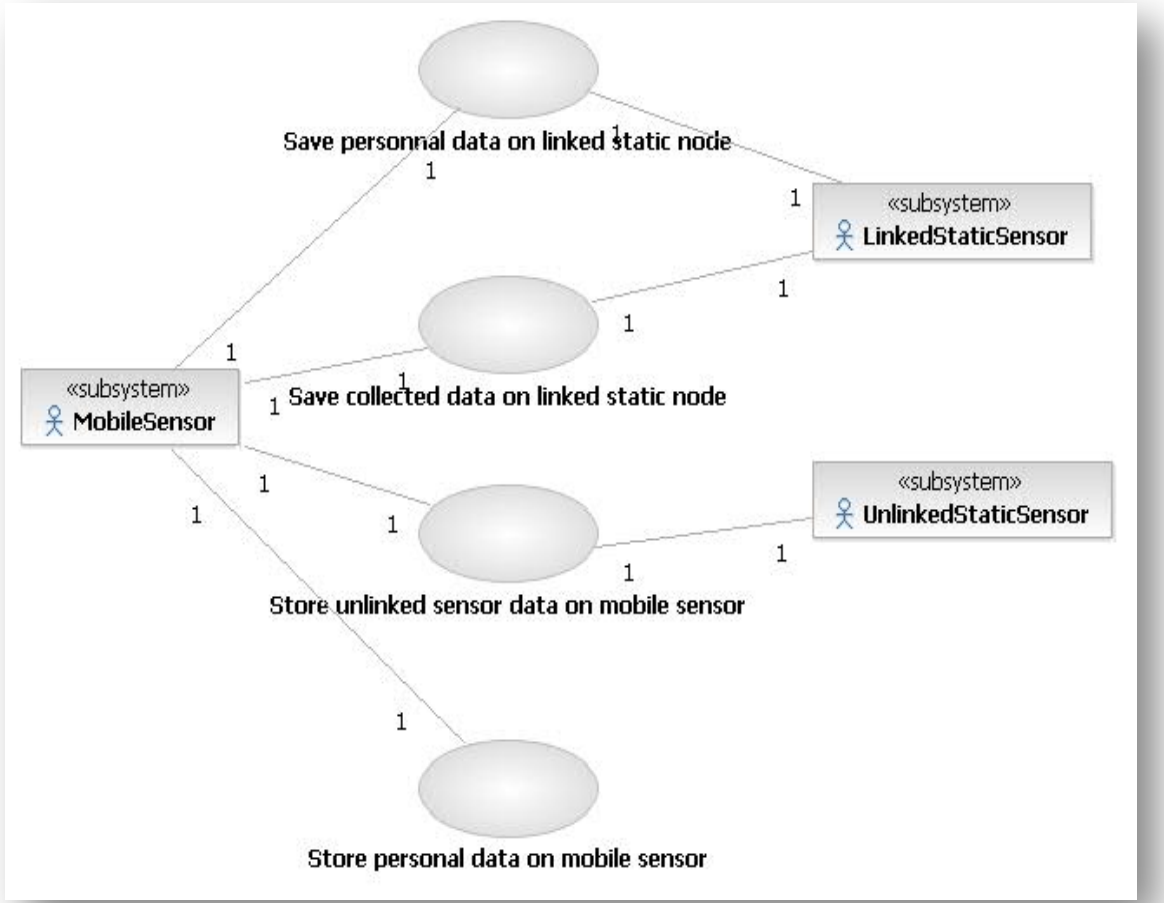
State Machine Diagrams (stm)

- Like UML



Use Case Diagrams (uc)

- Like UML



SysML cross-cutting constructs

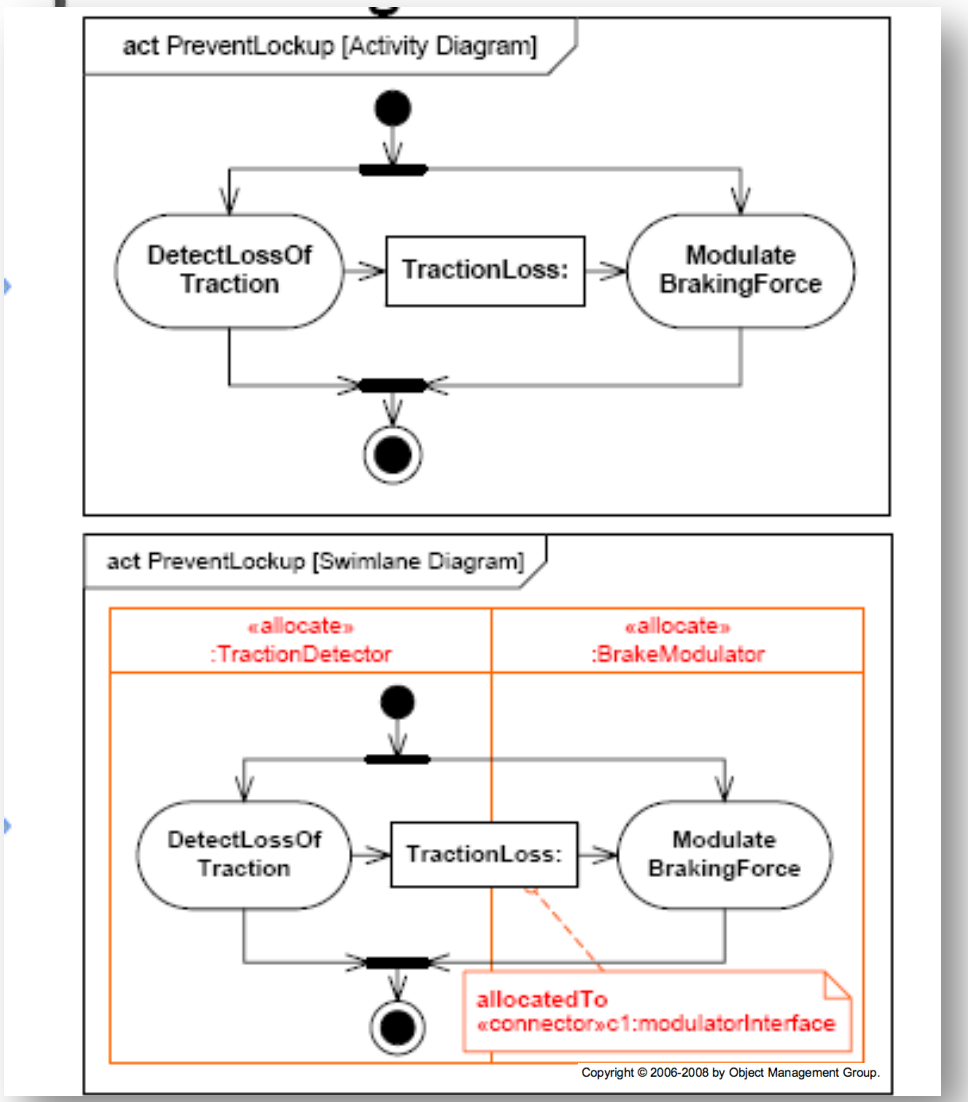
- Allocation
- Requirement Diagrams

Allocation

- General relationship between two elements of the model
- Different kinds of allocation:
  - Functionality – component
  - Logical component – physical component
  - Software – hardware
  - …
- Usable in a lot of different diagrams
- Usable under graphical or tabular representation

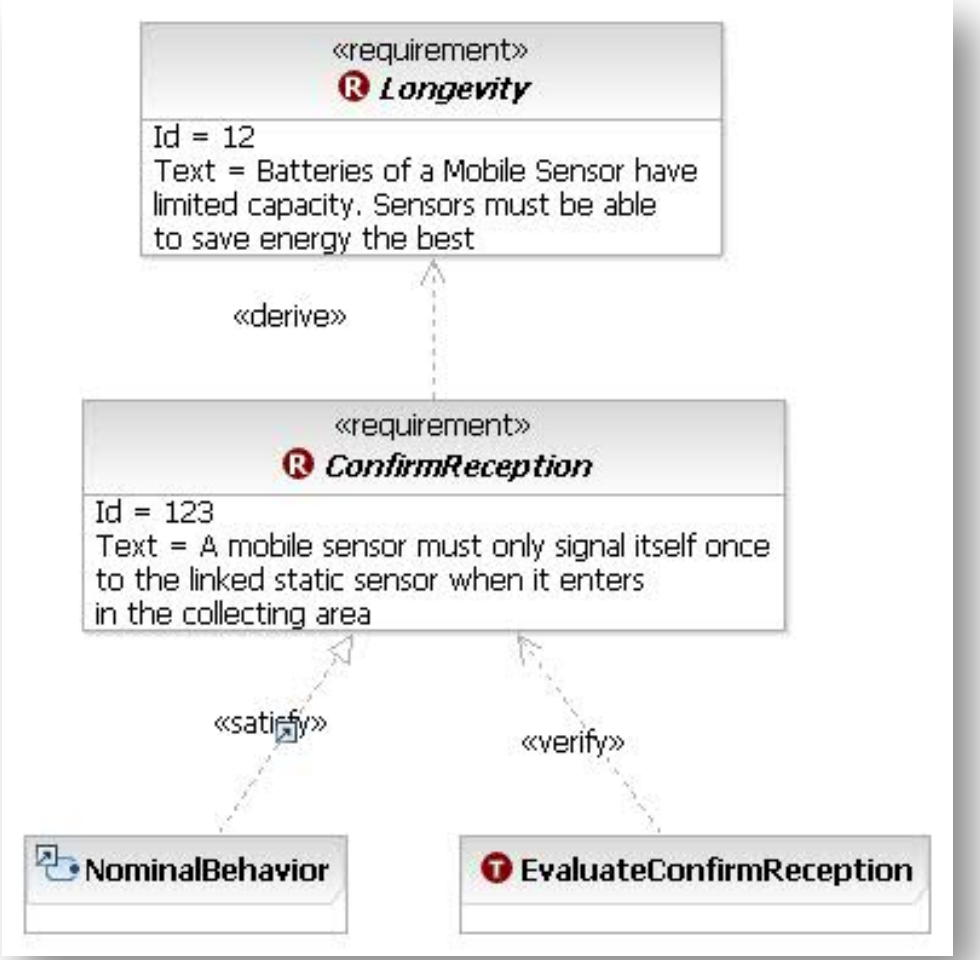
Allocation (e.g.)

- Use of swimlanes

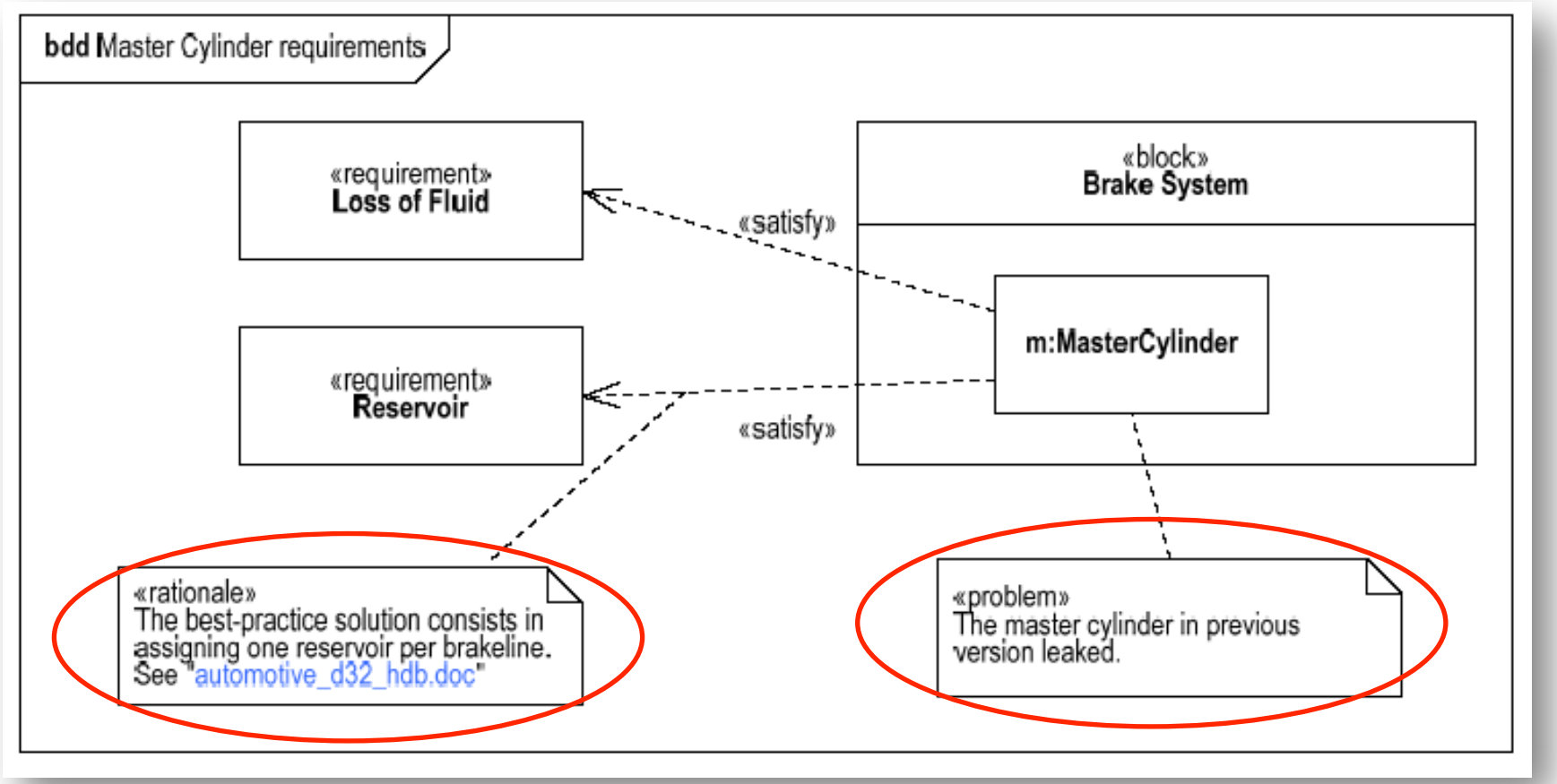


RequirementDiagrams (req)

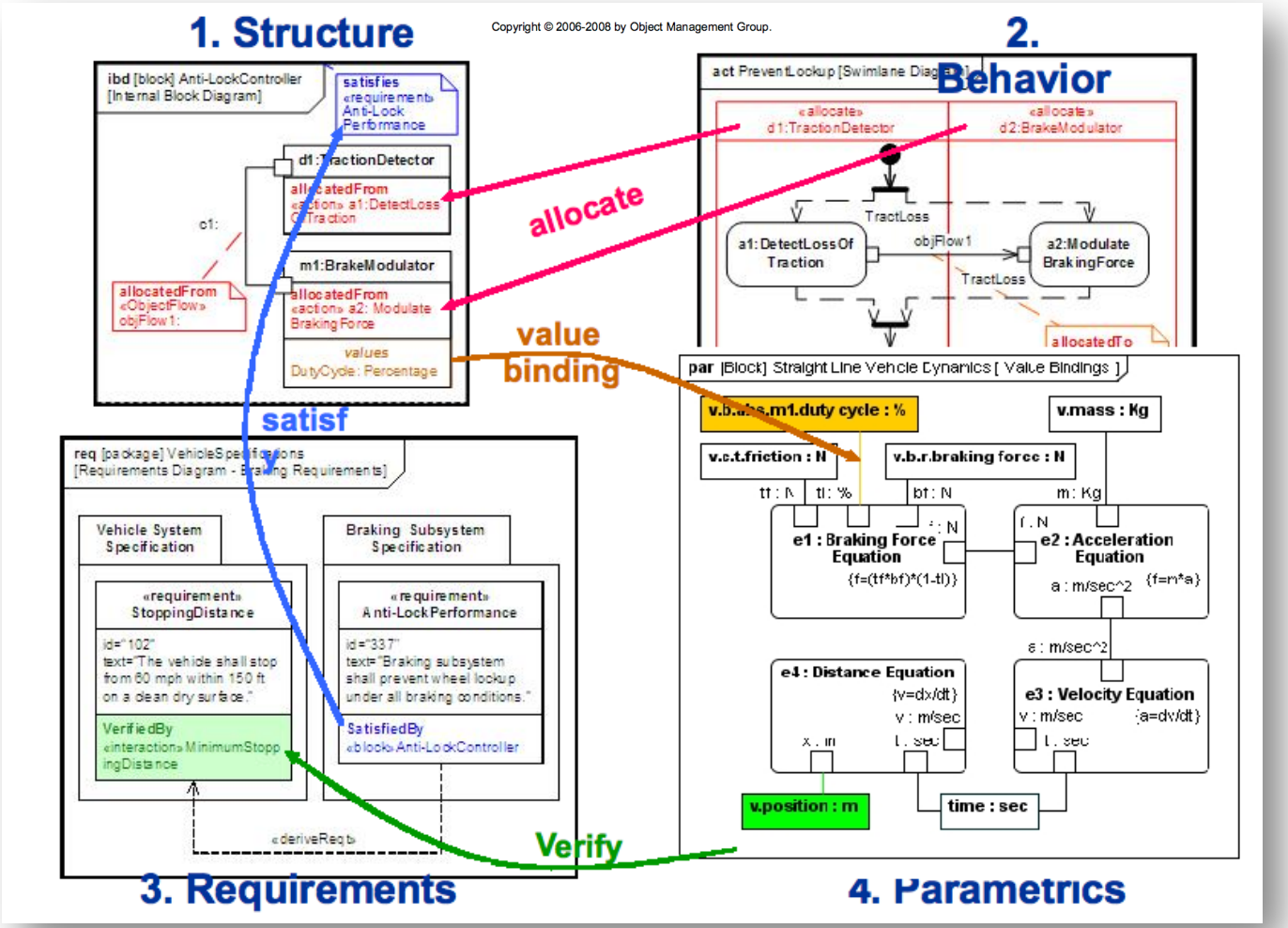
- We will focuss on them soon…



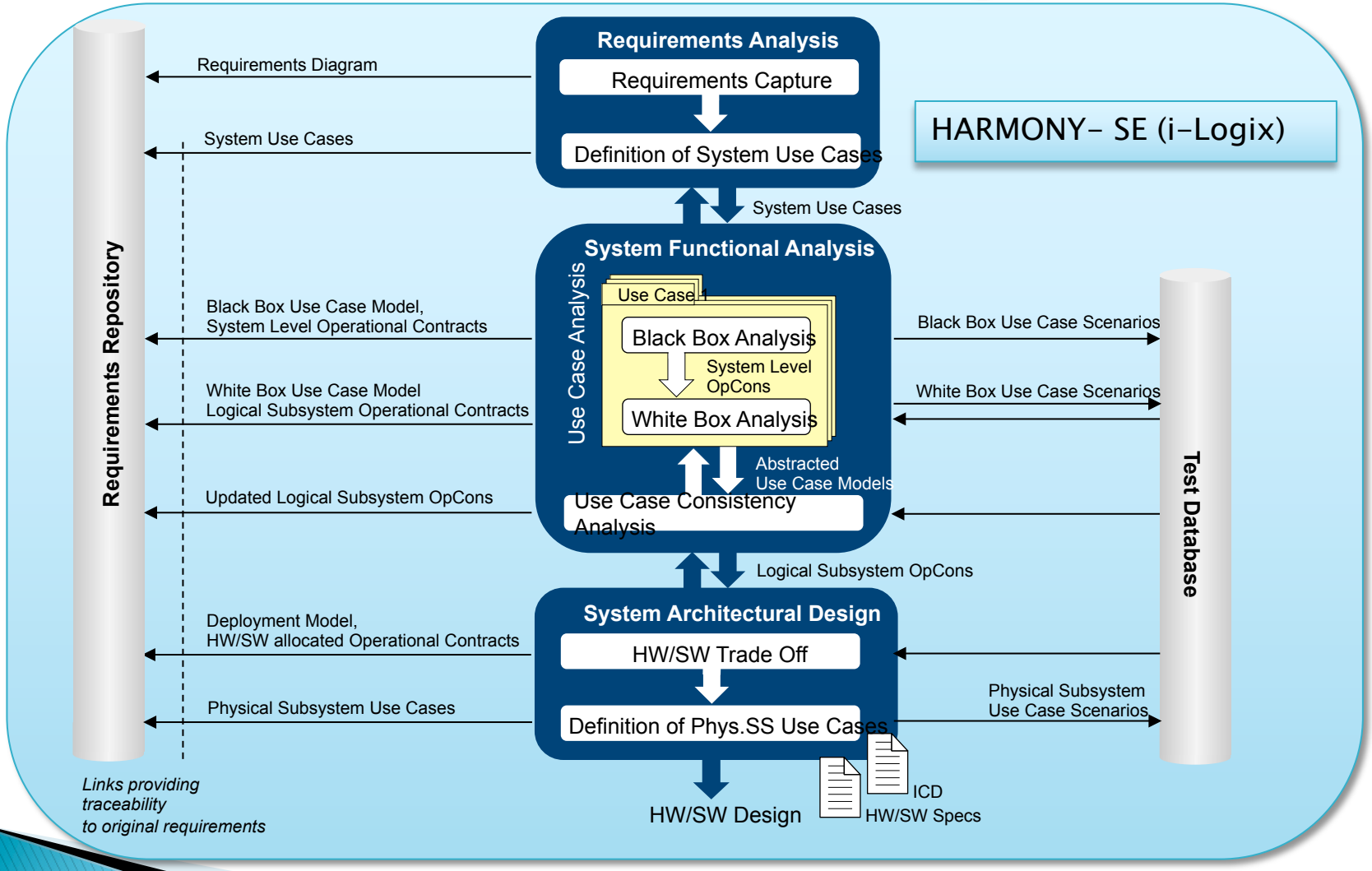
Rational and Problems



Cross-connecting elements



Example of methodology



Summary

- Introduction
- System Engineering
- Systems Requirements
- Requirements elicitation process
- KAOS overview
- SysML overview
- Requirements in SysML
- Mapping KAOS models into SysML models
- Practical case study

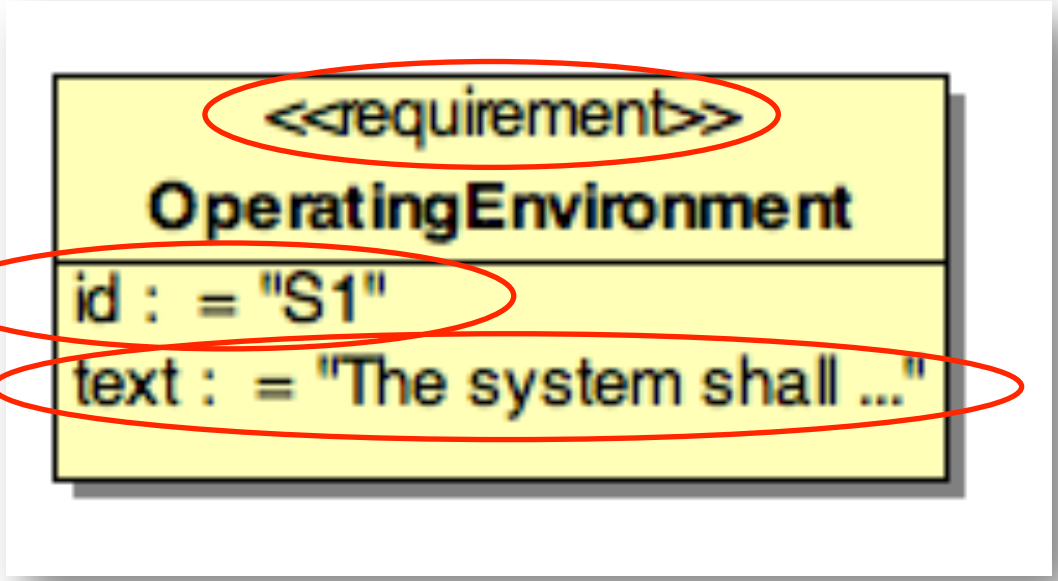
Requirements in SysML

- Requirement diagram
- Other diagrams
  - To link use cases in a Use Case diagram
  - To allocate them to block in a Block Def. Diag.
  - Etc.
- In tables

RequirementDiagrams (req)

- *<<requirement>>* allows to represent a text based requirement
  - Includes one identifier *id* and some textual properties
  - Can add user defined properties
  - Can add user defined requirement categories
- Requirements can be
  - decomposed
  - specialized
- Requirement relationships
  - « deriveRqt », « refine »
  - « satisfy », « verify »
  - « trace », « copy »

Basic definition



- Stereotype
- Id
- text

Requirements Tables

table [requirement] Performance [Decomposition of Performance Requirement]

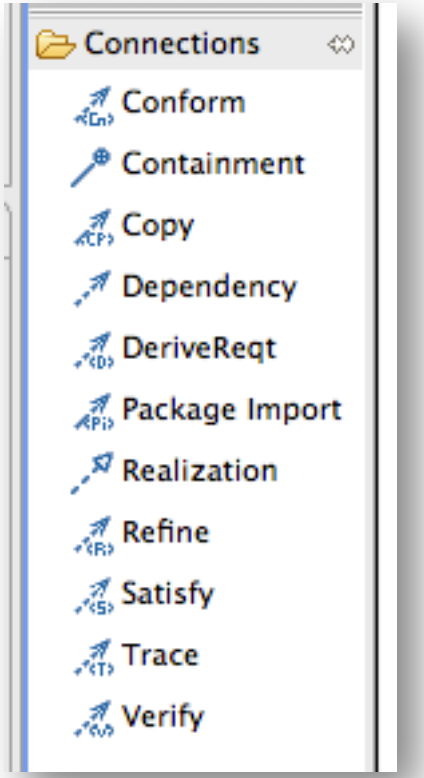
id	name	text
2	Performance	The Hybrid SUV shall have the braking, acceleration, and off-road capability of a typical SUV, but have dramatically better fuel economy.
2.1	Braking	The Hybrid SUV shall have the braking capability of a typical SUV.
2.2	FuelEconomy	The Hybrid SUV shall have dramatically better fuel economy than a typical SUV.
2.3	OffRoadCapability	The Hybrid SUV shall have the off-road capability of a typical SUV.
2.4	Acceleration	The Hybrid SUV shall have the acceleration of a typical SUV.

table [requirement] Performance [Tree of Performance Requirements]

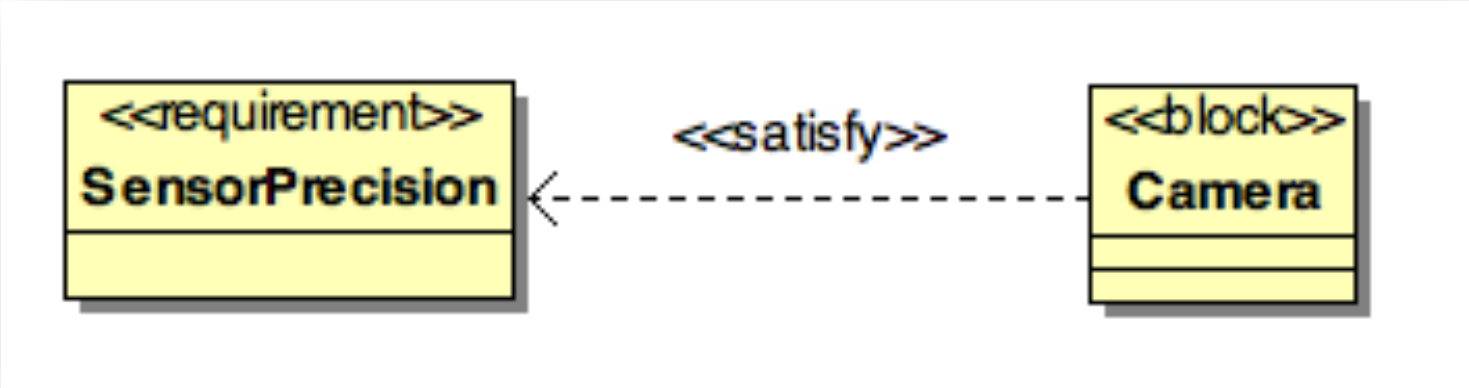
id	name	relation	id	name	relation	id	name
2.1	Braking	deriveReqt	d.1	RegenerativeBraking			
2.2	FuelEconomy	deriveReqt	d.1	RegenerativeBraking			
2.2	FuelEconomy	deriveReqt	d.2	Range			
4.2	FuelCapacity	deriveReqt	d.2	Range			
2.3	OffRoadCapability	deriveReqt	d.4	Power	deriveReqt	d.2	PowerSourceManagement
2.4	Acceleration	deriveReqt	d.4	Power	deriveReqt	d.2	PowerSourceManagement
4.1	CargoCapacity	deriveReqt	d.4	Power	deriveReqt	d.2	PowerSourceManagement

Requirements Relationships

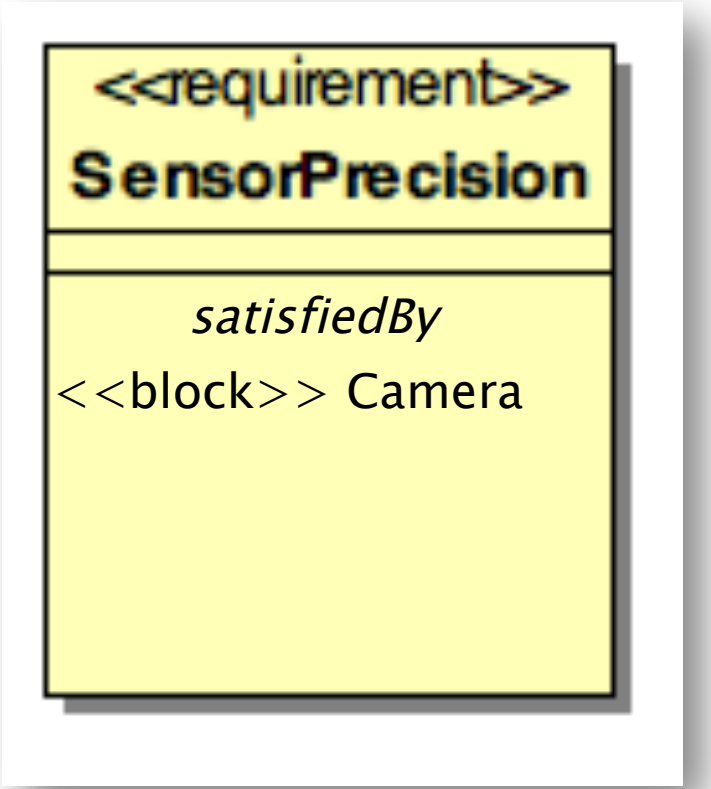
- Between requirements
  - Containment
  - Refine
  - Derive
  - Specialize
  - Copy
  - Trace
- Between requirements and others
  - Satisfy
  - Verify
  - Refine
  - Trace



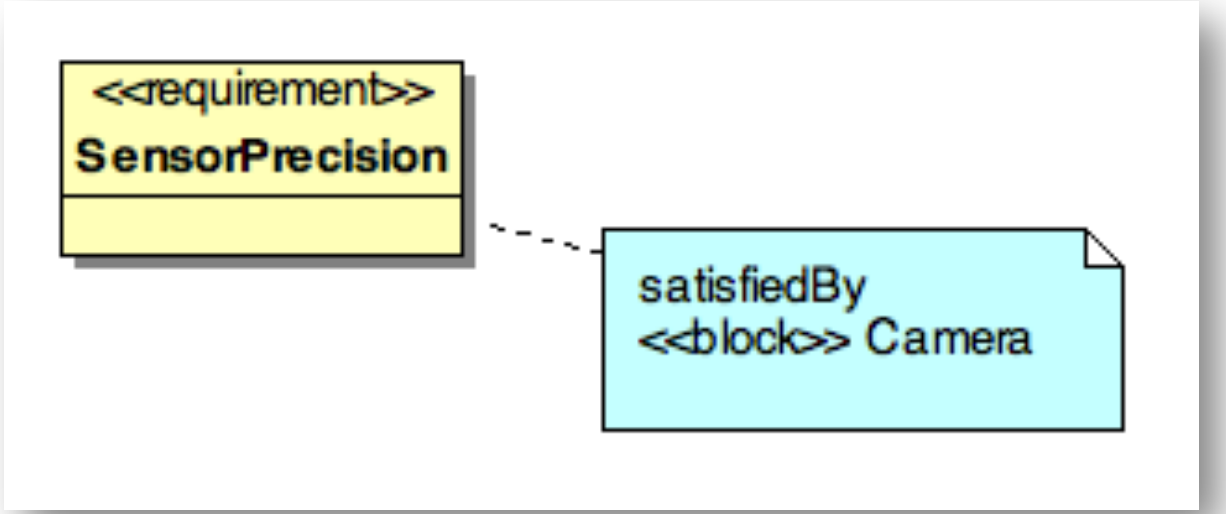
Depicting relationship directly



Relationship in compartments



Relationship with callout



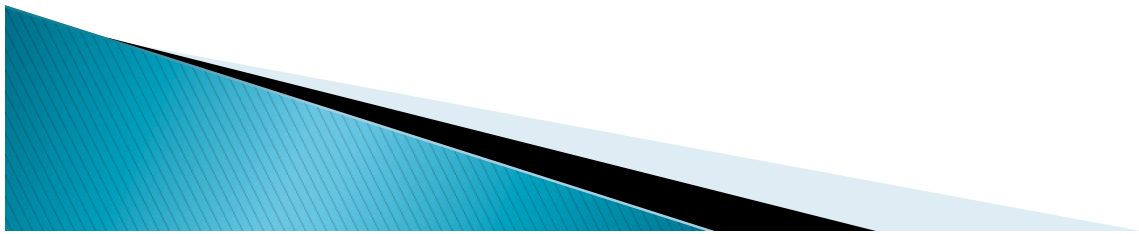
Relationship with tables

Id	Name	RelatesTo	RelatesHow	Type
TMC17	Traffic management configuration information	{TMC15, TMC16}	deriveReqt	Functional
TMC20	Make function/context obvious	{TMC18, TMC19}	deriveReqt	Functional
TMC21	Education material	{TMC18, TMC19}	deriveReqt	Functional

http://ojs.academypublisher.com/index.php/jsw/article/viewFile/03065768/988

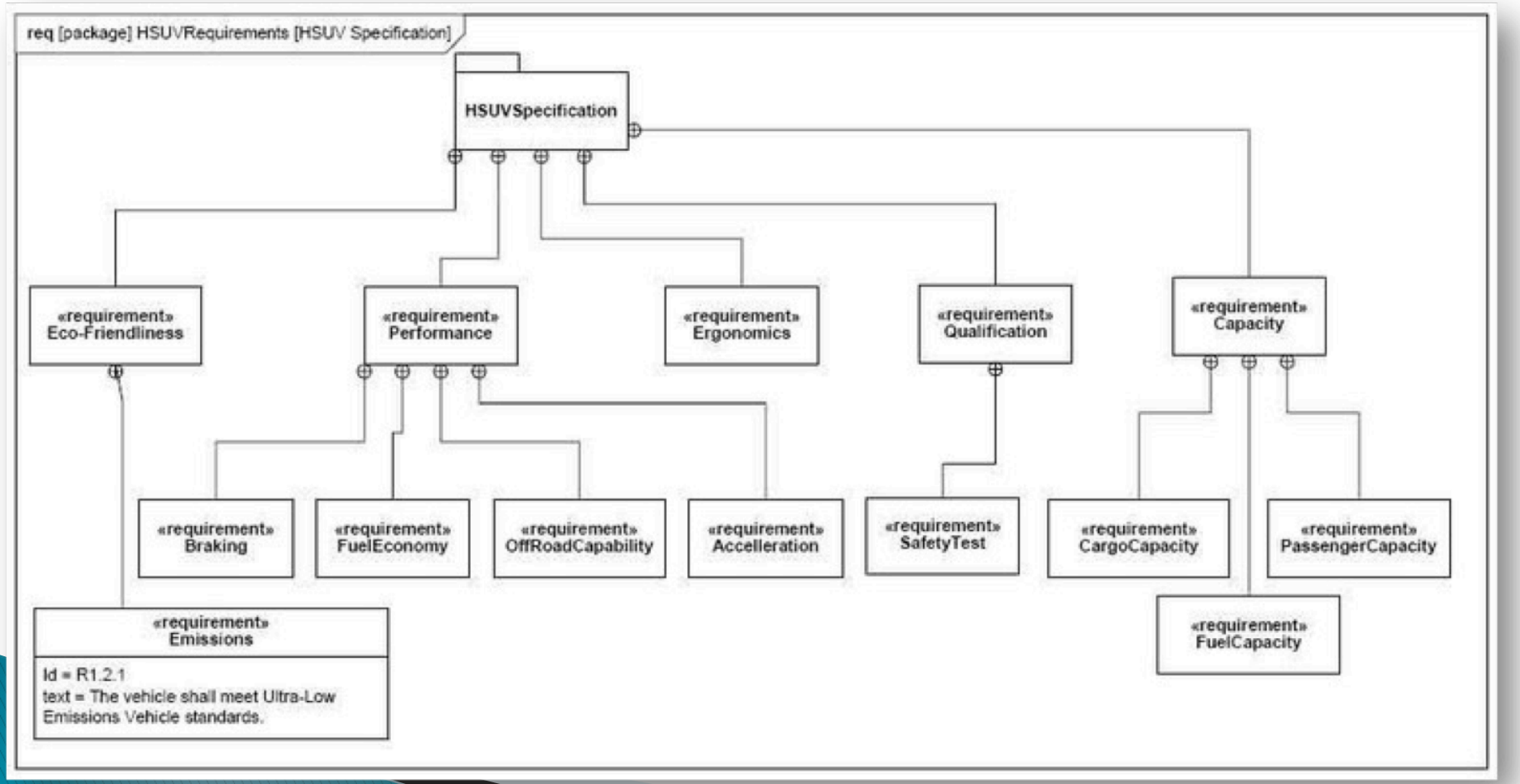
Requirements hierarchies

- Requirements organized into package structure



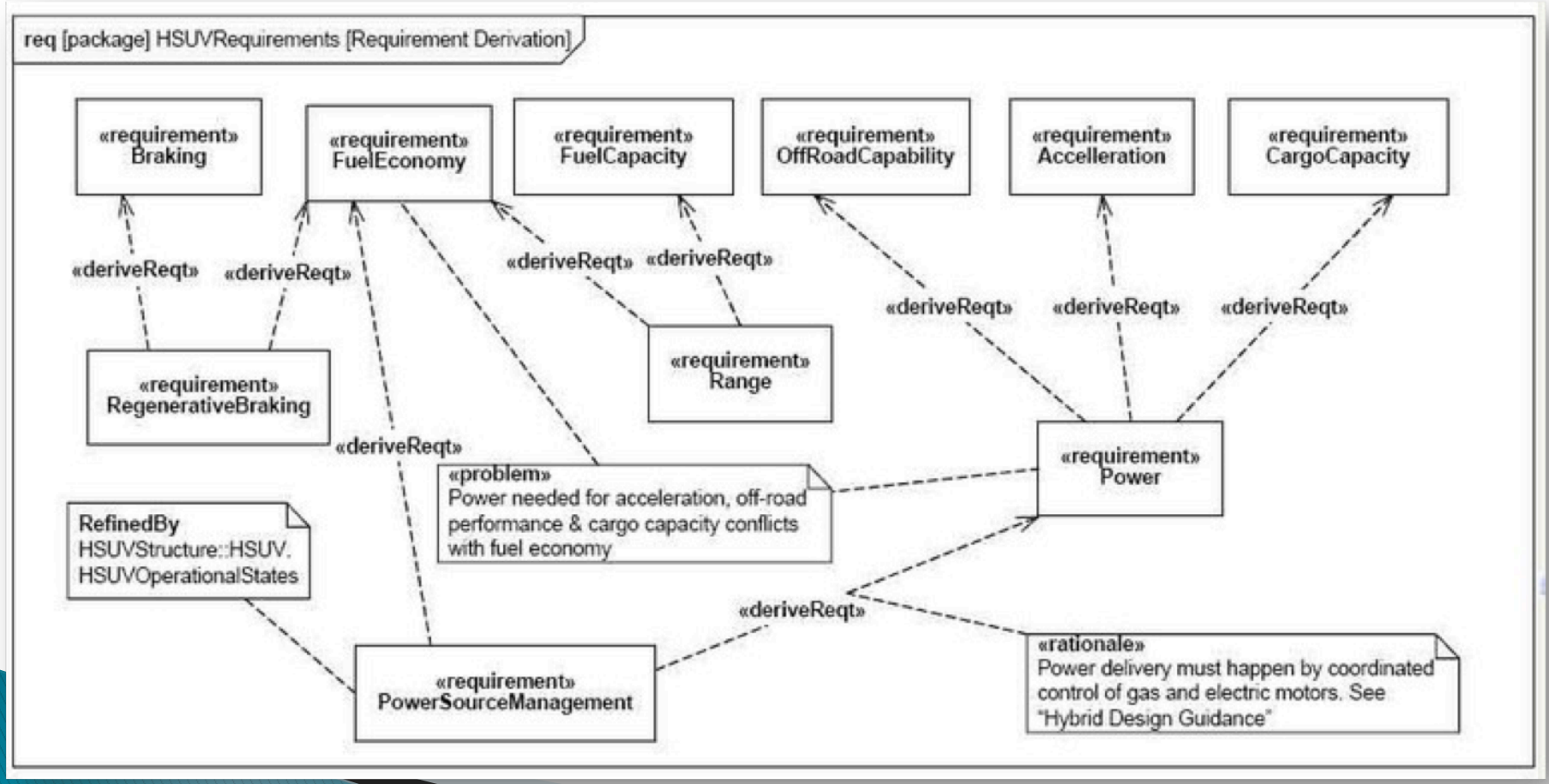
Requirements hierarchies

- Requirements organized using **containment**

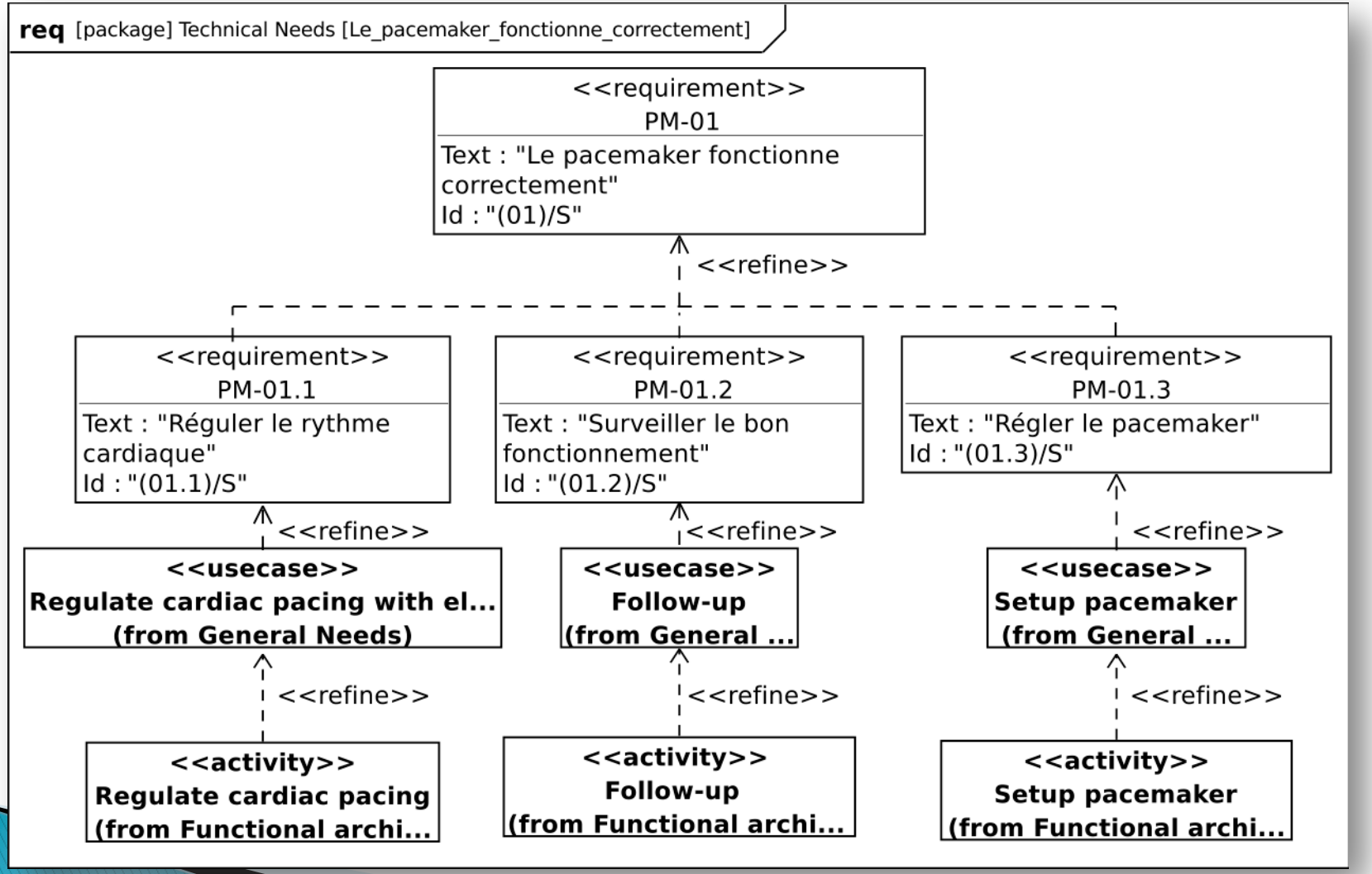


Requirement Derivation

- Implies an analysis



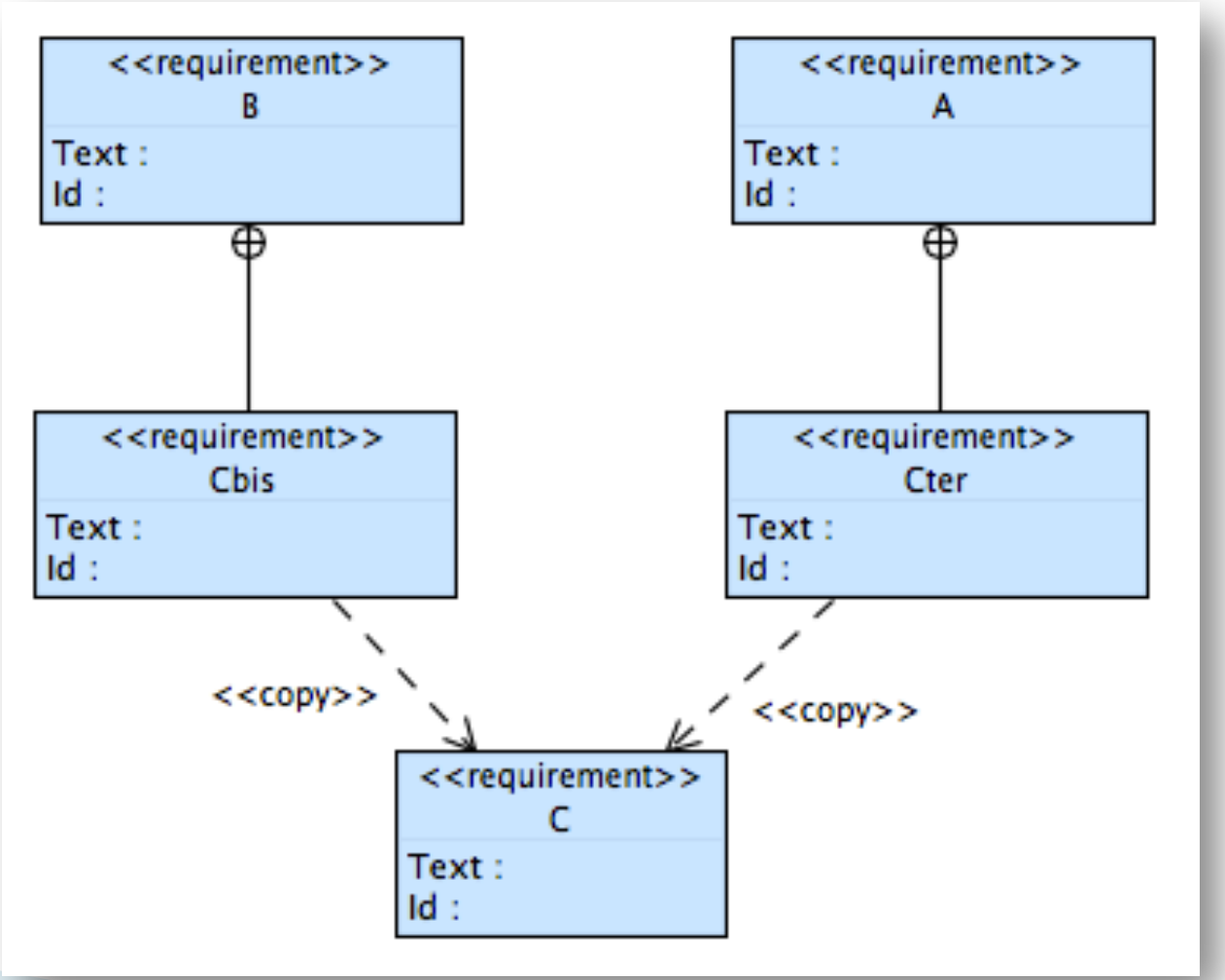
Reducing ambiguity



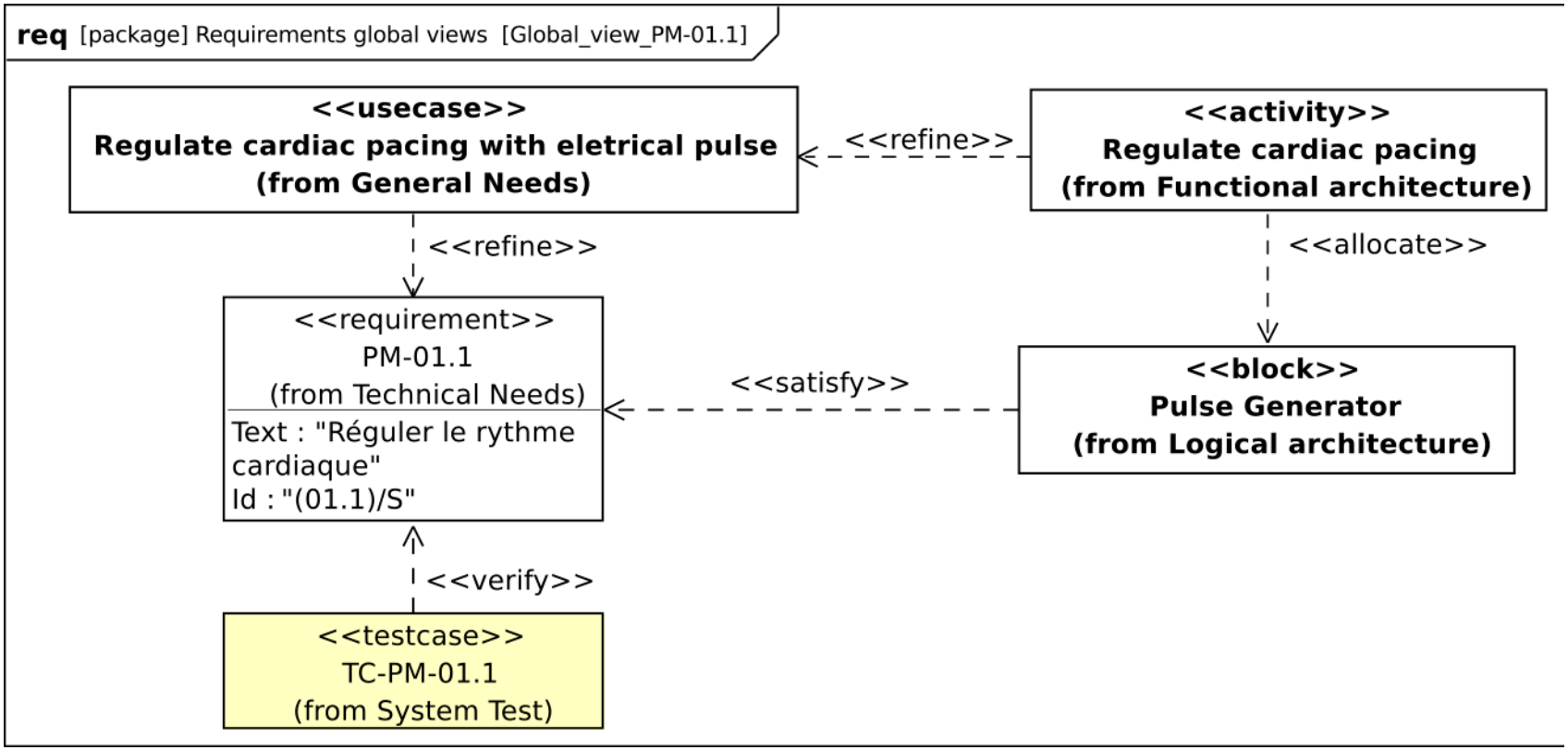
Tracing between requirements

- When you don't want to be more precise

Copying requirements



Example of traceability links



Why requirements in SysML ?

- Not too much if used only for requirements
- Added value when related to other elements
- Easy import/export

Summary

- Introduction
- System Engineering
- Systems Requirements
- Requirements elicitation process
- KAOS overview
- SysML overview
- Requirements in SysML
- Mapping KAOS models into SysML models
- Practical case study

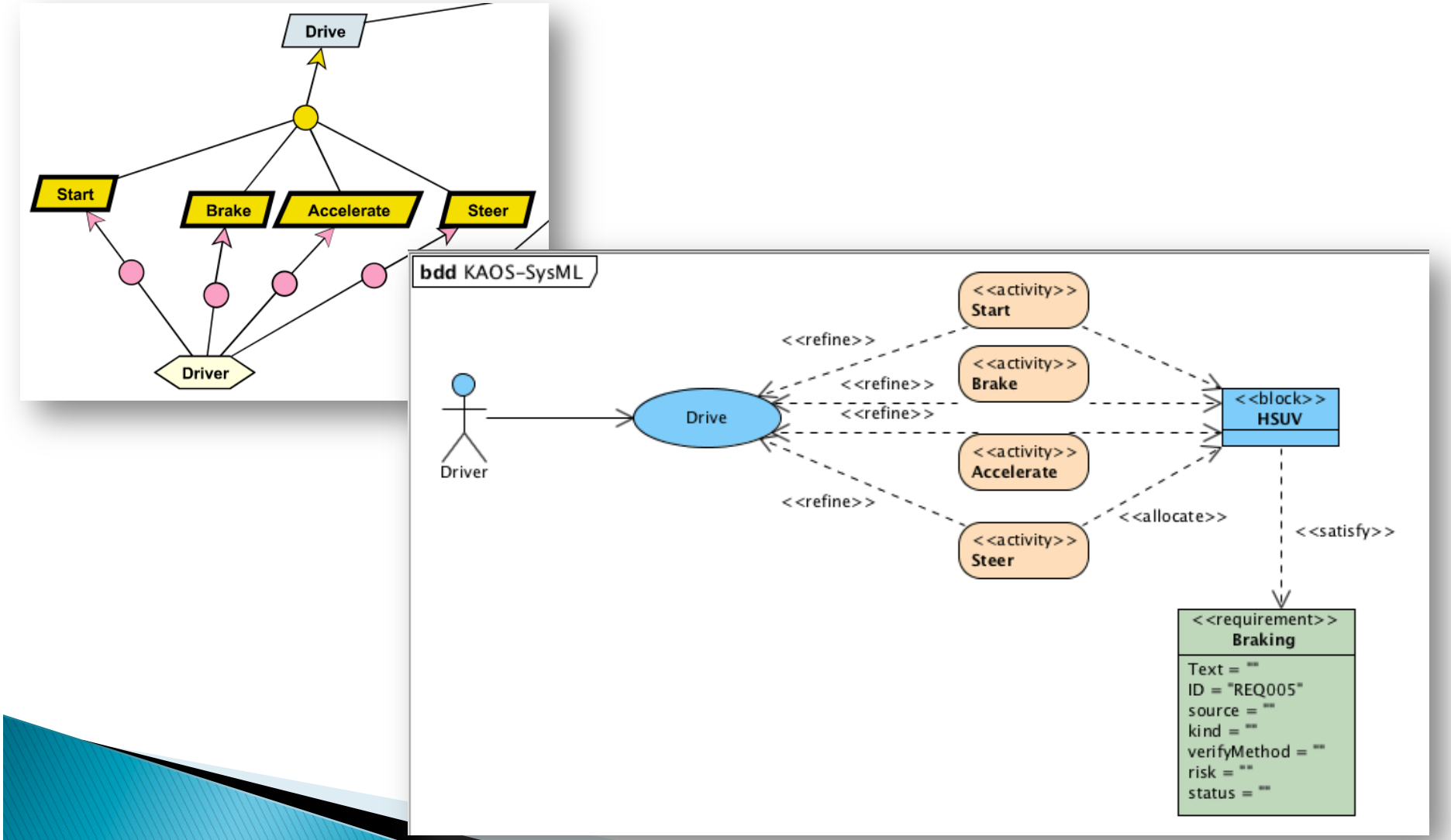
Mapping KAOS models into SysML models

- Mapping Modeling concepts
  - Goal → <<requirement>>
  - Requirement → <<requirement>> (system)
  - Expectation → <<requirement>> (user)
  - Resolutions → <<requirement>> (system or user)
  - Entity → Block
  - Operation → activity or Block operation
  - Environment Agents → Actors
  - System Agents → Blocks/components

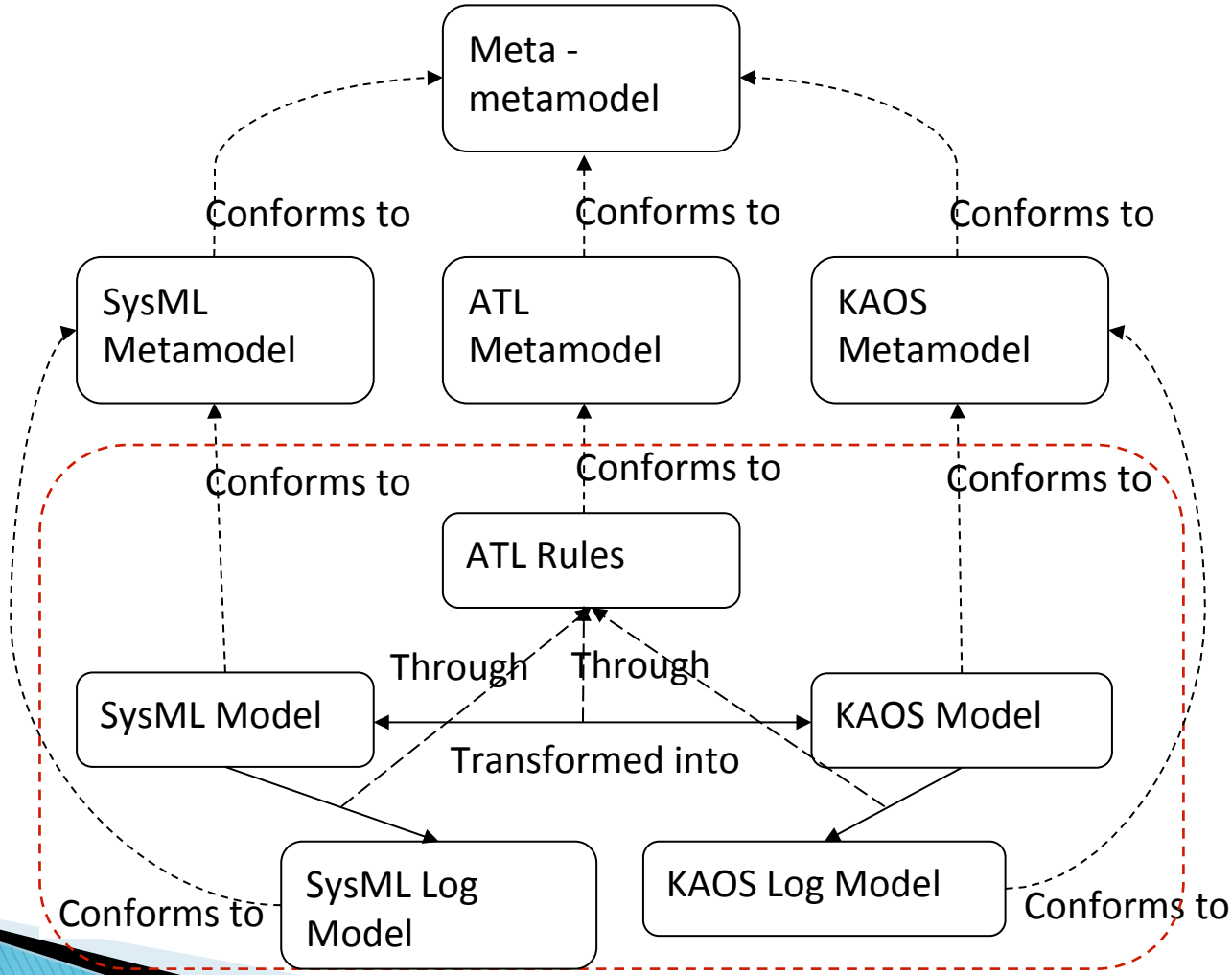
Mapping KAOS models into SysML models (ctd.)

- Relationships
  - Decomposition
    - Or → multiple <<refine>>
    - And → composition
  - Concerns → <<satisfy>>
- No direct mapping
  - Obstacles
  - Conflicts

Mapping KAOS/SysML: example



Transformation Framework



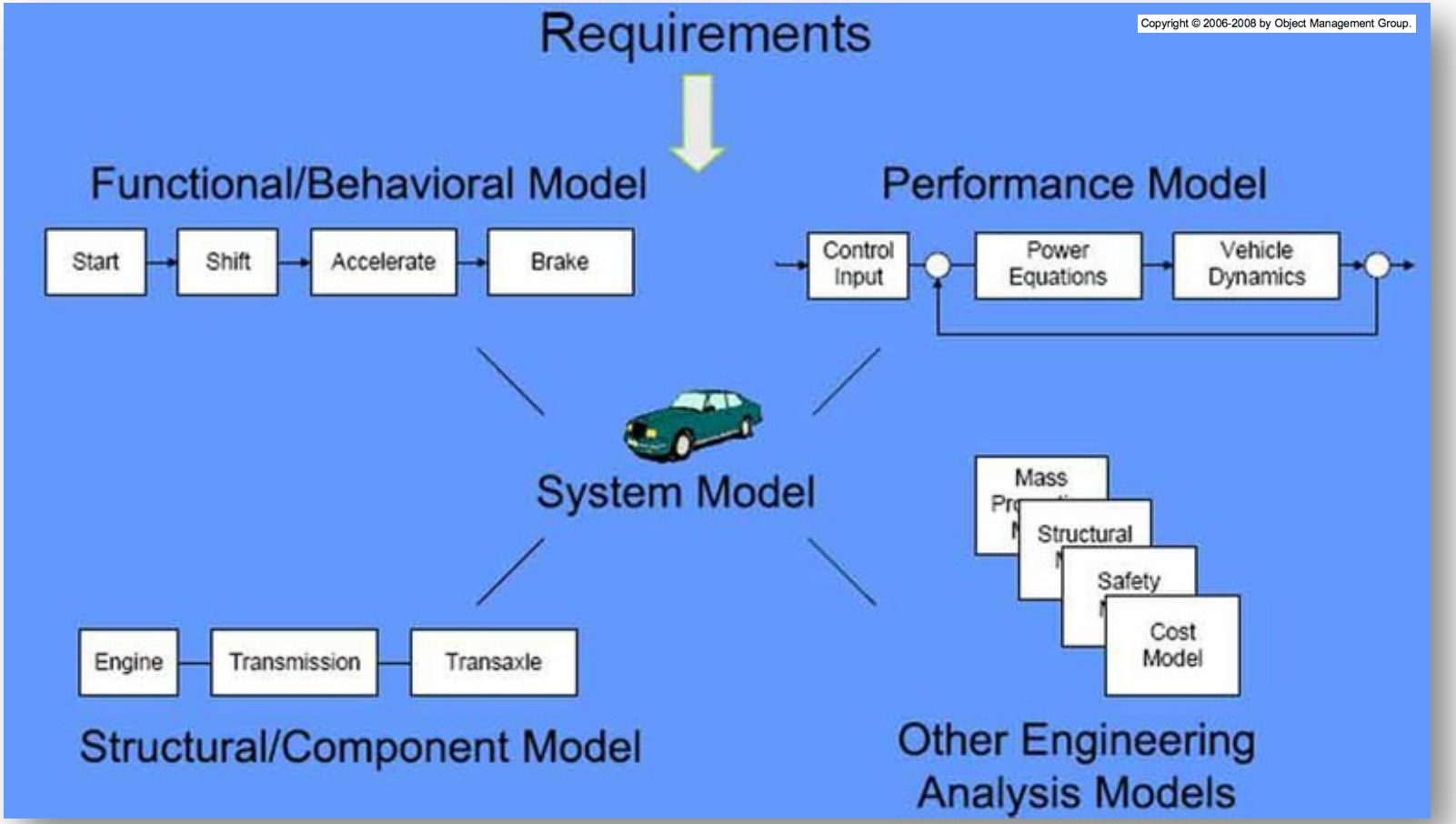
Summary

- Introduction
- System Engineering
- Systems Requirements
- Requirements elicitation process
- KAOS overview
- SysML overview
- Requirements in SysML
- Mapping KAOS models into SysML models
- Practical case study

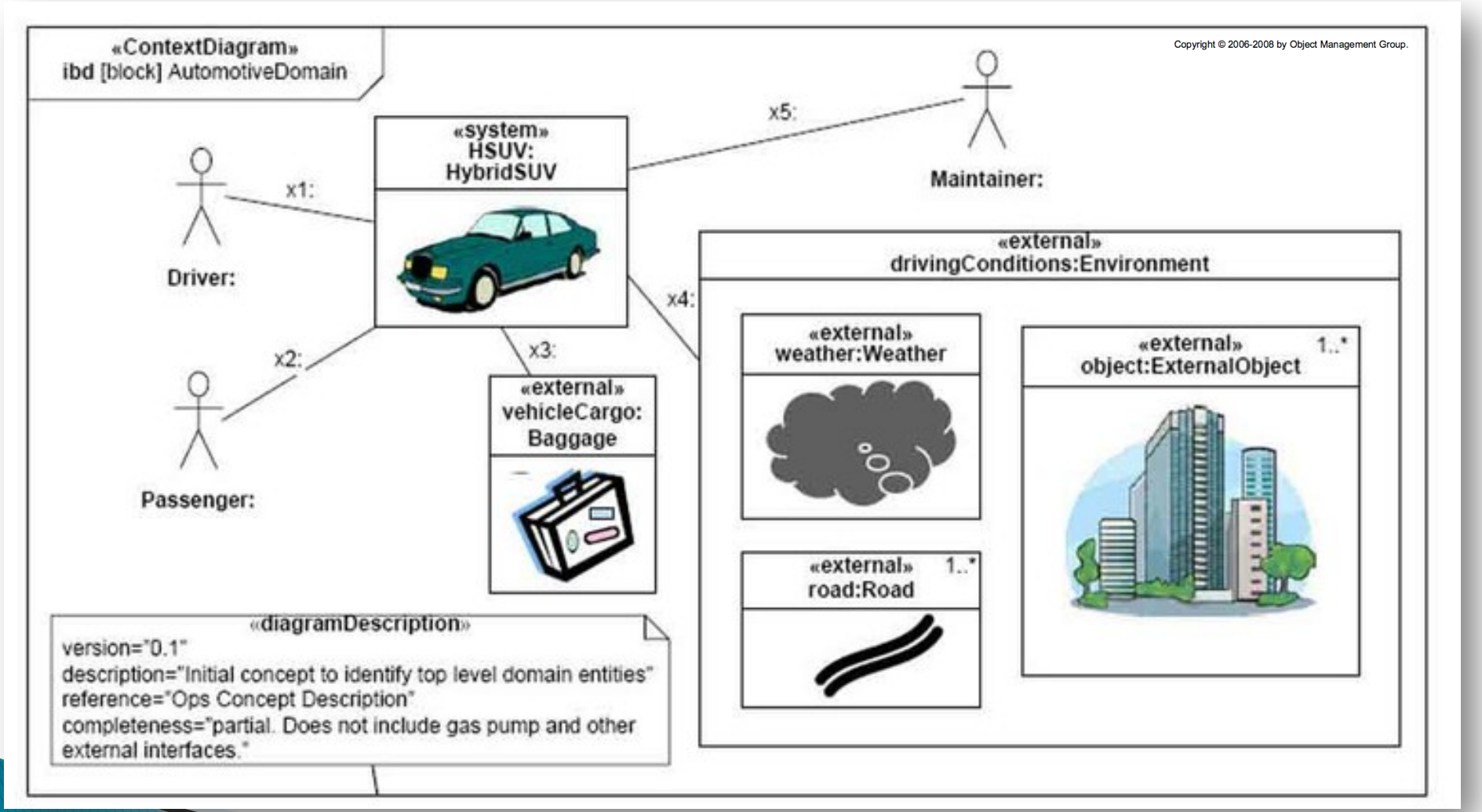
Practical case study

- Hybrid Utility Vehicle (HUV)
  - From http://www.uml-sysml.org/sysml
- Cable TV

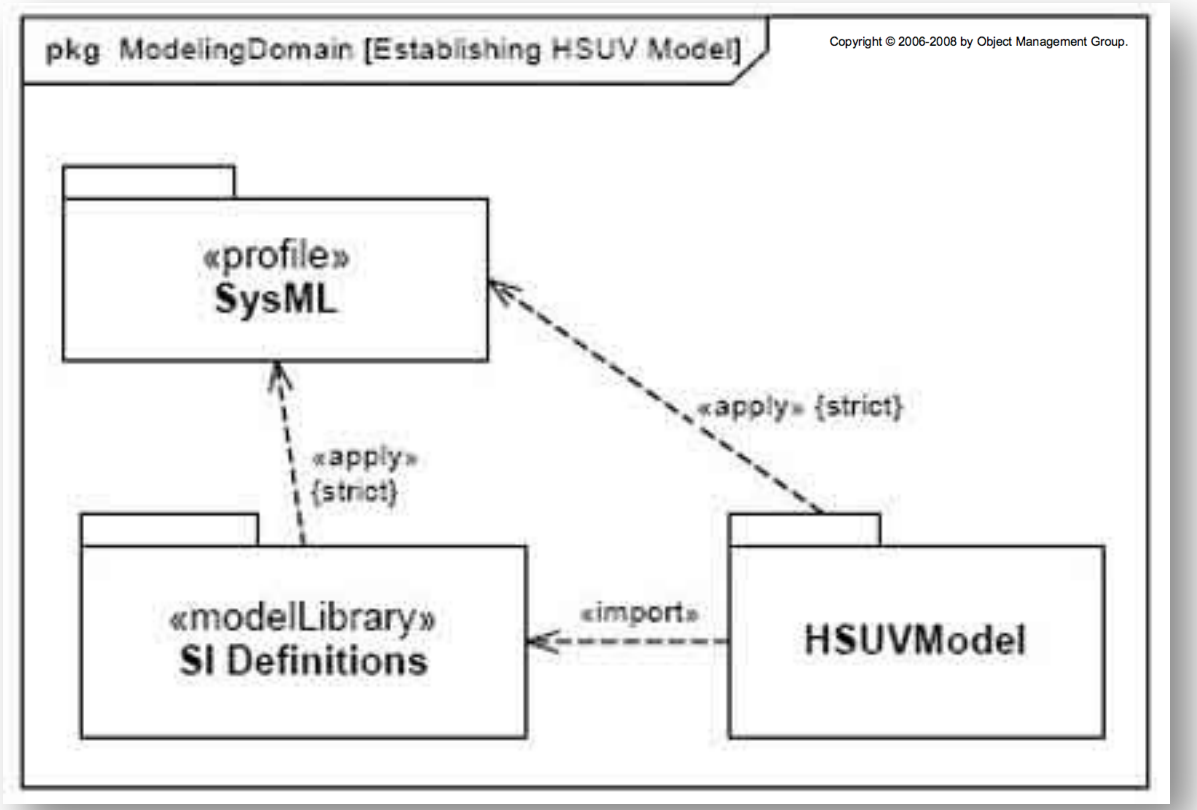
HSUV (cont.)


Requirements
Copyright © 2006-2008 by Object Management Group.
Functional/Behavioral Model
Start
Shift
Accelerate
Brake
Performance Model
Control Input
Power Equations
Vehicle Dynamics
System Model
Mass
Structural
Safety
Cost Model
Engine
Transmission
Transaxle
Structural/Component Model
Other Engineering Analysis Models

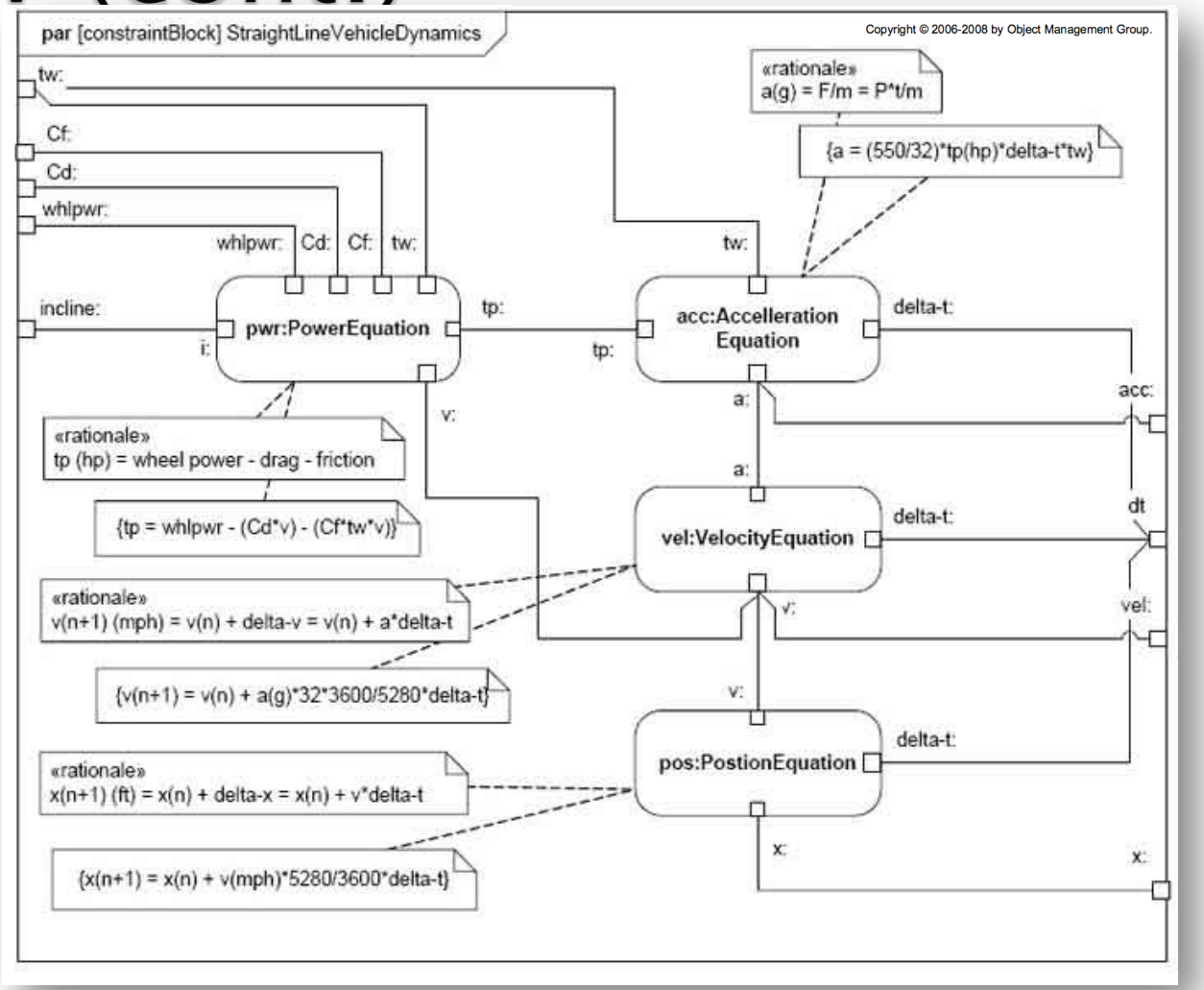
HSUV (cont.)


«ContextDiagram»
ibd [block] AutomotiveDomain
Copyright © 2006-2008 by Object Management Group.
«system»
HSUV:
HybridSUV
x1:
x2:
x3:
x4:
x5:
Driver:
Passenger:
Maintainer:
«external»
vehicleCargo:
Baggage
«external»
drivingConditions:Environment
«external»
weather:Weather
«external»
object:ExternalObject
1..*
«external»
road:Road
1..*
«diagramDescription»
version="0.1"
description="Initial concept to identify top level domain entities"
reference="Ops Concept Description"
completeness="partial. Does not include gas pump and other external interfaces."

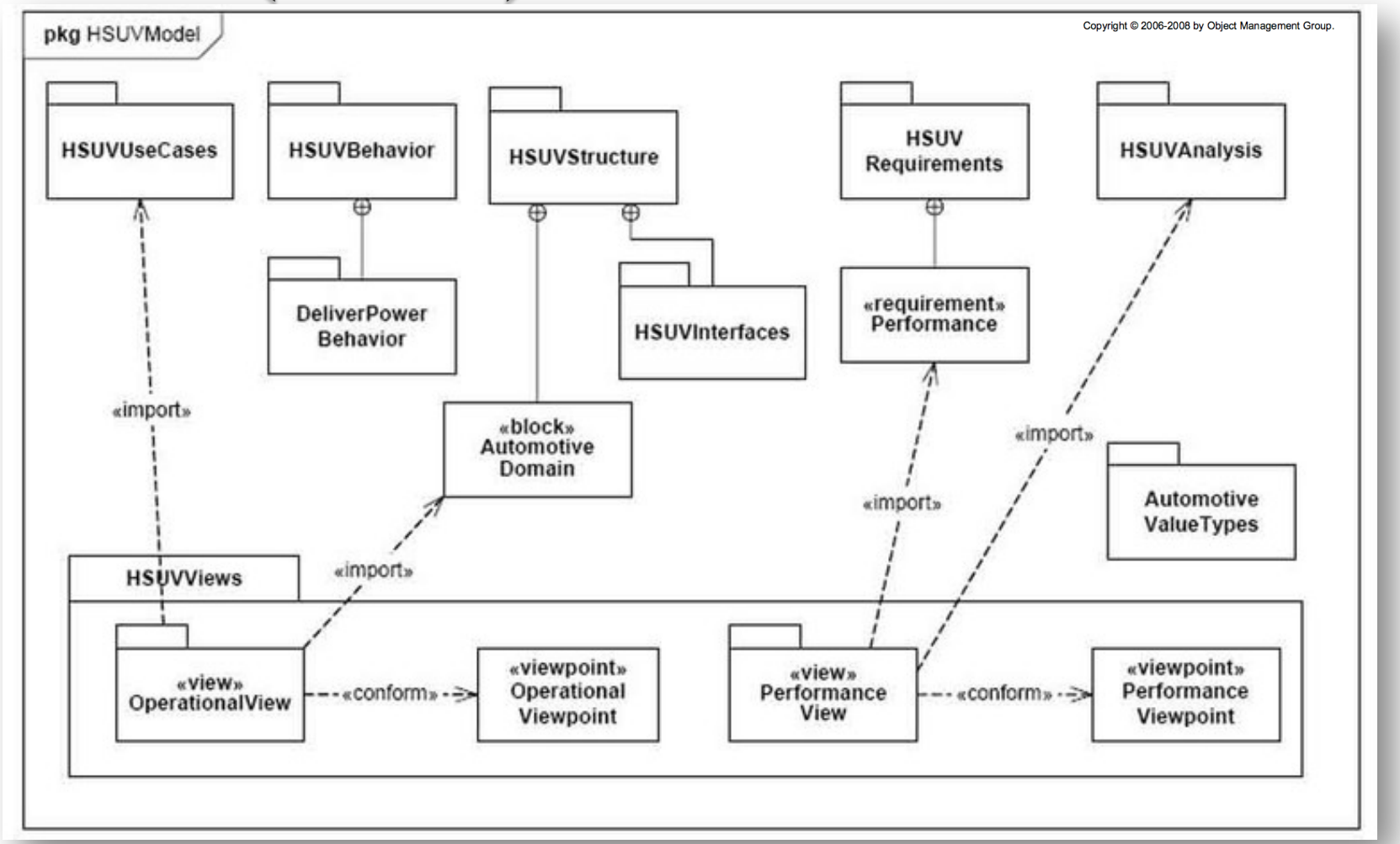
HSUV (cont.)


pkg ModelingDomain [Establishing HSUV Model]
Copyright © 2006-2008 by Object Management Group.
«profile»
SysML
«apply» {strict}
«apply»
{strict}
«modelLibrary»
SI Definitions
«import»
HSUVModel

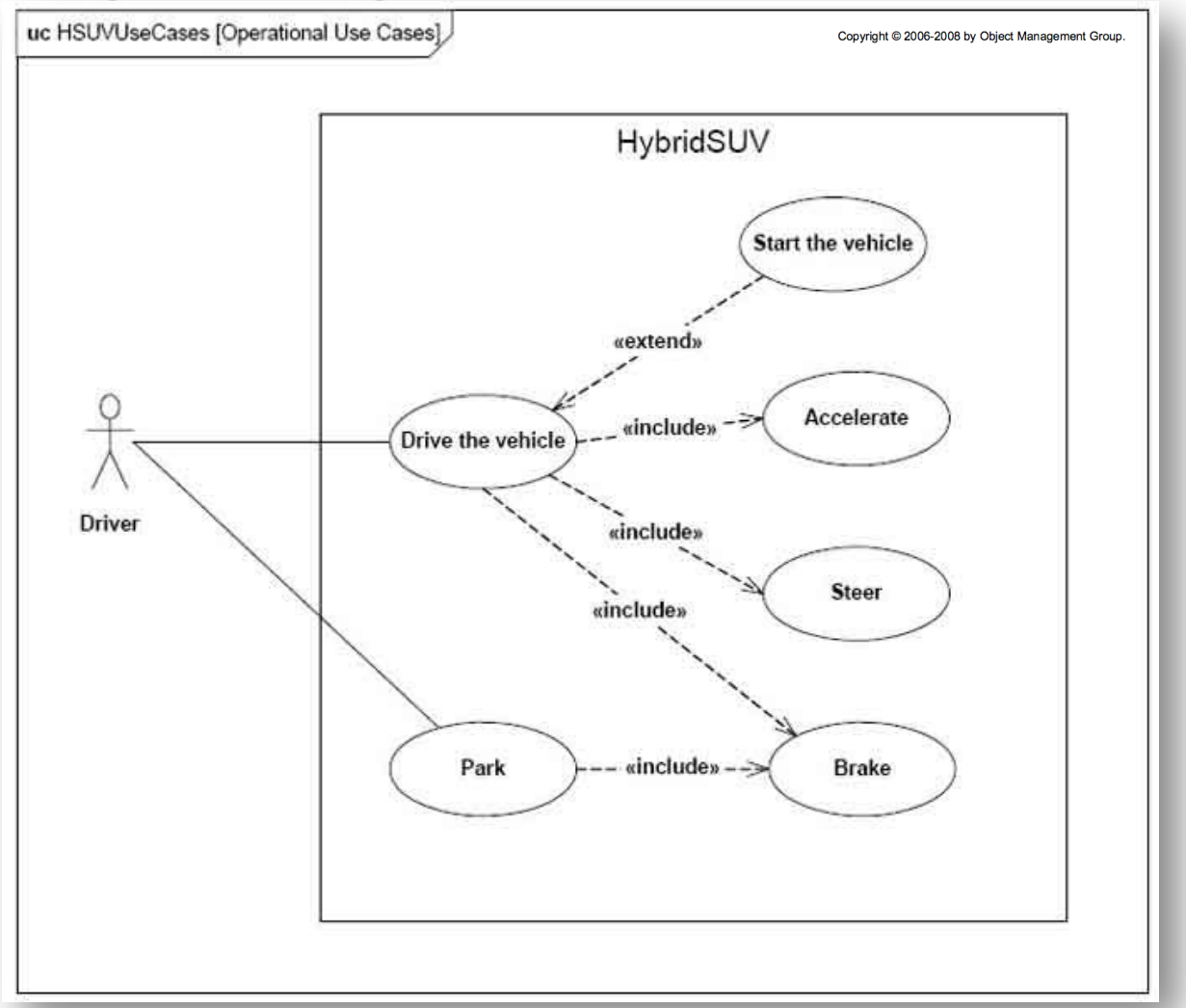
HSUV (cont.)


par [constraintBlock] StraightLineVehicleDynamics
Copyright © 2006-2008 by Object Management Group.
«rationale»
a(g) = F/m = P*t/m
{a = (550/32)*tp(hp)*delta-t*tw}
tw:
Cf:
Cd:
whlpwr:
whlpwr:
Cd:
Cf:
tw:
tw:
incline:
i:
pwr:PowerEquation
tp:
tp:
acc:Accelleration Equation
delta-t:
acc:
a:
v:
«rationale»
tp (hp) = wheel power - drag - friction
{tp = whlpwr - (Cd*v) - (Cf*tw*v)}
a:
vel:VelocityEquation
delta-t:
dt
«rationale»
v(n+1) (mph) = v(n) + delta-v = v(n) + a*delta-t
{v(n+1) = v(n) + a(g)*32*3600/5280*delta-t}
v:
vel:
v:
pos:PostionEquation
delta-t:
«rationale»
x(n+1) (ft) = x(n) + delta-x = x(n) + v*delta-t
{x(n+1) = x(n) + v(mph)*5280/3600*delta-t}
x:
x:

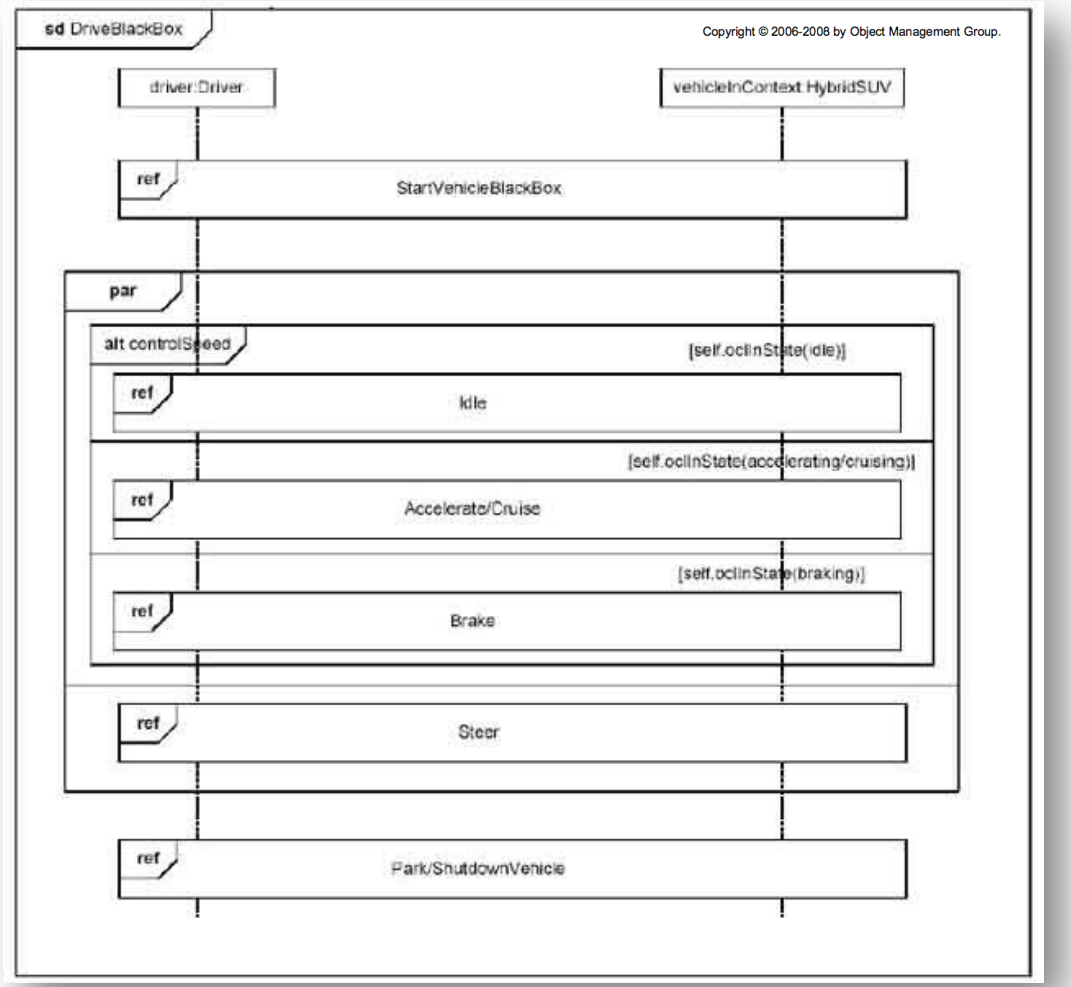
HSUV (cont.)


pkg HSUVModel
Copyright © 2006-2008 by Object Management Group.
HSUVUseCases
HSUVBehavior
HSUVStructure
HSUV Requirements
HSUVAnalysis
DeliverPower Behavior
HSUVInterfaces
«requirement» Performance
«import»
«block» Automotive Domain
«import»
«import»
Automotive ValueTypes
HSUVViews
«import»
«view» OperationalView
«conform»
«viewpoint» Operational Viewpoint
«view» Performance View
«conform»
«viewpoint» Performance Viewpoint

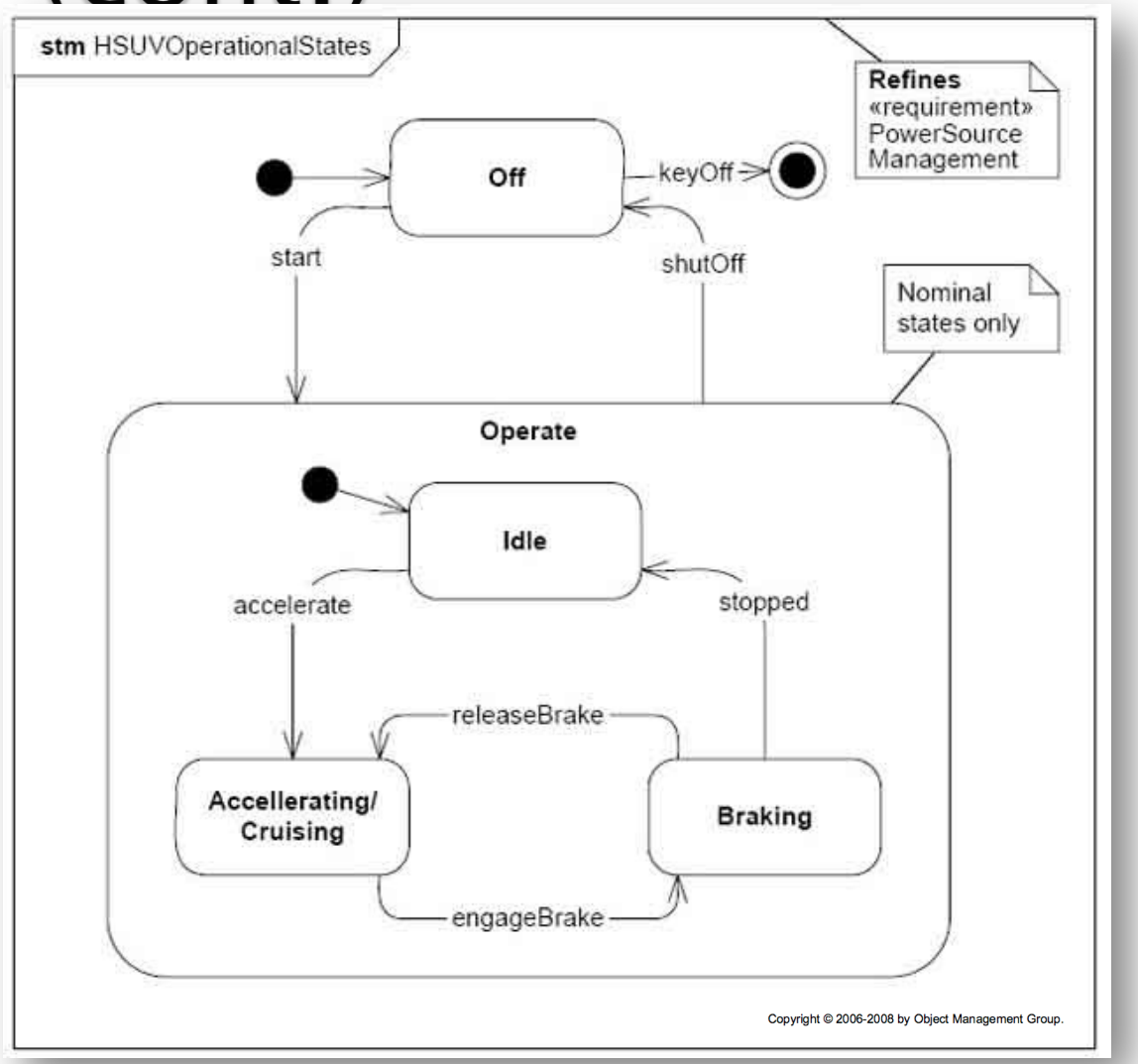
HSUV (cont.)


uc HSUVUseCases [Operational Use Cases]
Copyright © 2006-2008 by Object Management Group.
HybridSUV
Start the vehicle
«extend»
Drive the vehicle
«include»
Accelerate
«include»
Steer
«include»
Brake
Park
«include»
Driver

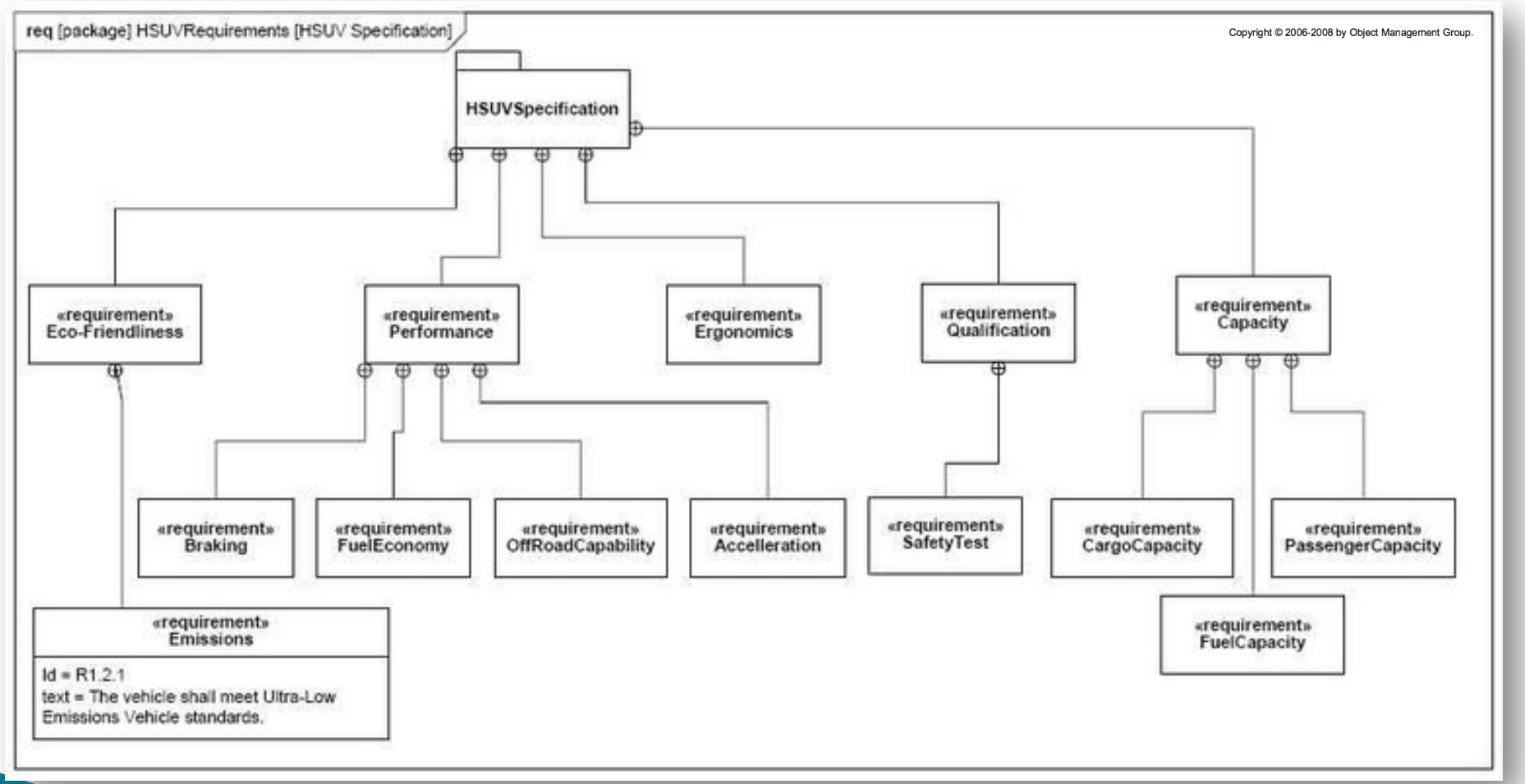
HSUV (cont.)


sd DriveBlackBox
Copyright © 2006-2008 by Object Management Group.
driver:Driver
vehicleInContext:HybridSUV
ref
StartVehicleBlackBox
par
alt controlSpeed
[self.oclInState(idle)]
ref
Idle
[self.oclInState(accelerating/cruising)]
ref
Accelerate/Cruise
[self.oclInState(braking)]
ref
Brake
ref
Steer
ref
Park/ShutdownVehicle

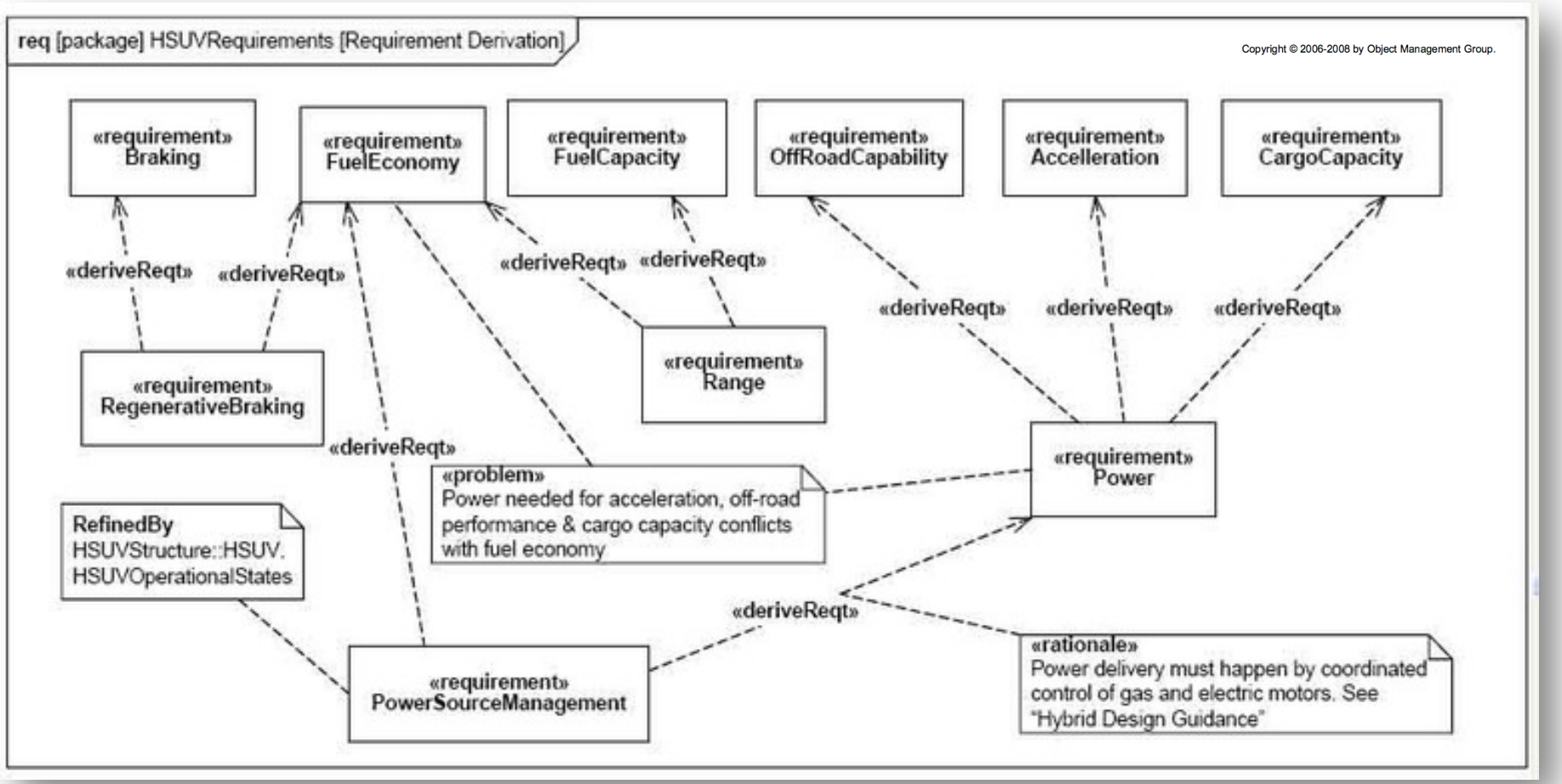
HSUV (cont.)


stm HSUVOperationalStates
Refines
«requirement»
PowerSource
Management
Off
keyOff
start
shutOff
Nominal
states only
Operate
Idle
accelerate
stopped
releaseBrake
Accellerating/
Cruising
Braking
engageBrake
Copyright © 2006-2008 by Object Management Group.

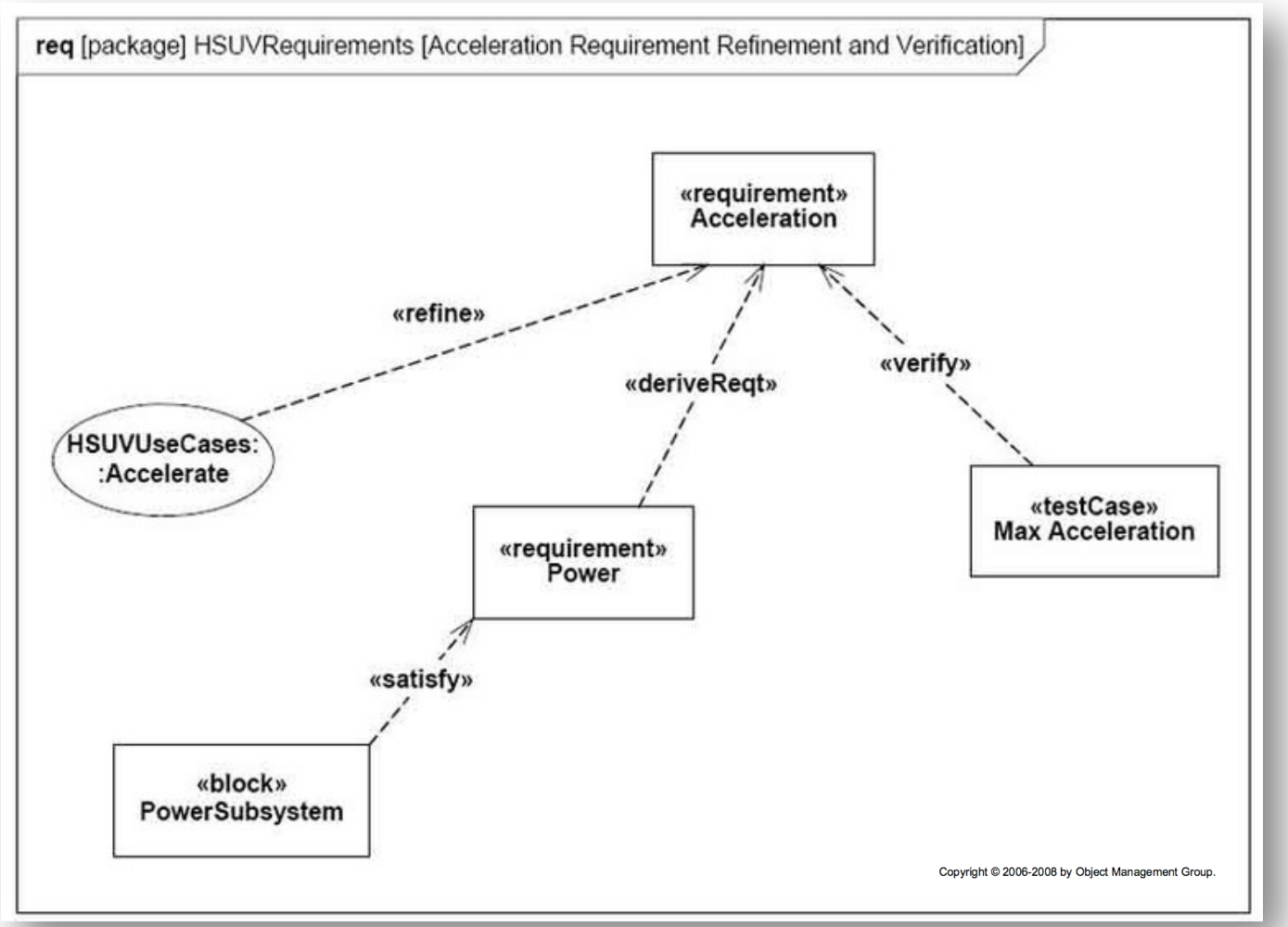
HSUV (cont.)


req [package] HSUVRequirements [HSUV Specification]
Copyright © 2006-2008 by Object Management Group.
HSUVSpecification
«requirement» Eco-Friendliness
«requirement» Performance
«requirement» Ergonomics
«requirement» Qualification
«requirement» Capacity
«requirement» Braking
«requirement» FuelEconomy
«requirement» OffRoadCapability
«requirement» Accelleration
«requirement» SafetyTest
«requirement» CargoCapacity
«requirement» PassengerCapacity
«requirement» FuelCapacity
«requirement» Emissions
Id = R1.2.1
text = The vehicle shall meet Ultra-Low Emissions Vehicle standards.

HSUV (cont.)


req [package] HSUVRequirements [Requirement Derivation]
Copyright © 2006-2008 by Object Management Group.
«requirement» Braking
«requirement» FuelEconomy
«requirement» FuelCapacity
«requirement» OffRoadCapability
«requirement» Accelleration
«requirement» CargoCapacity
«deriveReqt»
«requirement» RegenerativeBraking
«requirement» Range
«requirement» Power
«problem» Power needed for acceleration, off-road performance & cargo capacity conflicts with fuel economy
RefinedBy HSUVStructure::HSUV. HSUVOperationalStates
«requirement» PowerSourceManagement
«rationale» Power delivery must happen by coordinated control of gas and electric motors. See "Hybrid Design Guidance"

HSUV (cont.)


req [package] HSUVRequirements [Acceleration Requirement Refinement and Verification]
«requirement»
Acceleration
«refine»
«deriveReqt»
«verify»
HSUVUseCases:
:Accelerate
«requirement»
Power
«testCase»
Max Acceleration
«satisfy»
«block»
PowerSubsystem
Copyright © 2006-2008 by Object Management Group.

HSUV (cont.)

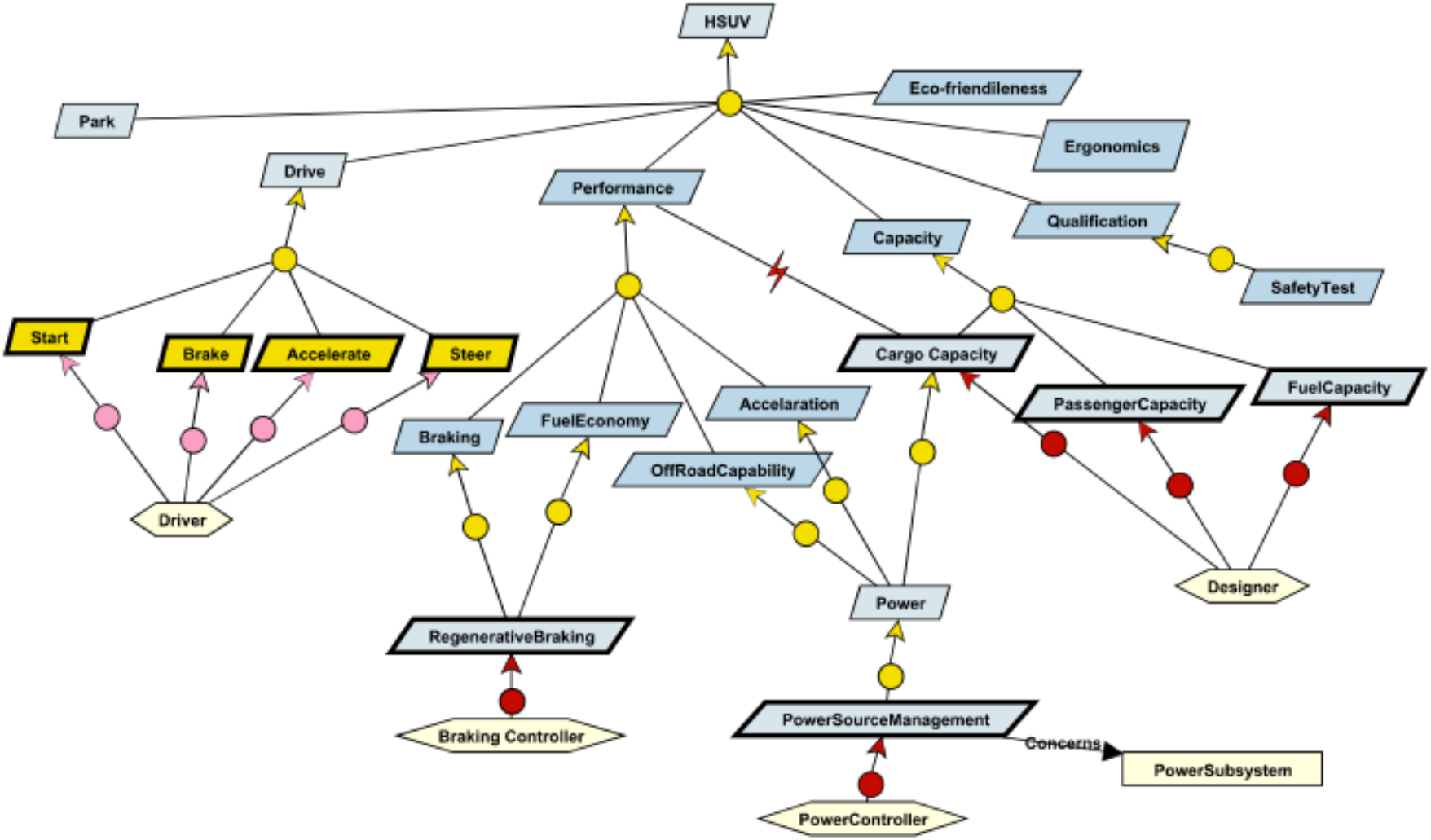
table [requirement] Performance [Decomposition of Performance Requirement]

id	name	text
2	Performance	The Hybrid SUV shall have the braking, acceleration, and off-road capability of a typical SUV, but have dramatically better fuel economy.
2.1	Braking	The Hybrid SUV shall have the braking capability of a typical SUV.
2.2	FuelEconomy	The Hybrid SUV shall have dramatically better fuel economy than a typical SUV.
2.3	OffRoadCapability	The Hybrid SUV shall have the off-road capability of a typical SUV.
2.4	Acceleration	The Hybrid SUV shall have the acceleration of a typical SUV.

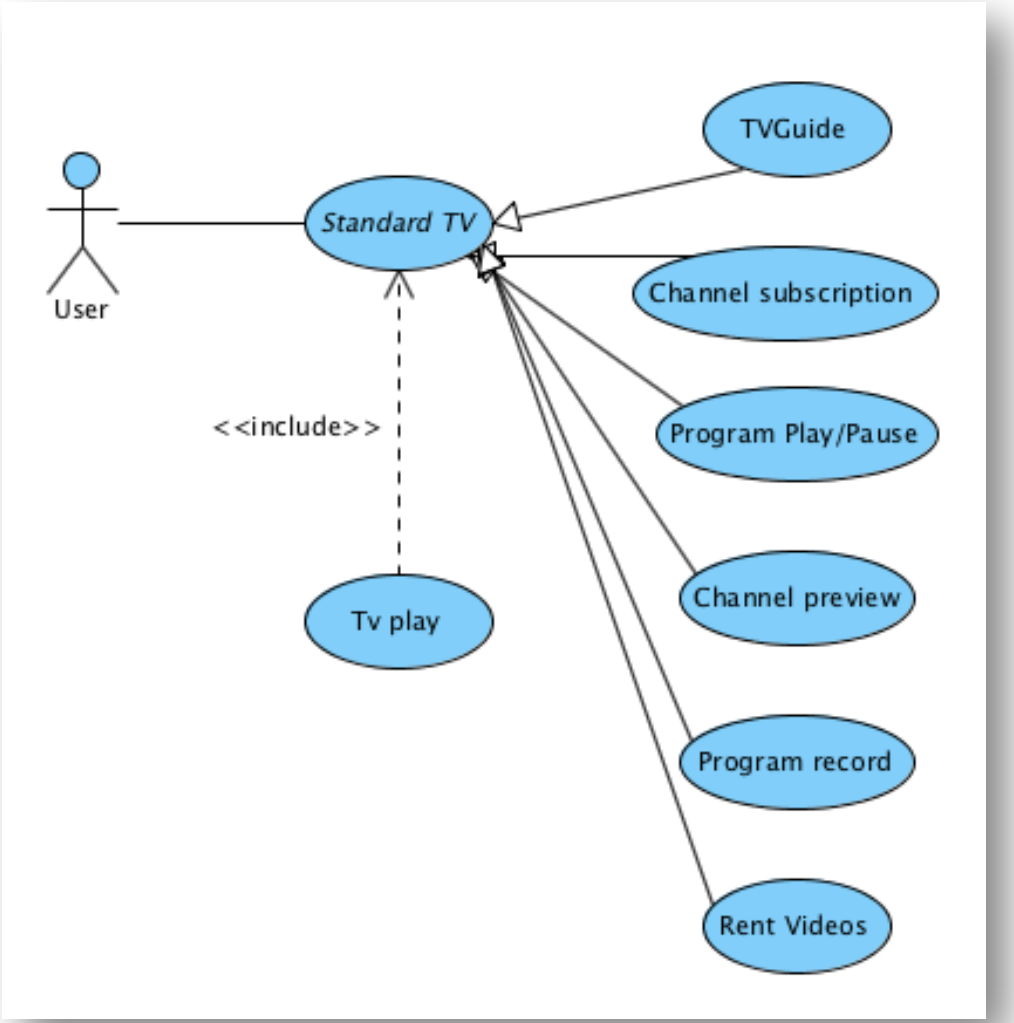
table [requirement] Performance [Tree of Performance Requirements]

id	name	relation	id	name	relation	id	name
2.1	Braking	deriveReqt	d.1	RegenerativeBraking			
2.2	FuelEconomy	deriveReqt	d.1	RegenerativeBraking			
2.2	FuelEconomy	deriveReqt	d.2	Range			
4.2	FuelCapacity	deriveReqt	d.2	Range			
2.3	OffRoadCapability	deriveReqt	d.4	Power	deriveReqt	d.2	PowerSourceManagement
2.4	Acceleration	deriveReqt	d.4	Power	deriveReqt	d.2	PowerSourceManagement
4.1	CargoCapacity	deriveReqt	d.4	Power	deriveReqt	d.2	PowerSourceManagement

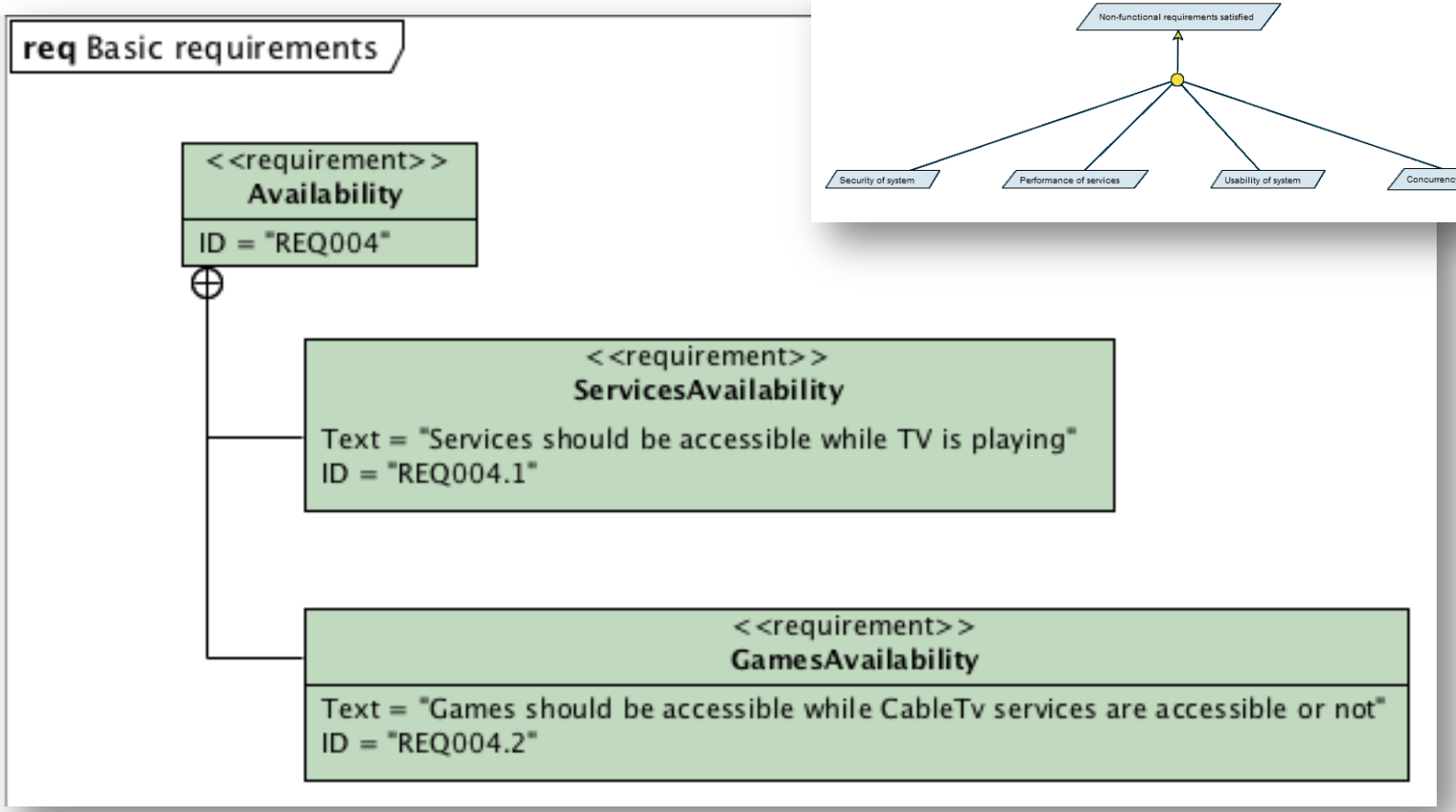
HSUV (cont.)


HSUV
Park
Drive
Performance
Eco-friendileness
Ergonomics
Capacity
Qualification
SafetyTest
Start
Brake
Accelerate
Steer
Cargo Capacity
PassengerCapacity
FuelCapacity
Braking
FuelEconomy
Accelaration
OffRoadCapability
Driver
Designer
Power
RegenerativeBraking
Braking Controller
PowerSourceManagement
Concerns
PowerSubsystem
PowerController

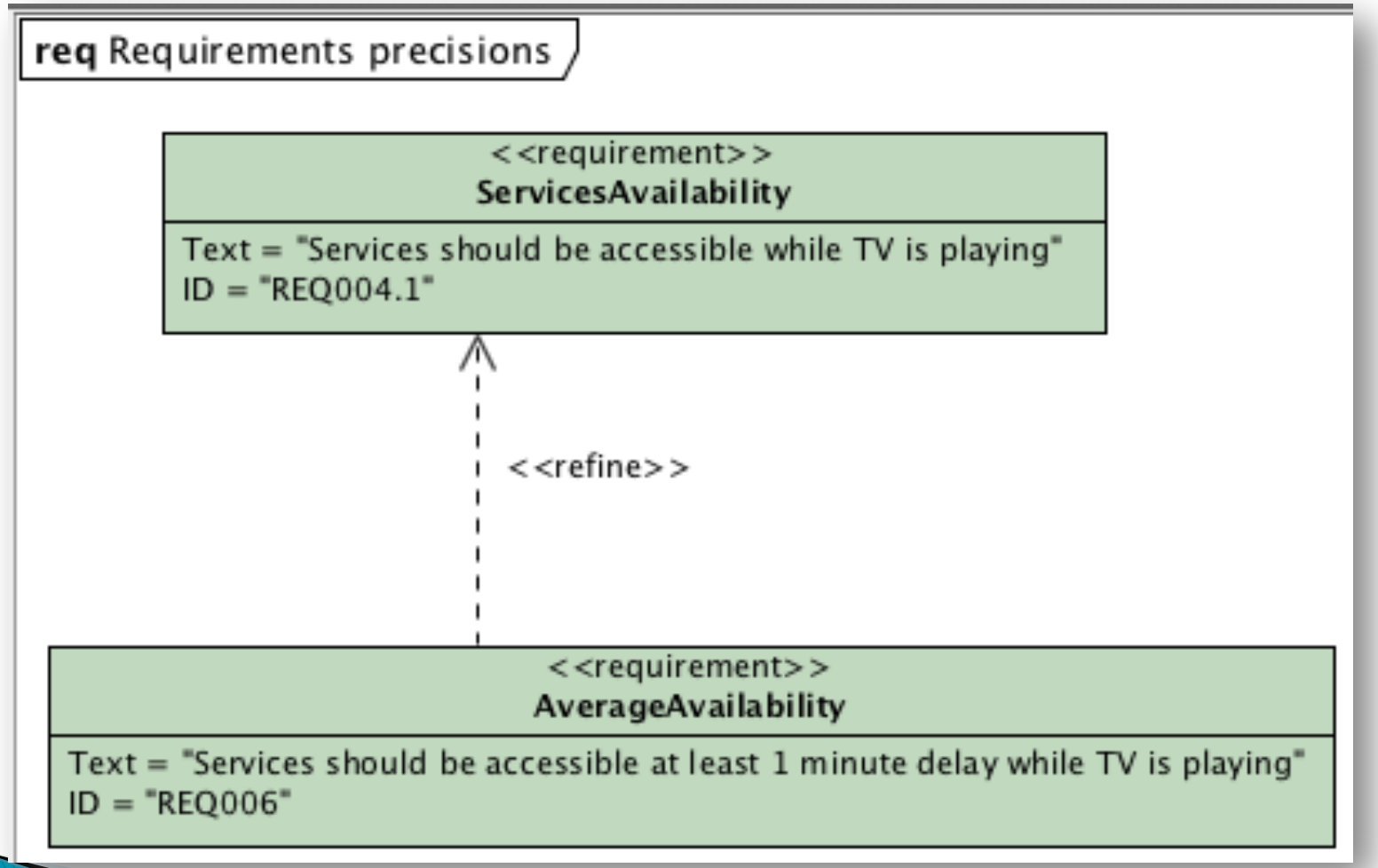
Cable TV (UC diagram)



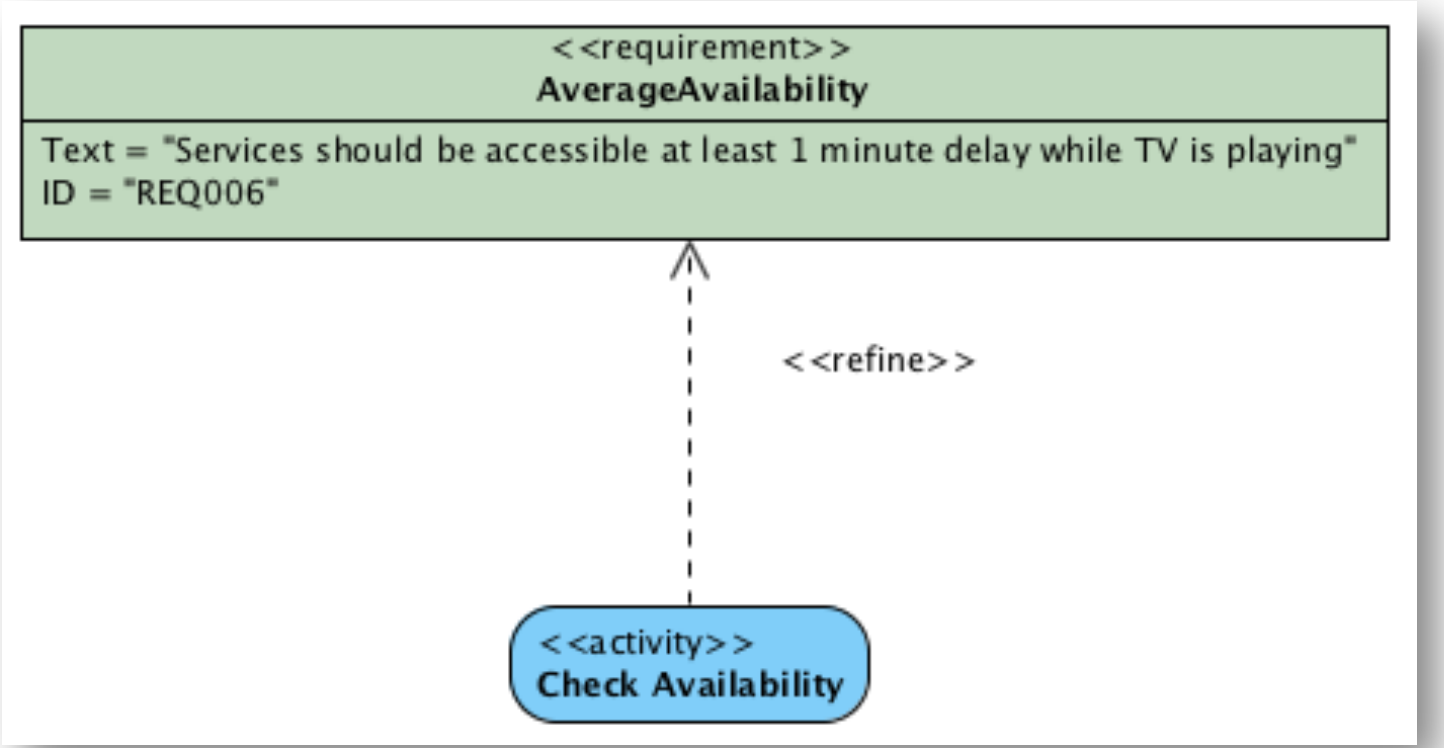
Cable TV (Requirements)



Cable TV (Requirements links)



Cable TV (Traceability links)



Cable TV (KAOS/SysML)

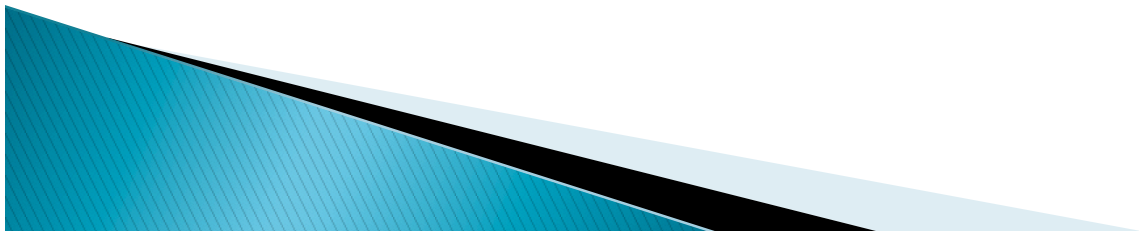
- Requirement → <<requirement>> (system)
- Expectation → <<requirement>> (user)
- Object → Block
- Operation → activity or Block operation
- Decomposition
  - Or → multiple <<refine>>
  - And → composition
- Satisfy → <<satisfy>>

Conclusions

- KAOS
  - Goal-oriented modeling language
  - Special role at Requirements elicitation
- SysML is:
  - a specific language for complex systems
  - strongly UML-Based
  - focusing on analysis
- SysML is not:
  - a method
  - just a UML profile
  - sufficient in itself
- Synergy between KAOS and SysML!

Future challenges

- Develop an MDD framework to automatically derive SysML requirement and block models from KAOS models and vice-versa.
- How to address the KAOS model elements that do not have direct correspondence to SysML
  - E.g. Obstacles and resolutions, conflicts



Thanks for your attention! Question / Discussions

- Any question?
- Anycomments?

No? Well,We've some for you…

- What is the SysML?
- Why do systems engineer need yet-another-modeling-language?
- What is the relationship between open source SysML and OMG SysML™?
- What is the current version of SysML and how can I obtain it?
- What changes were made during the last revision?
- What is the roadmap for OMG SysML 2.0?
- Who are the SysML Partners?
- What is the relationship between UML and SysML?
- Can SysML and UML be used together?
- Can SysML be customized?
- Which language is easier to learn, SysML or UML?

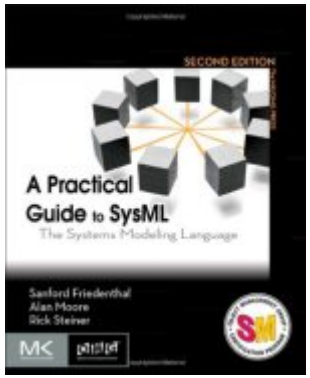
=> see http://www.sysmlforum.com/FAQ.htm

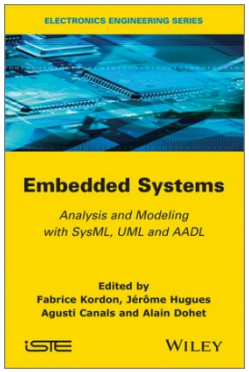
References

- A. v. Lamsweerde. "GoalOriented Requirements Engineering: A Guided Tour", presented at the 5th IEEE International Symposium on Requirements Engineering, Toronto, Canada, 2001.
- A. v. Lamsweerde. Requirements Engineering: From System Goals to UML Models to Software Specifications. Hoboken, USA: John Wiley & Sons, Inc., 2009.
- Jean Michel Bruel and Pascal Roques. "Présentation des concepts de SysML. Chap. 4 of the book: "Modélisation et analyse de systèmes embarqués", Hermès Book, To be published in June 2013.
- Manzoor Ahmad, JeanMichel Bruel, R´egine Laleau, Christophe Gnaho. Using RELAX, SysML and KAOS for Ambient Systems Requirements Modeling. Procedia Computer Science 10 (2012) 474-481.
- Jon Whittle, Pete Sawyer, Nelly Bencomo, Betty H. C. Cheng and JeanMichel Bruel. RELAX: A Language to Address Uncertainty in SelfAdaptive Systems Requirements.
- I. Sommerville, Software Engineering, Addison-Wesley, 9th Edition, 2010

References and links

- Books
  - « A Practical Guide to SysML », A. Moore, R. Steiner, S. Friedenthal, The MK/OMG Press, MK/OMG Press, 2011 (2nd edition).
  - « Embedded Systems Analysis and Modelingwith SysML, UML and AADL », F. Kordon, J. Hugues, A. Canals, A. Dohet, Wiley, 2013.





- Internet
  - OMG, Object Management Group (http://www.omgsysml.org/)
  - AFIS, Association Françaised'IngénierieSystème, (http://www.afis.fr/)
  - INCOSE, International Council on Systems, (http://www.incose.org/)
  - The SysML spec: http://www.omg.org/spec/SysML/1.3/PDF

References and links (ctd.)

- Tools
  - Papyrus (http://www.papyrusuml.org)
  - TopCased (http://topcased.gforge.enseeiht.fr/)
  - Artisan Software / Real-time Studio (http://www.artisansw.com/)
  - Embedded Plus / SysML Toolkit for RSDP (http://www.embeddedplus.com/)
  - I-Logix / Rhapsody (http://www.ilogix.com/sublevel.aspx?id=53)
  - SparxSystems / Enterprise Architect (http://www.sparxsystems.com/sysml)
  - Telelogic / Tau G2 (http://www.telelogic.com/products/tau/index.cfm)
  - MagicDraw (http://www.nomagic.com/)