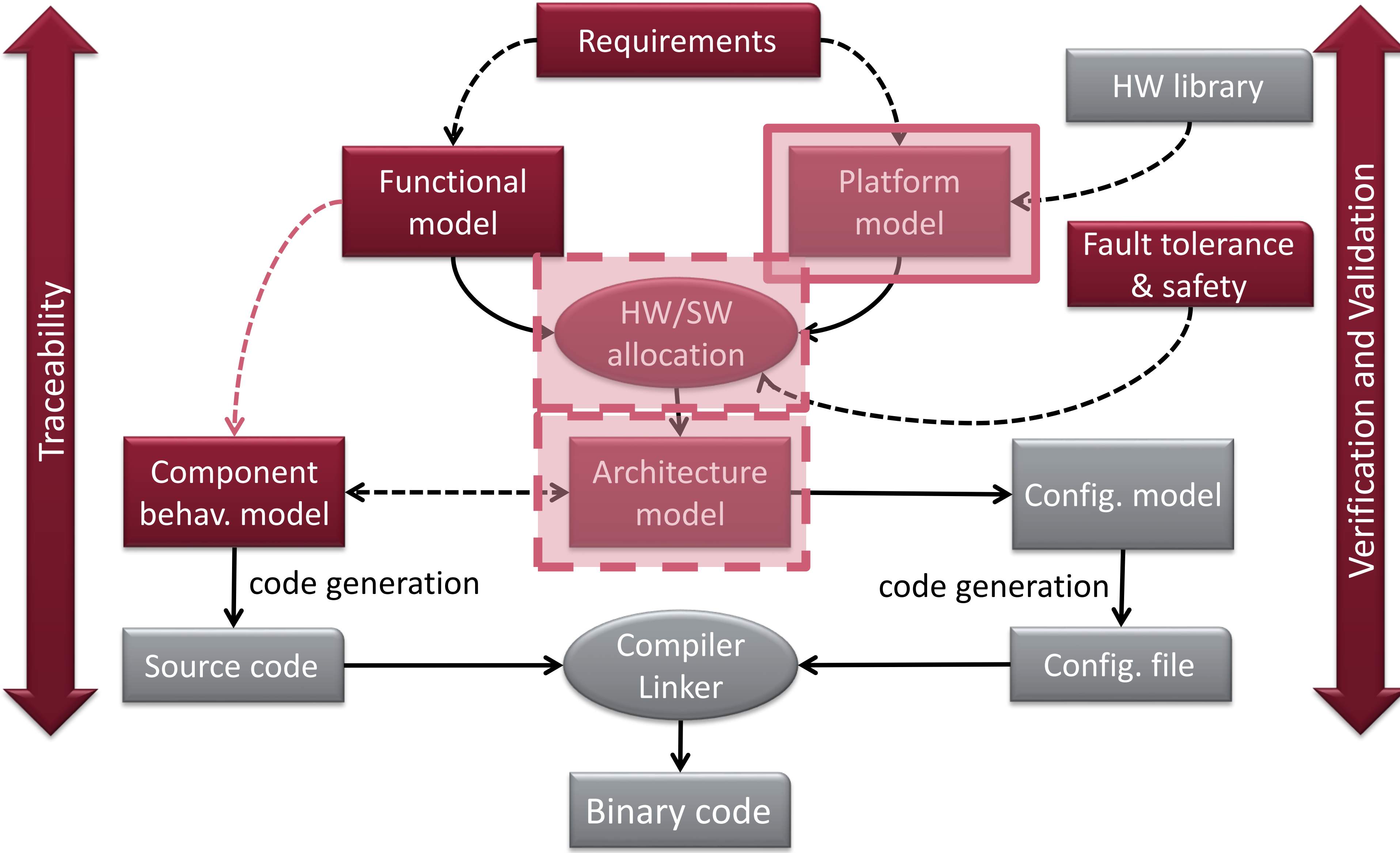
Platform model, allocation, integrated models and their analysis

Systems Engineering BSc Course

Budapest University of Technology and Economics
Department of Measurement and Information Systems

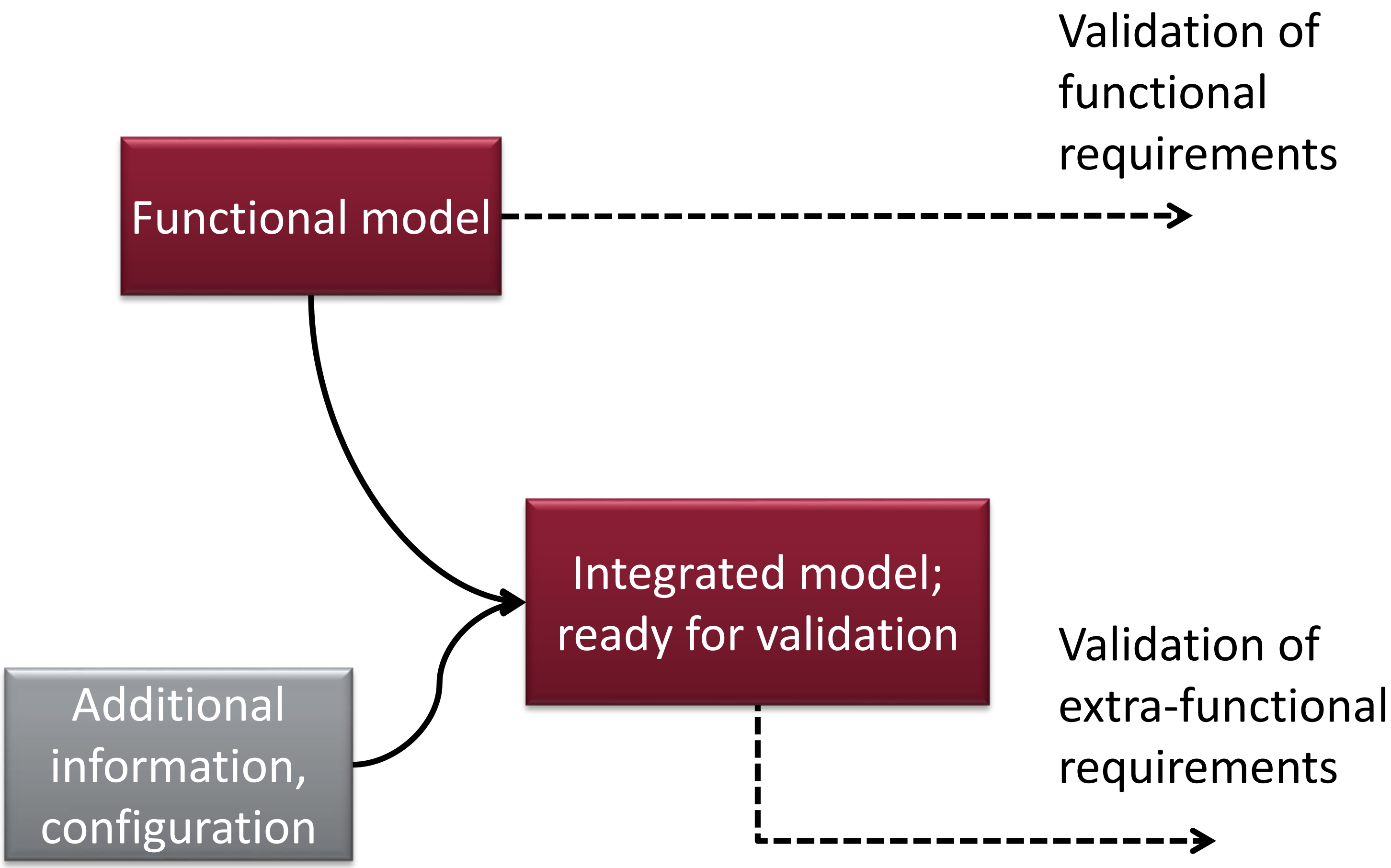
Platform-based systems design



System properties

- **Functional requirements → Functional properties**: functions that the system is able to perform
  - including how the system behaves while operating – also called operational properties.
- **Extra-functional requirements → Extra-functional properties**: they do not have a bearing on the functionality of the system, but describe attributes, constraints, performance considerations, design, quality of service, environmental considerations, failure and recovery.

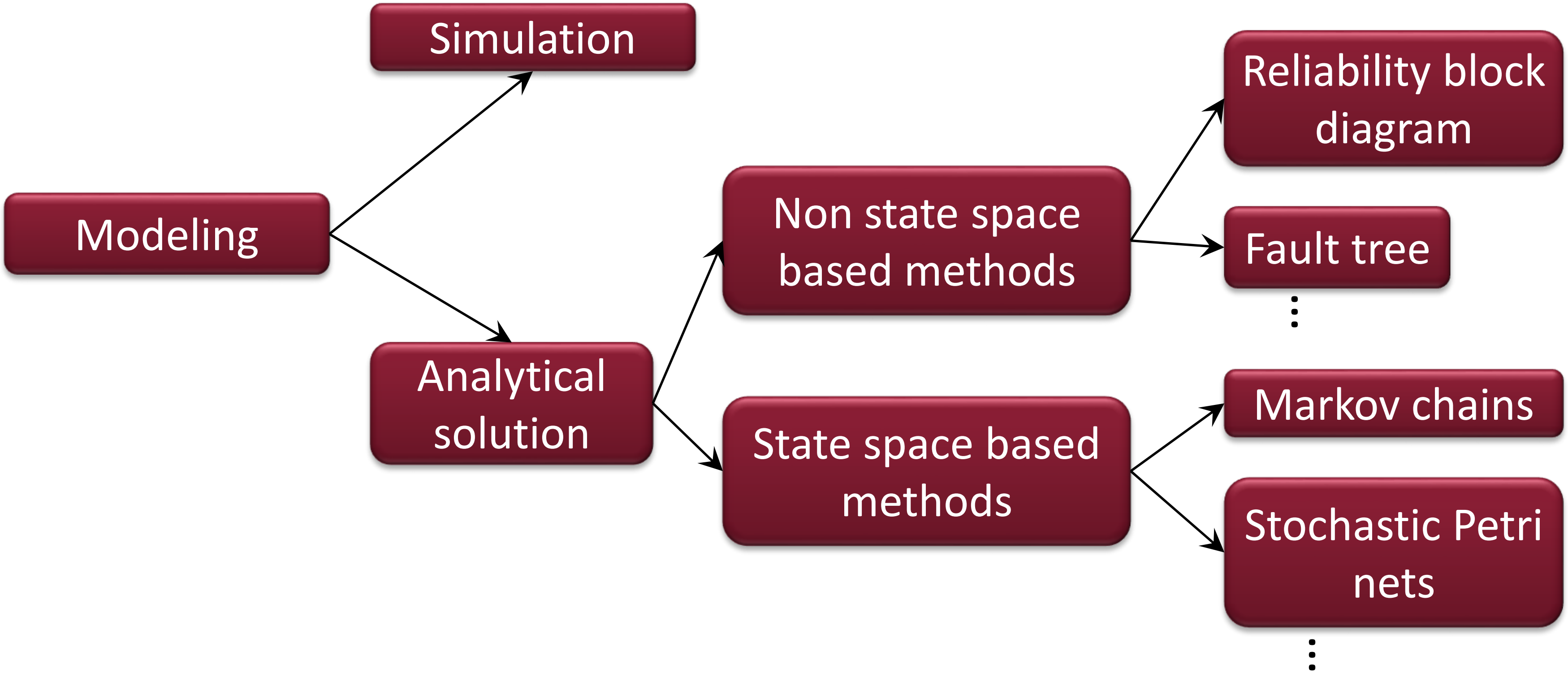
Approach


Validation of functional requirements
Functional model
Integrated model; ready for validation
Additional information, configuration
Validation of extra-functional requirements

Extra-functional properties

- **Dependability**: the ability to deliver service that can justifiably be *trusted*.
- Attributes of dependability:
  - **availability**: readiness for correct service.
  - **reliability**: continuity of correct service.
  - **safety**: absence of catastrophic consequences on the user(s) and the environment.
  - **integrity**: absence of improper system alterations.
  - **maintainability**: ability to undergo modifications and repairs
- **Performability**: If the performance of a computing system is "degradable" performance and reliability issues must be dealt with simultaneously in the process of evaluating system effectiveness. For this purpose, a unified measure, called "performability" is introduced and the foundations of performability modeling and evaluation are established.

Example: dependability analysis taxonomy



Why platform models are needed

Runtime platform

- Systems provide functions
- Functions are defined using
  - Functional models
  - Component behavior models
- How to realize these functions?

Runtime platform

- Systems provide functions
- Functions are defined using
  - Functional models
  - Component behavior models
- How to realize these functions? → in Software!

Runtime platform

- Systems provide functions
- Functions are defined using
  - Functional models
  - Component behavior models
- How to realize these functions? → in Software!
  - Maybe in hardware? (e.g., sensors, GPU, FPGA, etc.)
  - What will execute our software functions?
  - How will they be able to communicate

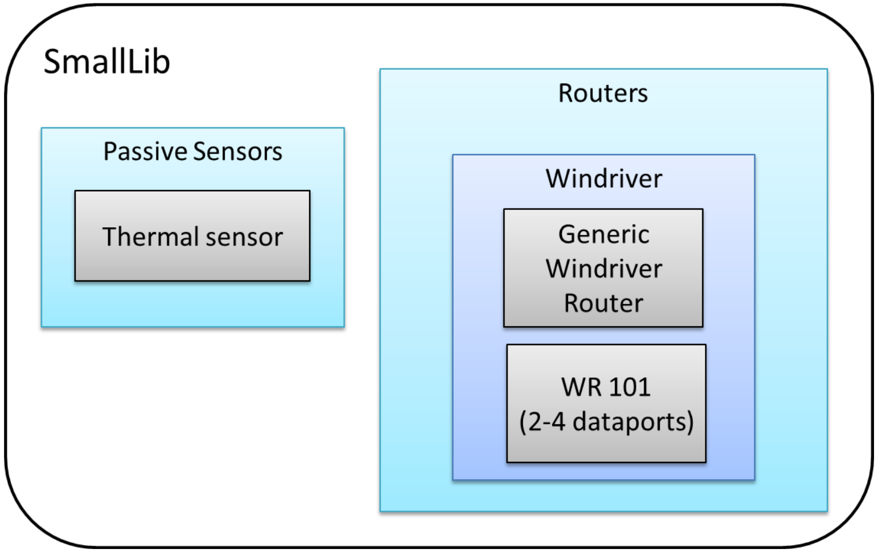
Platform model

- The platform model specifies the physical building blocks of the execution platform
  - the execution resources
    - memory, CPU, etc.
  - the available communication resources
    - Network interfaces, routers, etc.
  - the properties of the used HW elements
    - Weight
    - Availability
    - Size
    - etc.

Defining the platform model I.

- Resource capturing phase
  - Specification of reusable hardware entities
    - Coming from HW libraries/technical dictionaries
    - Defined by HW designers within the project

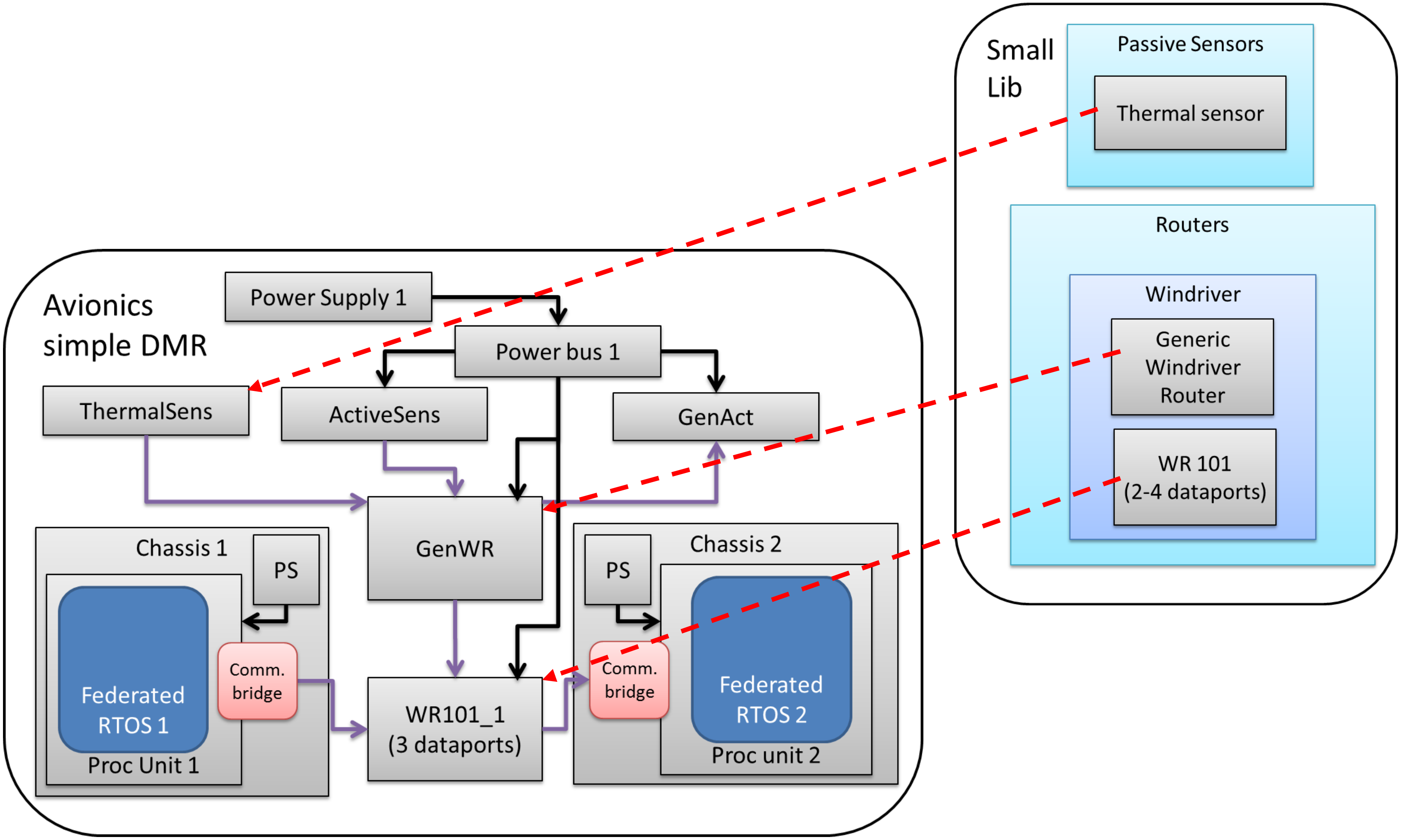
    → atomic hardware units of the execution platform
      - Embedded systems: Processor, Communication controller
      - Define hardware properties



Defining the platform model II.

- Platform composition phase
  - (Already available HW design → only modifications)
  - Definition from bottom-up based on the atomic building blocks
- Similar modeling task as the functional component definition BUT
  - Connecting blocks == physical linkage
  - Part-whole relationship == physical containment
  - Physical HW properties are needed to be taken into consideration
    - Size, weight, number of ports, etc.

Defining the platform model II.

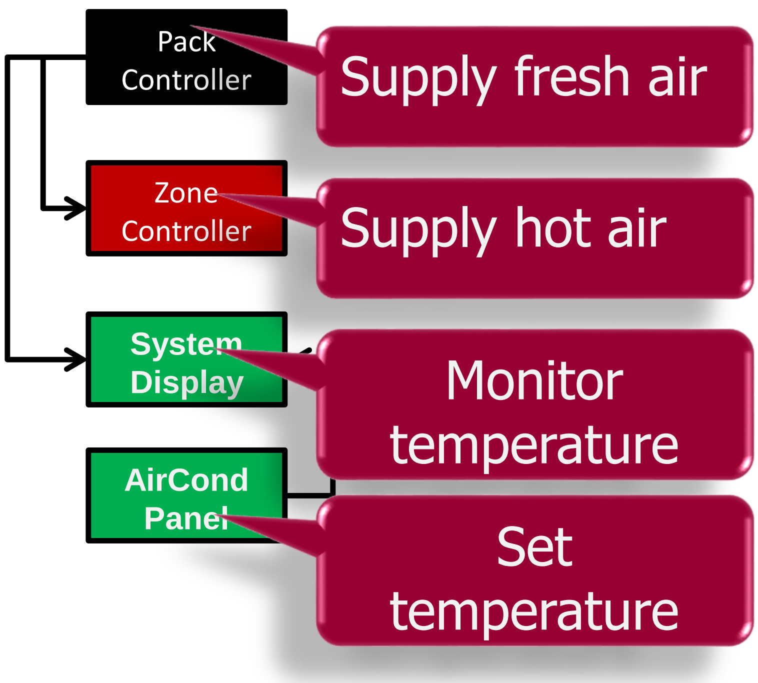


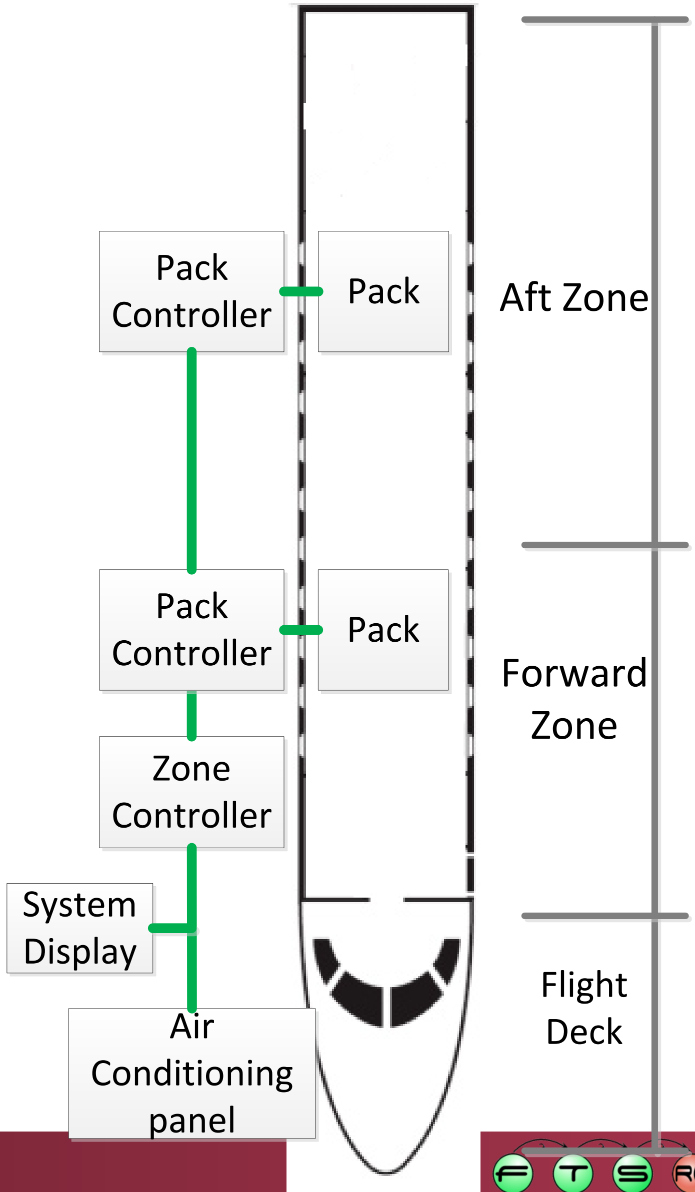
Functions to Platform allocation

Usually HW-SW allocation

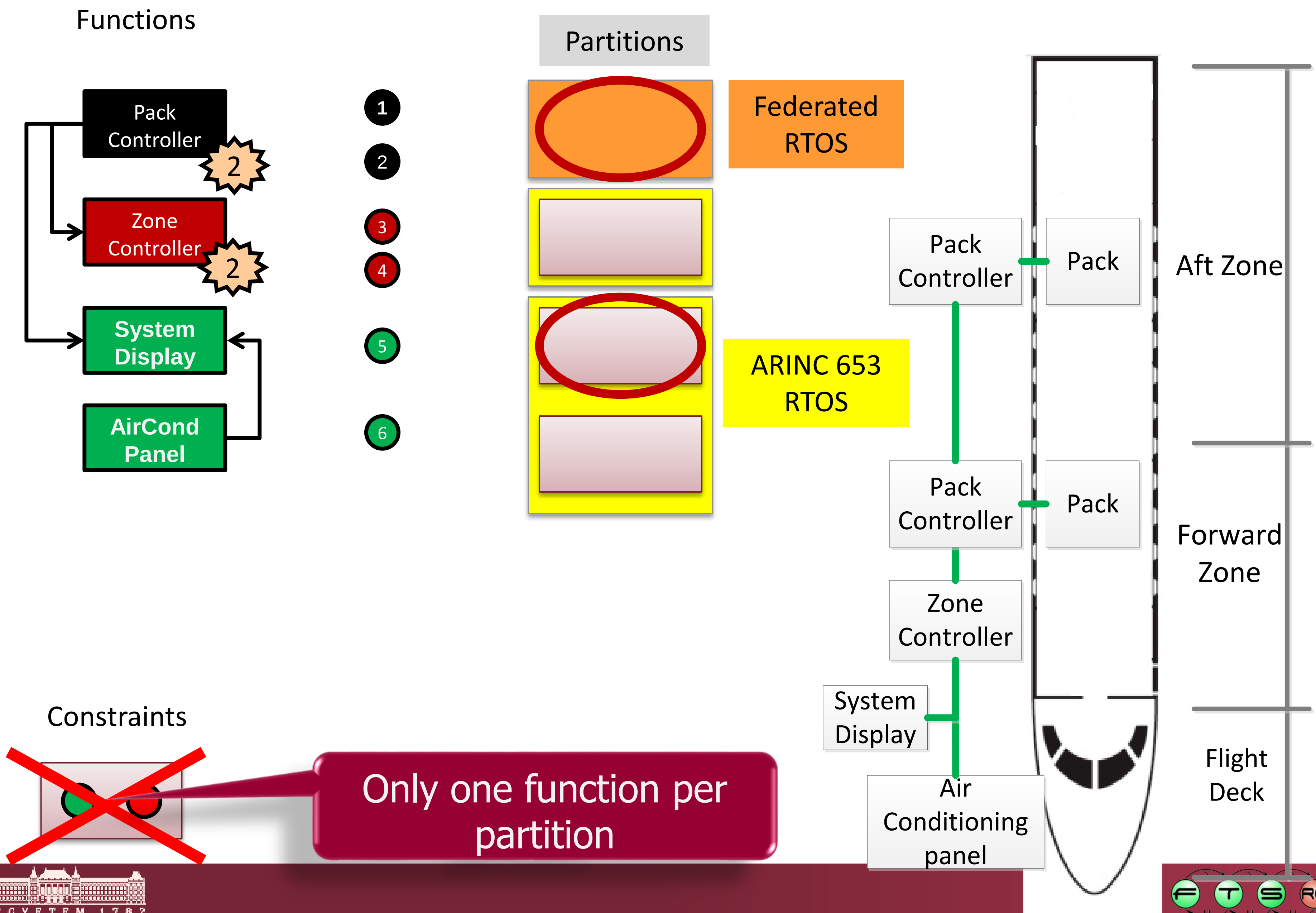
Allocation example

Functions

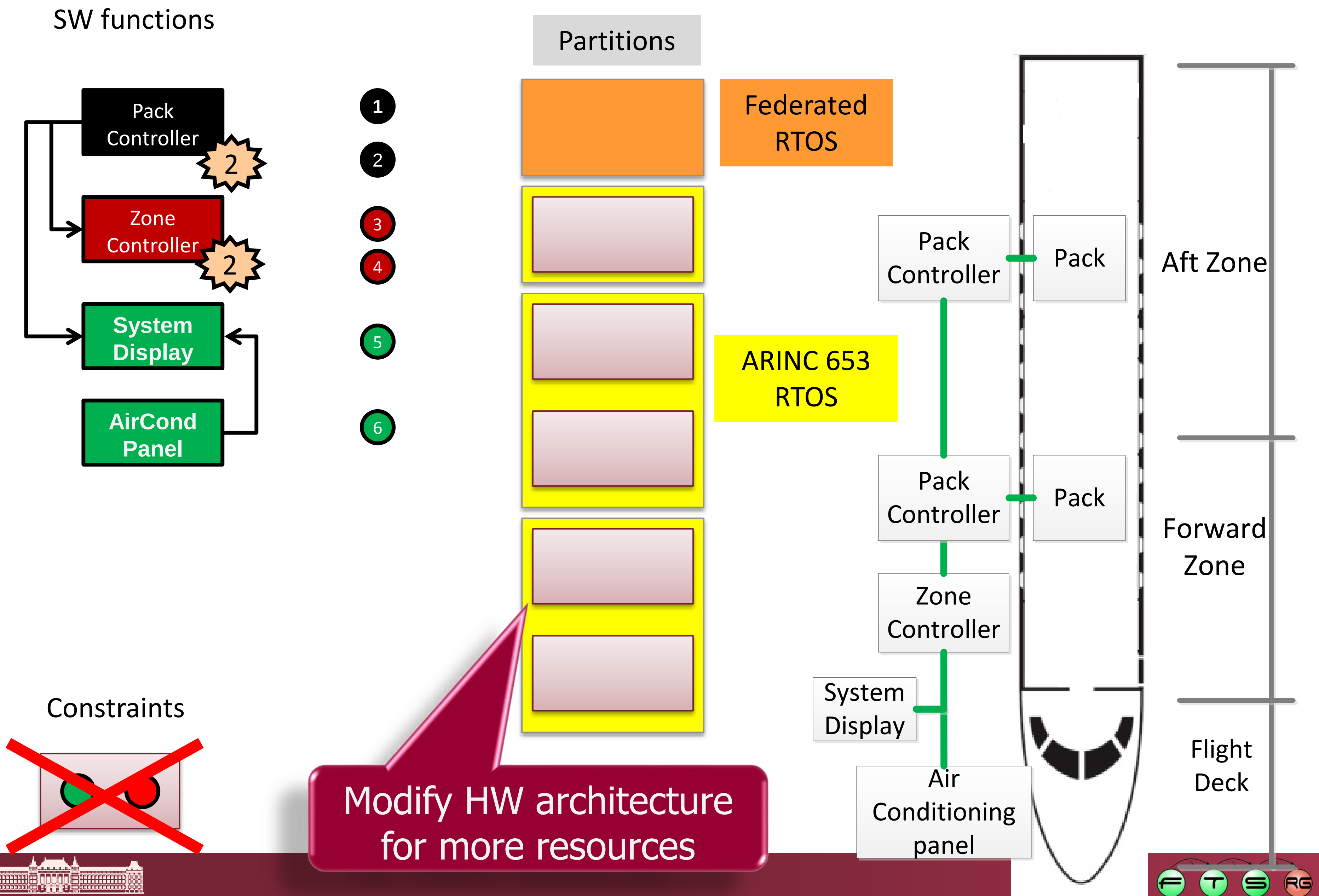




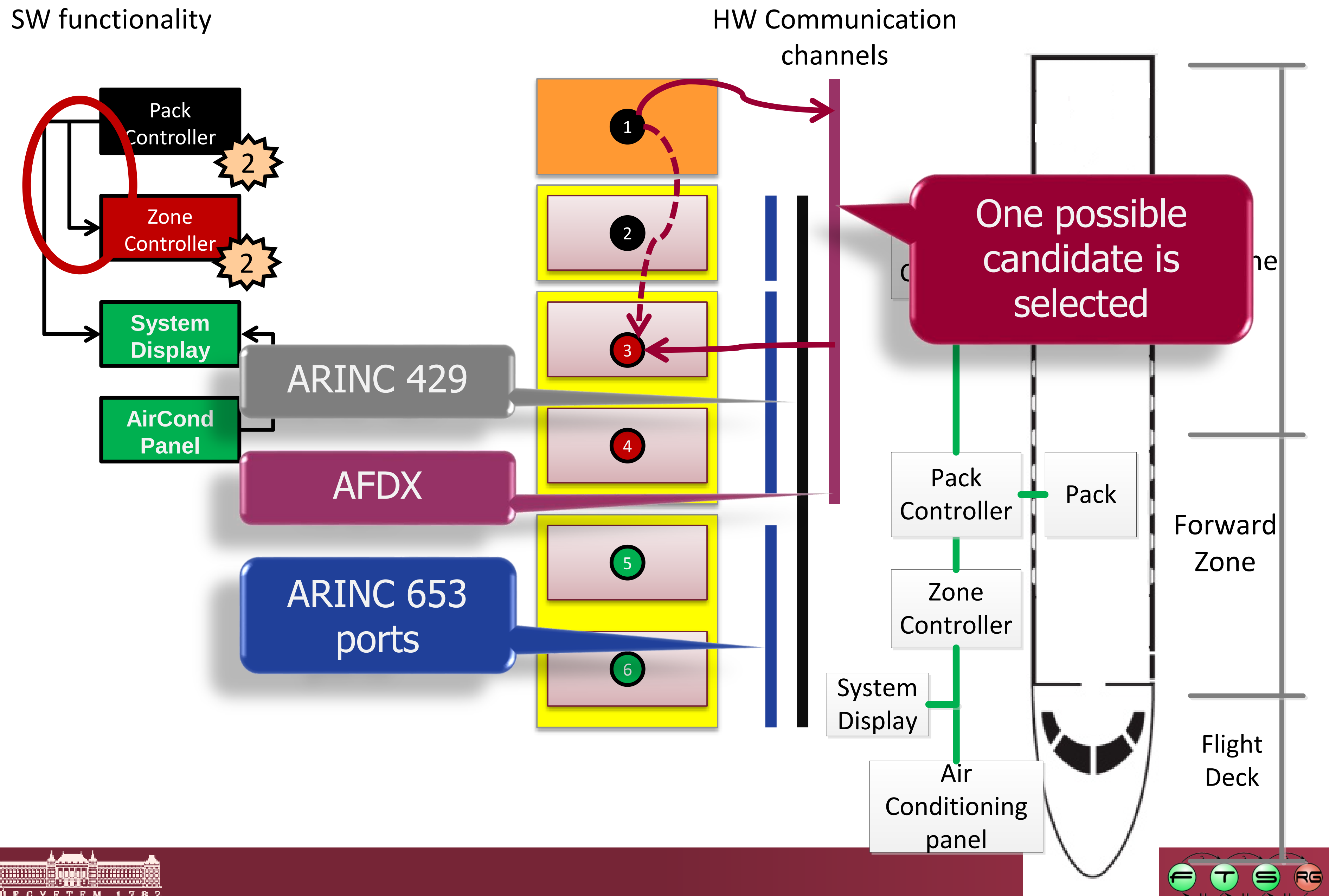
Allocation example – functions to partitions



Allocation example – functions to partitions



Allocation example – communication channels



Allocation

- **Input:**
  - Functional model + platform model
  - Additional non-functional constraints
- **Output:**
  - System Architecture

- The *System Architecture* defines for each instance of a Function
  - *where and when to execute*
  - *when to communicate*
  - *and on which bus*

Where and when to execute

- Allocate the functions to their designated execution resource
  - Processor, GPU, server, node, etc.
- Schedule the execution of functions
  - Based on their required execution window
    - **Major driver of the allocation process**
- Constraints (usually) taken into consideration

- Platform (HW)
  - Available memory
  - CPU performance
  - Redundancy

- Functional (SW)
  - Memory required
  - Execution window required
  - Safety aspects
    - E.g., criticality levels

When to communicate and on which bus

- Allocate Function model level communication means to platform communication resources
  - Information flow to bus mapping
  - Data/message mapping to platform representation
  - Scheduling
    - Messages, buses, routers
    - **Major driver of the allocation process**
  - Constraints (usually) taken into consideration
- Platform (HW)
  - Connectivity
    - comm. architecture
    - Routing
    - Supported modes
  - Bandwidth & Speed
  - Precision
    - Data mapping
  - Redundancy
    - Independent paths
- Functional (SW)
  - Message properties
    - size
    - priority
  - Communication mode
    - 1-1, 1-n, n-n
  - Safety aspects
    - WCET

Additional aspects of the allocation

- Multi-level allocation
  - Complexity is handled on multiple abstraction-level → allocation is handled between all hierarchies

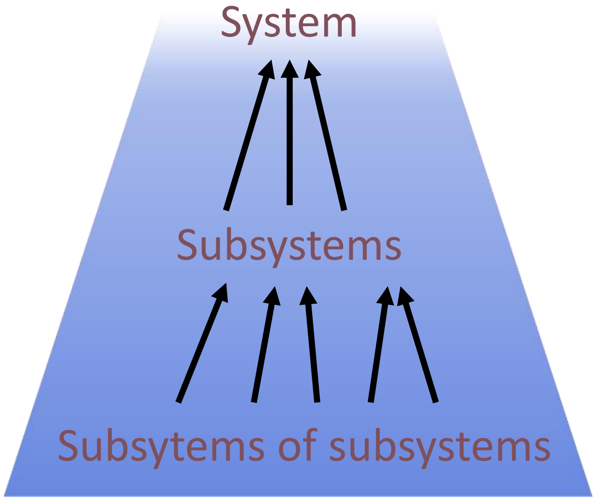
- Resulting System Architectures are used for validating system level functional/non-functional aspects
  - Timing requirements, safety requirements, etc.
  - Used methods: Static checks, simulations, HiL, etc.

- No perfect allocation → Multi-dimension optimization problem
  - Design Space Exploration

Modeling the platform in SysML

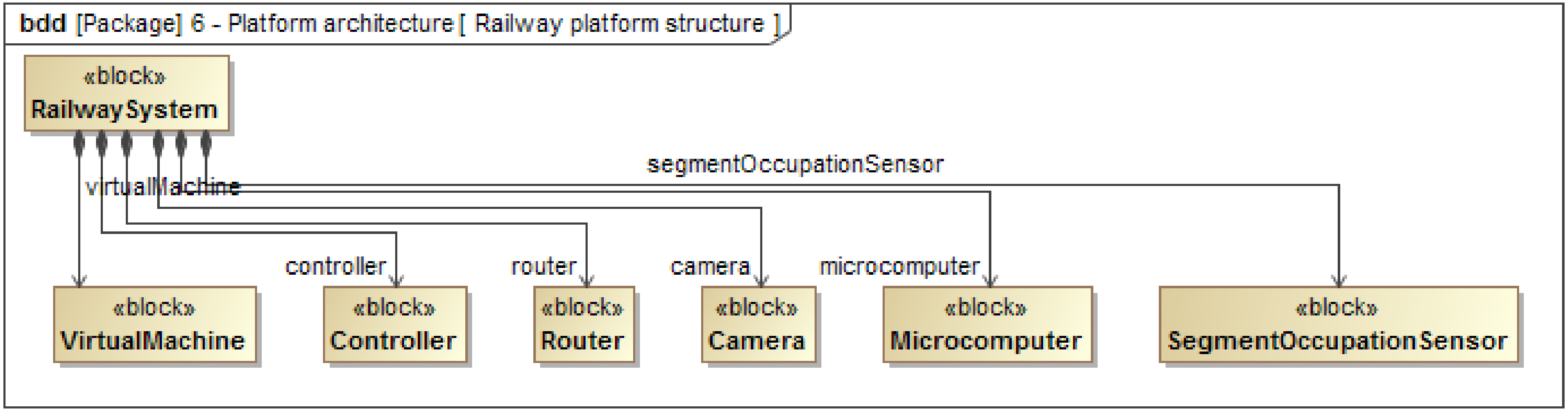
Platform modeling techniques

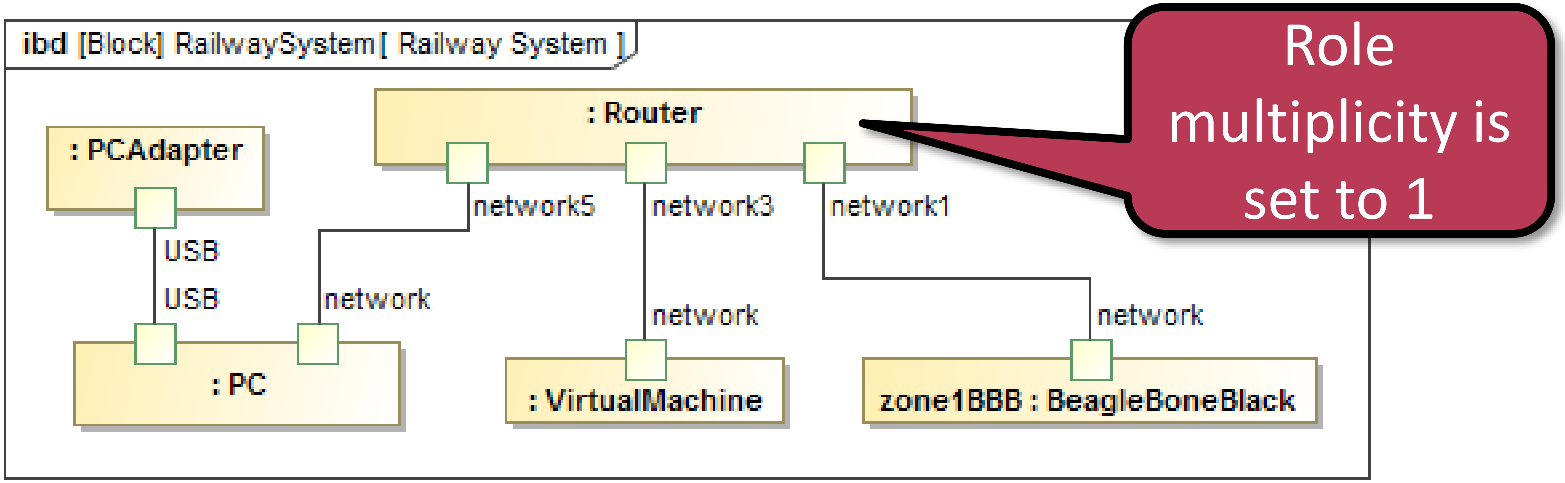
- Running platform is composed of existing (hardware) elements
- Approach: bottom-up using composition
  - ☺ Subsystems can be tested one-by-one
  - ☺ There are always some working parts during development
  - ☹ Exact roles of the subsystems are revealed late



Platform models in SysML

- Models composed of blocks → BDD, IBD are used.

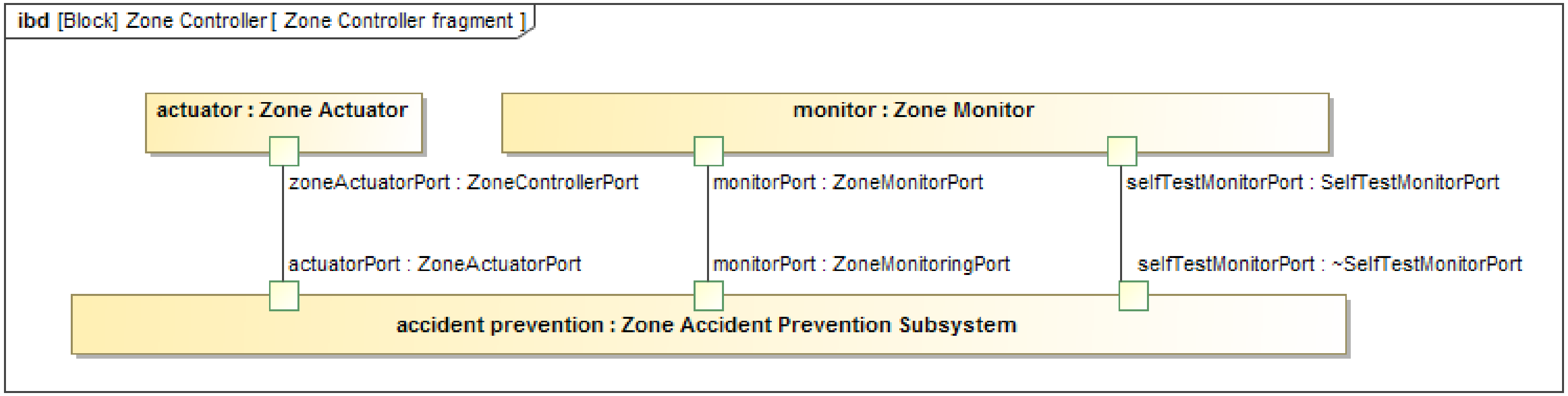


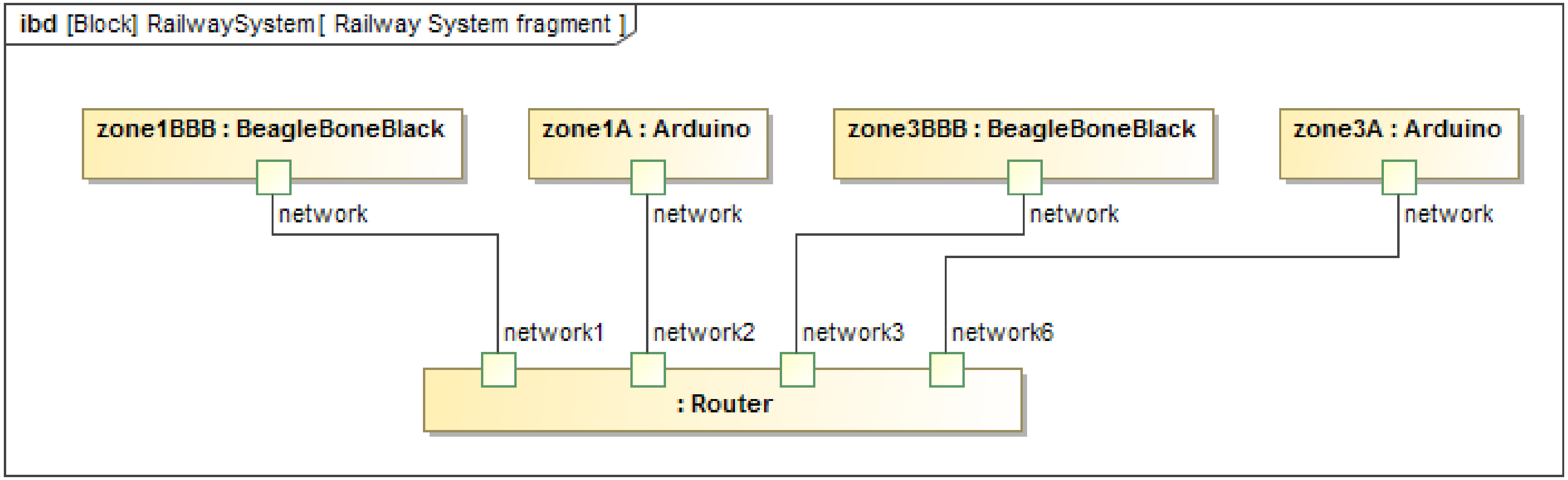


Modeling allocation in SysML

Allocation example: railway system

- Functional structure

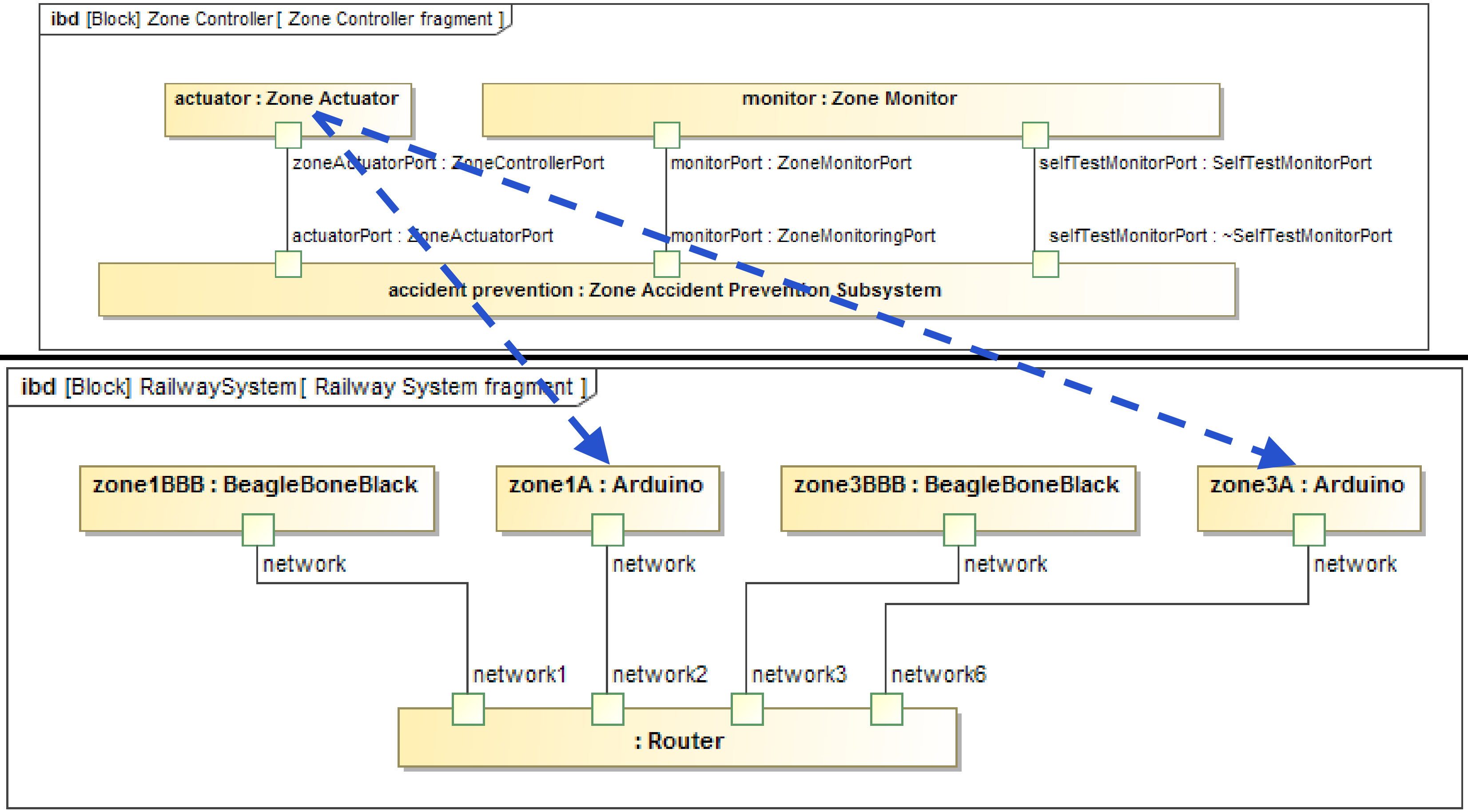




- Platform structure

Allocation example: railway system

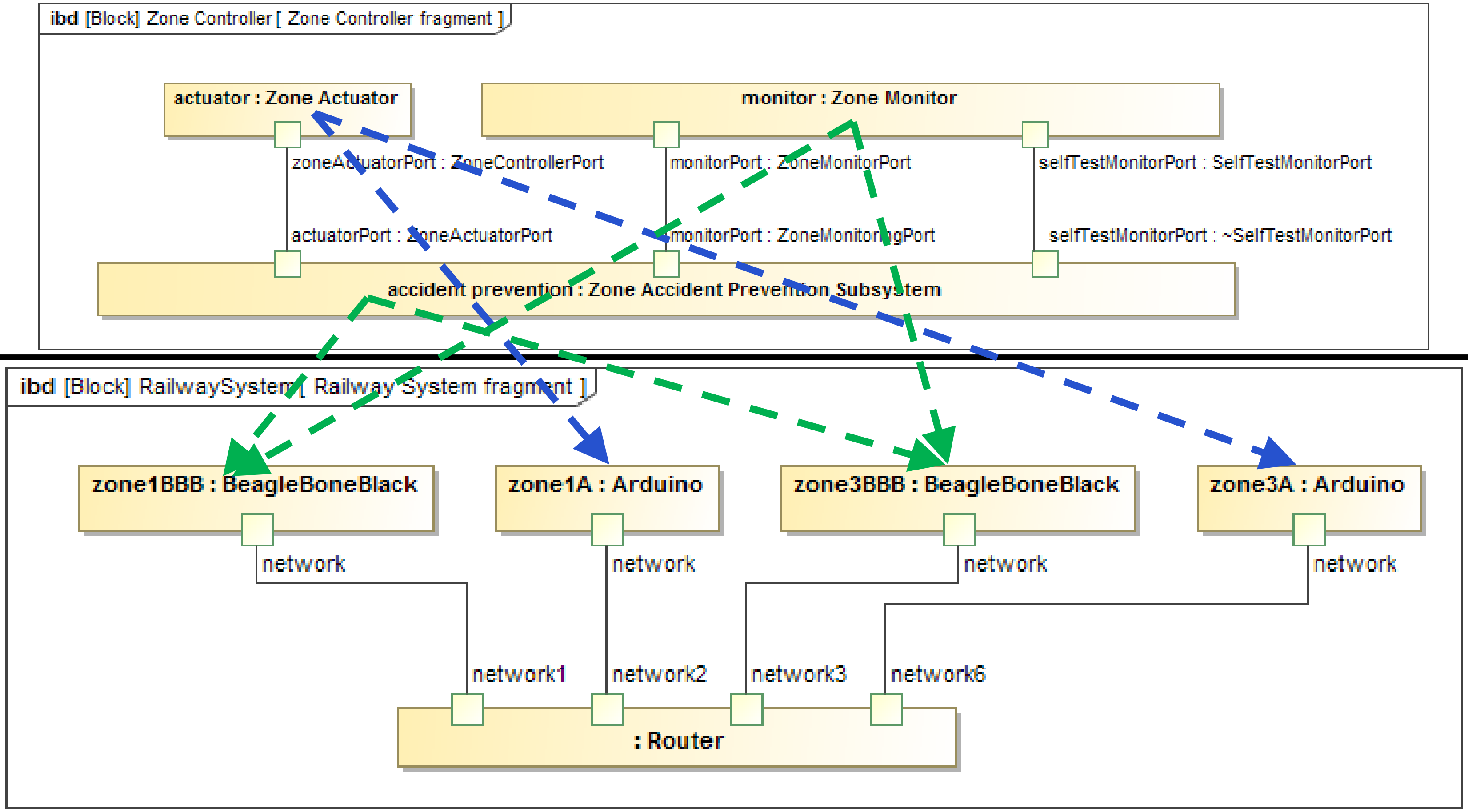
- Functional structure



- Platform structure

Allocation example: railway system

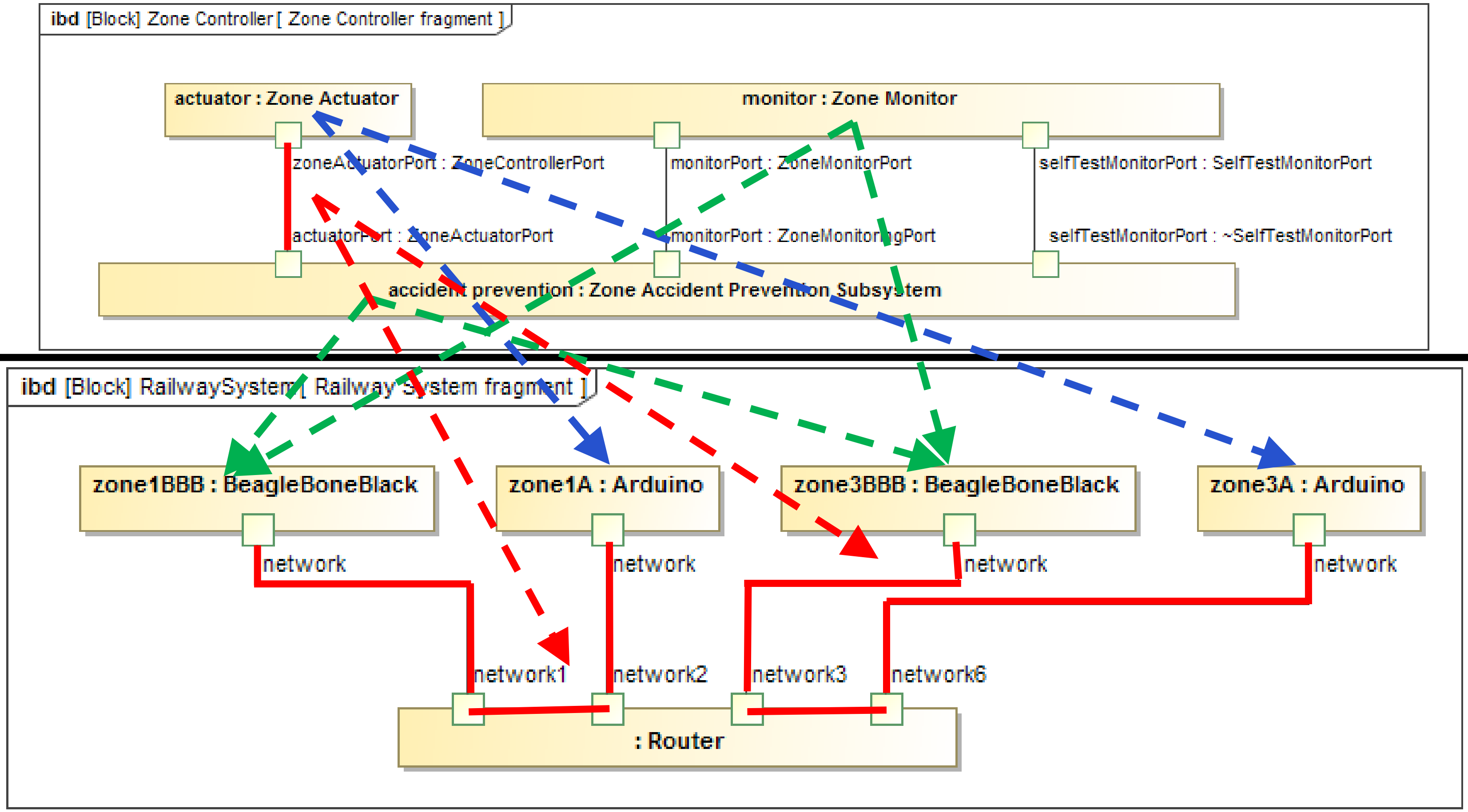
- Functional structure



- Platform structure

Allocation example: railway system

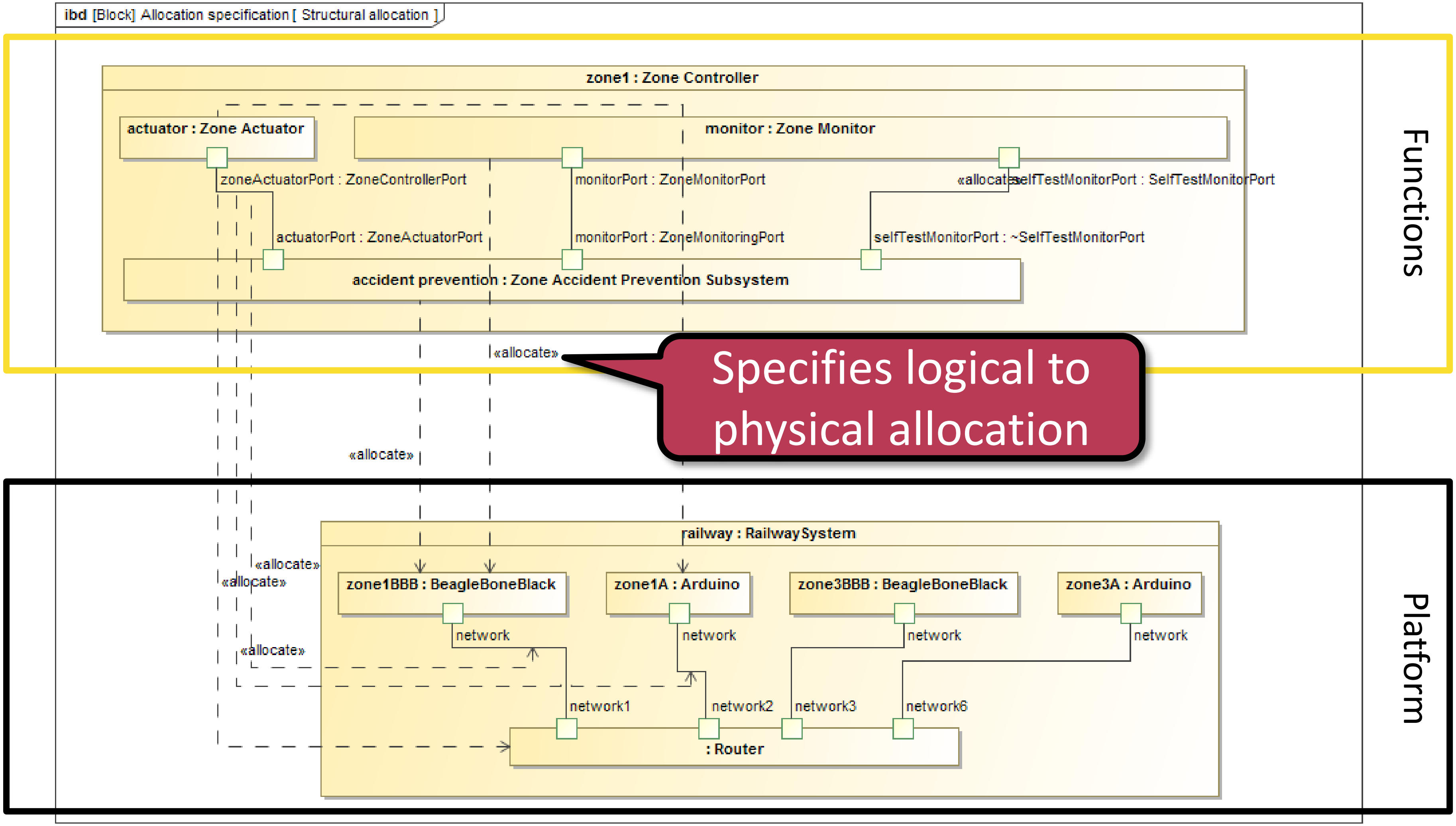
- Functional structure



- Platform structure

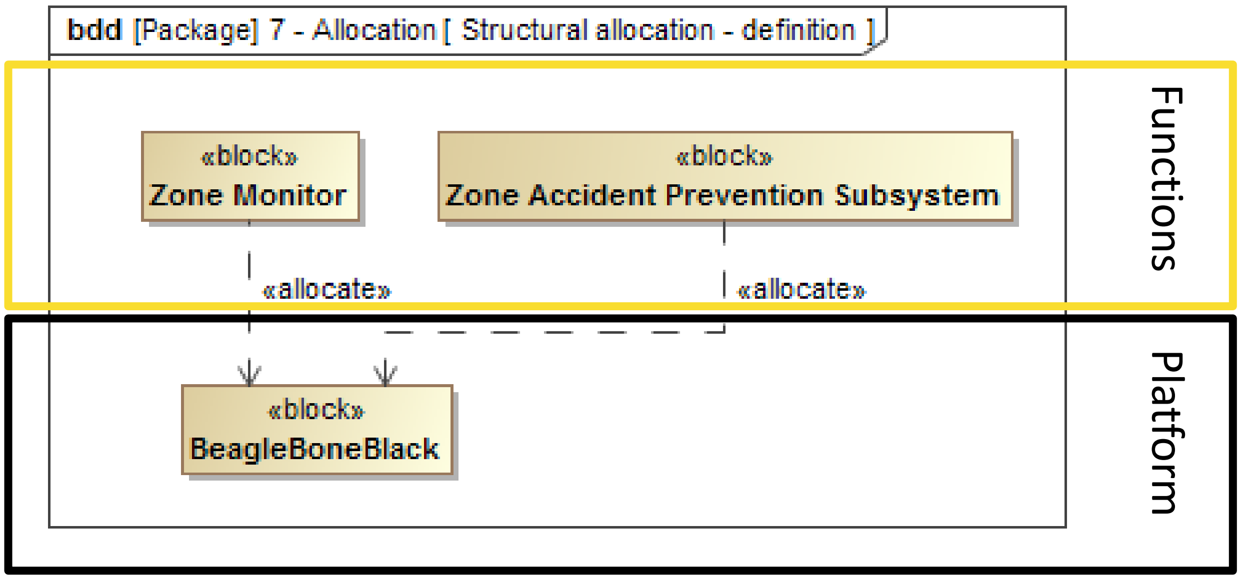
The allocation relation in SysML

- Structural allocation: usage



The allocation relation in SysML

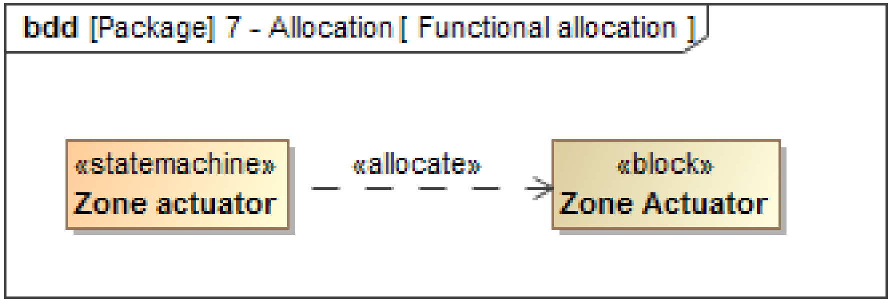
- Structural allocation: definition



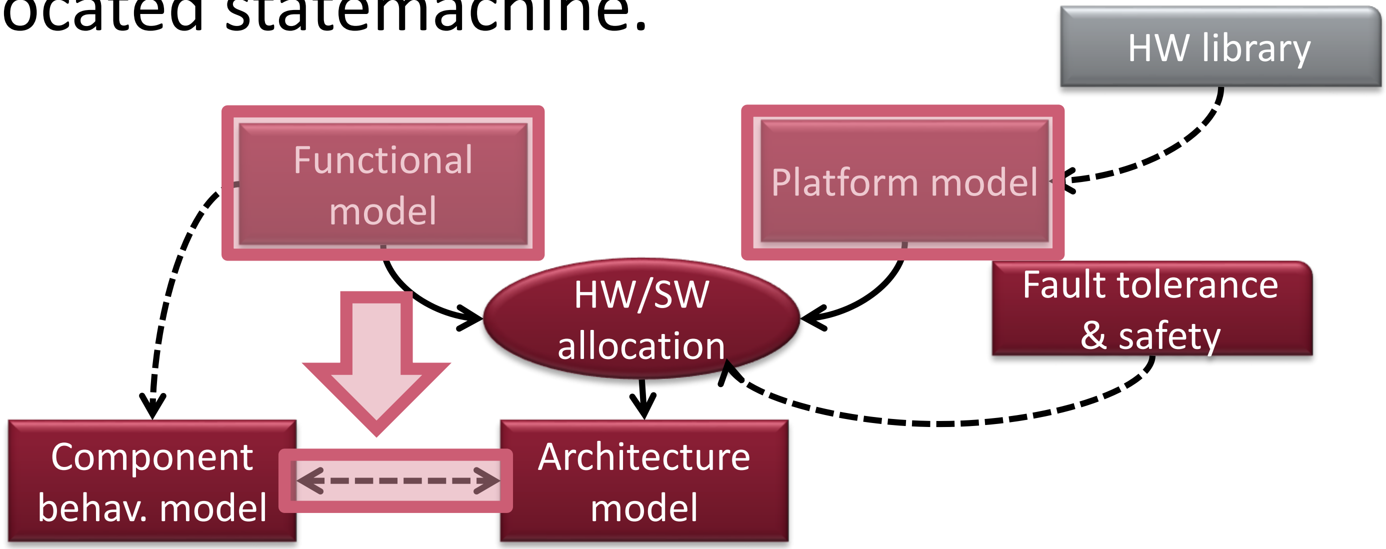
  - Wherever a BBB is used in the system, a zone monitor and an accident prevention subsystem is assumed to be allocated to it

The allocation relation in SysML

- Functional allocation: definition



  - A zone actuator behaves as it is described in the allocated statemachine.



SysML allocation matrix

Columns: functional elements

Rows: platform elements

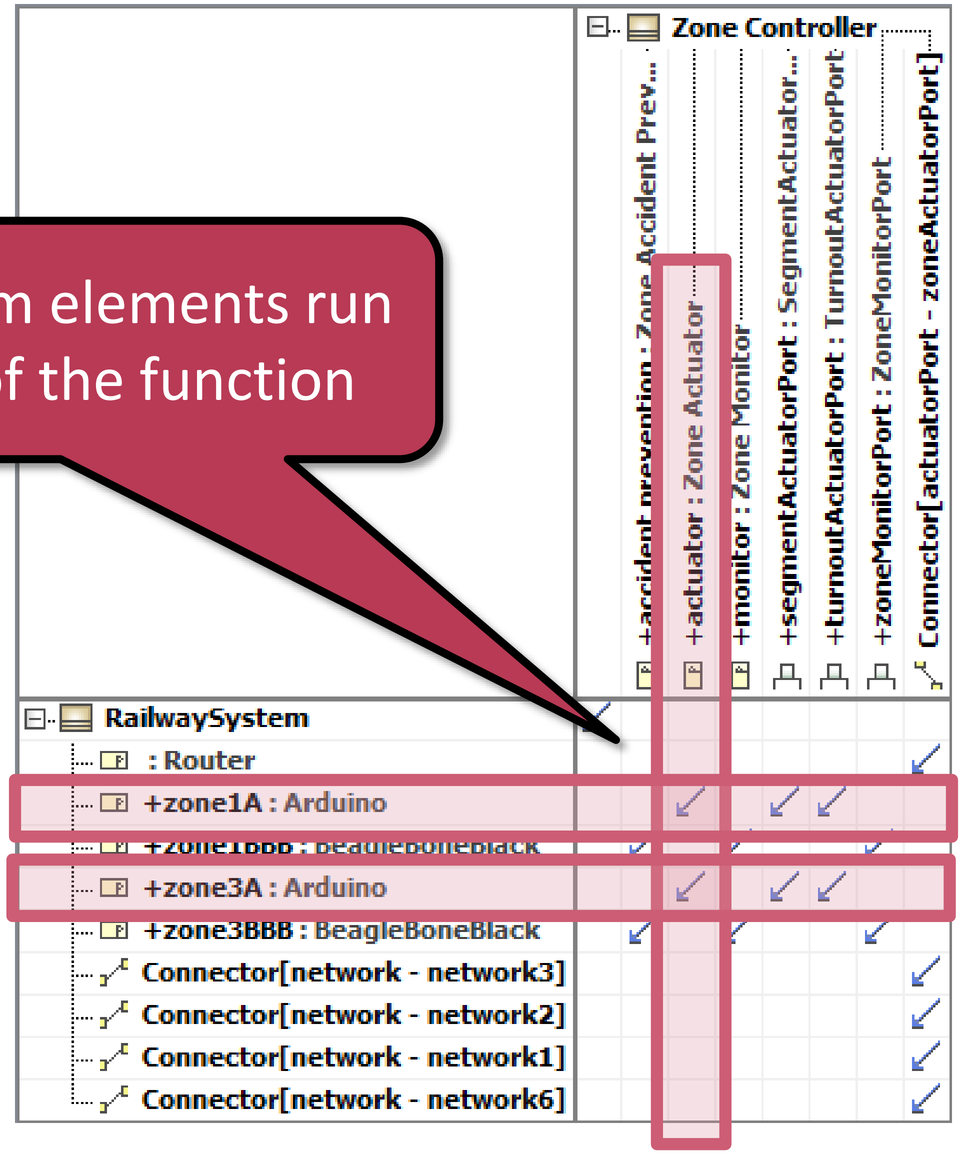
	Zone Controller	+accident prevention : Zone Accident Prev...	+actuator : Zone Actuator	+monitor : Zone Monitor	+segmentActuatorPort : SegmentActuator...	+turnoutActuatorPort : TurnoutActuatorPort	+zoneMonitorPort : ZoneMonitorPort	Connector[actuatorPort - zoneActuatorPort]
RailwaySystem	↙							
: Router								↙
+zone1A : Arduino			↙		↙	↙		
+zone1BBB : BeagleBoneBlack		↙		↙			↙	
+zone3A : Arduino			↙		↙	↙		
+zone3BBB : BeagleBoneBlack		↙		↙			↙	
Connector[network - network3]								↙
Connector[network - network2]								↙
Connector[network - network1]								↙
Connector[network - network6]								↙

SysML allocation matrix

Arrow pointing downward: function to platform allocation (colum to row)

	Zone Controller	+accident prevention : Zone Accident Prev...	+actuator : Zone Actuator	+monitor : Zone Monitor	+segmentActuatorPort : SegmentActuator...	+turnoutActuatorPort : TurnoutActuatorPort	+zoneMonitorPort : ZoneMonitorPort	Connector[actuatorPort - zoneActuatorPort]
RailwaySystem	↙							
: Router								↙
+zone1A : Arduino			↙		↙	↙		
+zone1BBB : BeagleBoneBlack		↙		↙			↙	
+zone3A : Arduino			↙		↙	↙		
+zone3BBB : BeagleBoneBlack		↙		↙			↙	
Connector[network - network3]								↙
Connector[network - network2]								↙
Connector[network - network1]								↙
Connector[network - network6]								↙

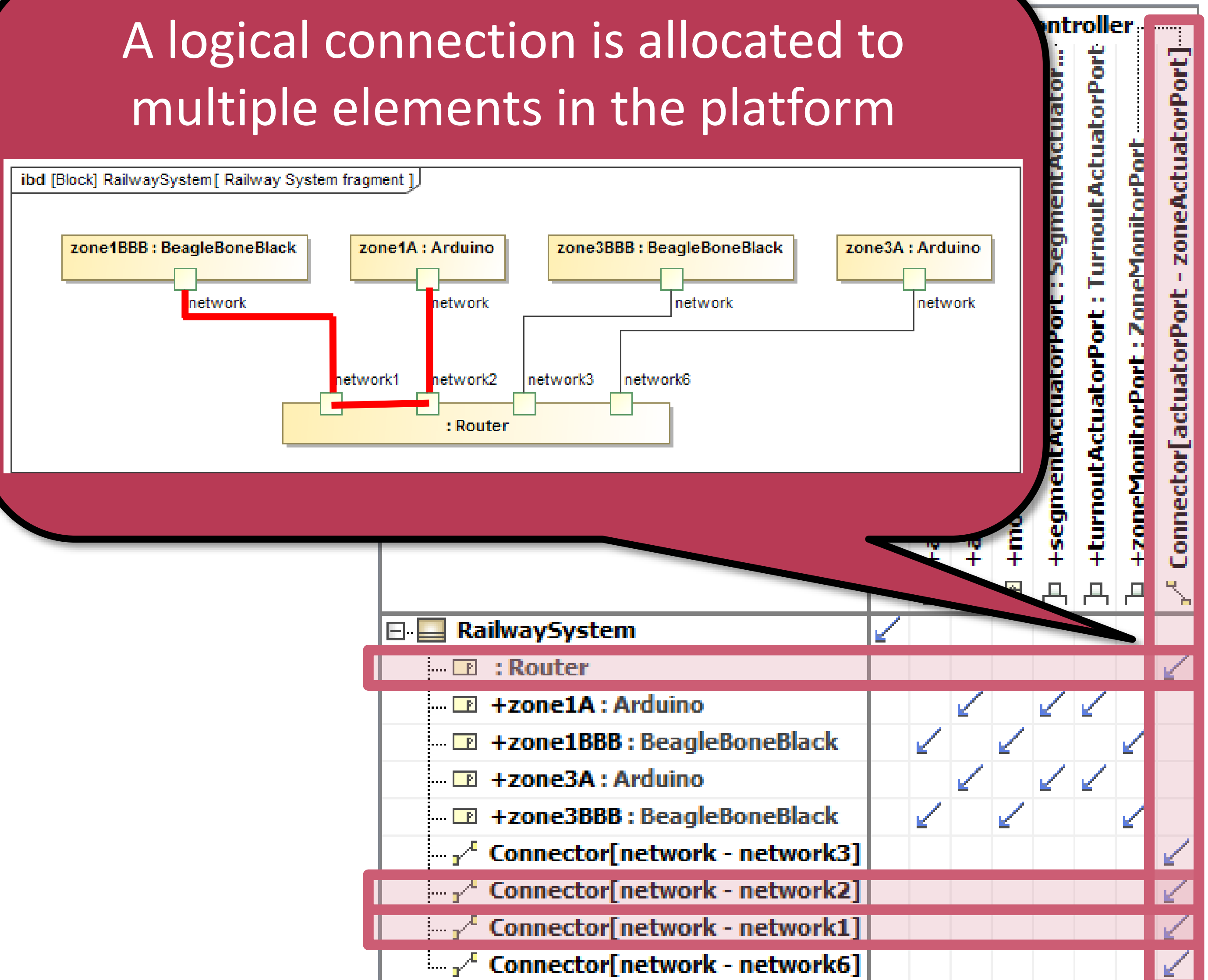
SysML allocation matrix



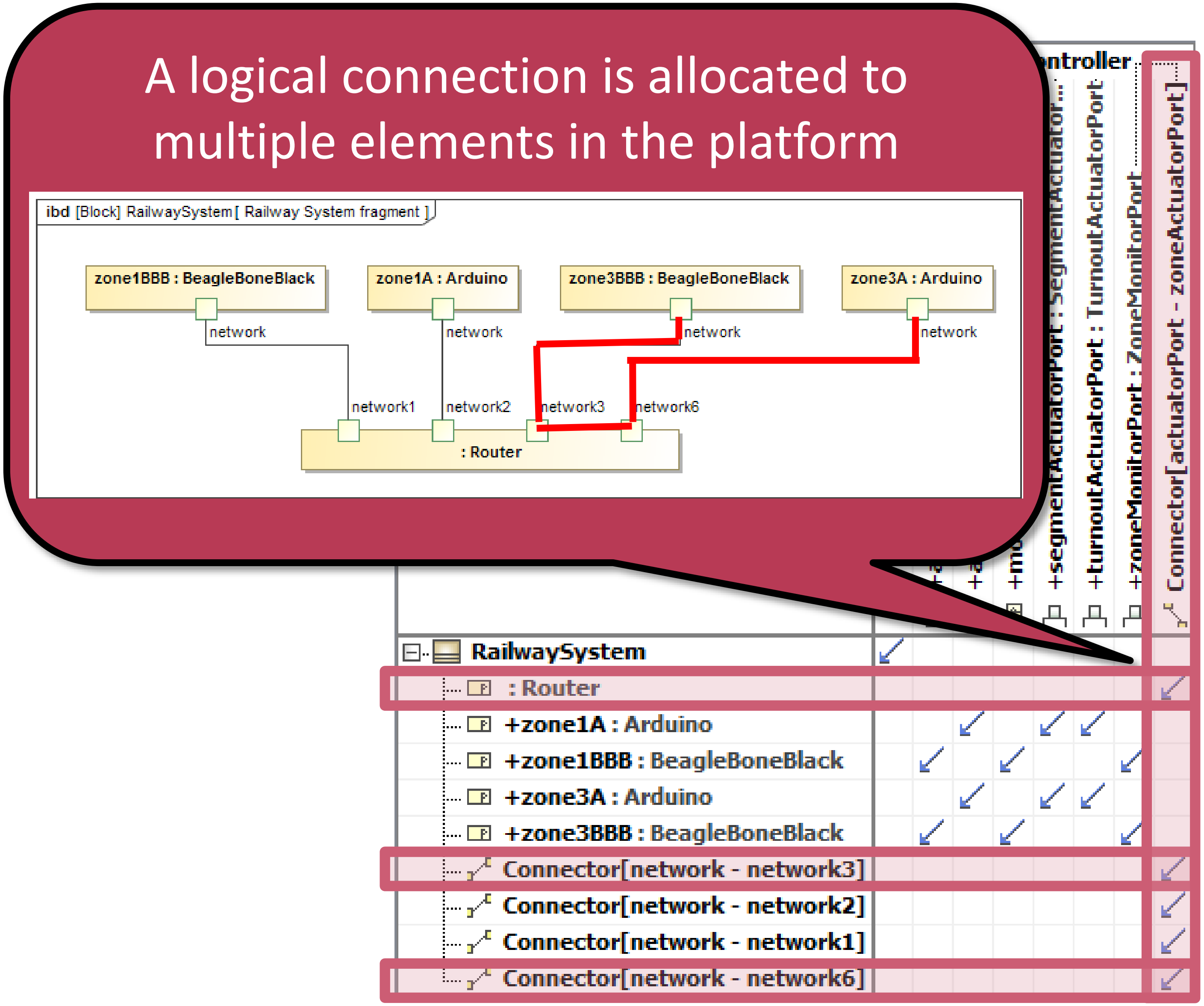
Multiple platform elements run the instances of the function

SysML allocation matrix

A logical connection is allocated to multiple elements in the platform



SysML allocation matrix



Allocation constraints

- Platform element capabilities
  - What kind of resources does the platform element have?
- Realization of connections
  - Are the connections between the functions supported by the platform?
- Standards and additional well-formedness rules
  - Such as „critical and non-critical functions shall not run on the same platform element”.

Advantages of allocation matrices

- A function cannot be deployed to the same device twice.
- Allocation of the logical connections can be validated by examining endpoints and continuity of the corresponding platform connection.
- By examining the safety levels of the allocated functions row by row, critical and non-critical functions cannot be allocated to the same device.

Best practices / Goals

- Avoid single point of failures
- Fault tolerant design patterns
  - See previous lecture on *Safety-critical systems: Architecture*
- Cost efficiency
  - Weight
  - Price

Extrafunctional properties

Analysis of extra-functional properties of a service

System properties (repetition)

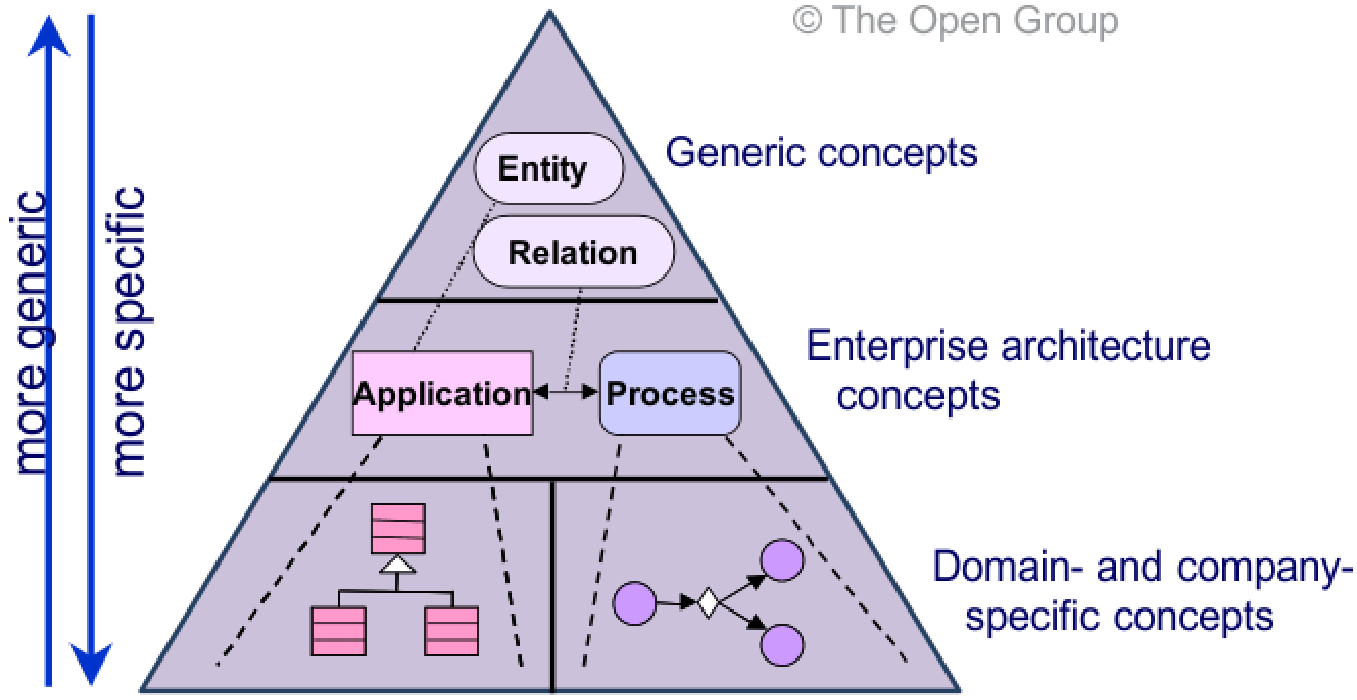
- **Functional requirements → Functional properties**: functions that the system is able to perform
  - including how the system behaves while operating – also called operational properties.
- **Extra-functional requirements → Extra-functional properties**: they do not have a bearing on the functionality of the system, but describe attributes, constraints, performance considerations, design, quality of service, environmental considerations, failure and recovery.

Case study

Modeling IT infrastructure using ArchiMate

IT system and infrastructure

- Challenge: find a modeling language that is not too general neither too specific for a given domain



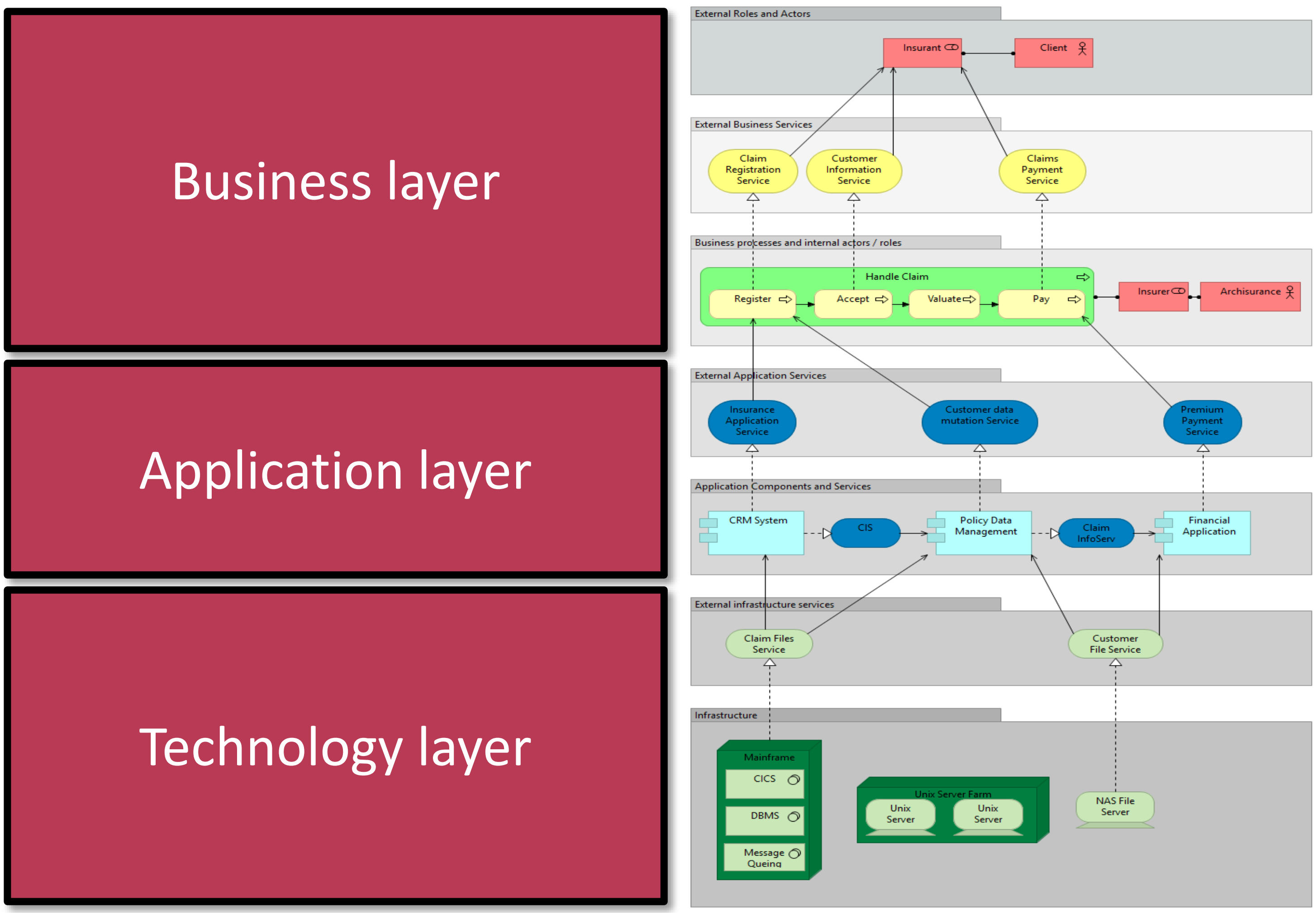
- Applies multi-level allocation

ArchiMate – infrastructure modeling

- The ArchiMate language defines three main layers
  - The *Business Layer* offers products and services to external customers, which are realized in the organization by business processes performed by business actors.
  - The *Application Layer* supports the business layer with application services which are realized by (software) applications.
  - The *Technology Layer* offers infrastructure services (e.g., processing, storage, and communication services) needed to run applications, realized by computer and communication hardware and system software.

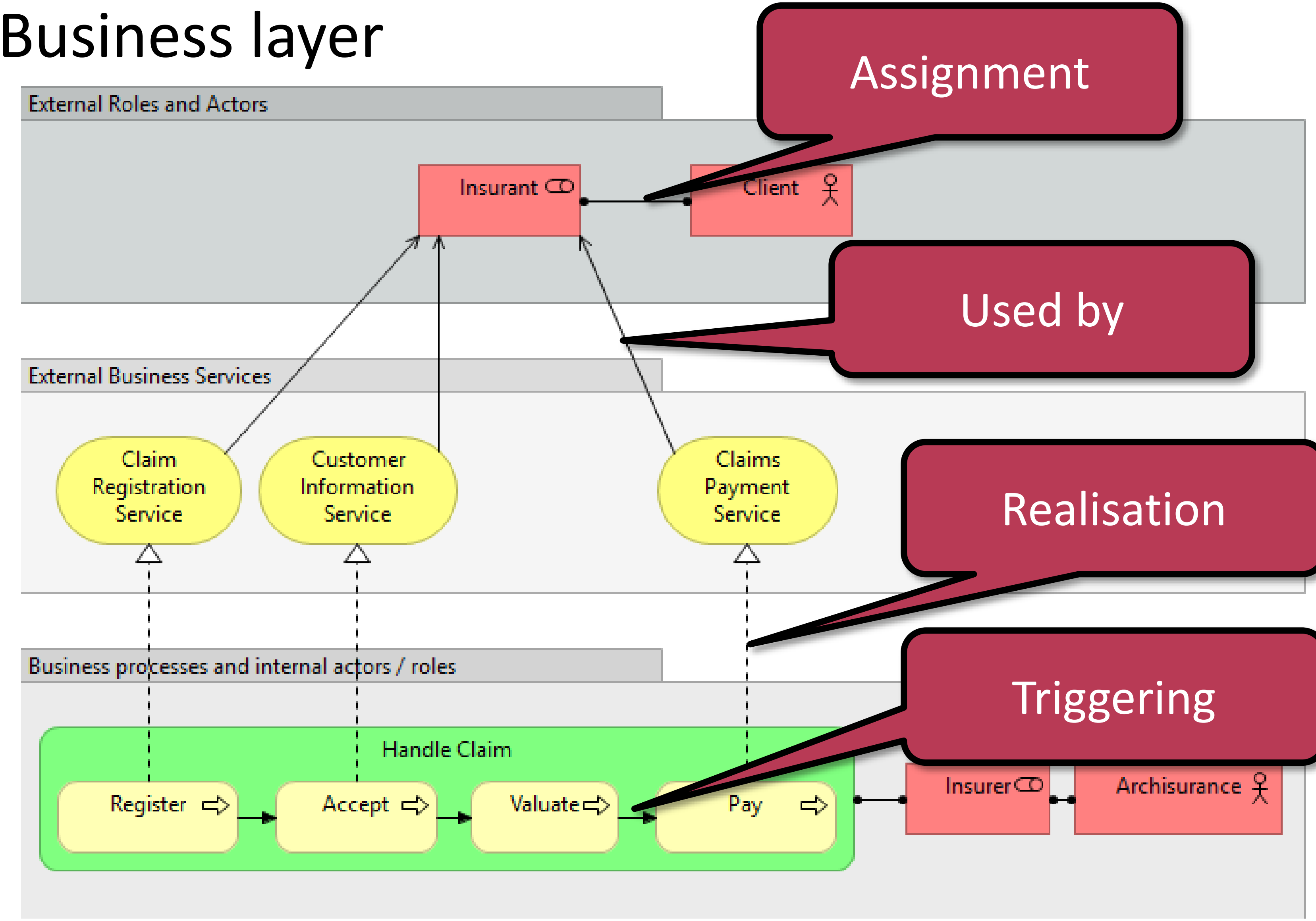
ArchiMate example – big picture

- An example of a fictional Insurance company.



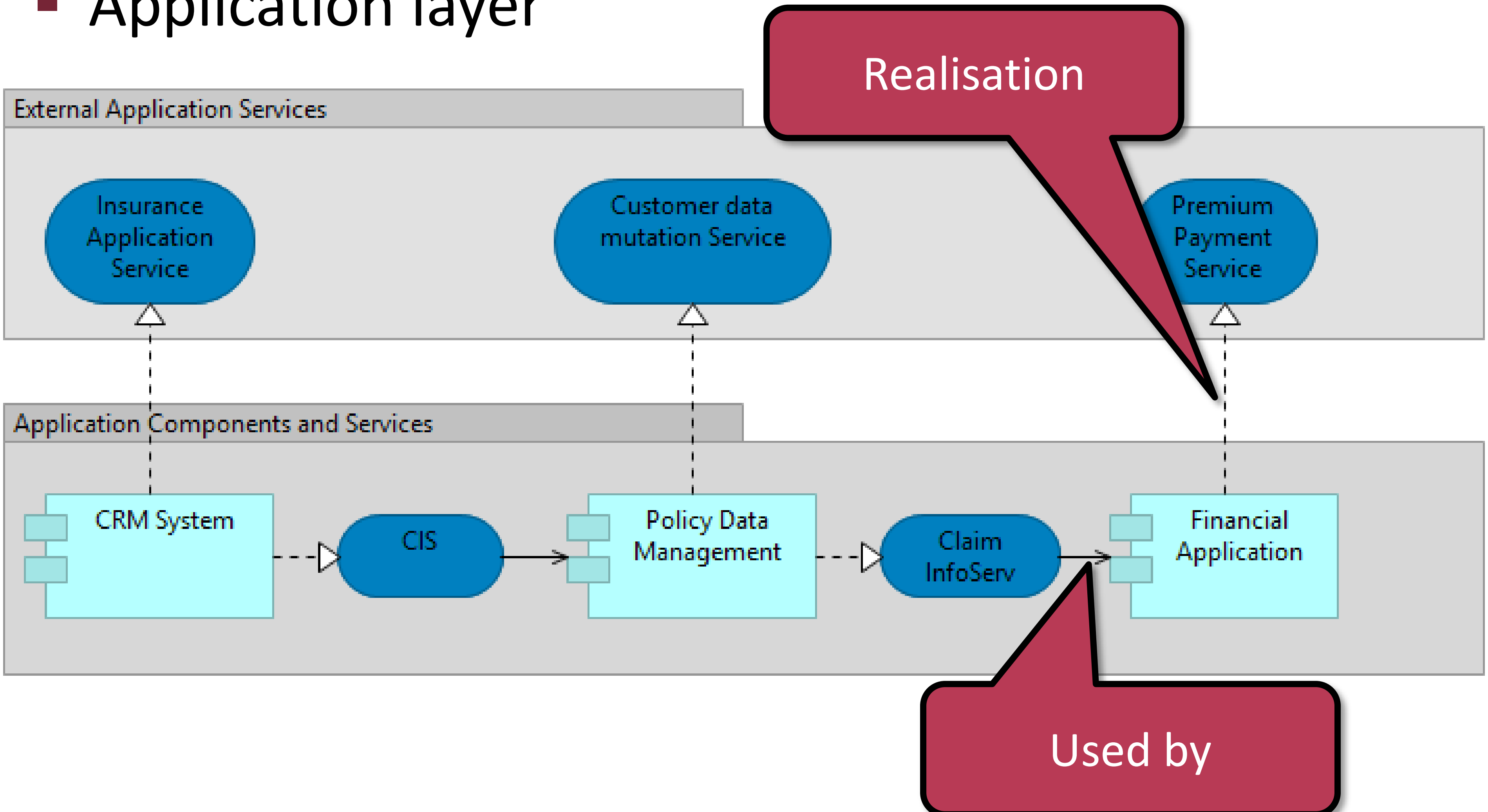
ArchiMate example: fictional Insurance company

- Business layer



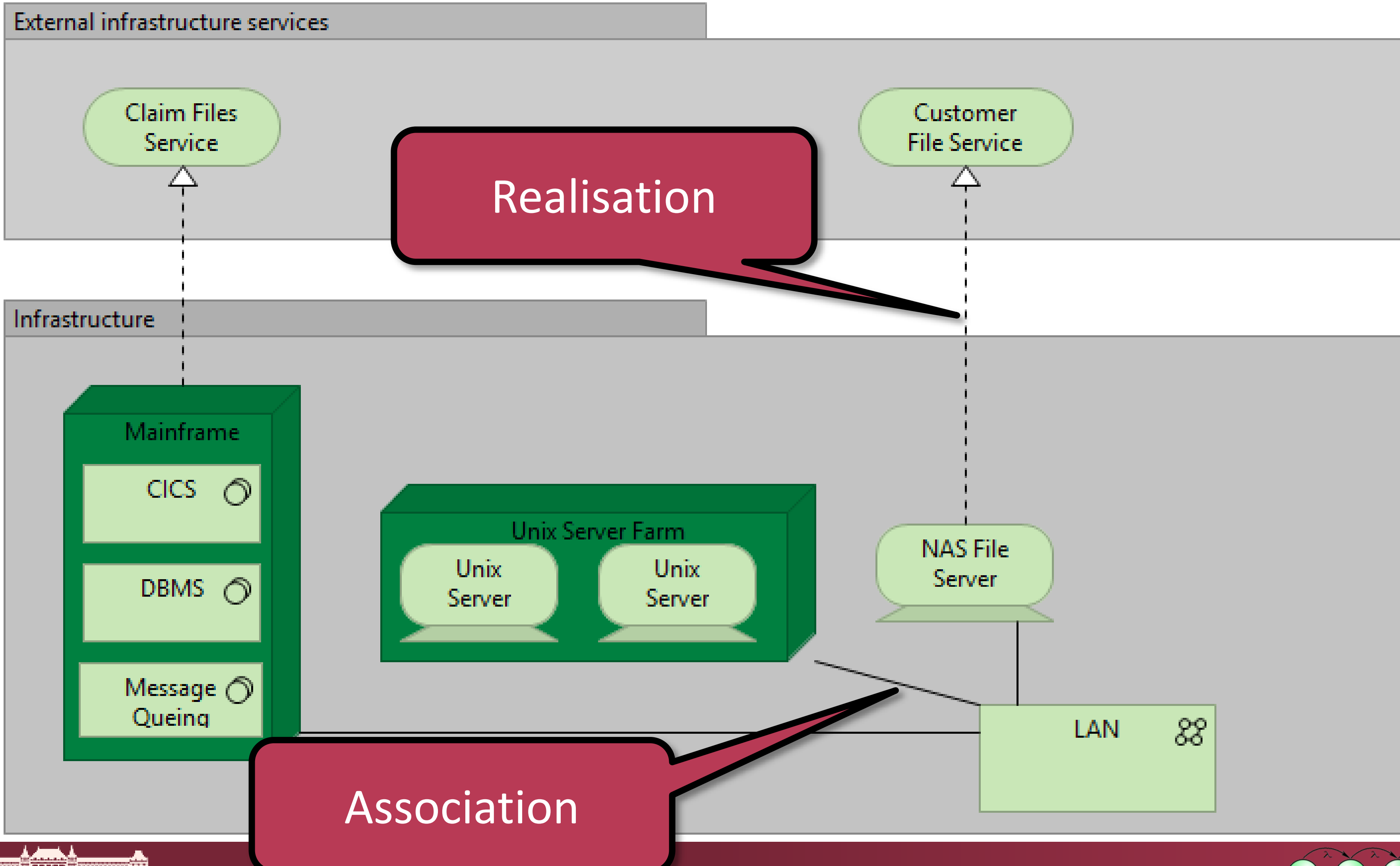
ArchiMate example: fictional Insurance company

- Application layer

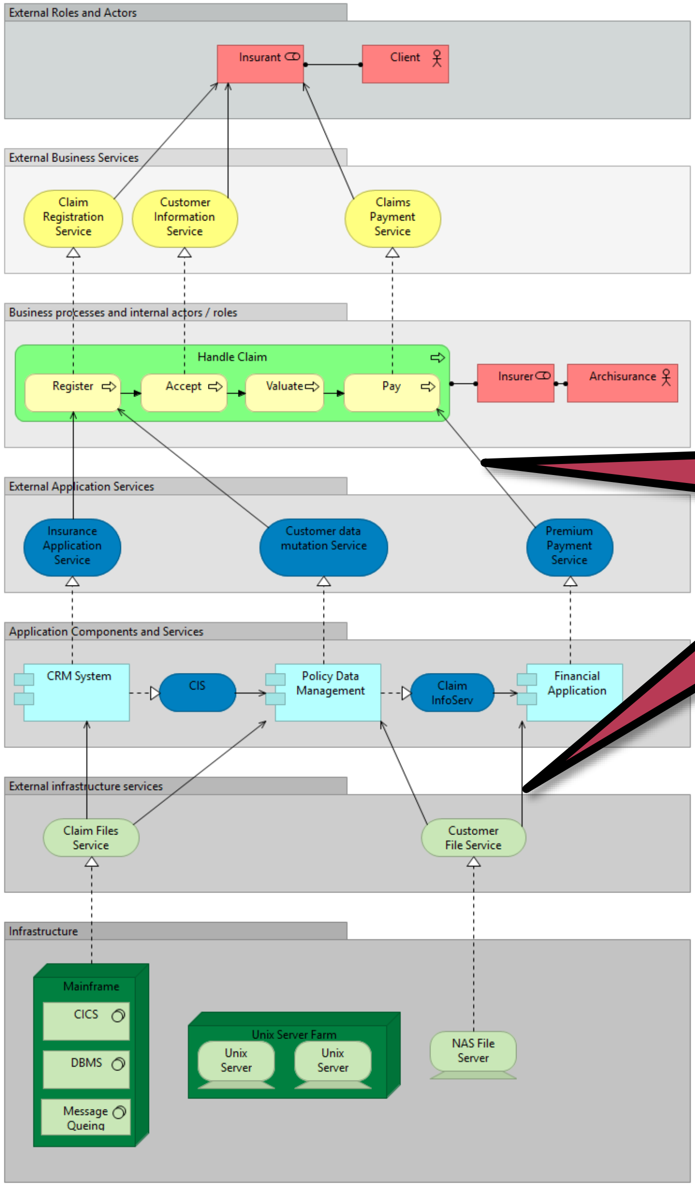


ArchiMate example: fictional Insurance company

- Technology layer



ArchiMate example – big picture



Used by – across layers

Case study

Analysis of extra-functional properties of a service

Validation of service configurations

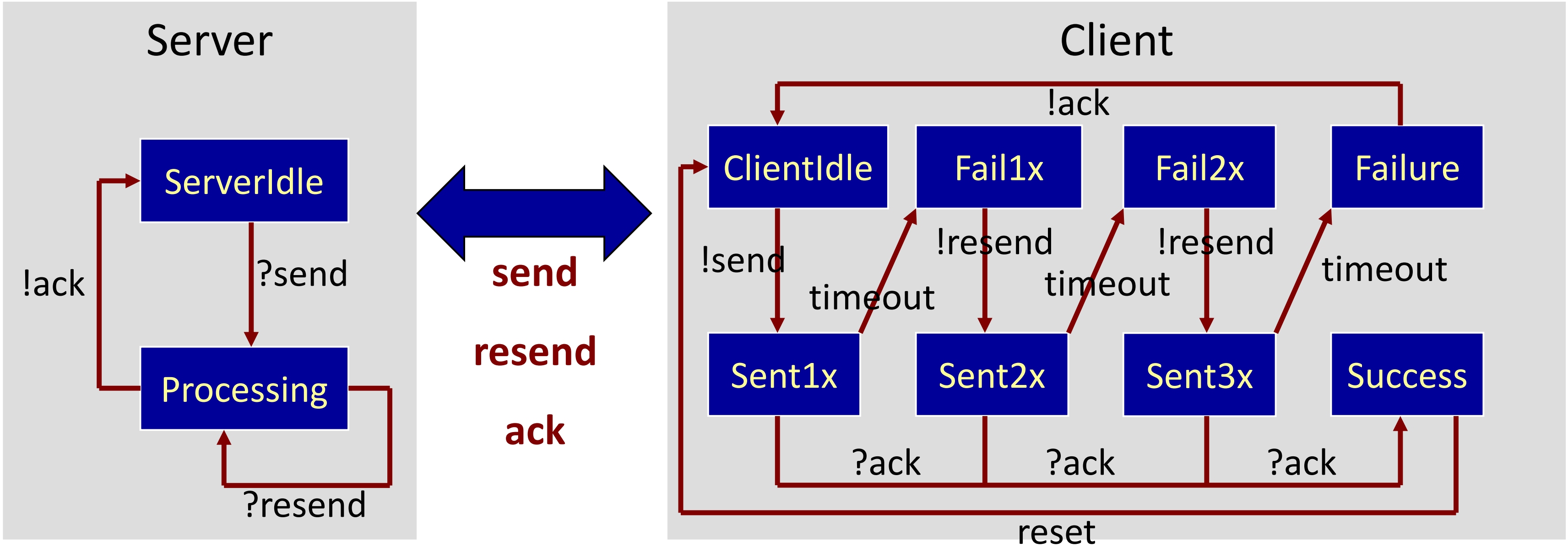
- Performability analysis
  - „Performability = Performance + Reliability”
- What happens in case of a failure?
  - E.g. the middleware responsible for reliable messaging resends the lost message → the guaranteed response time may increase (e.g. too low timeout → several false resends).
- What is the price of reliability? (performance-reliability *tradeoff*)
- How to set SLA parameters?

What do we model from all of this?

- Abstract behavior
  - Server
  - Client
- Message handling parameters (derived)
  - Method for handling messages
  - Number of resends
  - Parameters of **send**, **resend**, **ack**
    - (exponential distribution)

Middleware model

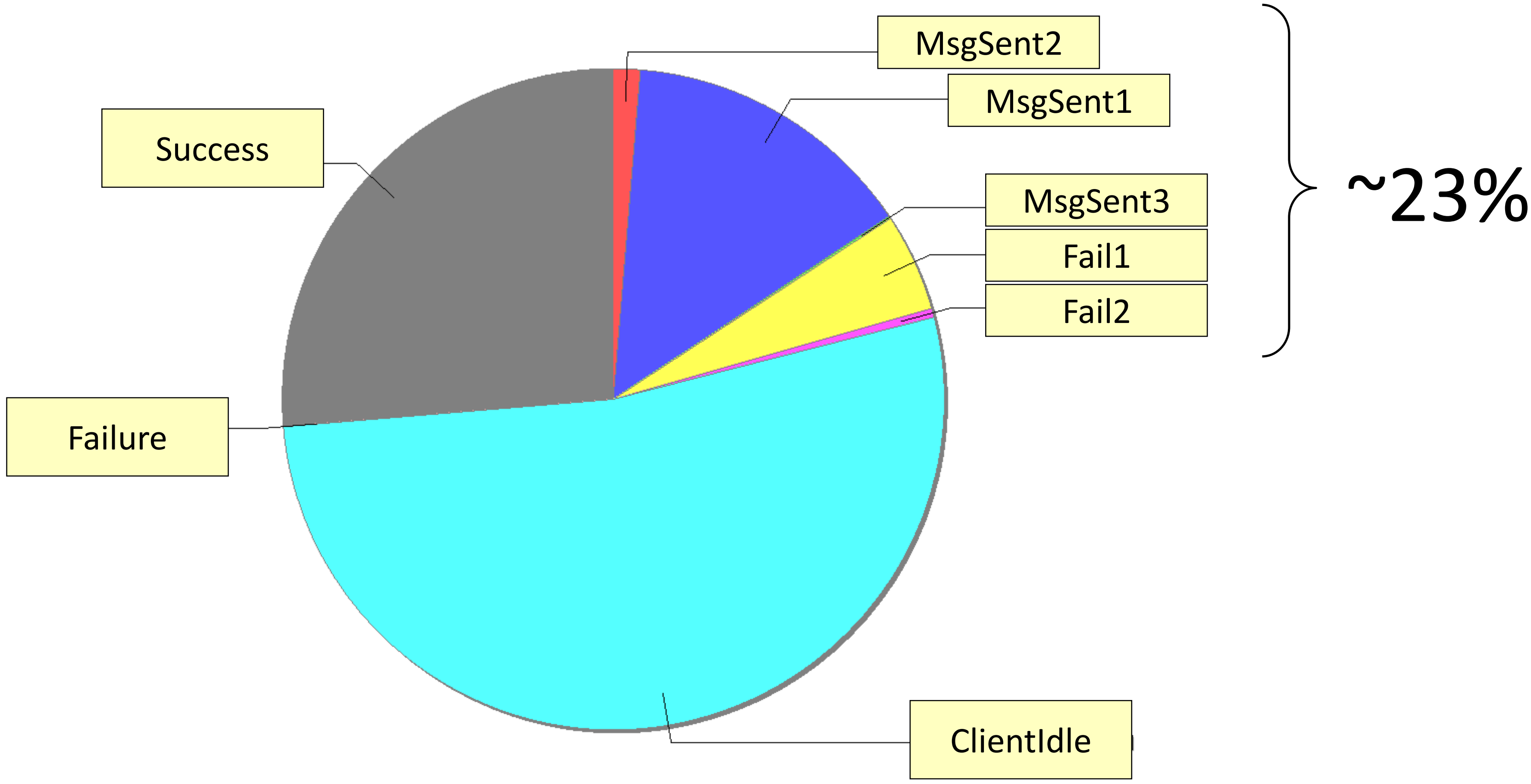
- Describes the platform
- Its parameters are included in the configuration model



Analysis results: utilization

Analysis in steady-state

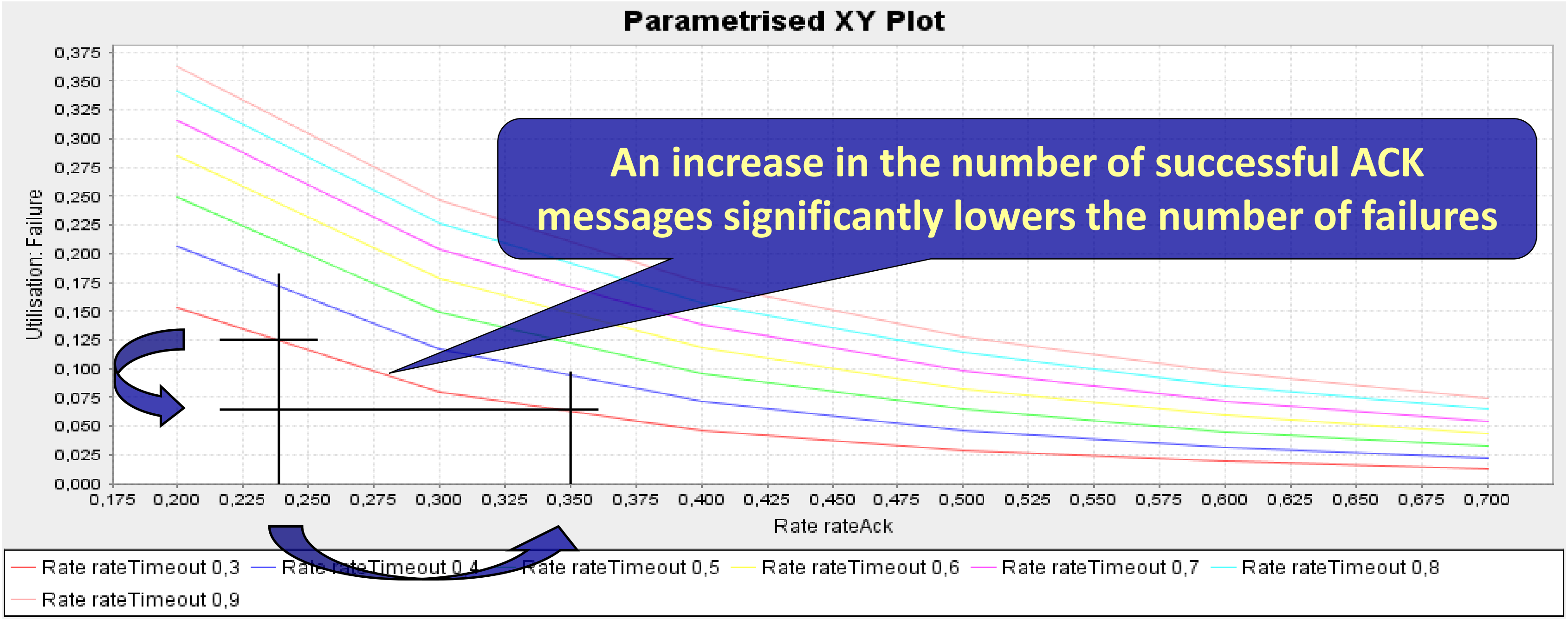
How much time does error handling take?



Sensitivity analysis results

Sensitivity analysis: what to change?

Probability of system level failures with respect to timing parameters of „resend”?



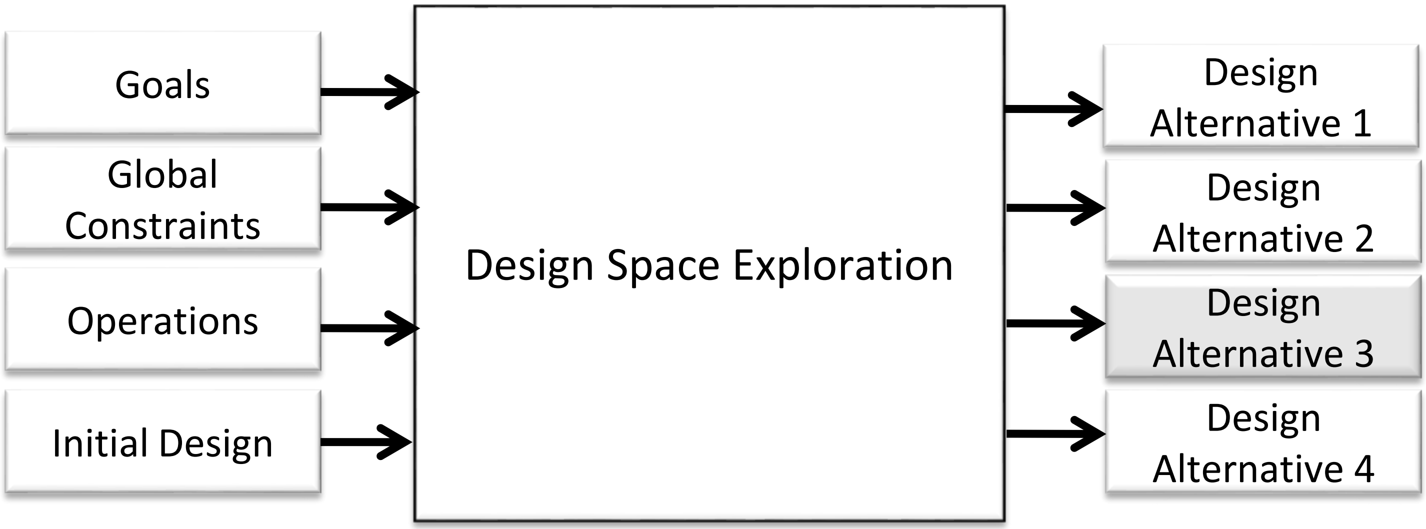
Case study

Application of DSE for allocation

Motivating example: Smart Building

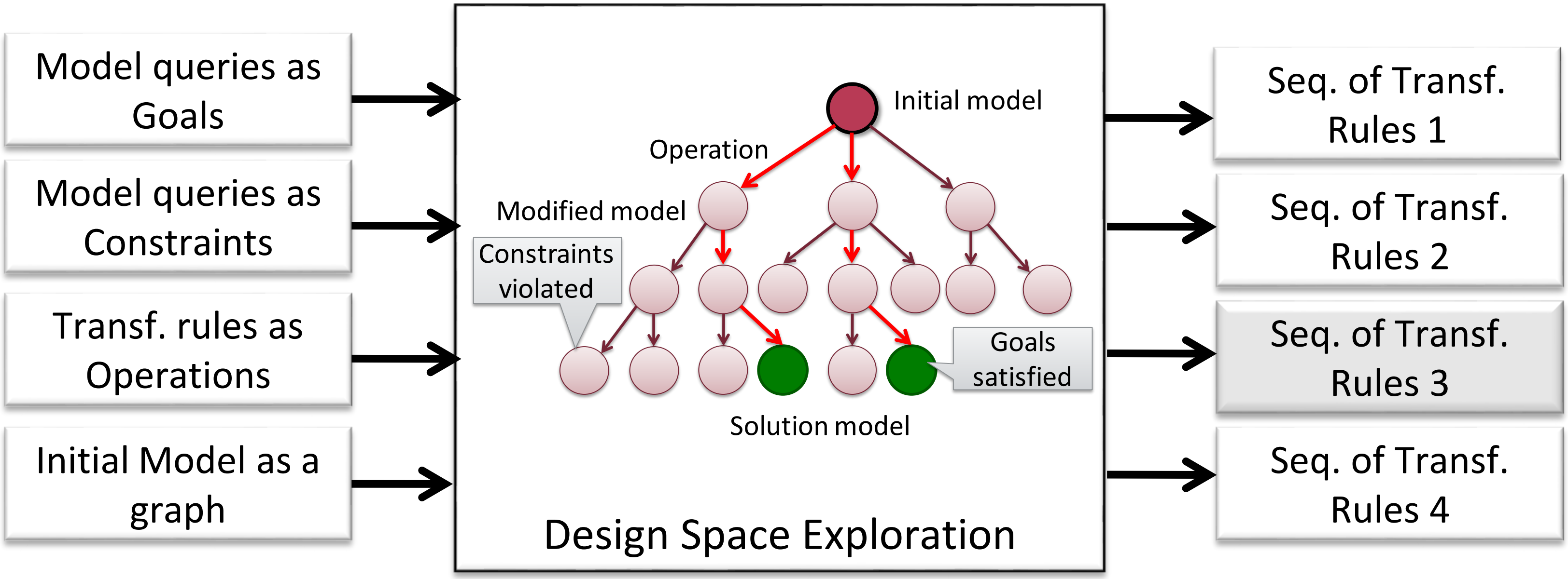
- Reconfiguration of supervising cyber-physical systems (CPS)
  - Offices to rent with highly configurable services
  - Services to deploy on both embedded and virtual computational units
  - Requests may change over time
  - Certain faulty devices may no longer function

Design Space Exploration (DSE)



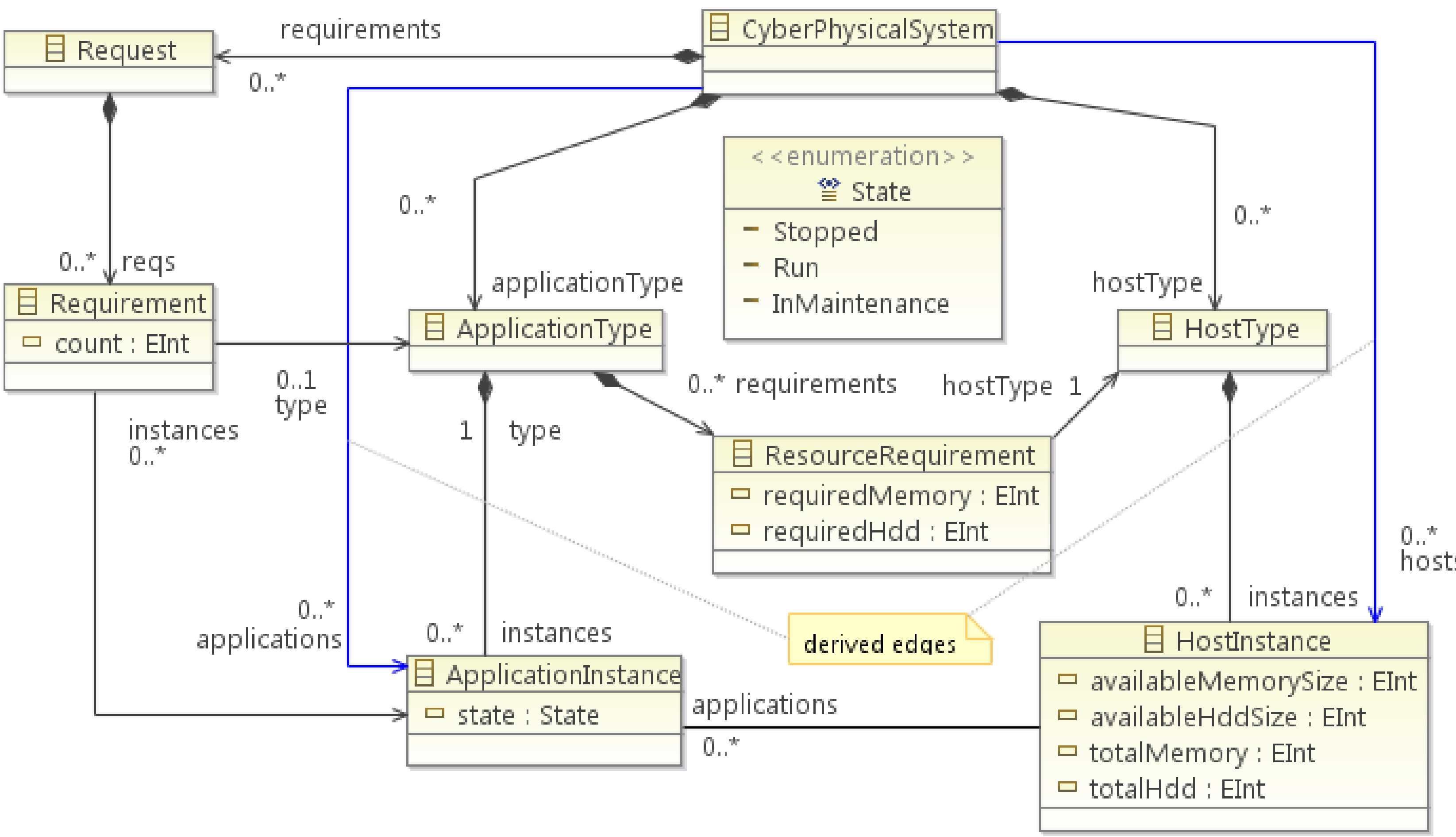
- Special state space exploration
  - Potentially infinite state space
  - cannot put upper bound on the number of model elements used in a design candidate (elements are created and deleted during exploration).

Rule-based Design Space Exploration

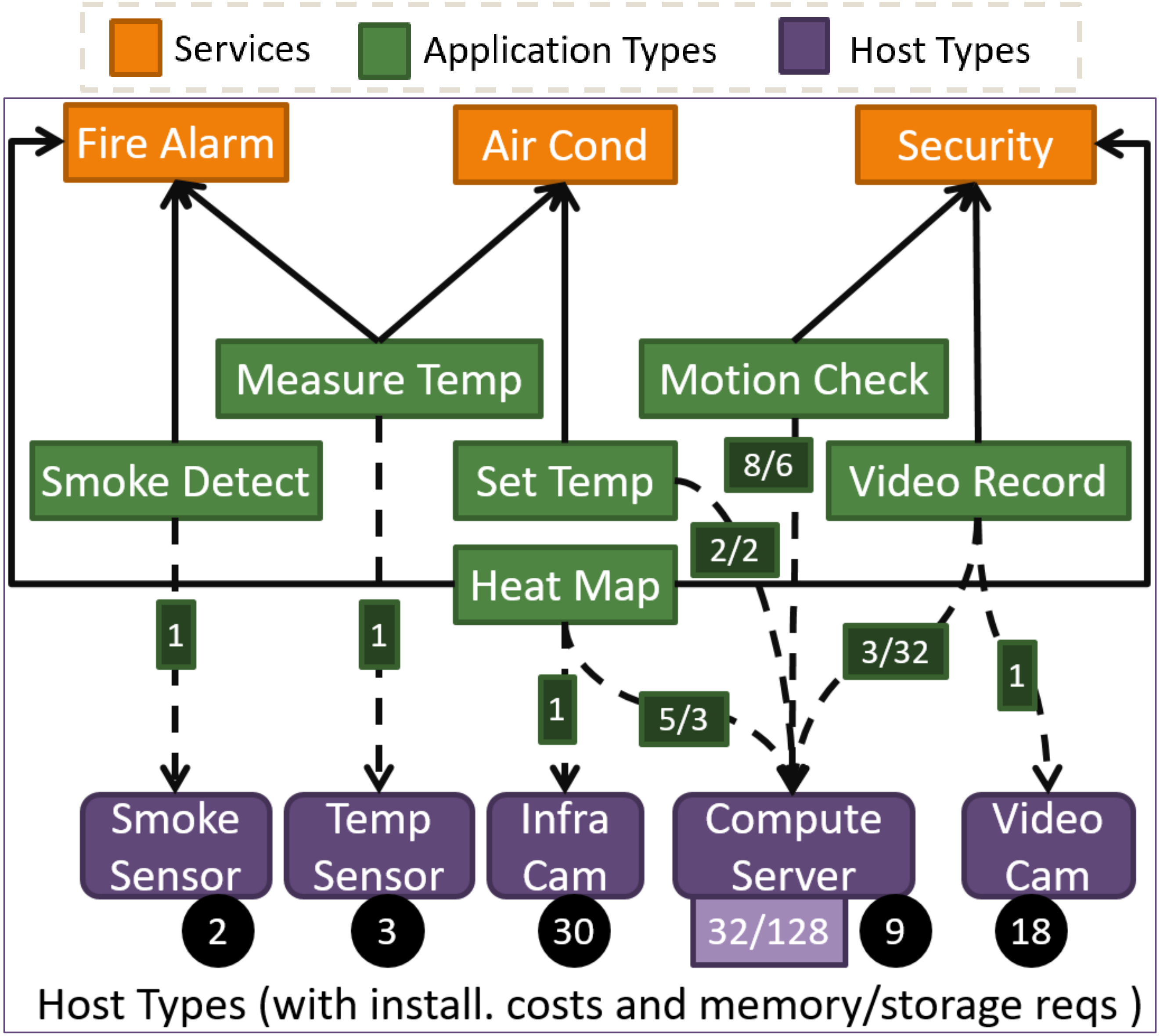


- Objectives : complex model metrics calculated by model queries
- Cost calculations may depend on the seq. of transf. rules
- Multiple objectives

Motivating example: Smart Building

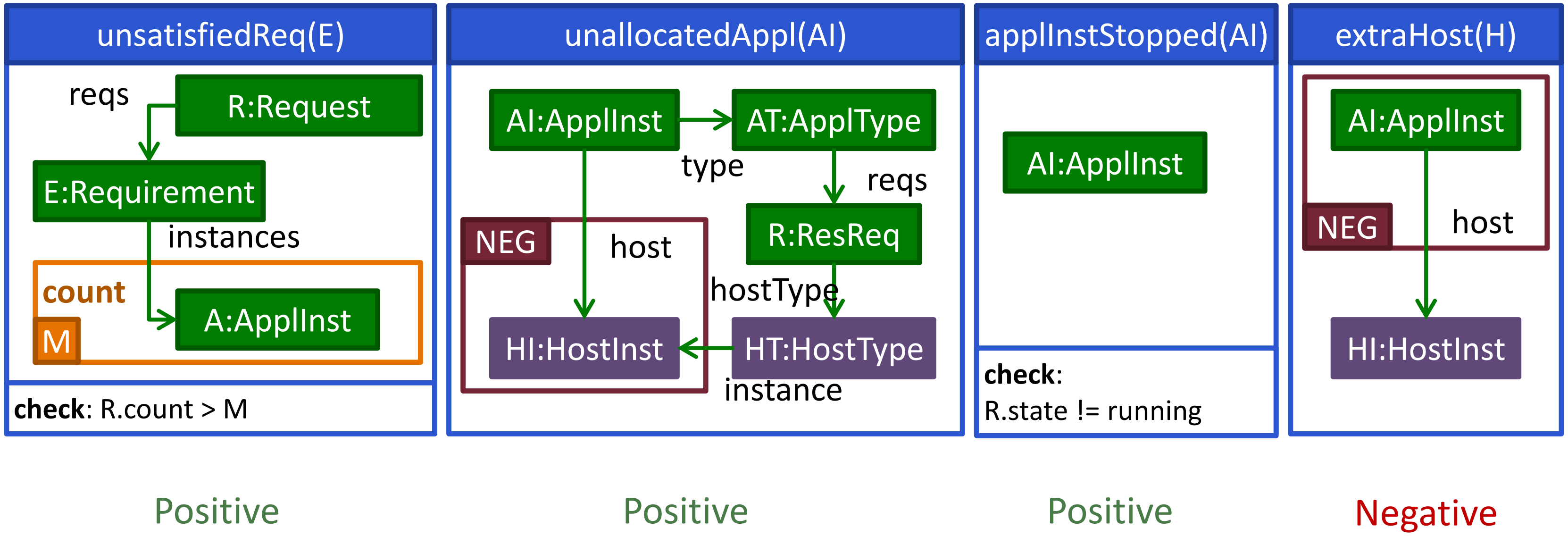


Motivating example: Smart Building



Smart building: constraints

- Constraints
  - Graph patterns to search for with model queries
  - For smart buildings
    - Constraints define valid or invalid configurations

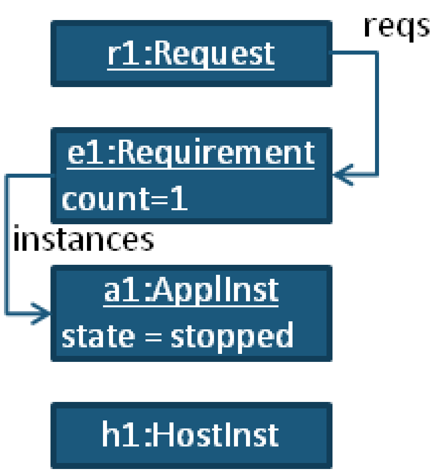


Smart building: constraints con't

- Constraints
  - Constraint fulfillment

$$ConstFulfillment(M) = \sum_{\forall p \in P} w_p \times matches(p, M)$$

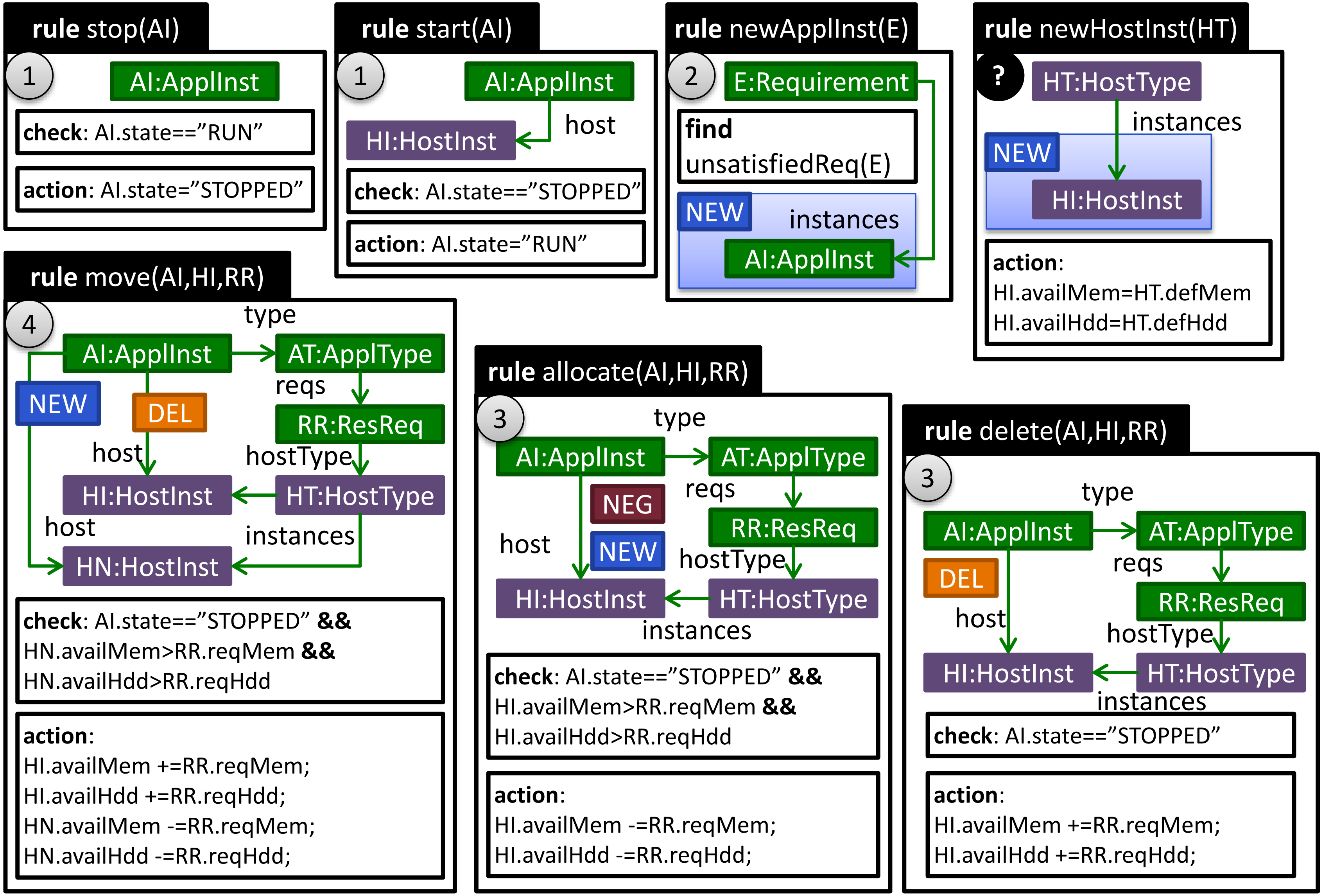
Positive for well-formedness constraints
Negative for ill-formedness constraints



Reqs	#	W	Score
satisfiedReq	1	2	2
allocatedAppl	0	3	0
applInstRun	0	4	0
extraHost	1	-1	-1
			1

$$ConstFulfillment(M) = 1 \times 2 + 0 \times 3 + 0 \times 4 + 1 \times -1 = 1$$

Smart building: rules



Smart Building: configuration model Services and Requests

Package	Services	Appl Types
Basic	Fire Alarm	Smoke Detect MeasureTemp
Comfort	+ Air Cond	+ SetTemp
Secure	+ Security	+MotionCheck +VideoRecord
Max		+HeatMap

(a) Services

R	Packages	ApplInst	HostInst
1	Comfort (2) Basic(1)	3xSD, 2xMT, 2xST	3xSS,6xTS, 2xCS,
2	Max (2)	2xSD, 6xMT, 2xST, 2xMC, 2xVR, 2xHM	2xSS,6xTS, 8xCS, 2xIC, 2xVC,

(b) Two examples on company requests

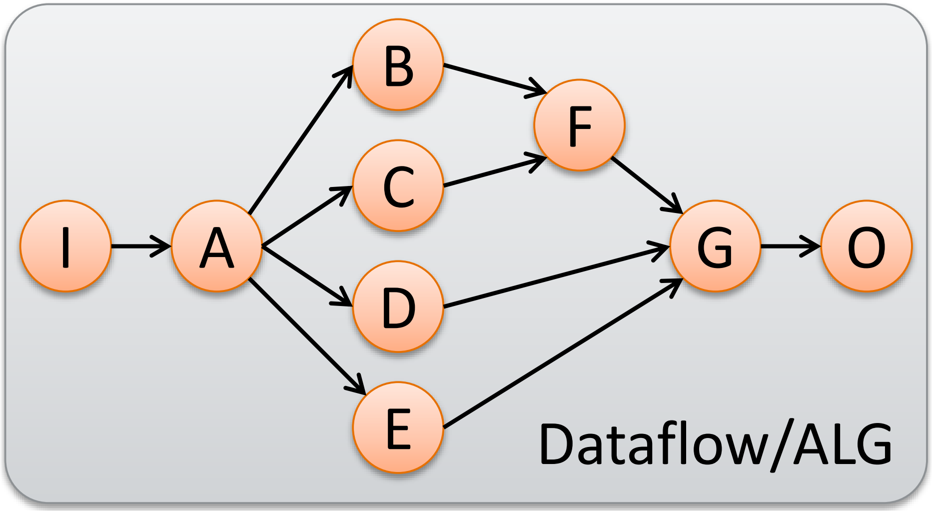
Case study

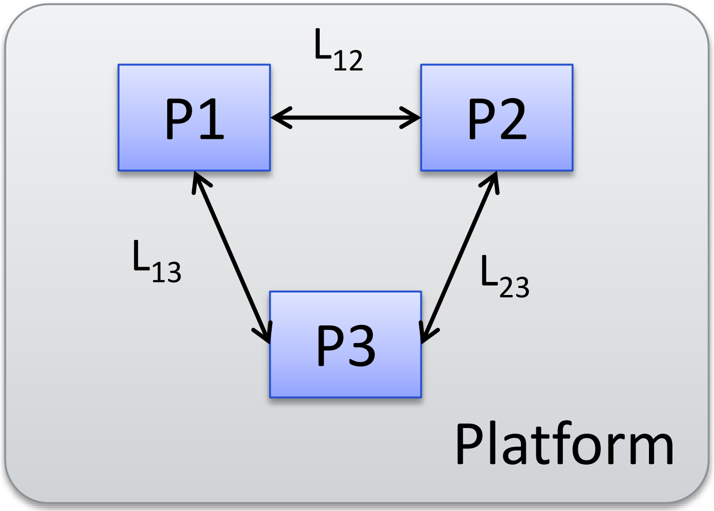
Schedule execution on a distributed platform

Scheduling

- **Platform model**: computation nodes and communication channels between them.
- **Algorithm model**: data-flow graph with operations as vertices and data-dependencies as edges.
- **Challenge**: schedule operations on the computation nodes for execution
  - Network communication takes time
  - Local results can be accessed instantly

Example [A. Girault]





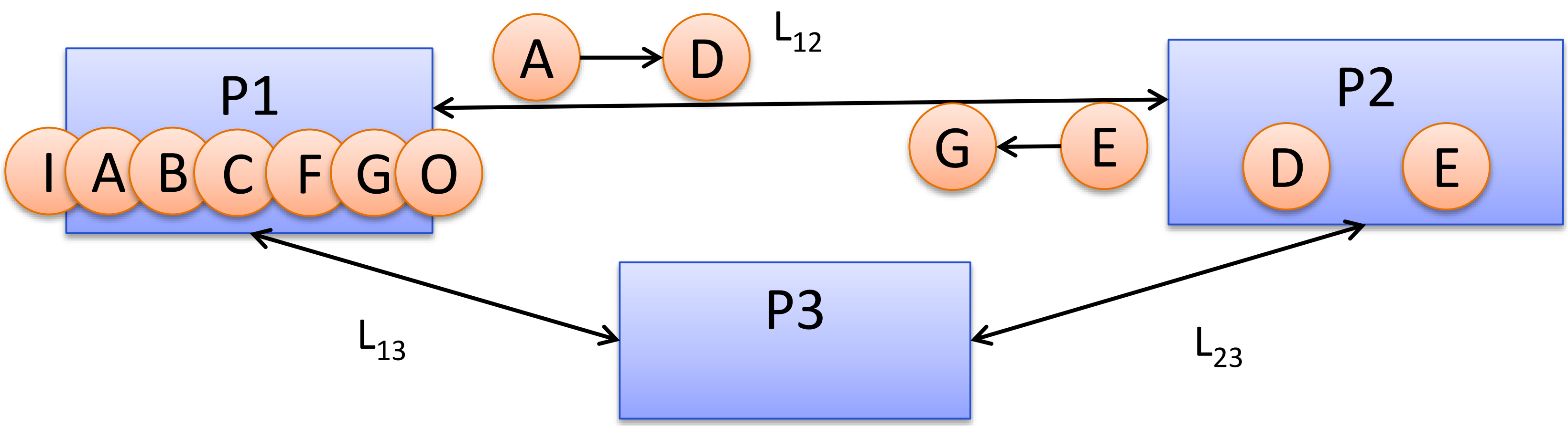
WCET	I	A	B	C	D	E	F	G	O
P1	10	20	30	20	30	10	20	14	14
P2	13	15	10	30	17	12	25	10	X
P3	X	10	15	10	30	20	10	15	18

Src/Trg	P1	P2	P3
P1	0	15	10
P2	15	0	20
P3	10	20	0

1) Create schedule (when and where to run what?)
2) Create fault-tolerant (FT) schedule if at most 1 proc may fail

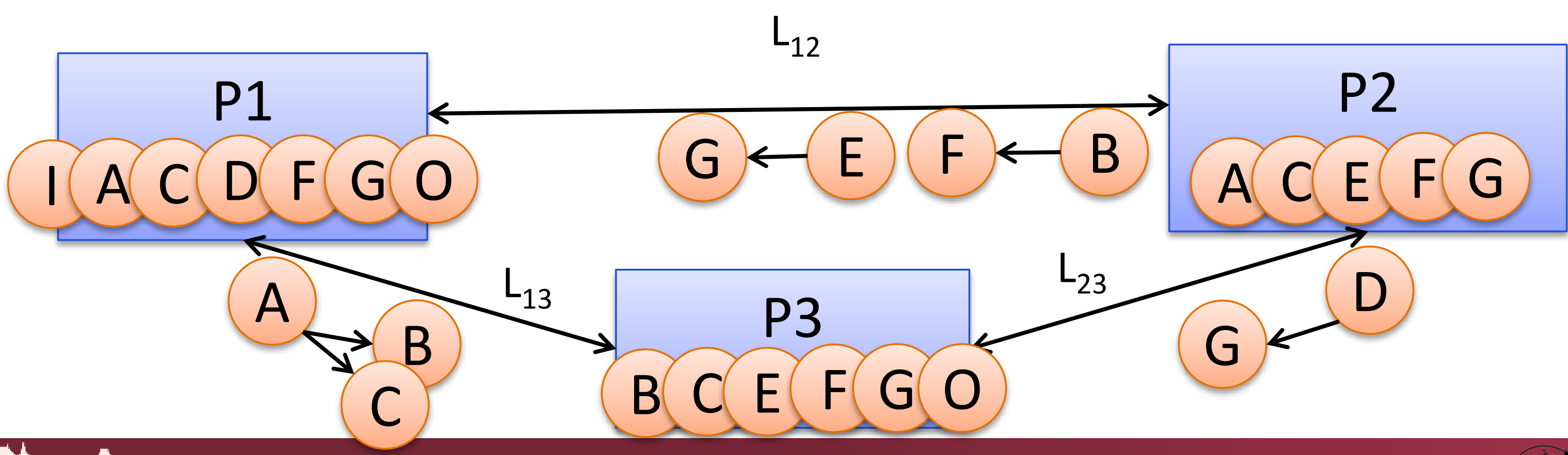
Naive solution (no FT)

	P1		L12		P2		L23		P3		L13	
	Start	End	Start	End	Start	End	Start	End	Start	End	Start	End
I	0	10										
A	10	30	30	45								
B	30	60										
C	60	80										
D					45	62						
E			74	89	62	74						
F	80	100										
G	100	114										
O	114	128										



FT Allocation and Schedule

	P1		L12		P2		L23		P3		L13	
	Start	End	Start	End	Start	End	Start	End	Start	End	Start	End
I	0	10			0	13						
A	10	30			13	28					30	40
B			38	53	28	38			40	55		
C	30	50							55	65		
D	50	80			38	55	55	75				
E			67	82	55	67			65	85		
F	80	100							85	95		
G	100	114							95	110		
O	114	128							110	128		



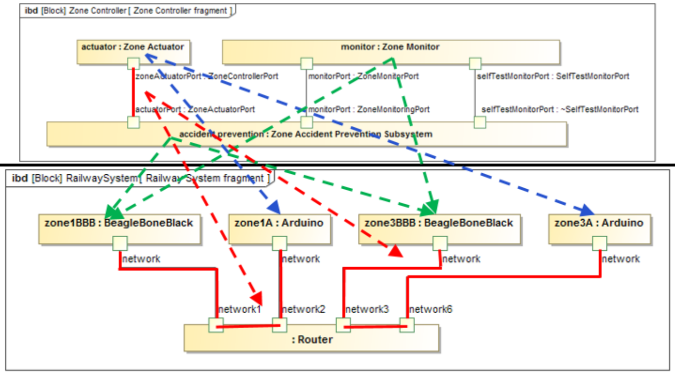
Summary

System properties

- **Functional requirements → Functional properties**: functions that the system is able to perform
  - including how the system behaves while operating – also called operational properties
- **Extra-functional requirements → Extra-functional properties**: they do not have a bearing on the functionality of the system, but describe attributes, constraints, performance considerations, design, quality of service, environmental considerations, failure and recovery.

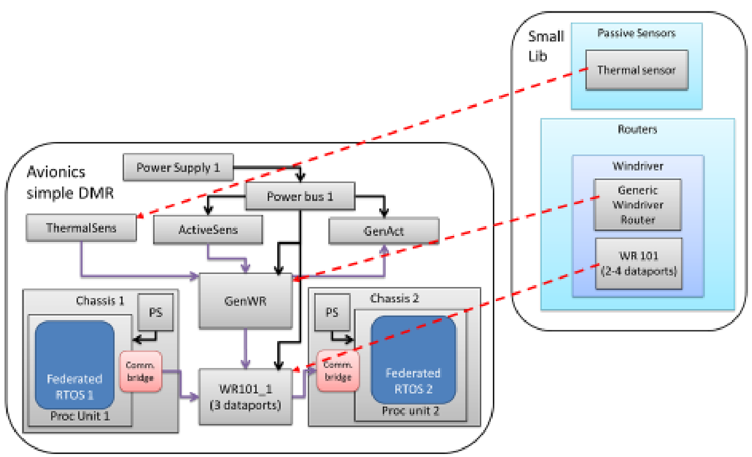
Allocation example: railway system

- Functional structure



- Platform structure

Defining the platform model II.



Analysis results: utilization

Analysis in steady-state

How much time does error handling take?