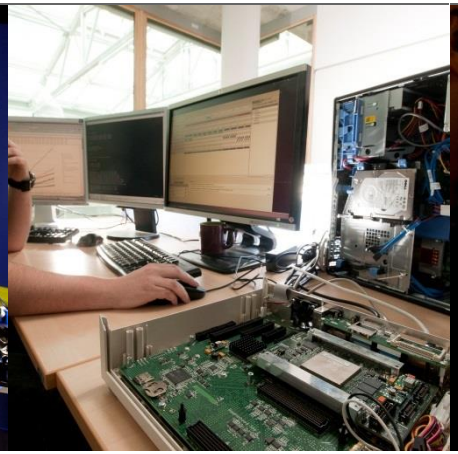
FROM RESEARCH TO INDUSTRY

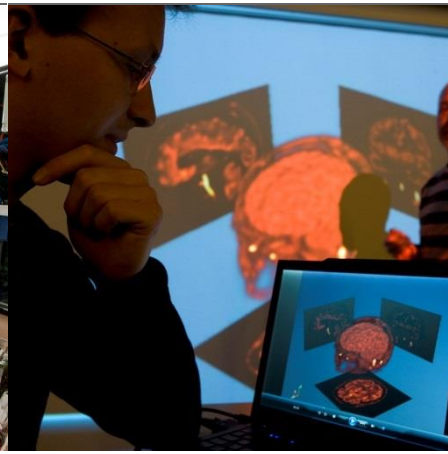
cea tech

list

REAL-TIME SOFTWARE ENGINEERING WITH MARTE

Shuai Li
Project manager / Research engineer
CEA LIST
shuai.li@cea.fr





INSTITUT CARNOT CEA LIST

digiteo

ACKNOWLEDGEMENT

Bran Selic, Malina Software, Canada

Sébastien Gérard, CEA LIST, France

CONTENT

CONTENT


Introduction


Real-Time Embedded System


Modeling with MARTE


MARTE in Practice: Schedulability Analysis


MARTE Support

Conclusion

Complex system modeling

- **UML can be a core modeling language**
  - Adapted for object-oriented software architectures
  - UML 2.x customizable for specific domains
- **SysML for system engineering concerns**
  - Requirements, mathematical expressions, continuous behavior, etc…
  - Good industrial acceptance level
- **What about concerns of the real-time embedded system domain?**

Real-Time Embedded System (RTES) concerns

- **Embedded: autonomous system with limited resources ➔ performance properties/constraints like energy consumption and resource contention**
- **Real-time: data processing correctness also depends on time ➔ time properties/constraints like execution time, deadline**

INTRODUCTION

We need a language for:
Modeling and **A**nalysis of **R**eal-**T**ime **E**mbedded systems

MARTE

What is MARTE?

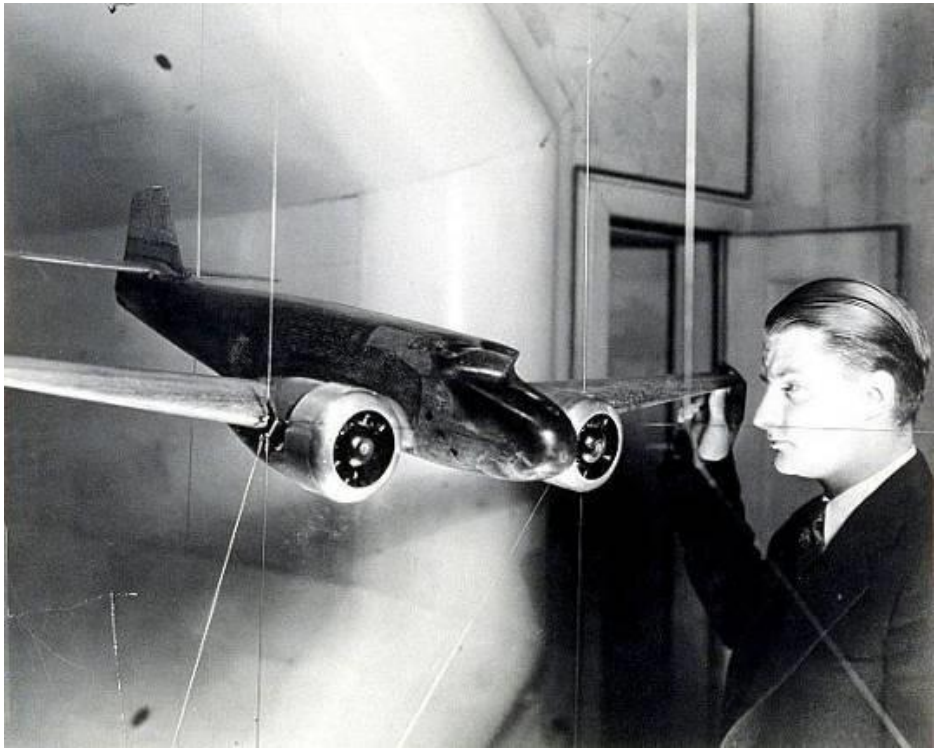
- **A modeling language for the RTES domain, based on several industrial systems and standards**
- **Support for specification of Quality of Service (QoS) characteristics (e.g., delays, memory capacities, CPU speeds, energy consumption)**
- **Support for annotations necessary for quantitative analyses and computer-aided design**
- **Implemented as a UML profile**
- **Standardized by the Object Management Group (who also standardizes UML among other things…)**

Why MARTE?

- **Remember why UML was proposed:**
  - Too many languages and tools for software design ➔ interoperability not guaranteed, engineering expertise difficult to manage
  - UML unifies all these languages around a standard language that shares many software concerns
- **Same situation for RTES:**
  - Too many languages and tools for RTES design ➔ interoperability not guaranteed, engineering expertise difficult to manage
  - Same problems as software design so same pattern to solve the problem!
  - An industry standard language potentially reduces cost and risks
    - Communication between stakeholders
    - Tools interoperability
    - Support and codify industry-wide best practices
    - Availability of trained experts and teaching material
    - Vendor independence

How is MARTE applied?

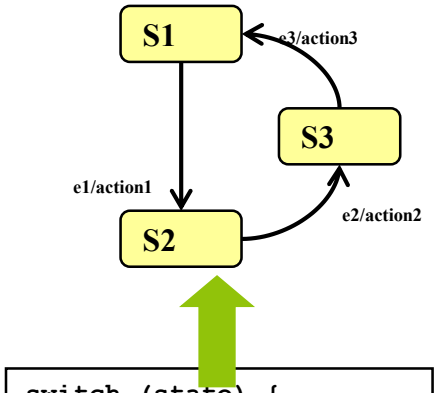
- **Model-driven engineering:**
  - An approach to system and software development in which computer-based models play a central role
  - Based on a time-proven idea as old as engineering:



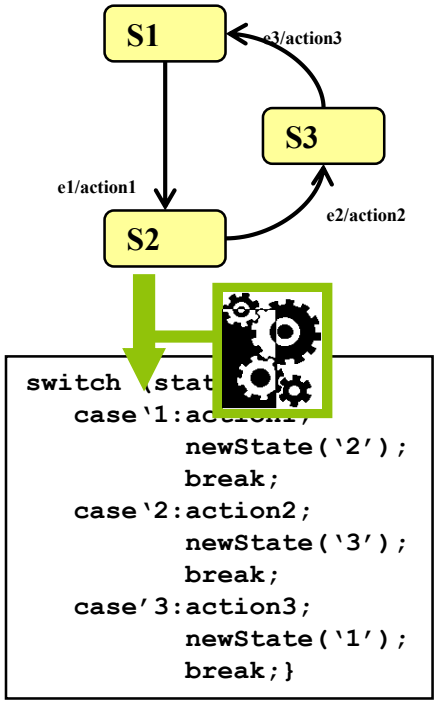
How is MARTE applied?

- **Model-driven engineering:**
  - Based on human nature:

Abstraction



Automation



CONTENT


Introduction


Real-Time Embedded System


Modeling with MARTE

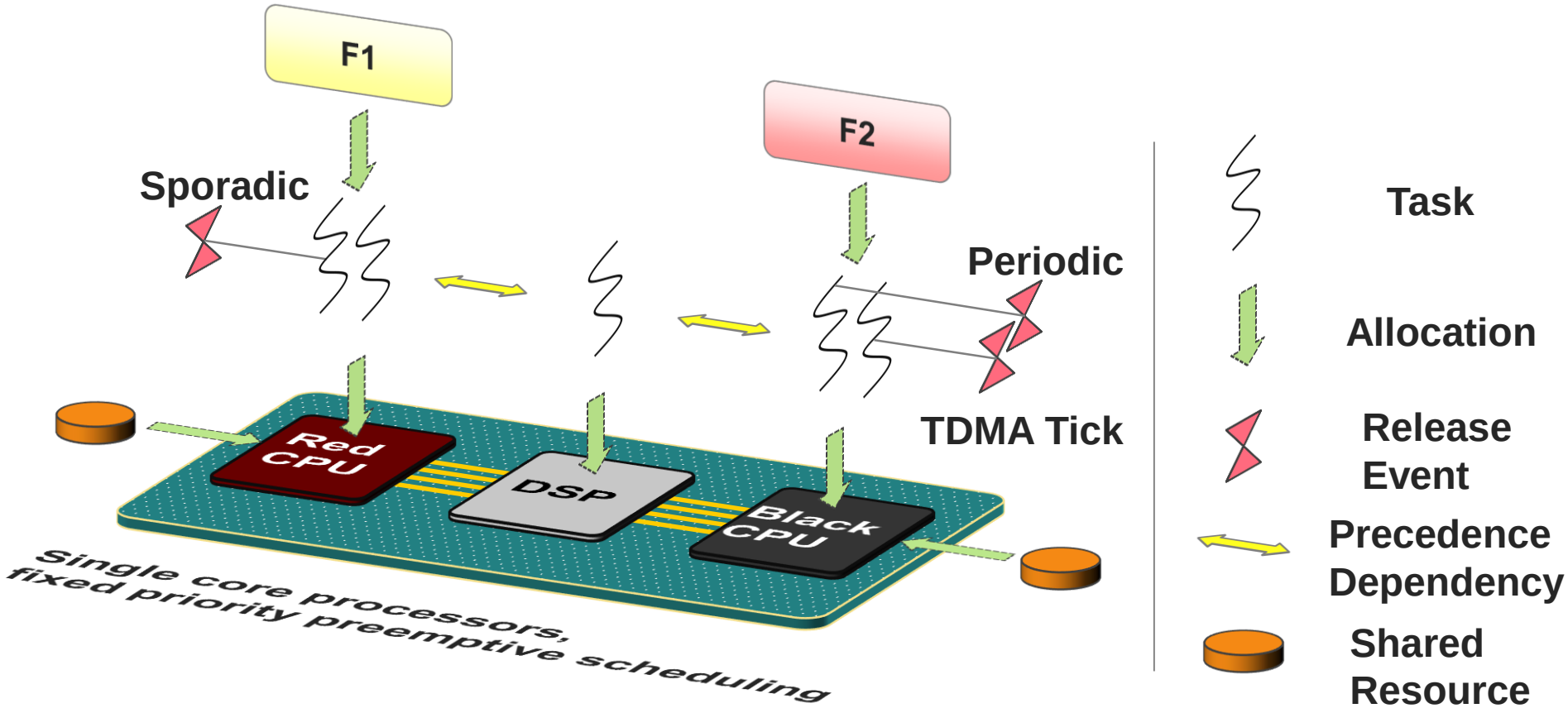

MARTE in Practice: Schedulability Analysis


MARTE Support

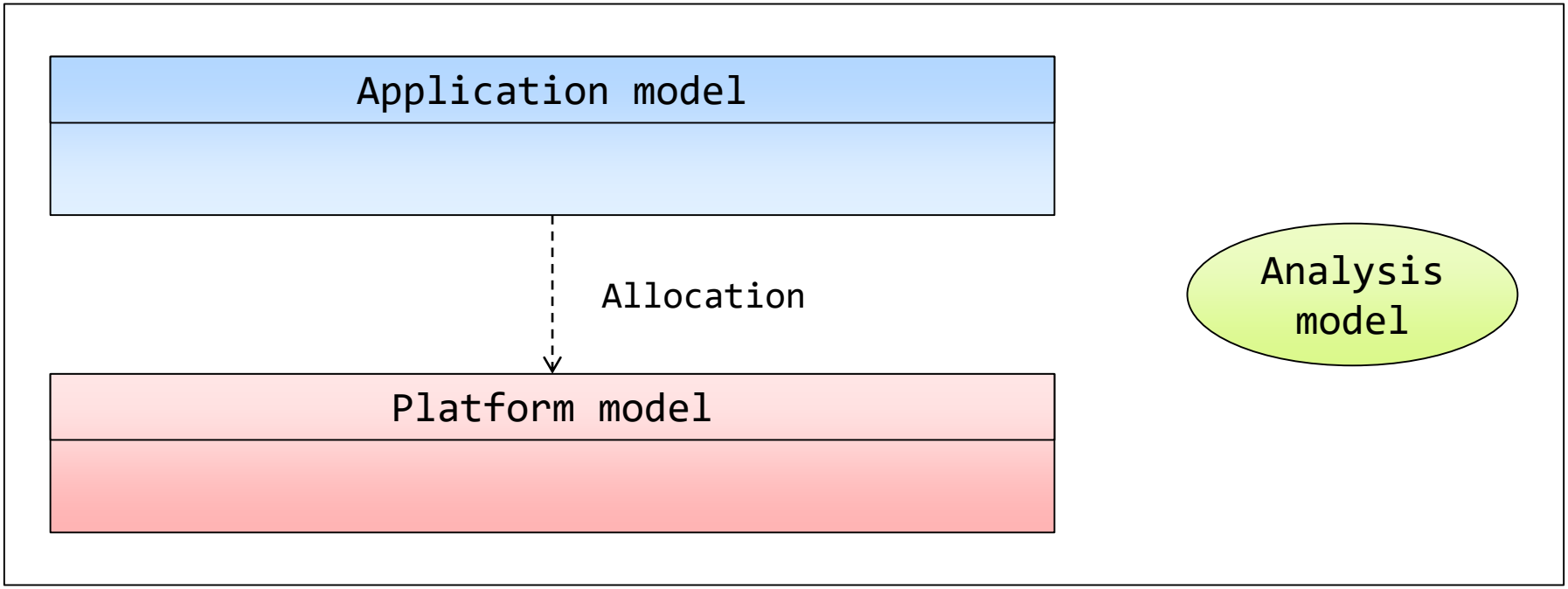
Conclusion

Example of an RTES

- A RTES can be characterized by its architecture



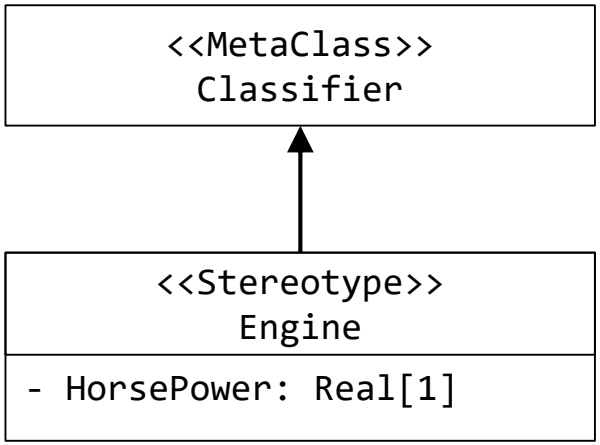
Modeling approach



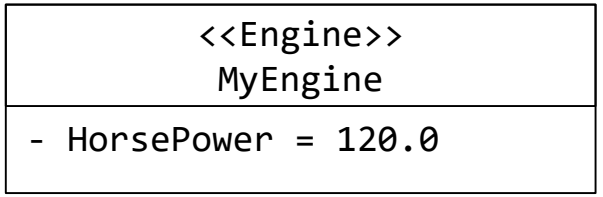
CONTENT

UML profile concept

- Profile = generic extension mechanism to customize UML for a specific domain
- Profiles are defined using stereotypes, tags (i.e. stereotype attributes in UML 2), and constraints
- A stereotype is applied to a UML meta-element; its attributes are then given values called tagged values
- MARTE is a UML profile for the RTES domain

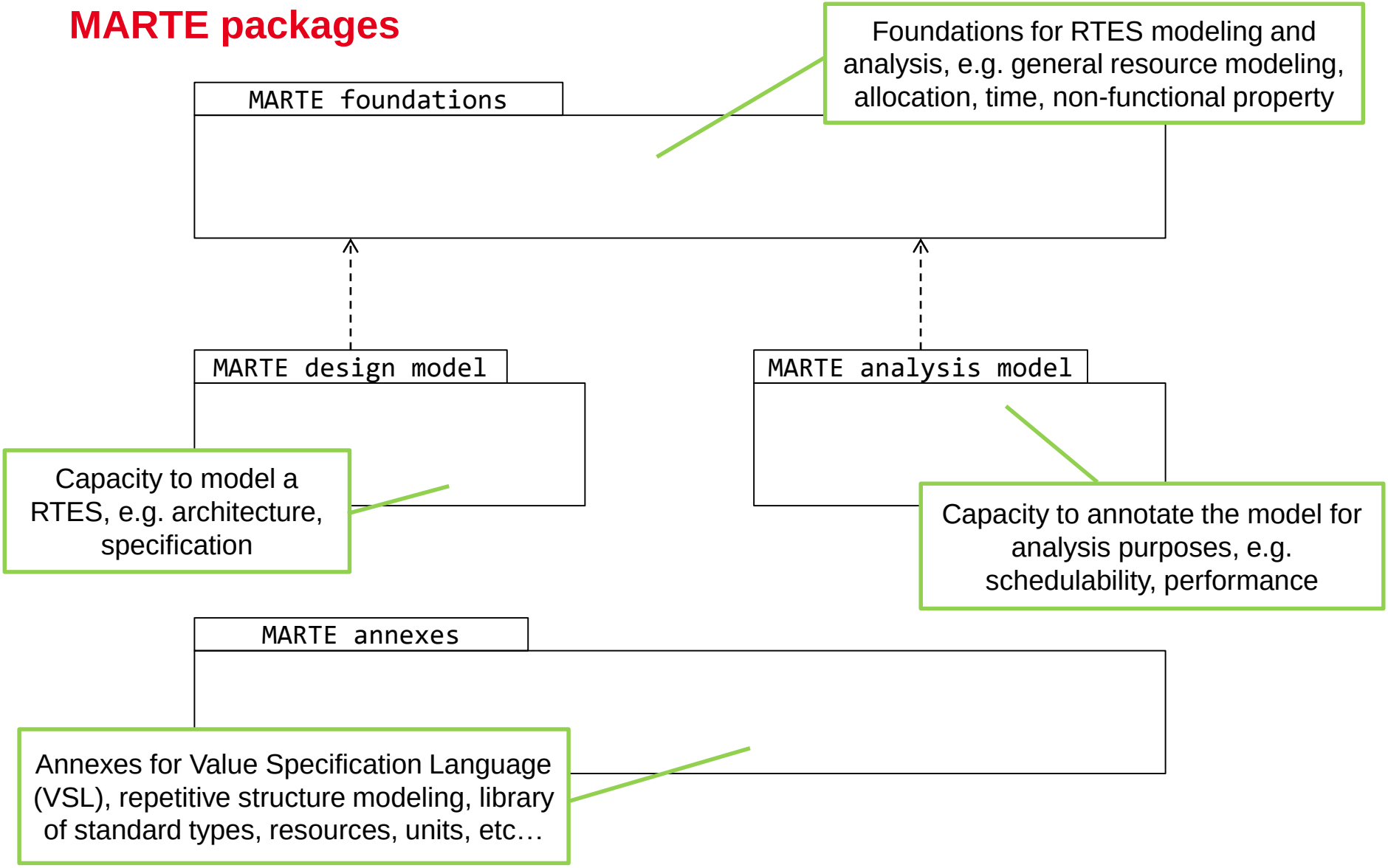


Profile definition



Profile usage

MARTE packages



CONTENT


Introduction


Real-Time Embedded System

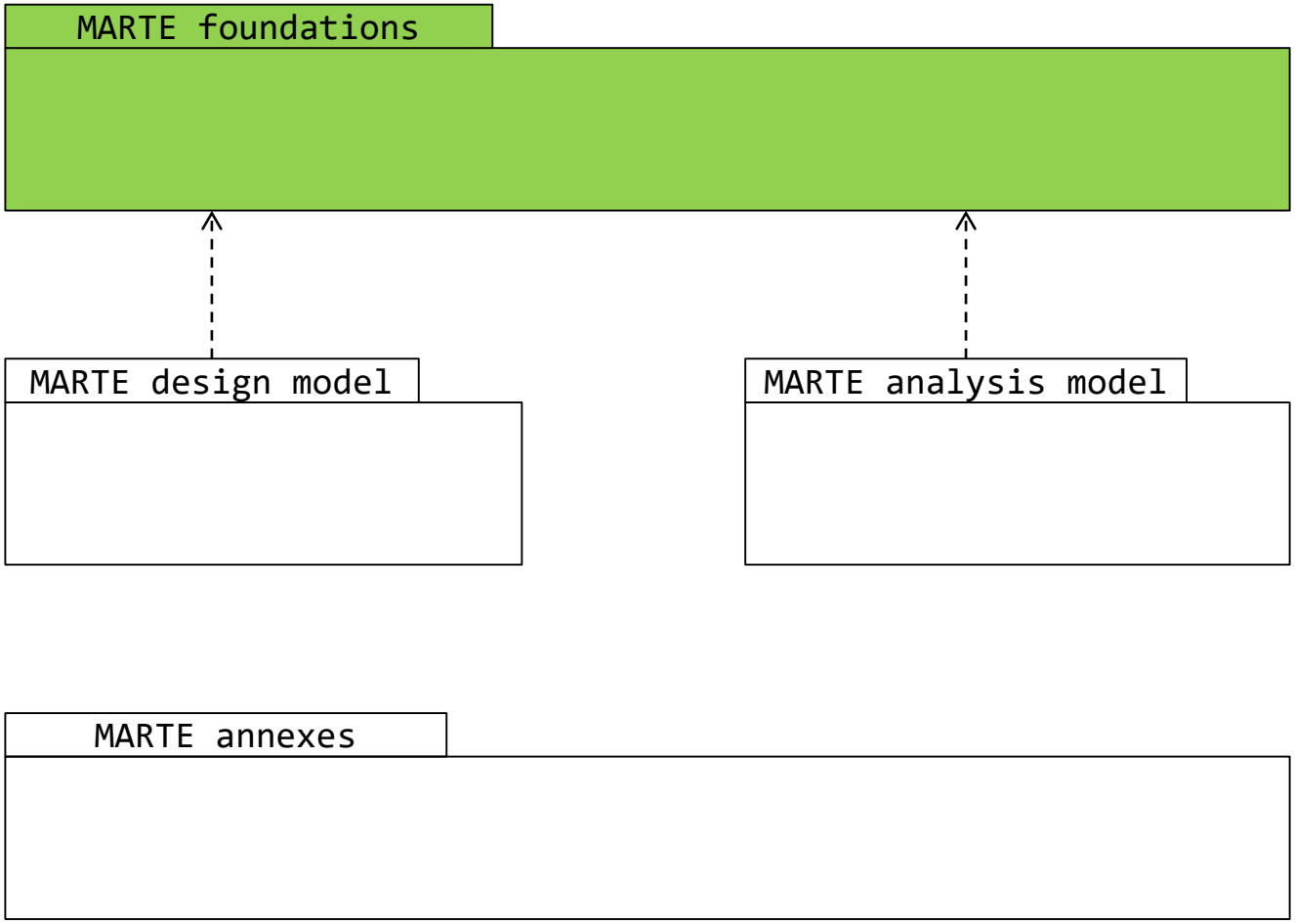

Modeling with MARTE: Foundations


MARTE in Practice: Schedulability Analysis

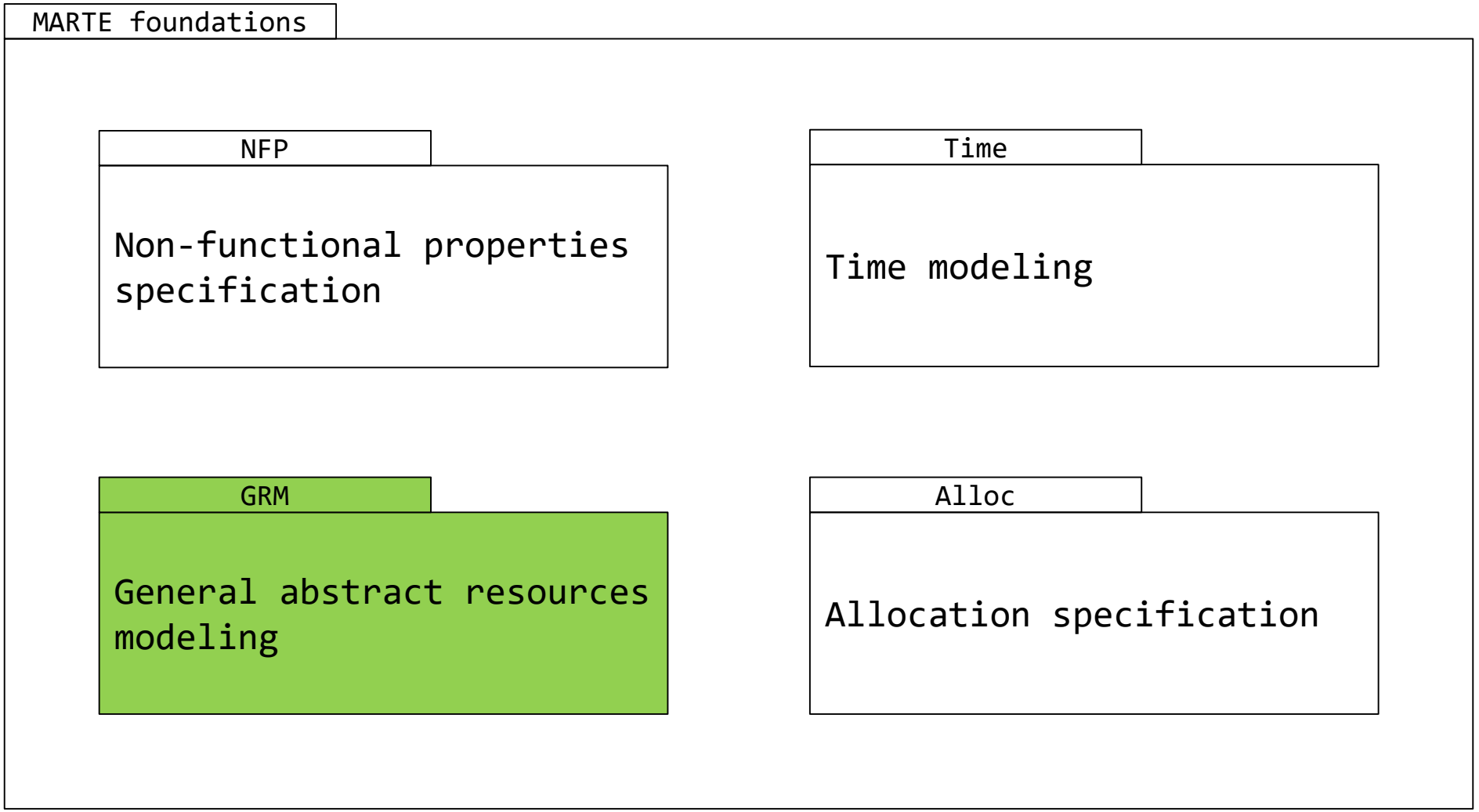

MARTE Support

Conclusion

MARTE packages

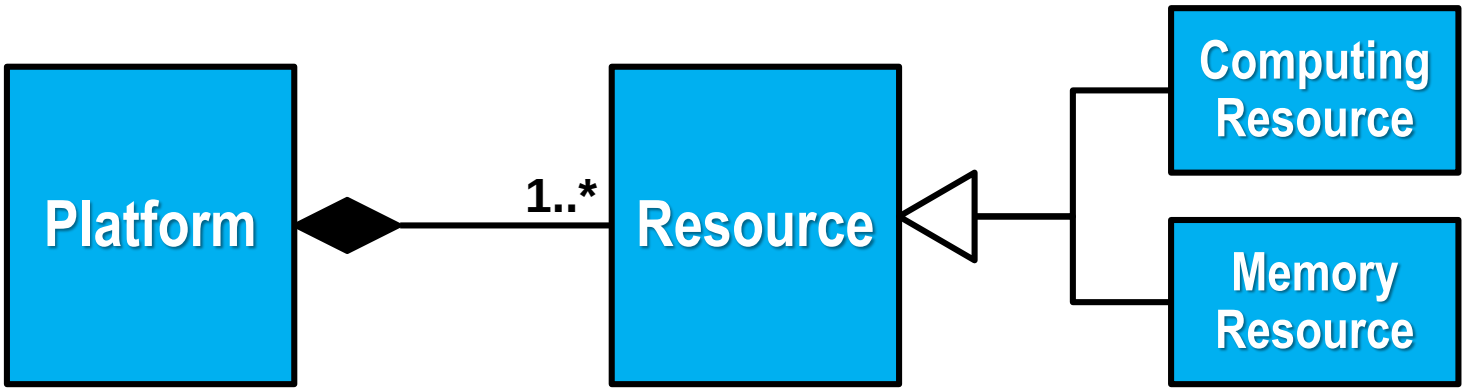


MARTE packages



CORE CONCEPT: RESOURCE

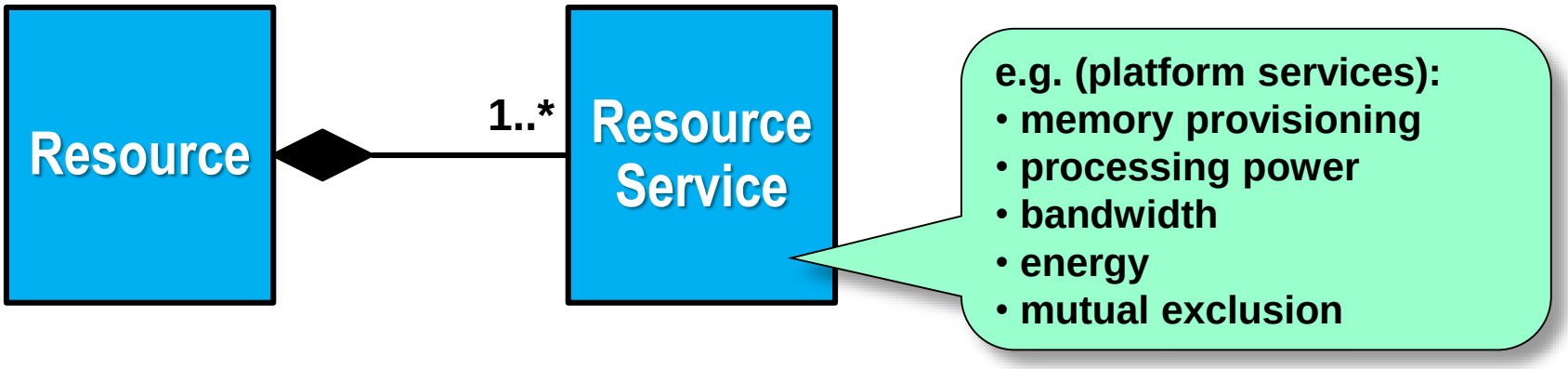
- ***Resource*:**
  - A source of supply of money, materials, staff and other assets that can be drawn upon…in order to function effectively [Oxford dictionary]
- **In MARTE, a platform is viewed as a *collection of different types of resources,* which can be drawn upon by applications**
  - The *finite nature of resources* reflects the physical nature of the underlying hardware platform(s)



etc.

CORE CONCEPT: RESOURCE SERVICES

- **In MARTE resources are viewed as *service providers***
  - Consequently, applications are viewed as *service clients*



- **Resource services are characteriszed by their**
  - Functionality
  - Quality of Service (QoS)

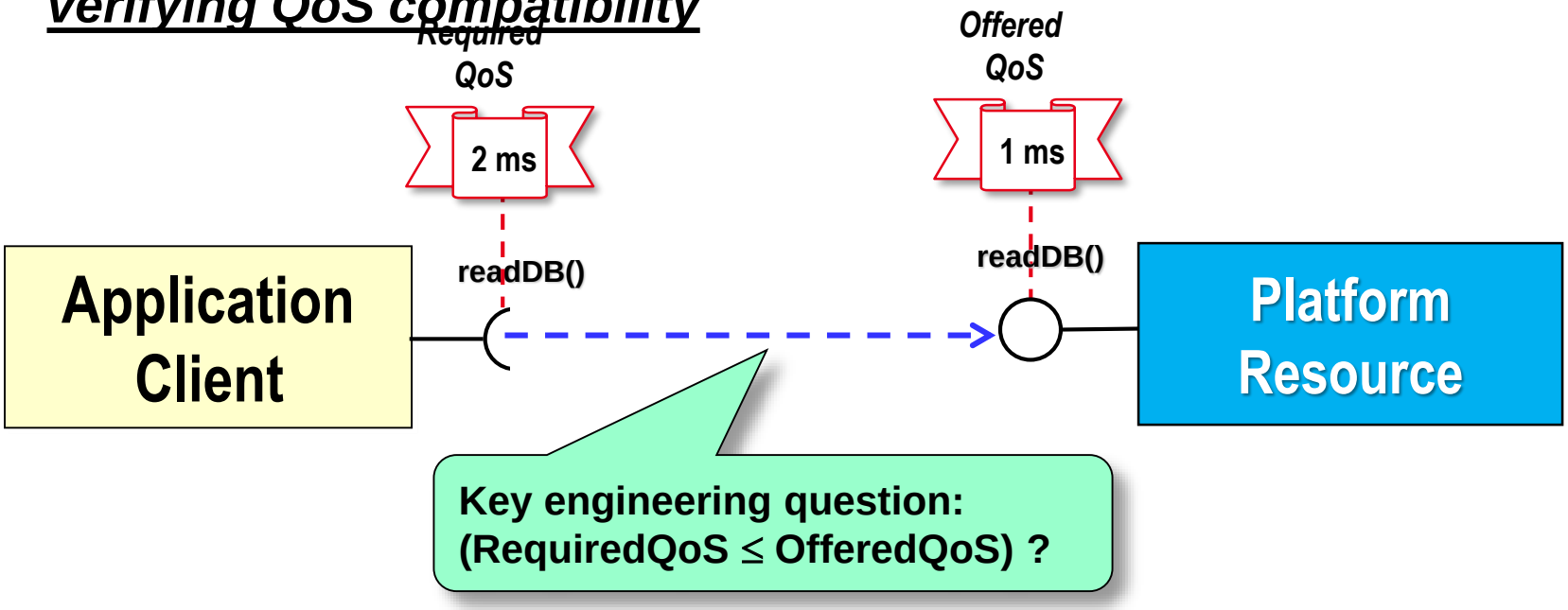
CORE CONCEPT: QUALITY OF SERVICE (QOS)

- ***Quality of Service (QoS):***
  - A measure of the effectiveness of service provisioning
- **Two complementary perspectives on QoS**
  - Required QoS: the demand side (what applications require)
  - Offered QoS: the supply side (what platforms provide)

Many engineering analyses consist of calculating whether (QoS) supply can meet (QoS) demand

QOS COMPATIBILITY

- **We have powerful mechanisms for verifying functional compatibility (e.g., type theory) but relatively *little support for verifying QoS compatibility***

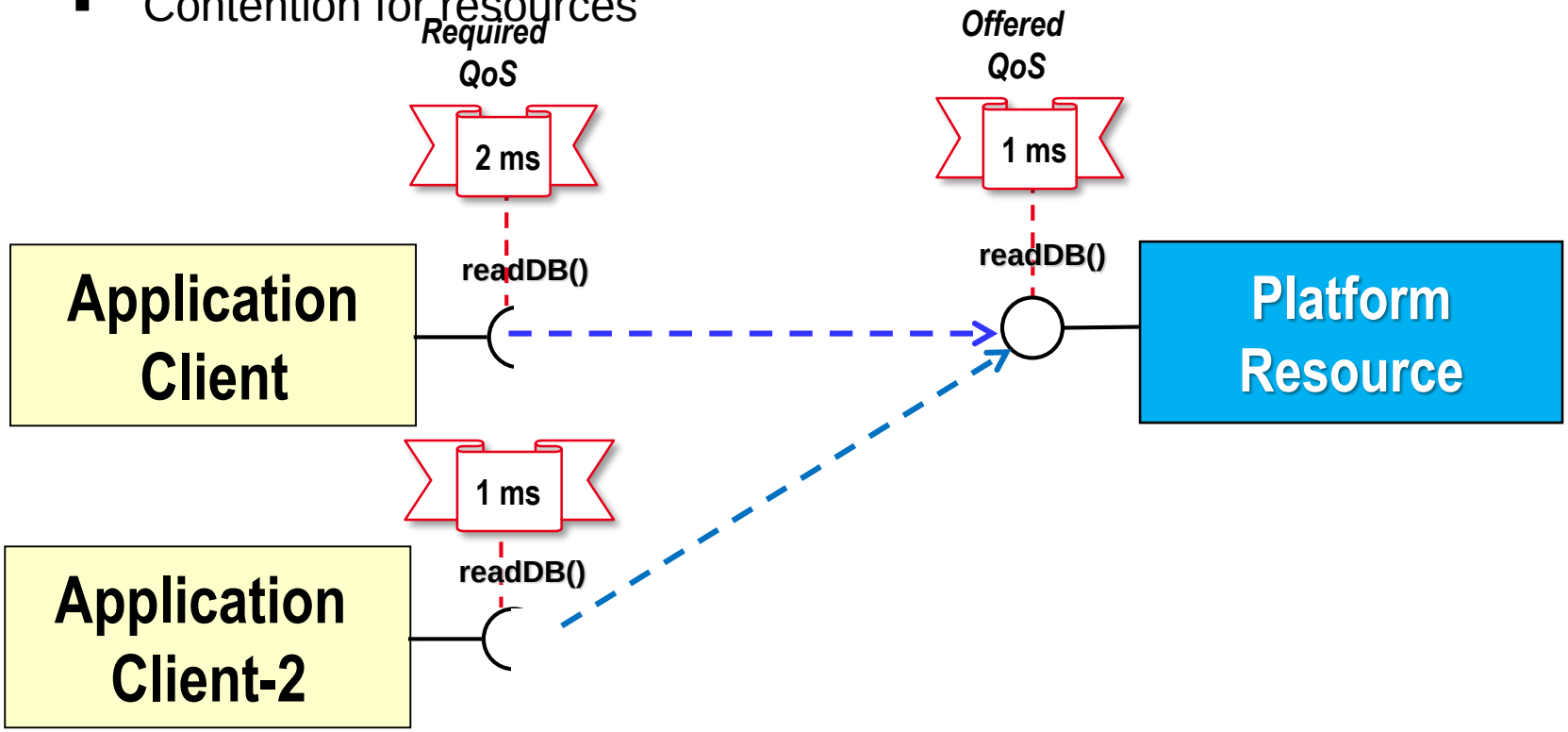


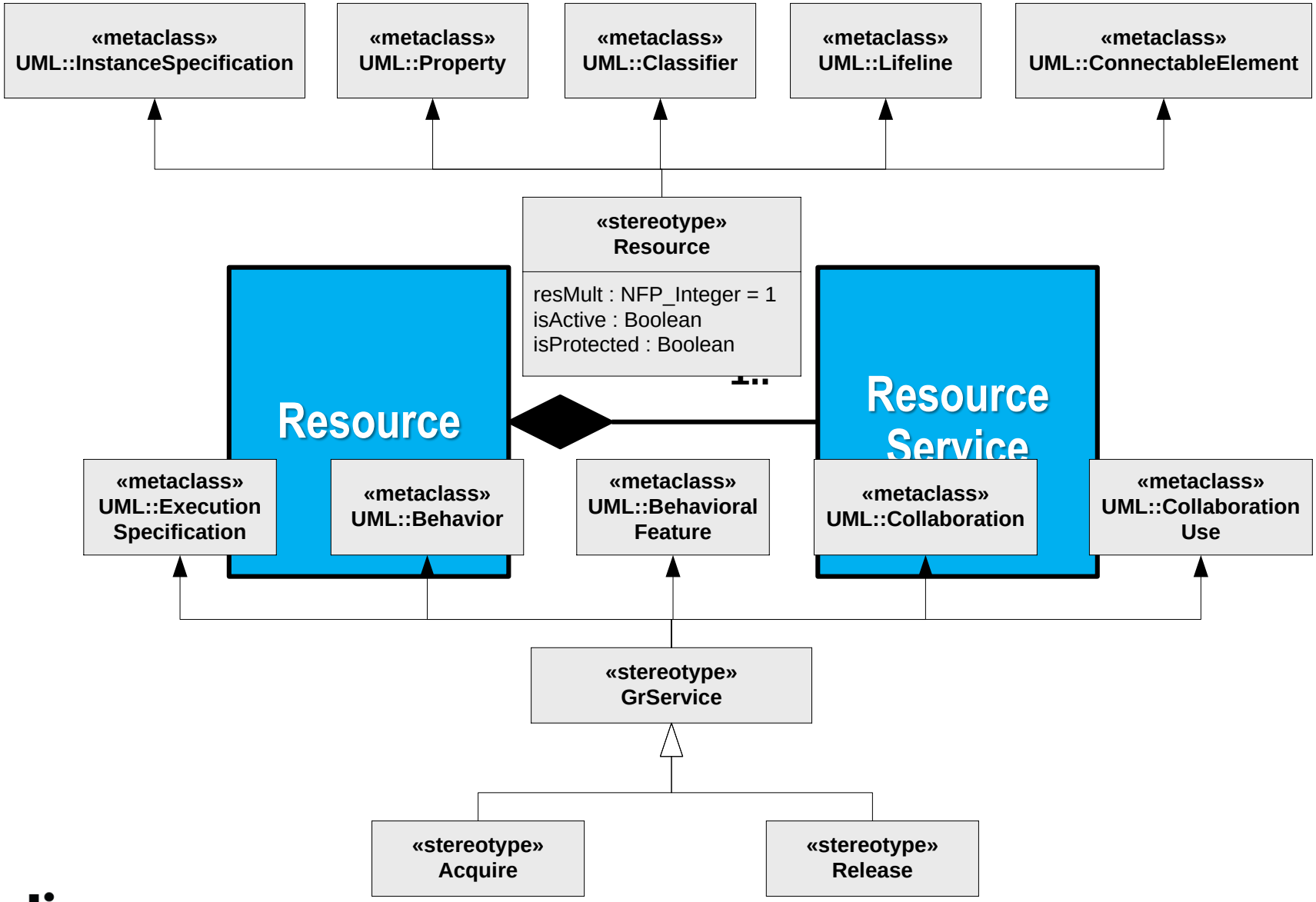
"Virtually every calculation an engineer performs…is a failure calculation…to provide the limits that cannot be exceeded"

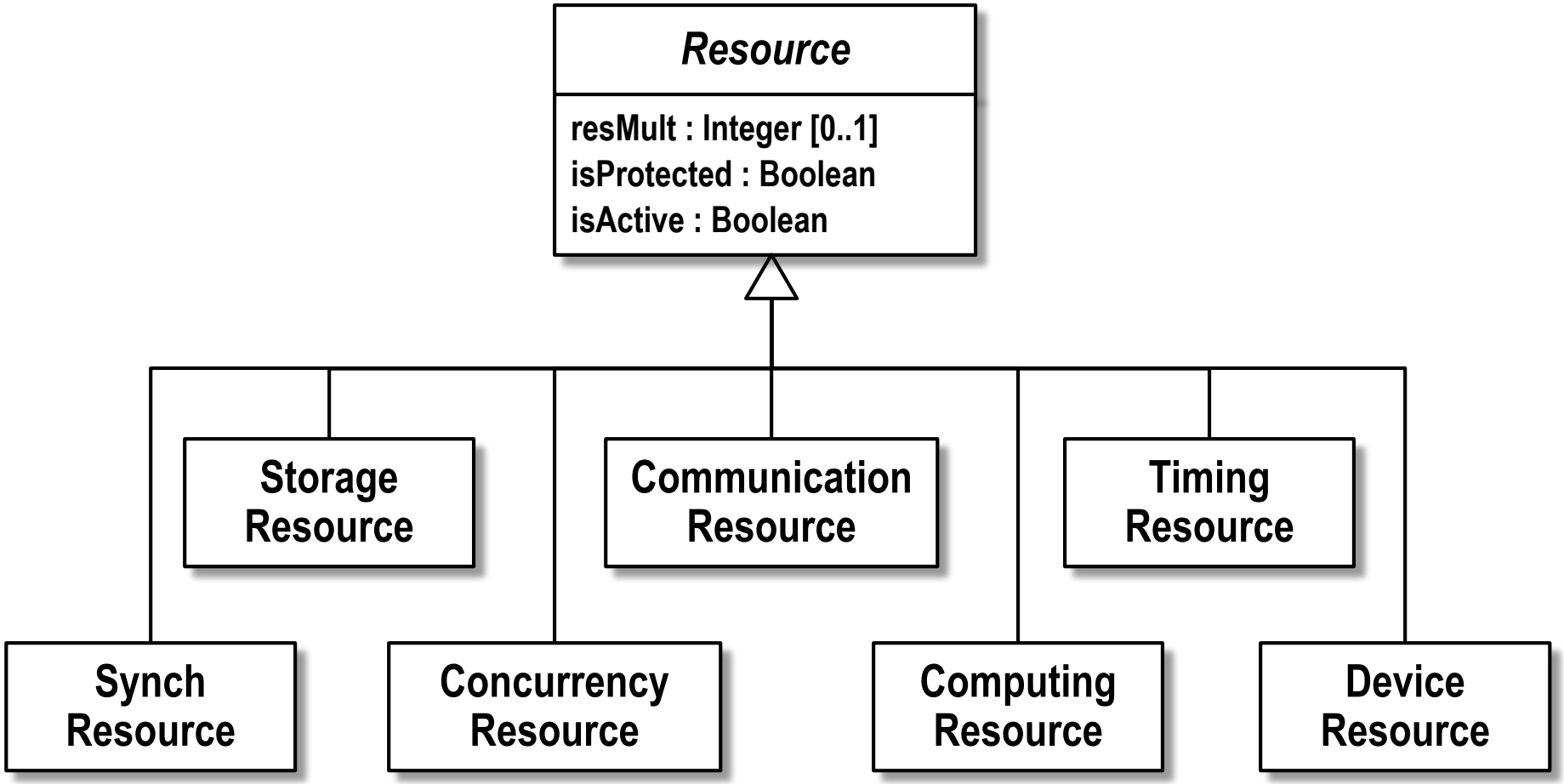
-- Henry Petroski

WHY IT IS DIFFICULT TO PREDICT SOFTWARE PROPERTIES?

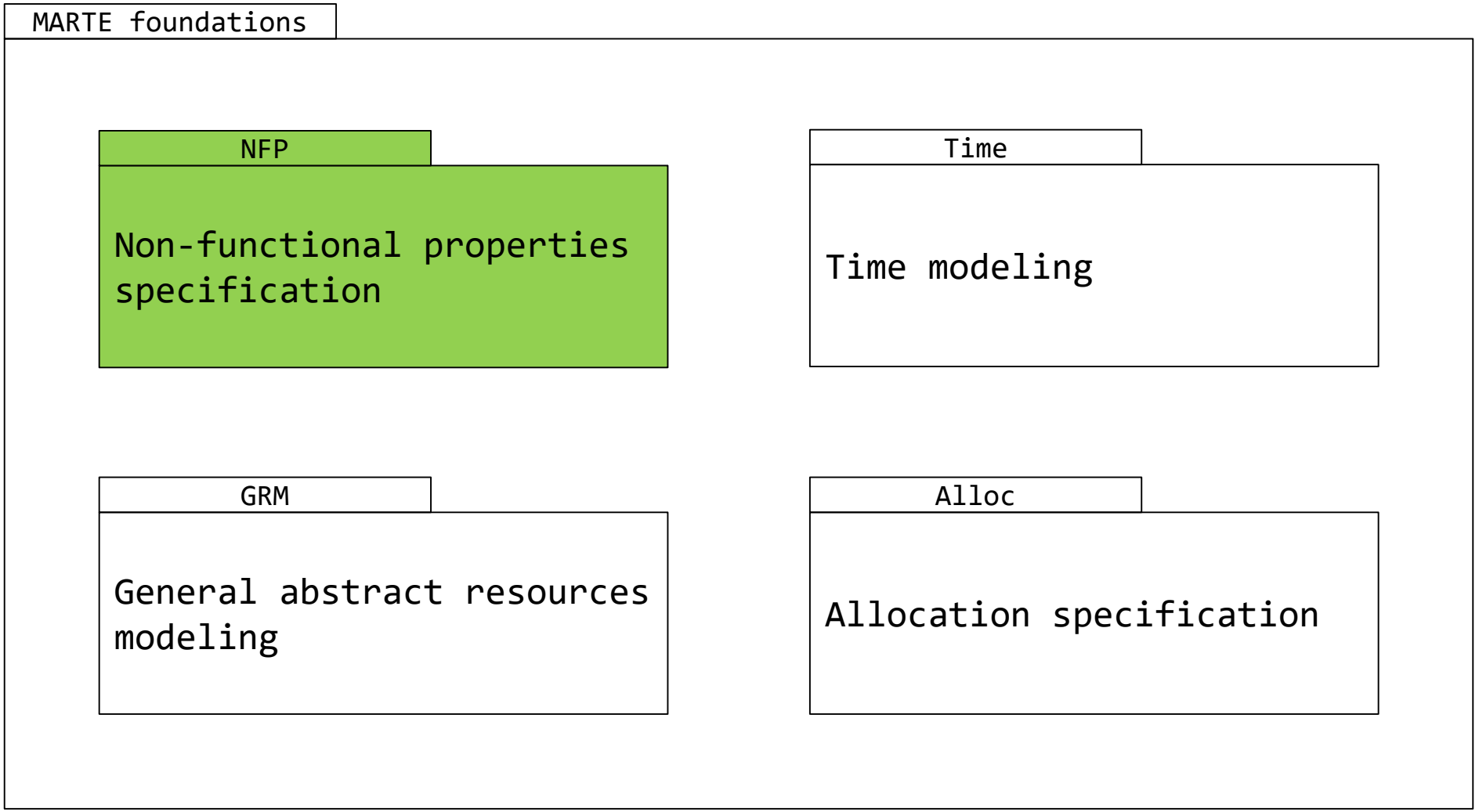
- **Because platform resources are often shared**
  - ..often by independently designed applications
  - Contention for resources





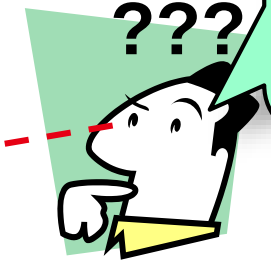


MARTE packages



THE SEMANTICS OF VALUES

- What do these numbers represent? (i.e., semantics)
- Which measurement units are used?
- How were they obtained (measured? estimated? computed?)

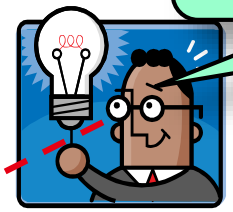


myCan: CAN_Bus

transmMode = Half-Duplex

Capacity = 4

- The system requires a CAN bus with a capacity of 4 kHZ max

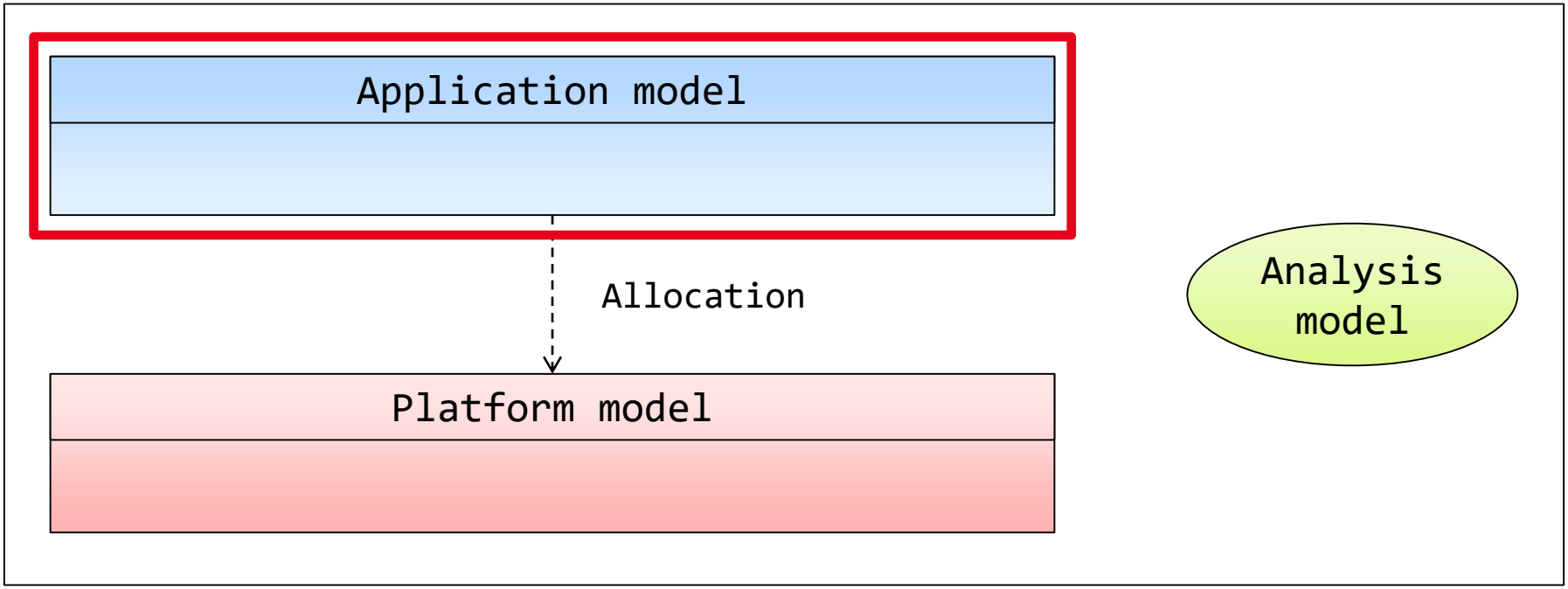


« CommHost »
can1: CAN_Bus
{ transMode= Half-Duplex,
capacity= (4, $capCan1, kHz, max, req),

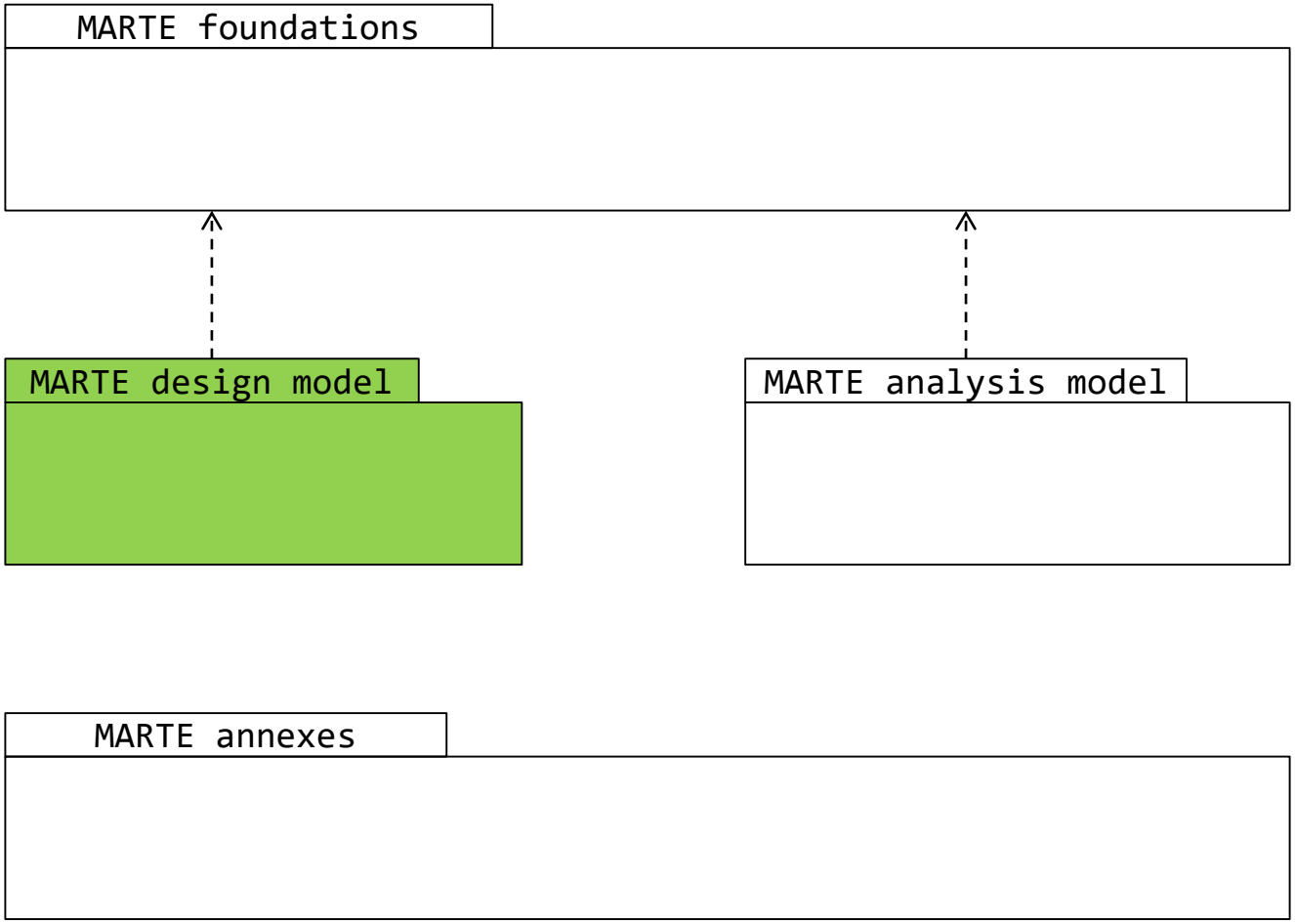
Providing necessary semantic information for values appearing in models is key to successful model-based automated analyses

CONTENT

Modeling approach



MARTE packages



MARTE packages

MARTE design model

GCM

Precise semantics for UML composite structures for enabling component-based MDE

HLAM

Further refinements of GRM concurrency-related concepts from the applications standpoint

HRM

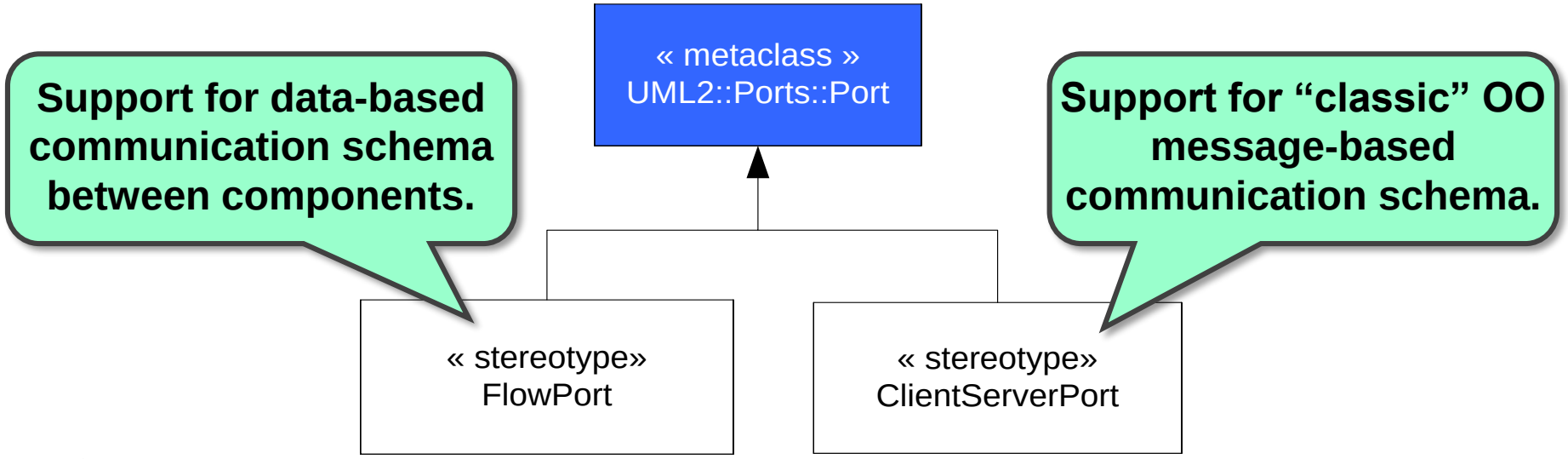
Hardware platform modeling

SRM

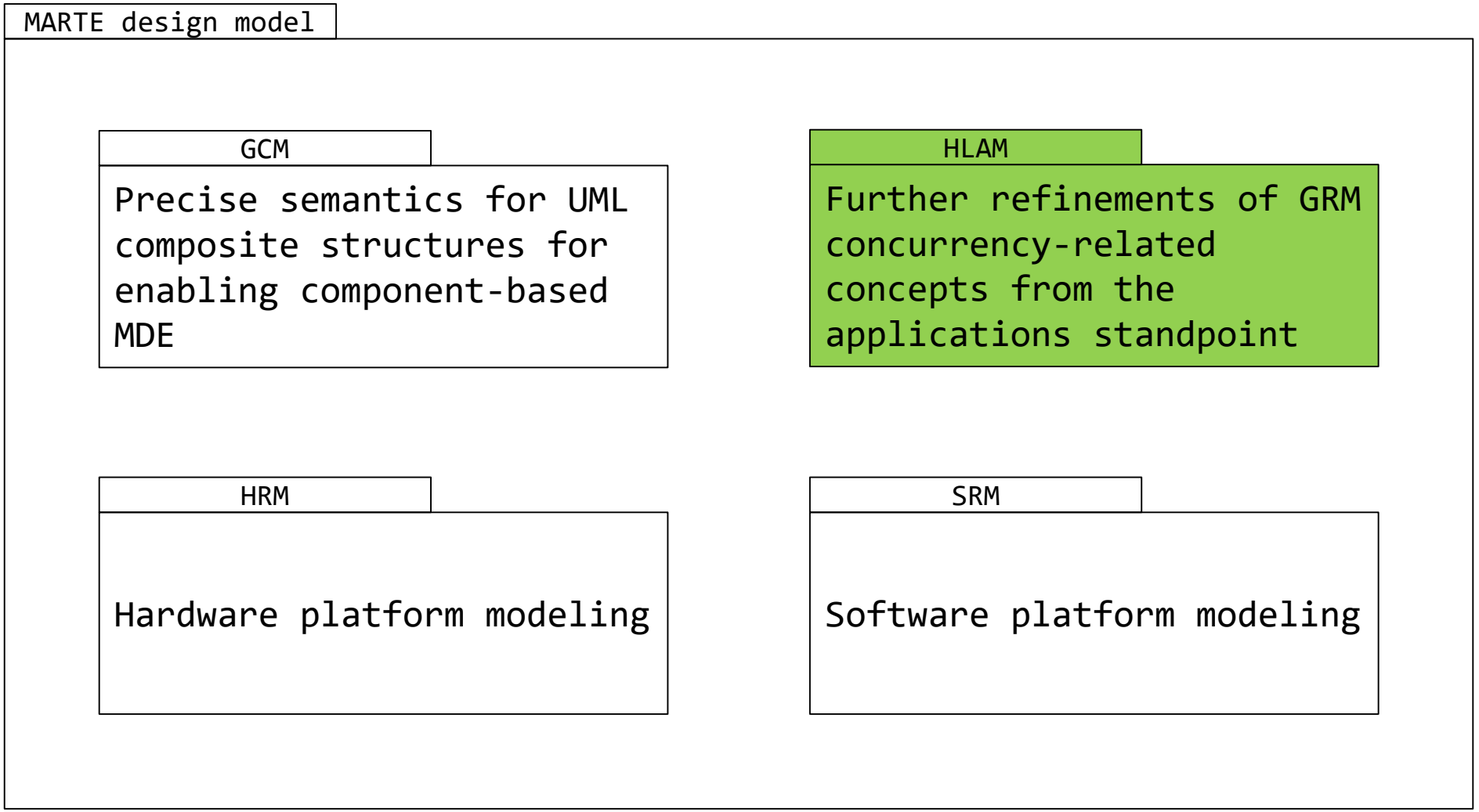
Software platform modeling

THE GENERIC COMPONENT MODEL OF MARTE

- **MARTE's GCM rationale**
  - Need to unify component models for CPS (e.g., AADL, Autosar, EAST-ADL2, Lightweight-CCM, and IP-XACT).
  - Lack of concepts for CPS:
    Needs for precise semantics (specially between behavior and structure)
    Needs for open semantics enabling various Models of Comm & Comp.
    Needs for data-flow communication at component-level



MARTE packages



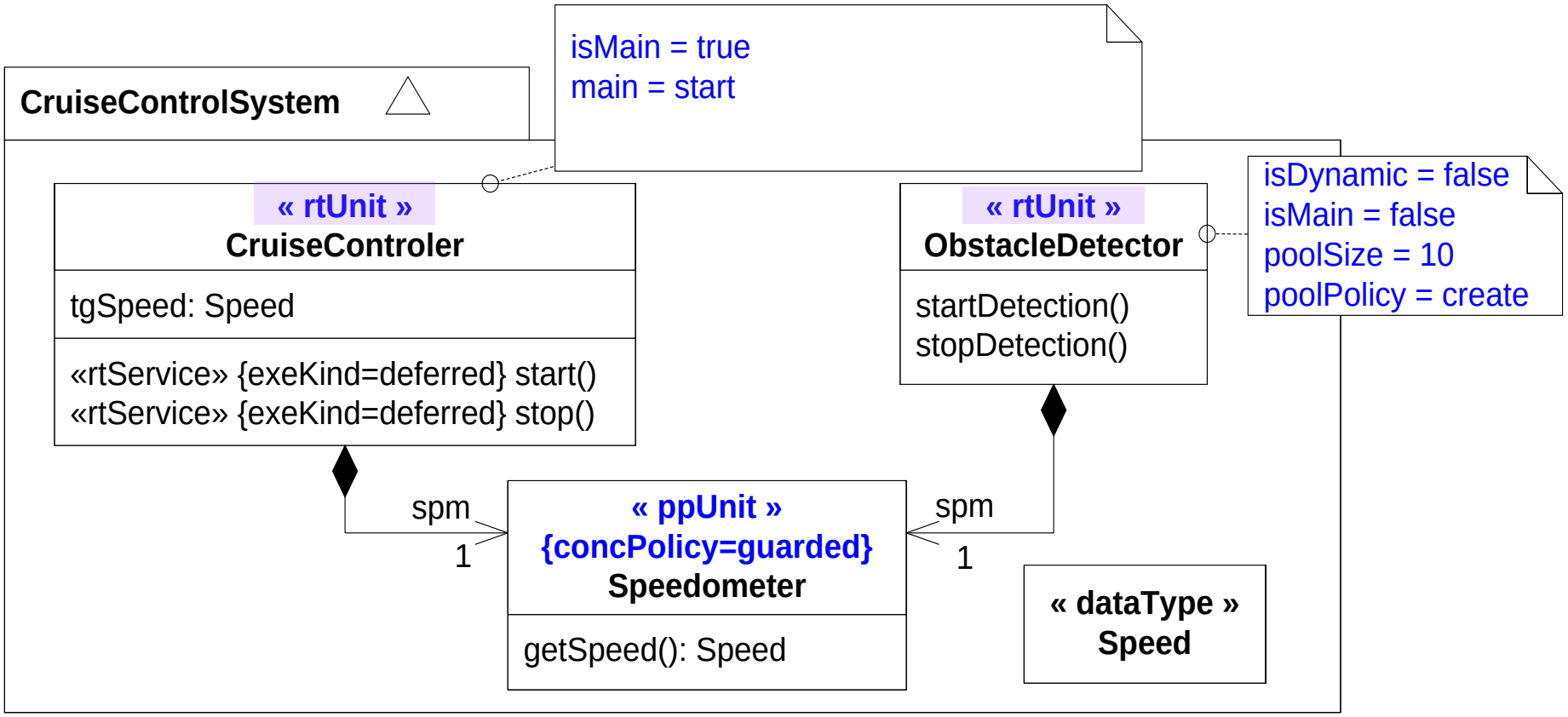
RTE MODEL OF COMPUTATION AND COMMUNICATION

- **Provides high-level concepts for modeling qualitative real-time features on classes / structured classes / components**
  - Real-Time Unit (RTUnit)
    Generalization of the Active Objects of the UML 2
    Owns at least one schedulable resource
    Resources are managed either statically (pool) or dynamically
    May have operational mode description (similar to AADL modes)
  - Protected Passive Unit (PPUnit)
    Generalization of the Passive Objects of the UML2
    Requires schedulable resources to be executed
    Supports different concurrency policies (e.g. sequential, guarded)
    Policies are specified either locally or globally
    Execution is either immediateRemote or deferred

RTE MODEL OF COMPUTATION AND COMMUNICATION

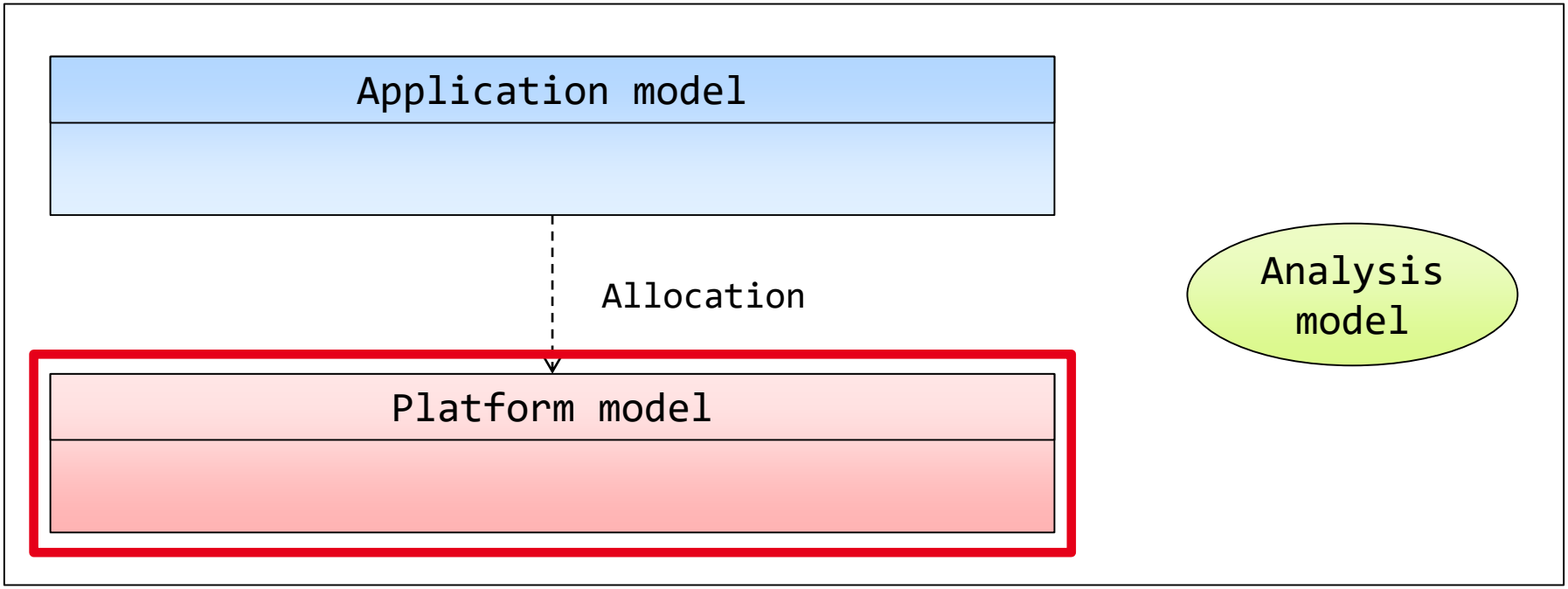
- **Provides high-level concepts for modeling quantitative real-time features on classes / structured classes / components**
  - Real-Time Behavior (RtBehavior)
    Message Queue size and policy bound to a provided behavior
  - Real-Time Feature (RTF)
    Relative/absolute/bound deadlines, ready time and miss ratio
    Apply to UML Action, Message, Signal, BehavioralFeature
  - Real-Time Connector (RteConnector)
    Throughput, transmission mode and max blocking/packet Tx time
    Apply to UML Connector

EXAMPLE OF «RTUNIT» AND «PPUNIT»

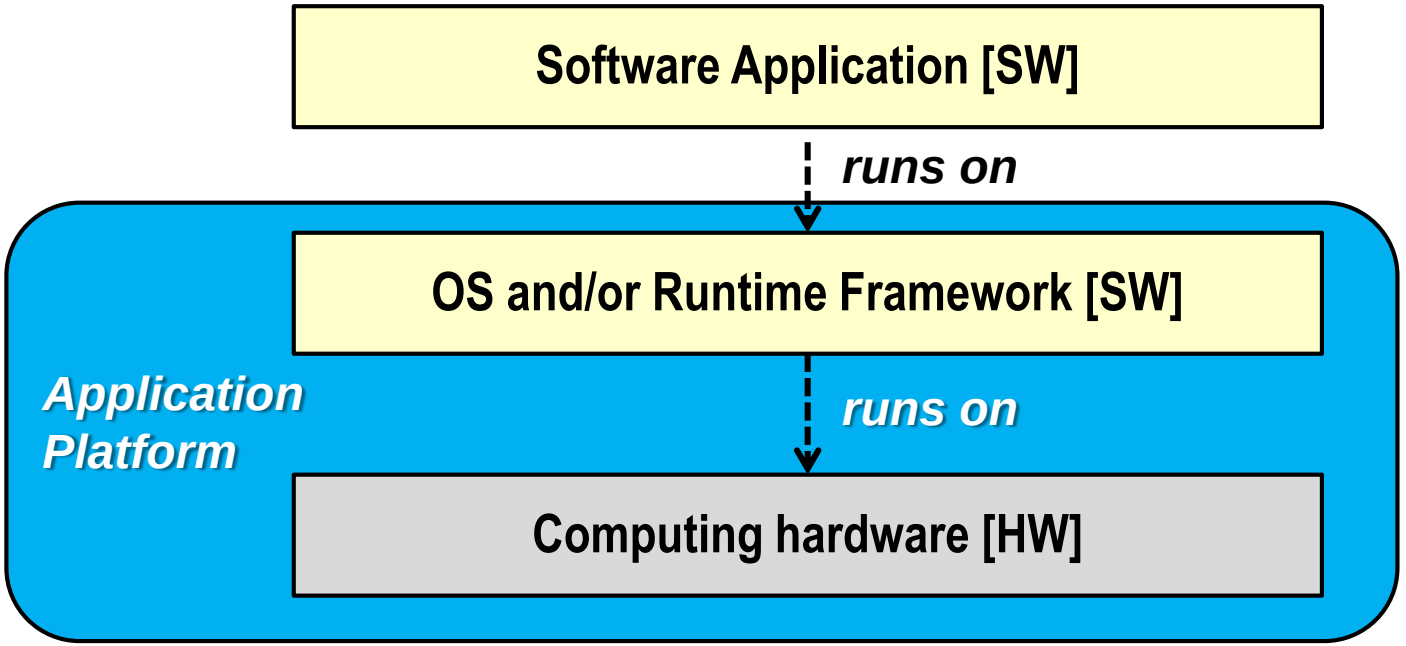


CONTENT

Modeling approach



PLATFORMS: WHERE SOFTWARE MEETS PHYSICS

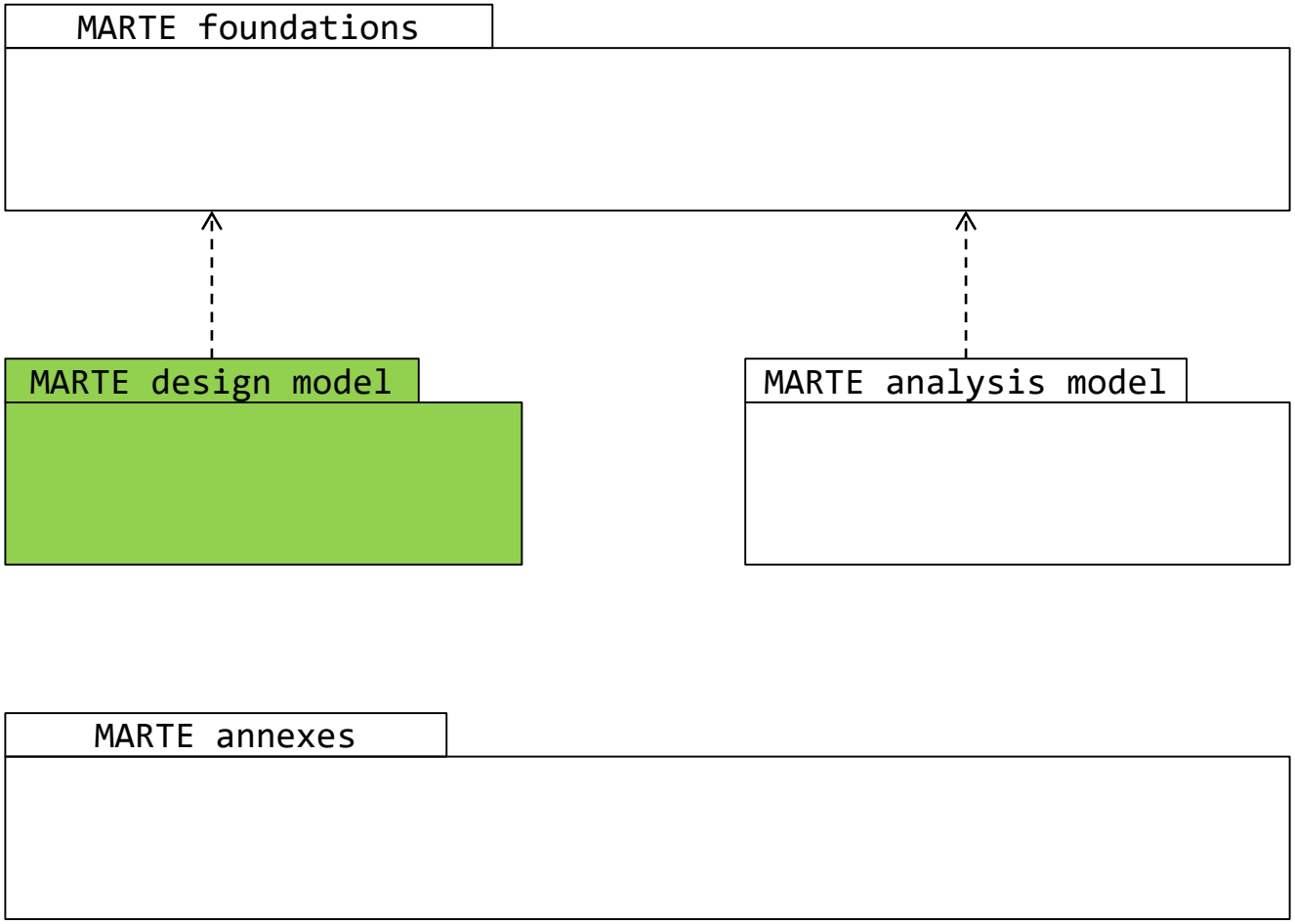


- **Platform def.:**

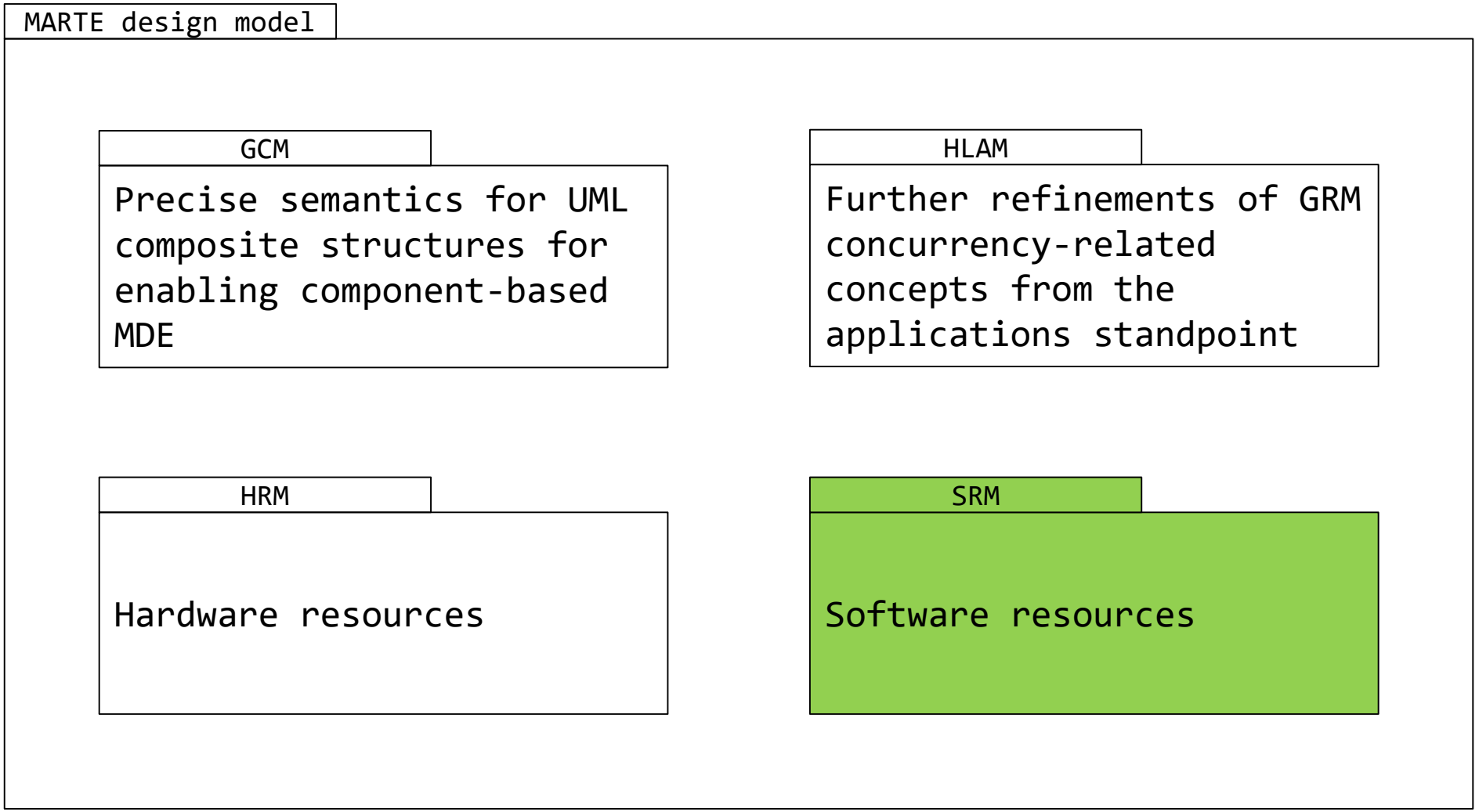
  "the full complement of software and/or hardware required for a given application program to execute"

The physical characteristics of the platform have a fundamental impact on the quality characteristics of software applications

MARTE packages



MARTE packages

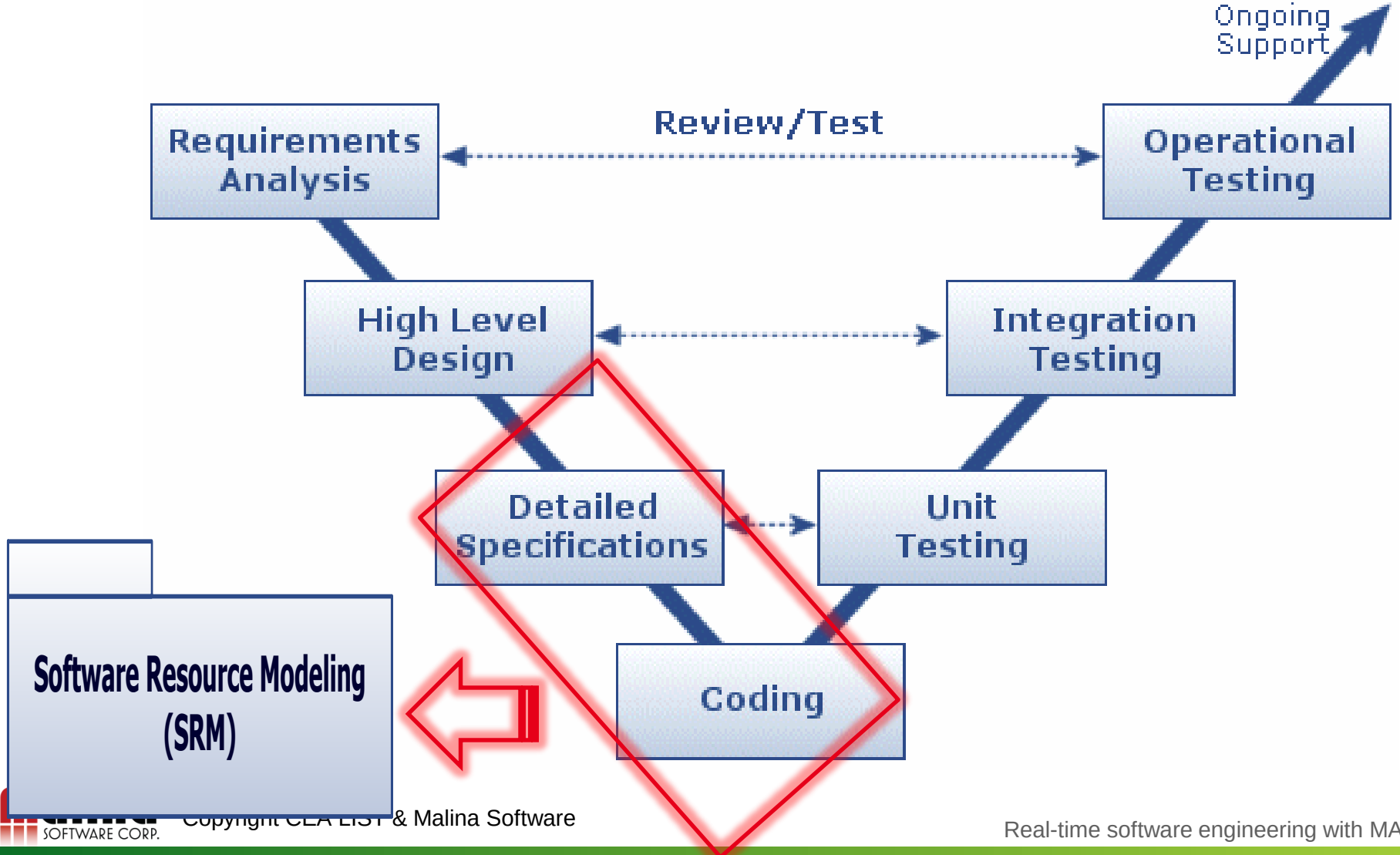


WHAT IS THE SOFTWARE RESOURCE MODELING PROFILE (SRM) ?

- **A UML profile for modeling APIs of RT/E sw execution supports**
  - Real Time Operating Systems (e.g., ARINC653-like OS)
  - Dedicated Language Libraries (e.g., ADA)
- **BUT it is NOT a new API standard dedicated to the RT/E domain!**
  - SRM is the result of a very deep state of the art and of the practices in the domain of RTOS, including:
    e.g., POSIX, ARINC 653, SCEPTRE, Linux RT, …

SRM = a unified mean to model RTOS (or equivalent language constructs) APIs

IN WHICH STEPS SHALL I USE SRM ?



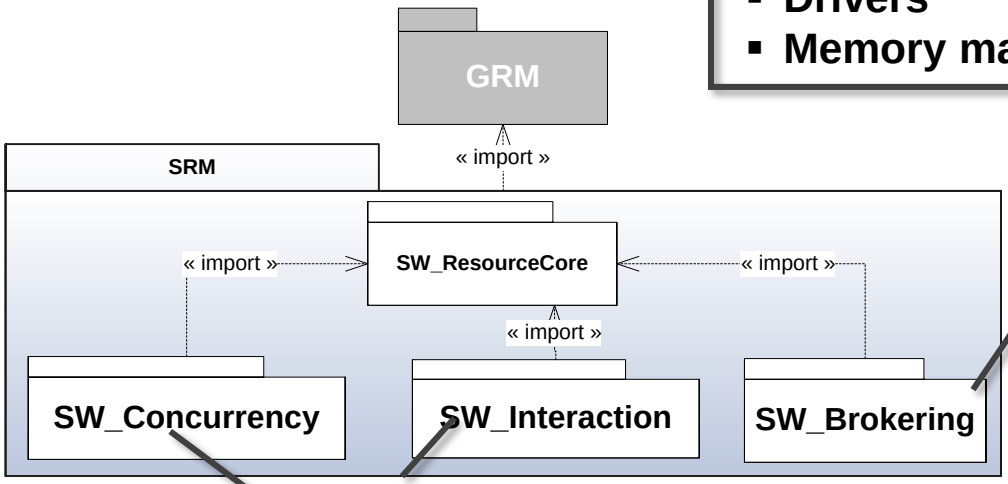
WHY USE SRM FOR MODELING RTOS API(S)?

- **RTOS API modeling with standard UML is already possible**
  - But, generic UML lacks RTE native artifacts!
    No modeling artifacts to describe specific concepts
    - e.g. tasks, semaphores and mailboxes
  - Consequently, models rely purely on naming conventions
    Not possible to define generic tools using these models
    - e.g. code generator or model transformations for analysis.
- **Hence, the SRM profile enables:**
  - Precise modeling of multitasking designs
  - Specification of generic generative tools
    e.g., code generators or model transformations for schedulability analysis.

WHAT IS SUPPORTED BY THE SRM PROFILE ?

Hardware and software resources brokering:

- **Drivers**
- **Memory management**



Concurrent execution contexts:

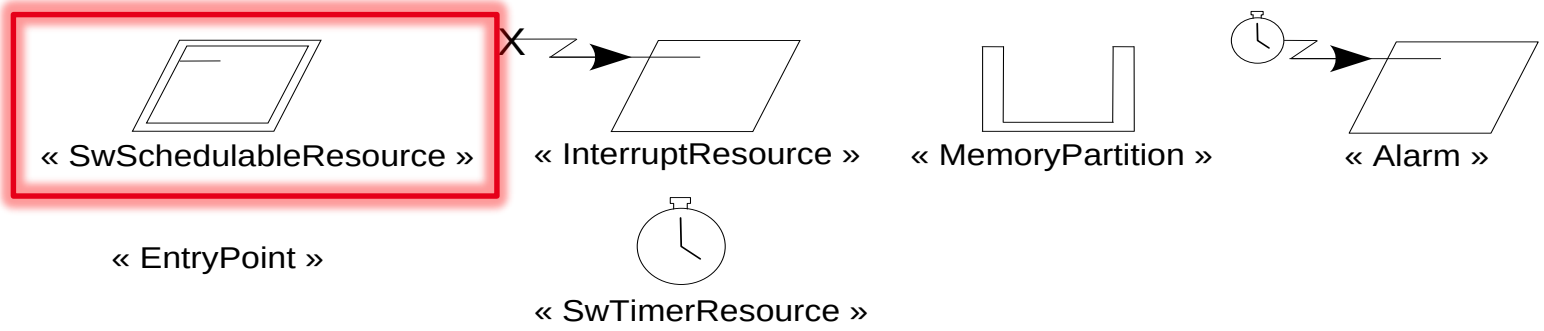
- **Schedulable Resource (~Task)**
- **Memory Partition (~Process)**
- **Interrupt Resource**
- **Alarm**

Interactions between concurrent contexts:

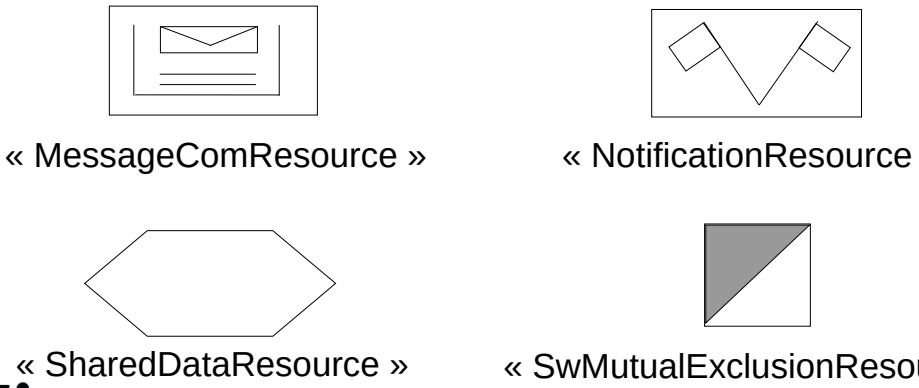
- **Communication**
  - Shared data
  - Message (~Message queue)
- **Synchronization**
  - Mutual Exclusion (~Semaphore)
  - Notification Resource (~Event mechanism)

SNAPSHOT OF THE UML EXTENSIONS PROVIDED BY SRM

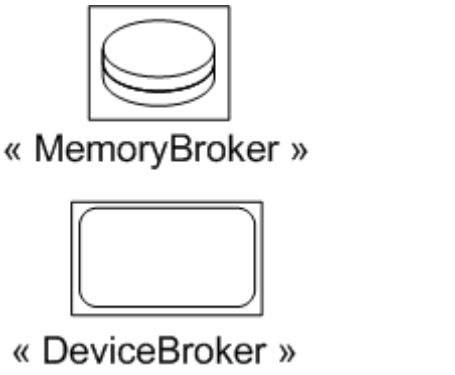
SRM::SW_Concurrency



SRM::SW_Interaction



SRM::SW_Brokering



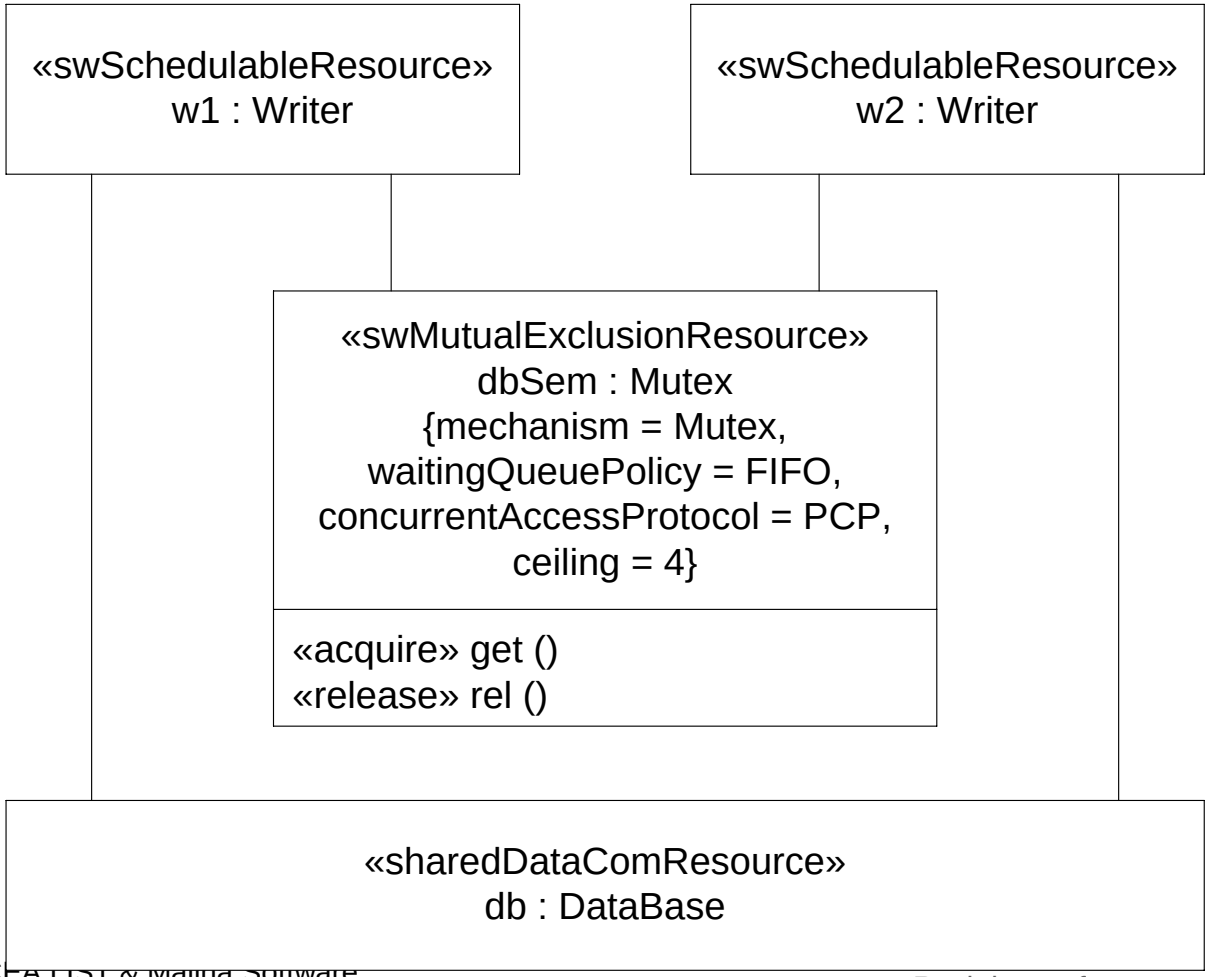
DETAILS OF «SWSCHEDULABLERESOURCE»

- **Semantic (from MARTE::SRM::Concurrency package)**
  - Resource which executes, periodically or not, concurrently to other concurrent resources

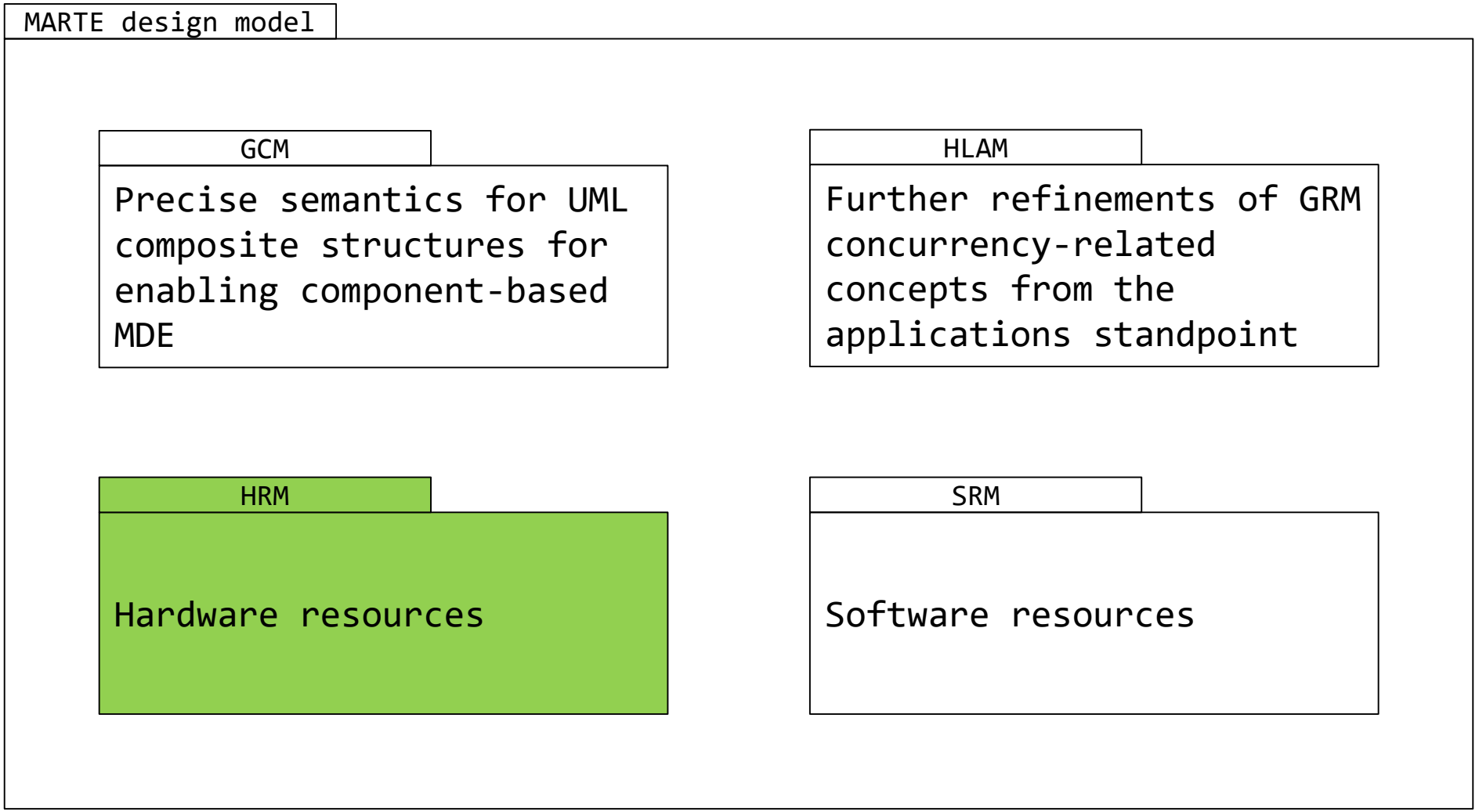
- **Main features**
  - Owns an entry point referencing the application code to execute
  - May be restricted to execute in a given address space (i.e. a memory partition)
  - Owns properties: e.g., Priority, Deadline, Period and StackSize
  - Provides services: e.g., activate, resume and suspend

SRM EXAMPLE: APPLICATION MODELING

- **Two writers accessing a shared database resource using a mutex**



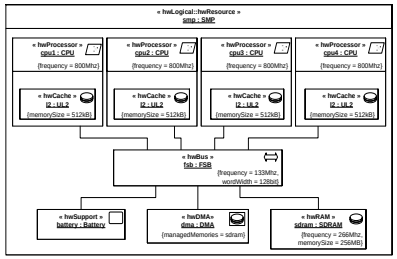
MARTE packages



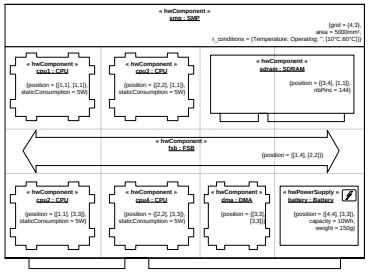
HARDWARE PLATFORM-BASED MODELLING

- **HRM: for describing structure of hardware platform**
  - Different abstraction levels.
  - Two possible views:

Logical view (functionality)



Physical view (layouts)



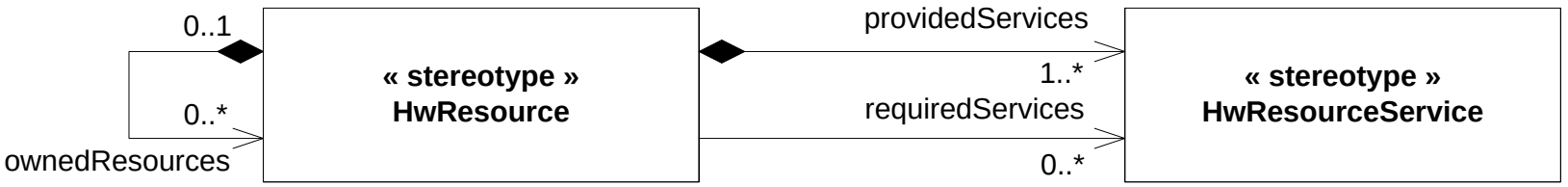
Usage examples of a HW platform models

- More precise model-based analysis, e.g. for WCET analysis.
- More precise model-based simulation

HRM STRUCTURE

- **Hierarchical taxonomy of hardware concepts**
  - Successive **inheritance** layers
  - **From** generic concepts (GRM-like)
    e.g., HwComputingResource, HwMemory and HwCommunicationResource
  - **To** specific and detailed resources
    e.g., HwProcessor, HwBranchPredictor, HwCache, HwMMU and HwBus

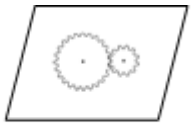
All HRM concepts are HwResource(s)



HRM LOGICAL MODELING

- **Provides a functional description based on a functional classification of hardware resources:**

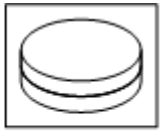
HRM::HwComputing



« HwProcessor », « HwPLD », « HwASIC »

HRM::HwStorage

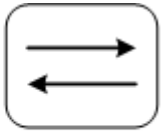
« HwCache », « HwRAM », « HwROM », « HwDrive »



« HwMMU », « HwDMA »

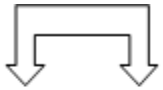
HRM::HwDevice

« HwDevice », « HwSupport »

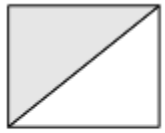


« HwI/O »

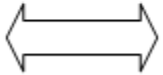
HRM::HwCommunicatio n



« HwBridge »



« HwArbiter »



« HwMedia », « HwBus »

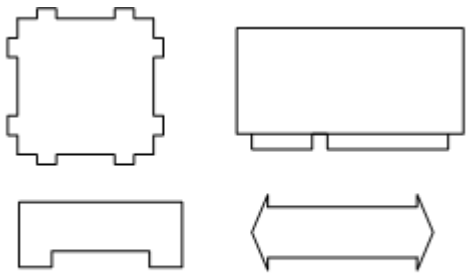
HRM::HwTiming

« HwClock », « HwTimer »

HRM PHYSICAL MODELING

- **Enable modeling of physical properties of HW resources, via two viewpoints:**
  - HwLayout
    Forms: Chip, Card, Channel…
    Dimensions, area and arrangement mechanism within rectilinear grids
    Environmental conditions: e.g. temperature, vibration, humidity…
  - HwPower
    Power consumption and heat dissipation

HRM::HwLayout



« HwComponent »
kind : {Card, Channel, Chip, Port}

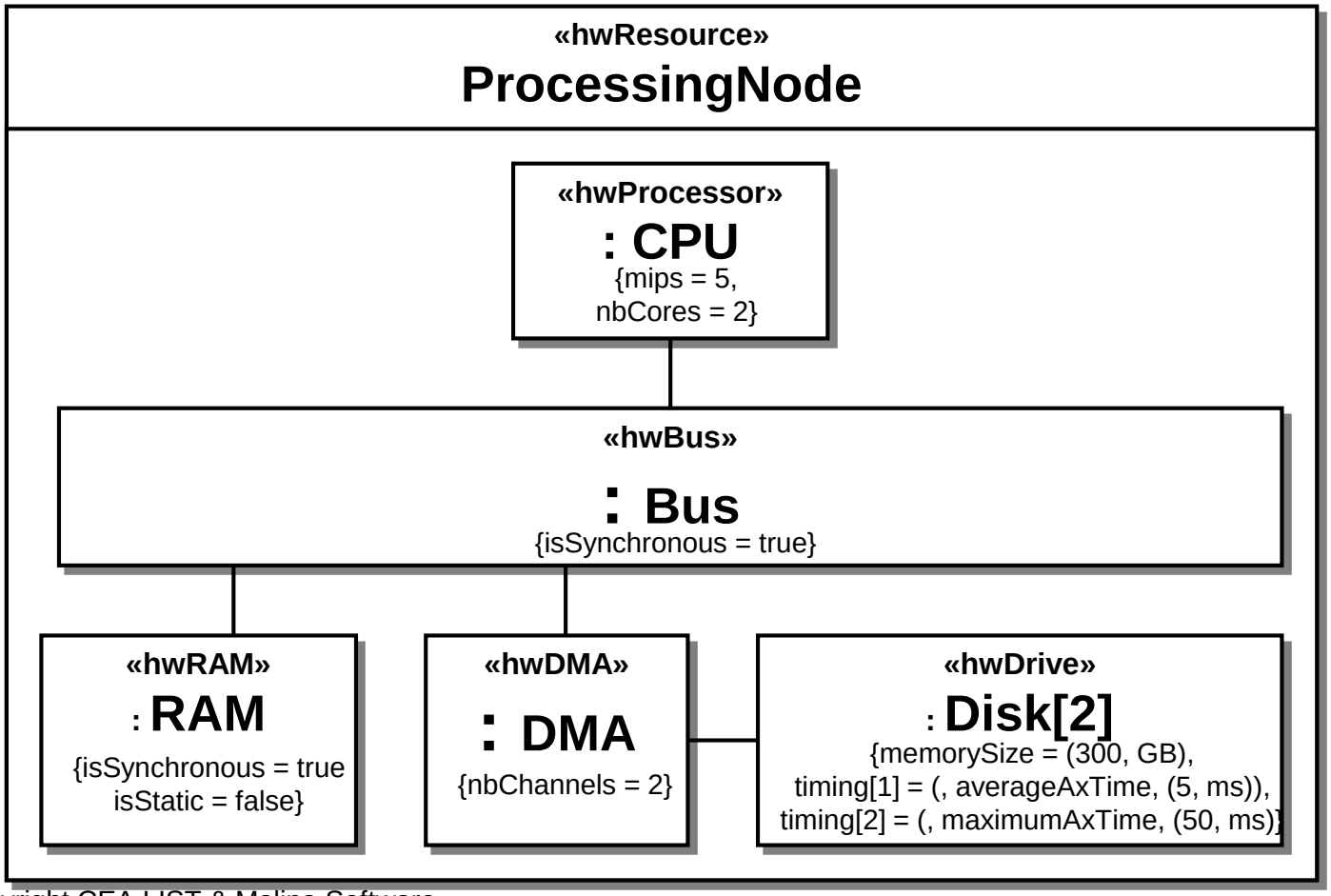
HRM::HwPower

« HwPowerSupply »

« HwCoolingSupply »

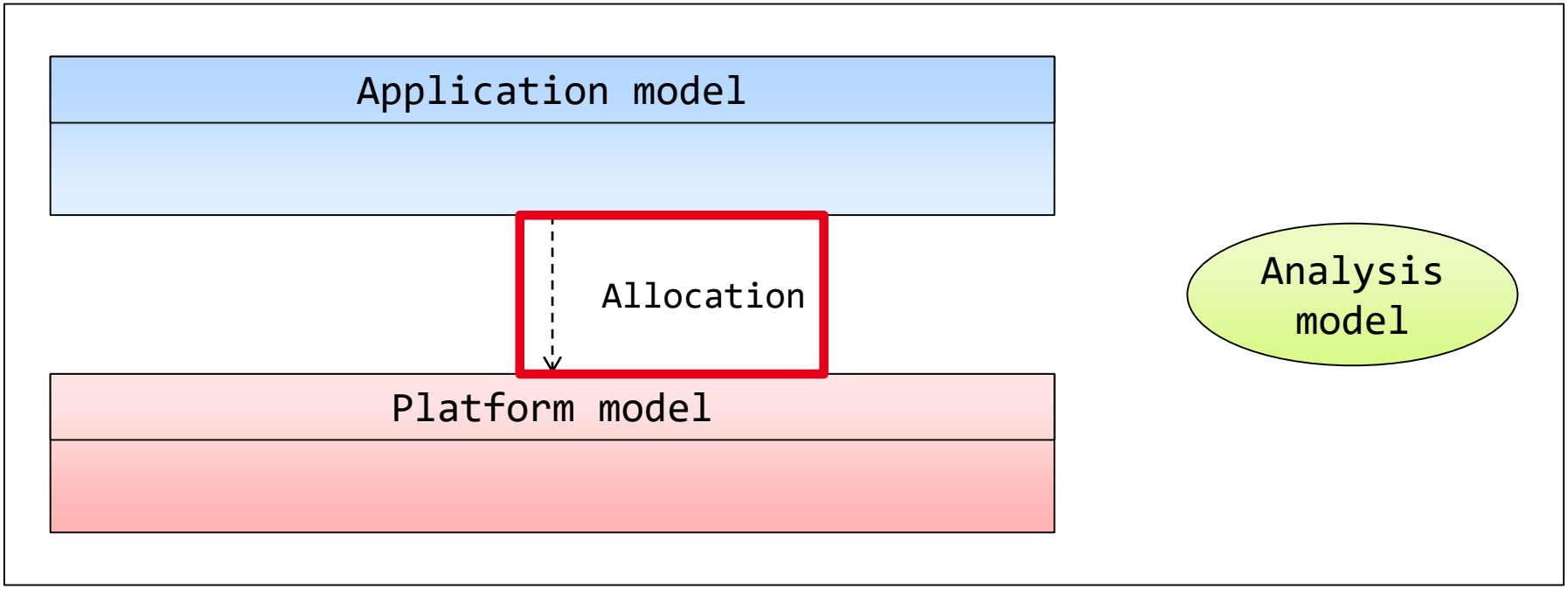
EXAMPLE: MODELING HARDWARE WITH MARTE

- **A hardware platform with specified QoS values**



CONTENT

Modeling approach



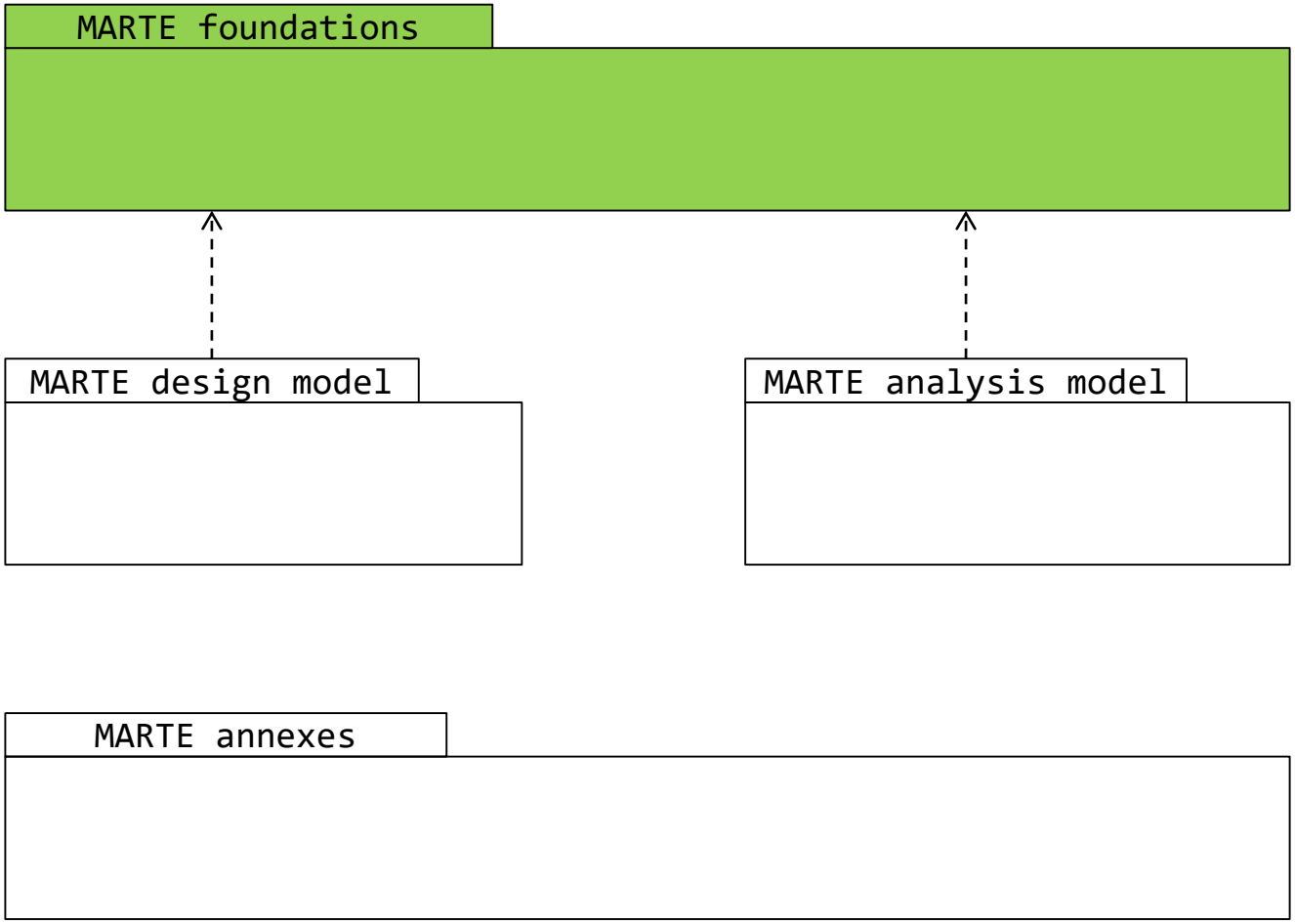
OUTLINE OF THE ALLOCATION CONCEPT

- **Purpose**
  - Provide support for denoting the "mapping/association" of the functional parts of a system (also called application) onto its resources (also called platform).

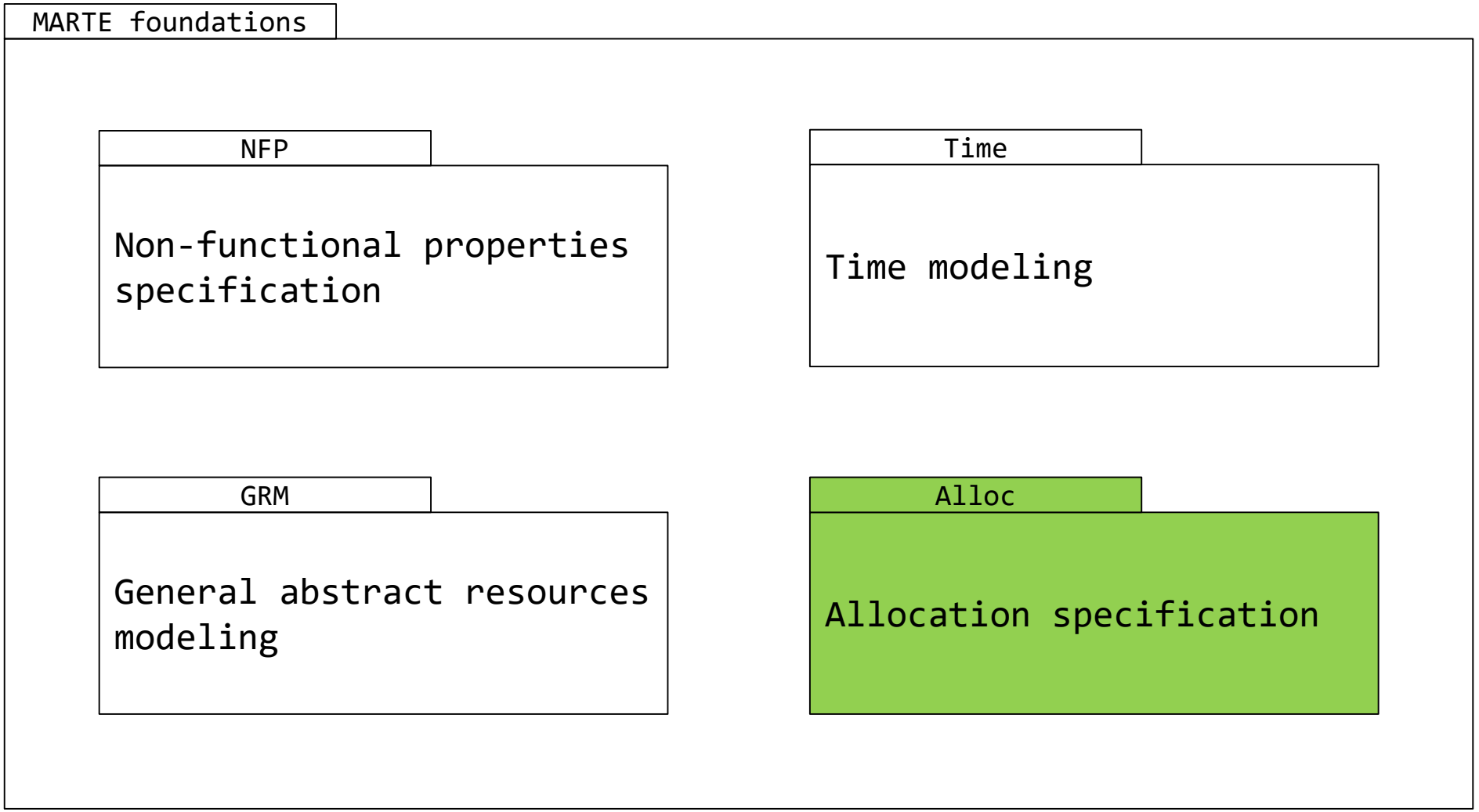
- **Similar to the SysML Allocation, but…**
  - More restrictive
    Focus on application allocation on its underlying platform
  - 2 natures of allocation:
    Spatial distribution
    - e.g., a variable allocated to a given memory resource.

    Temporal scheduling
    - e.g., a function computed/executed on a given processor resource.
  - Possibly associated with QoS constraints

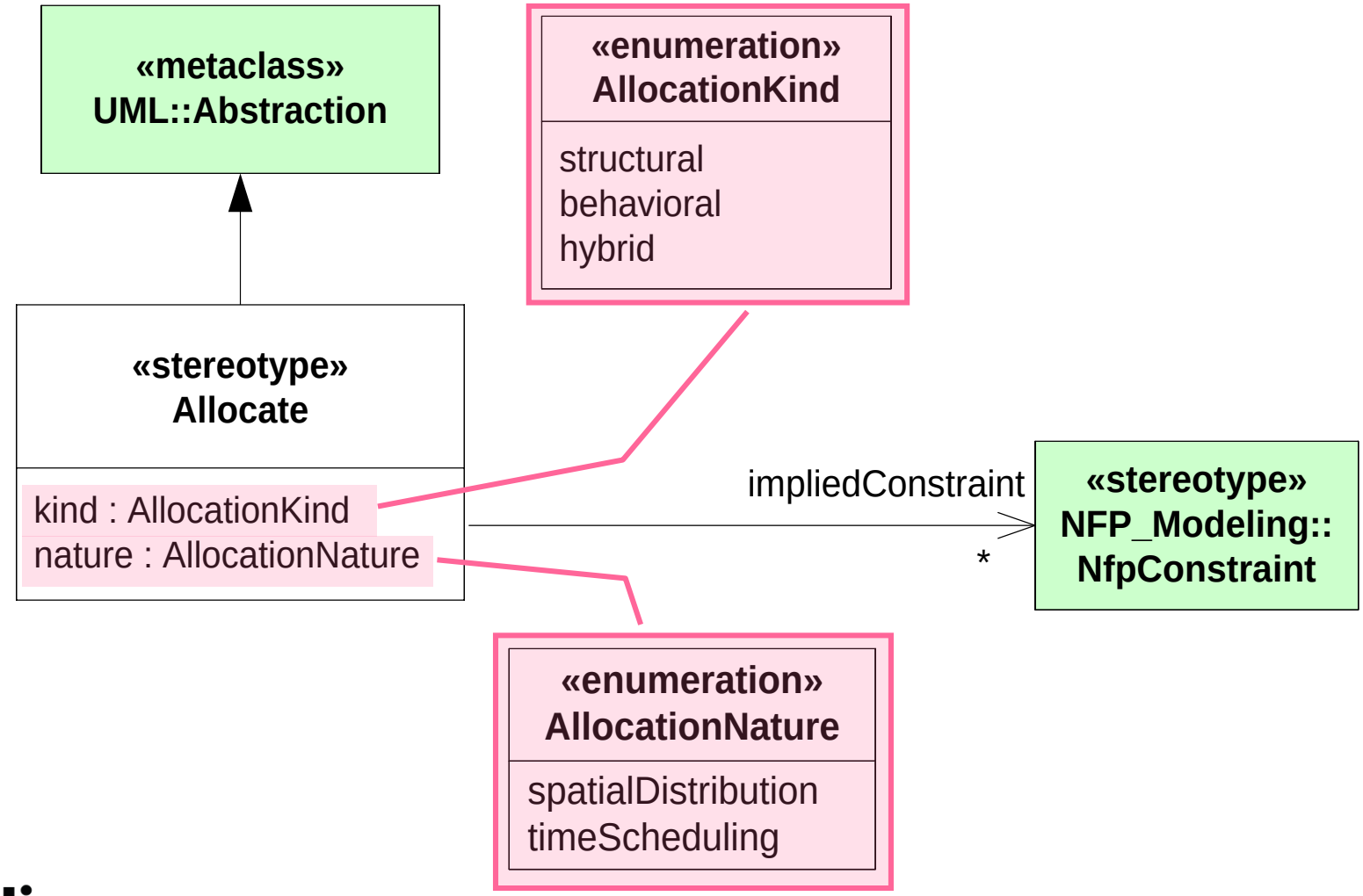
MARTE packages



MARTE packages

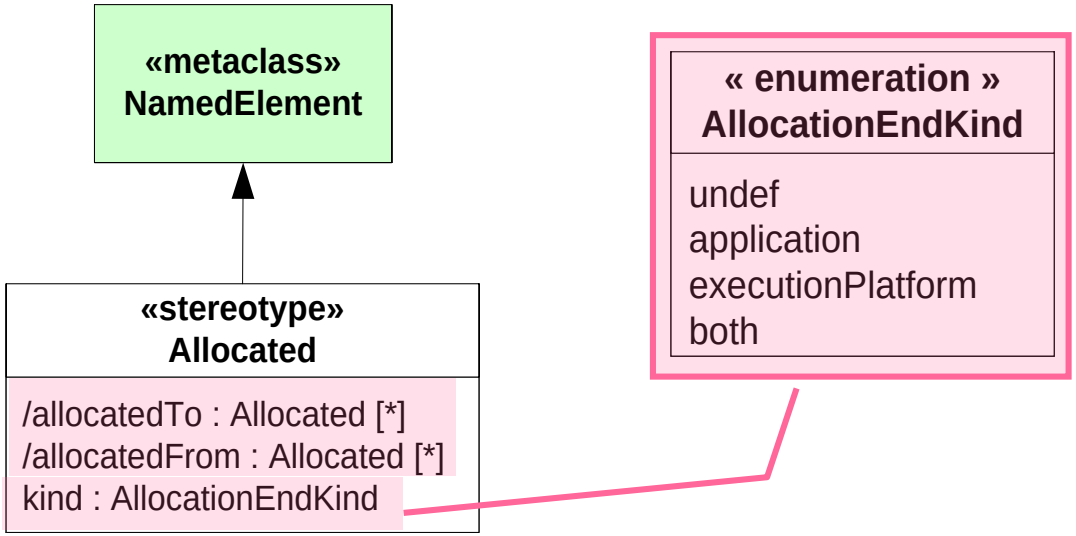


OUTLINE OF THE ALLOCATION STEREOTYPE

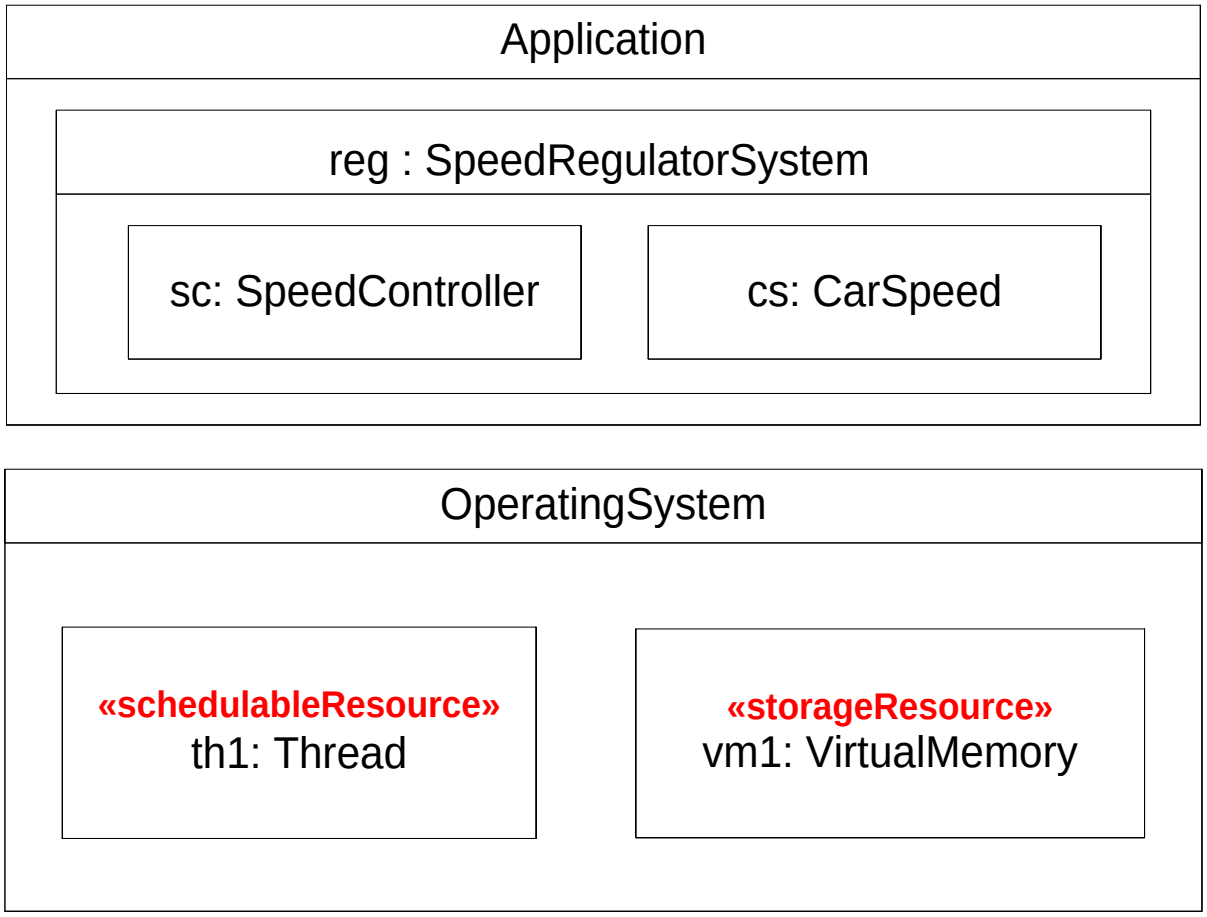


DEFINING THE ROLES OF MODEL ELEMENTS WITHIN ALLOCATION

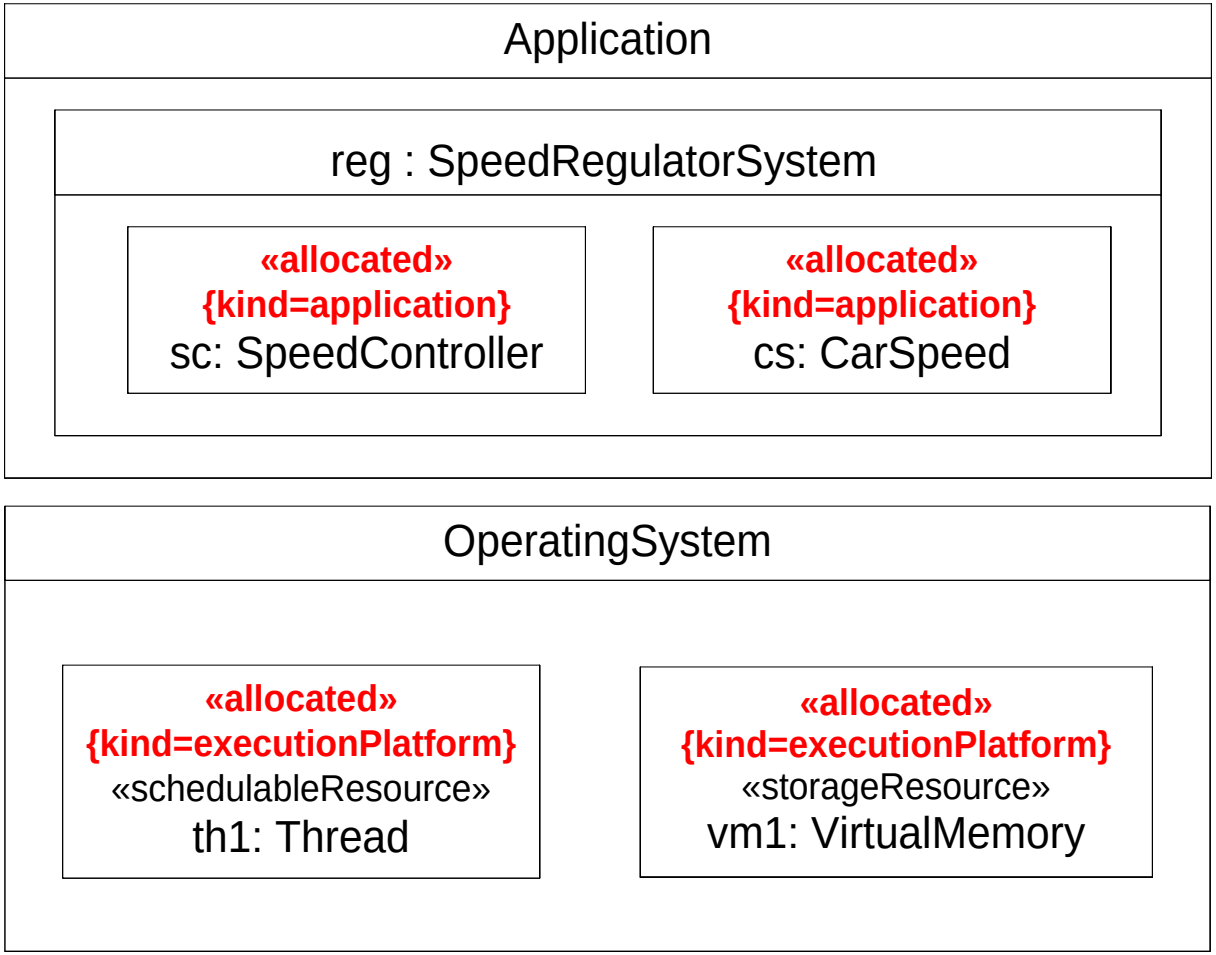
- **MARTE allocation rely on the distinction between:**
  - Model elements of the application
  - Model elements of the platform
- **« allocated »**



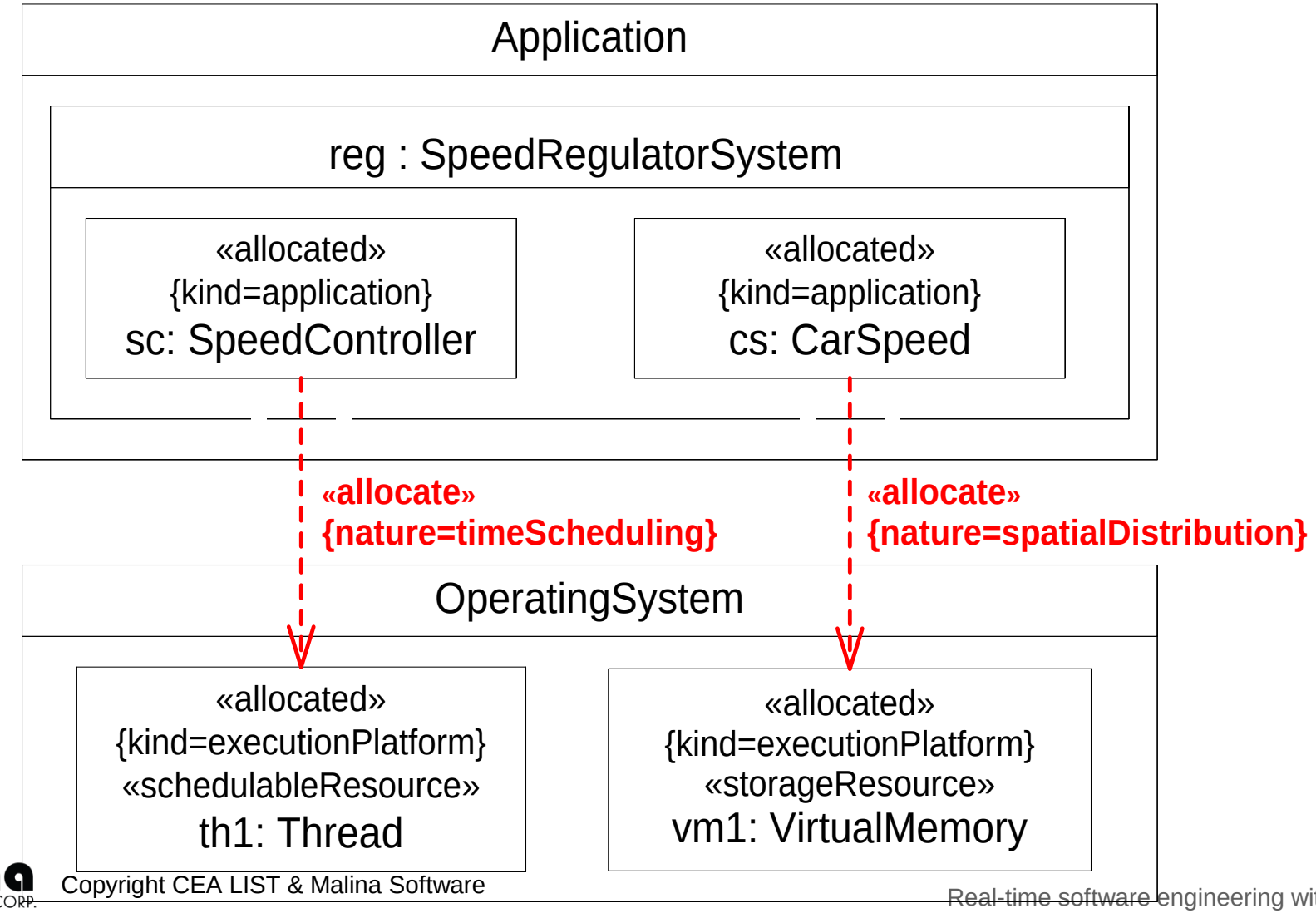
ALLOCATION EXAMPLE: ORIGINAL SITUATION

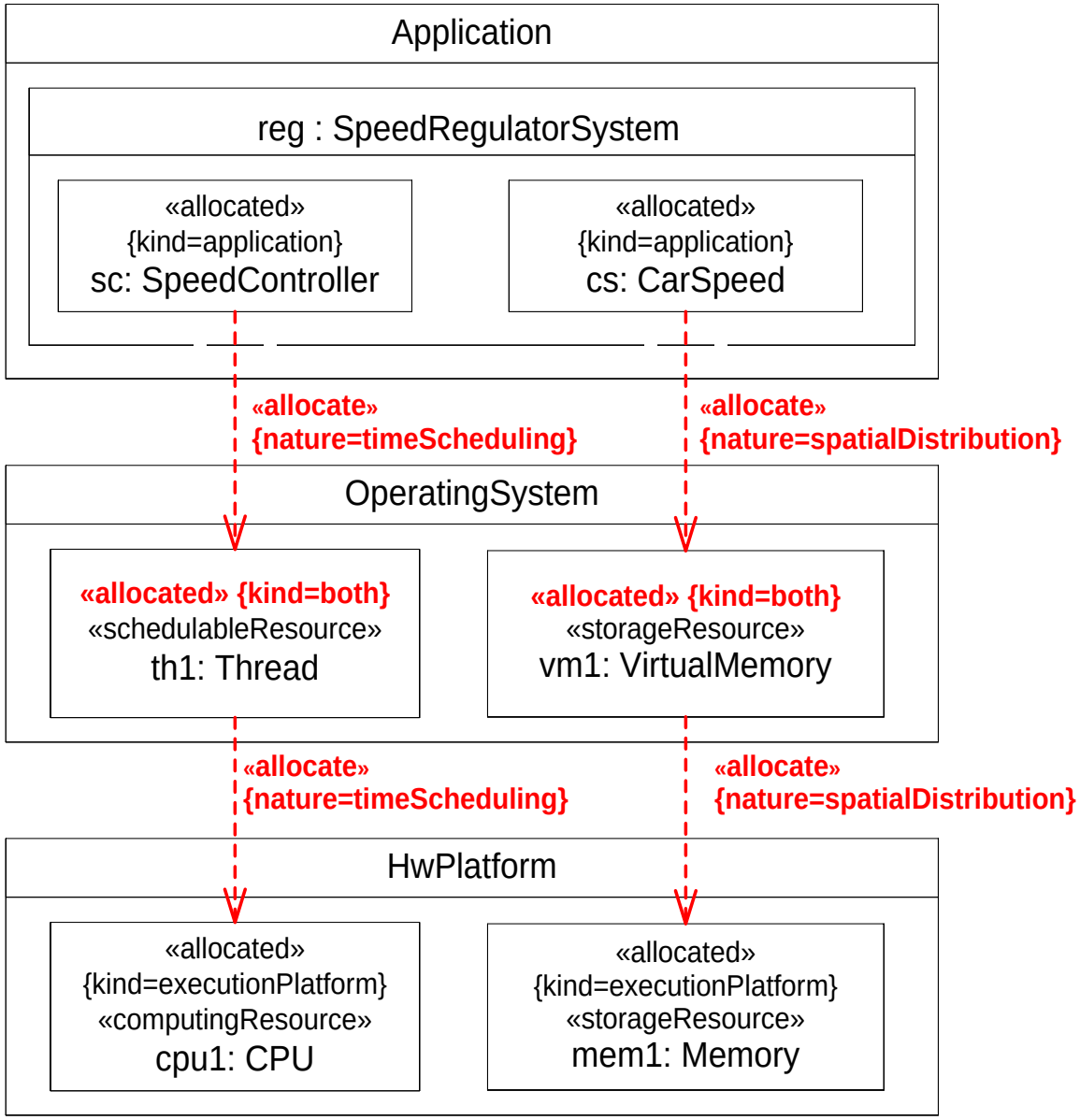


STEP1 - IDENTIFICATION OF ALLOCATION ROLES



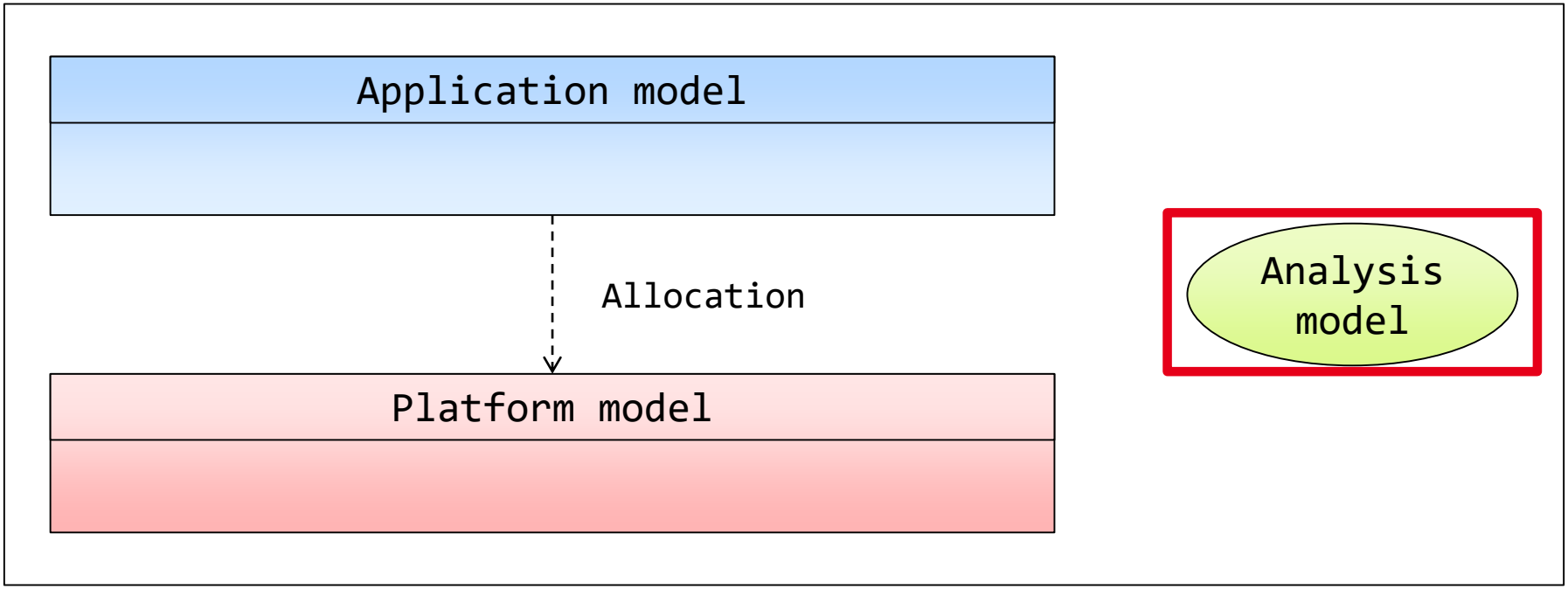
STEP 2 – DENOTE ALLOCATIONS

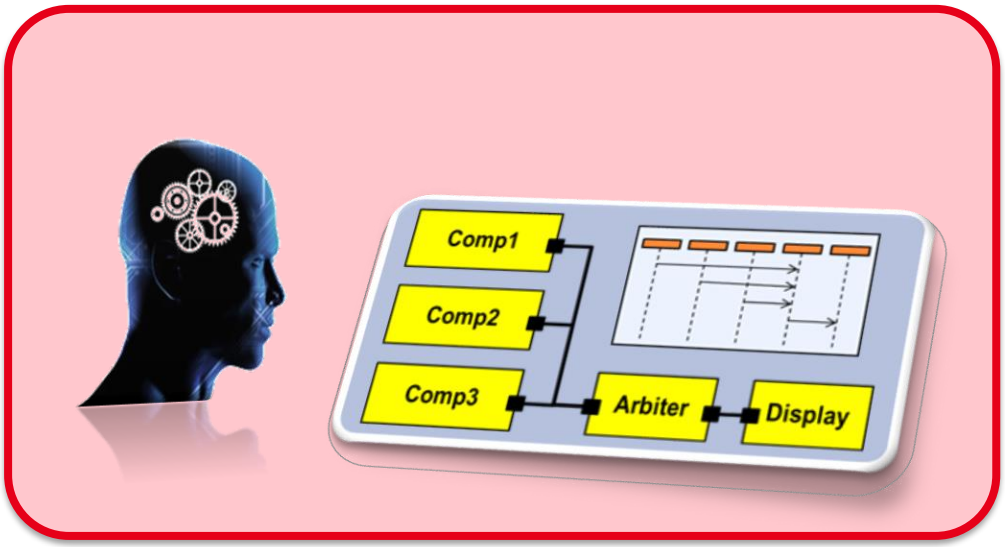




CONTENT

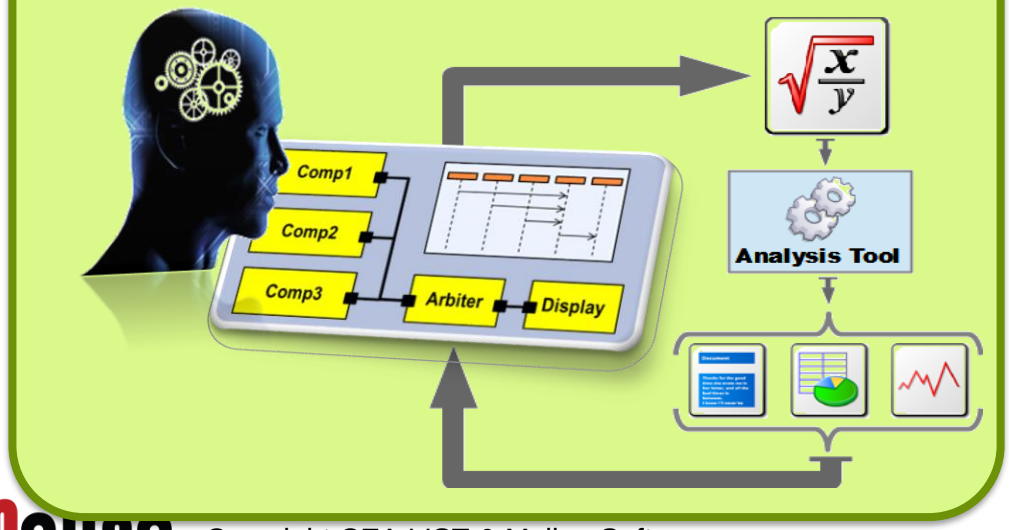
Modeling approach



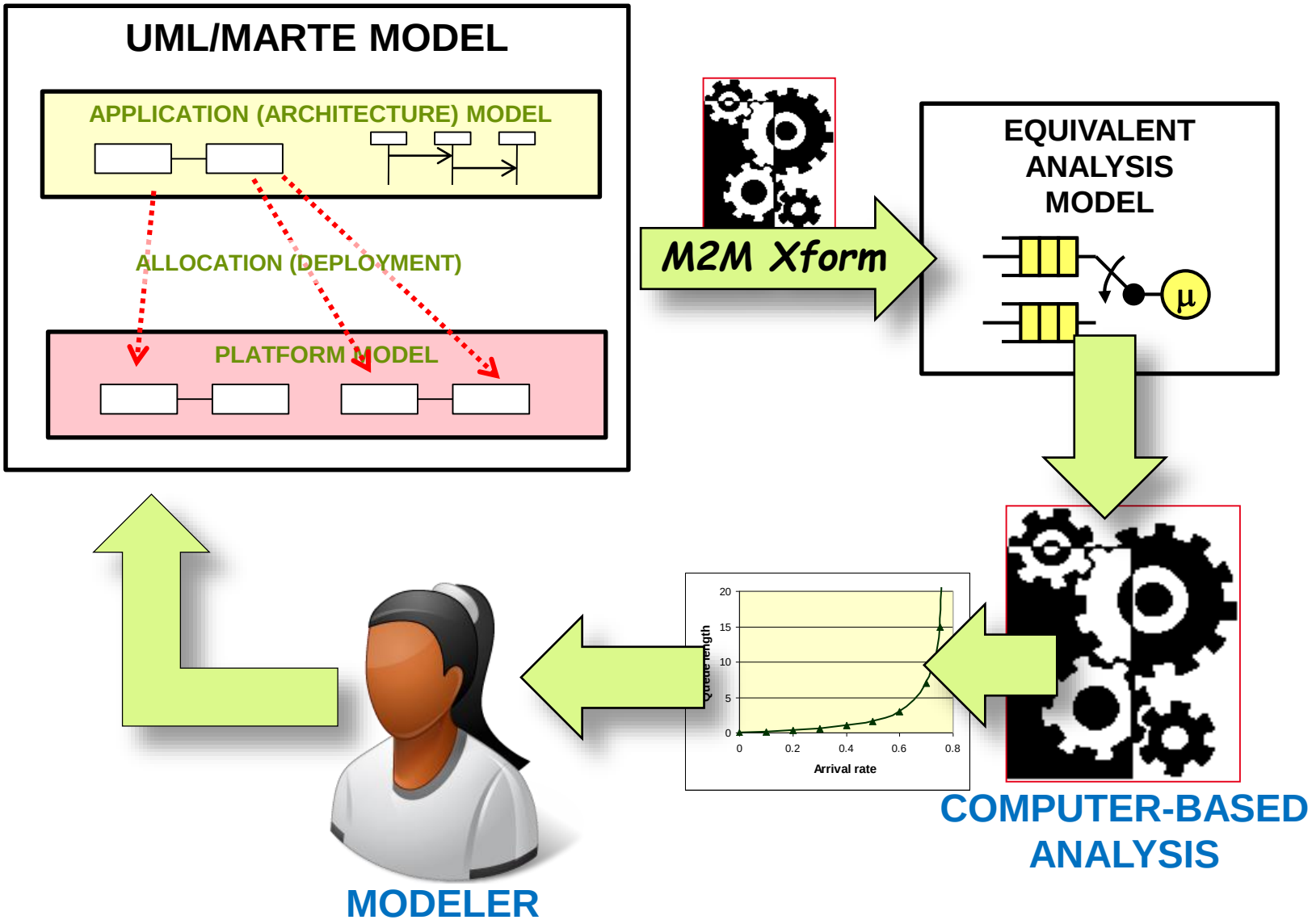


High-expertise
Unreliable
Not-scalable

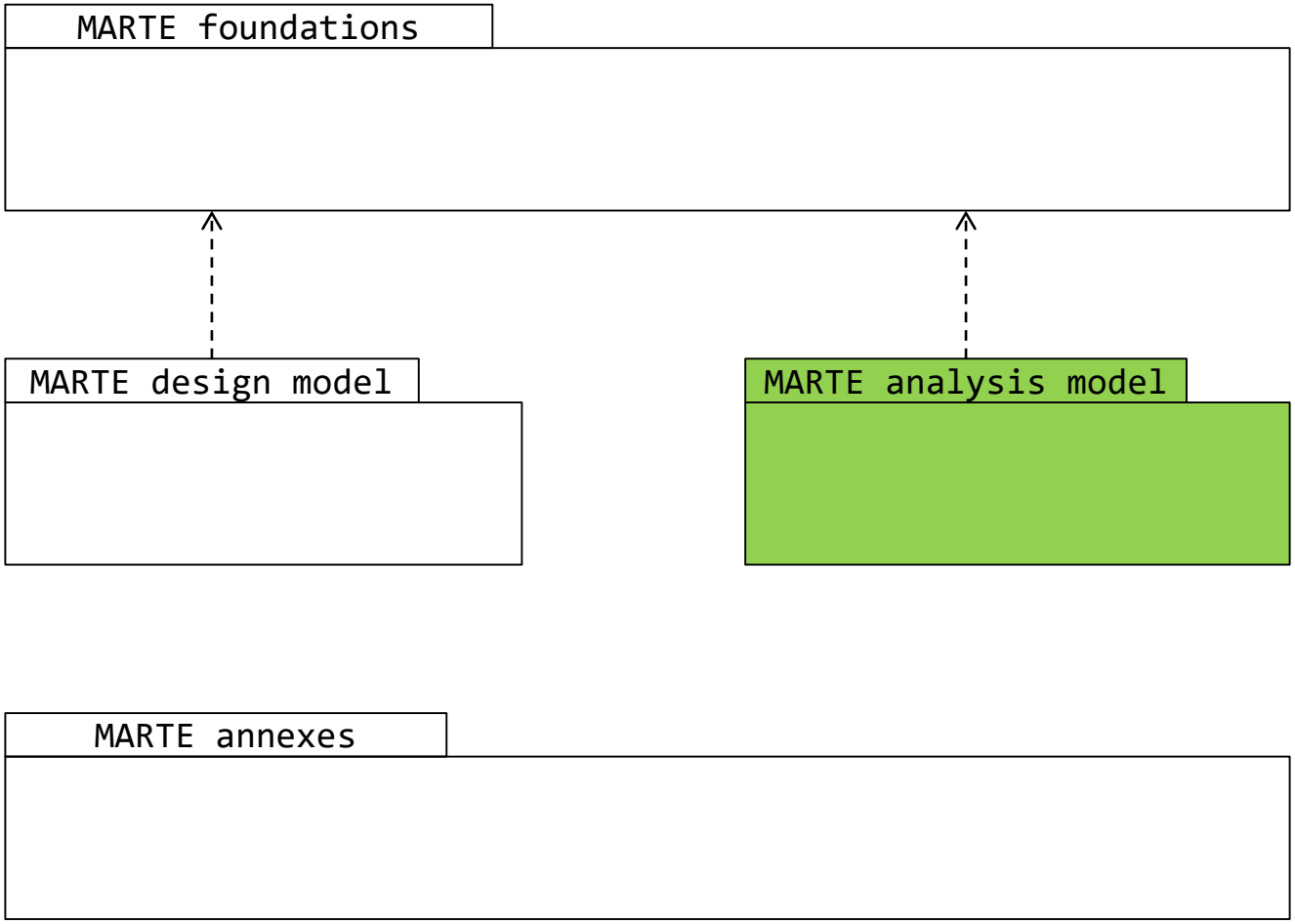
Computer-aided model analysis



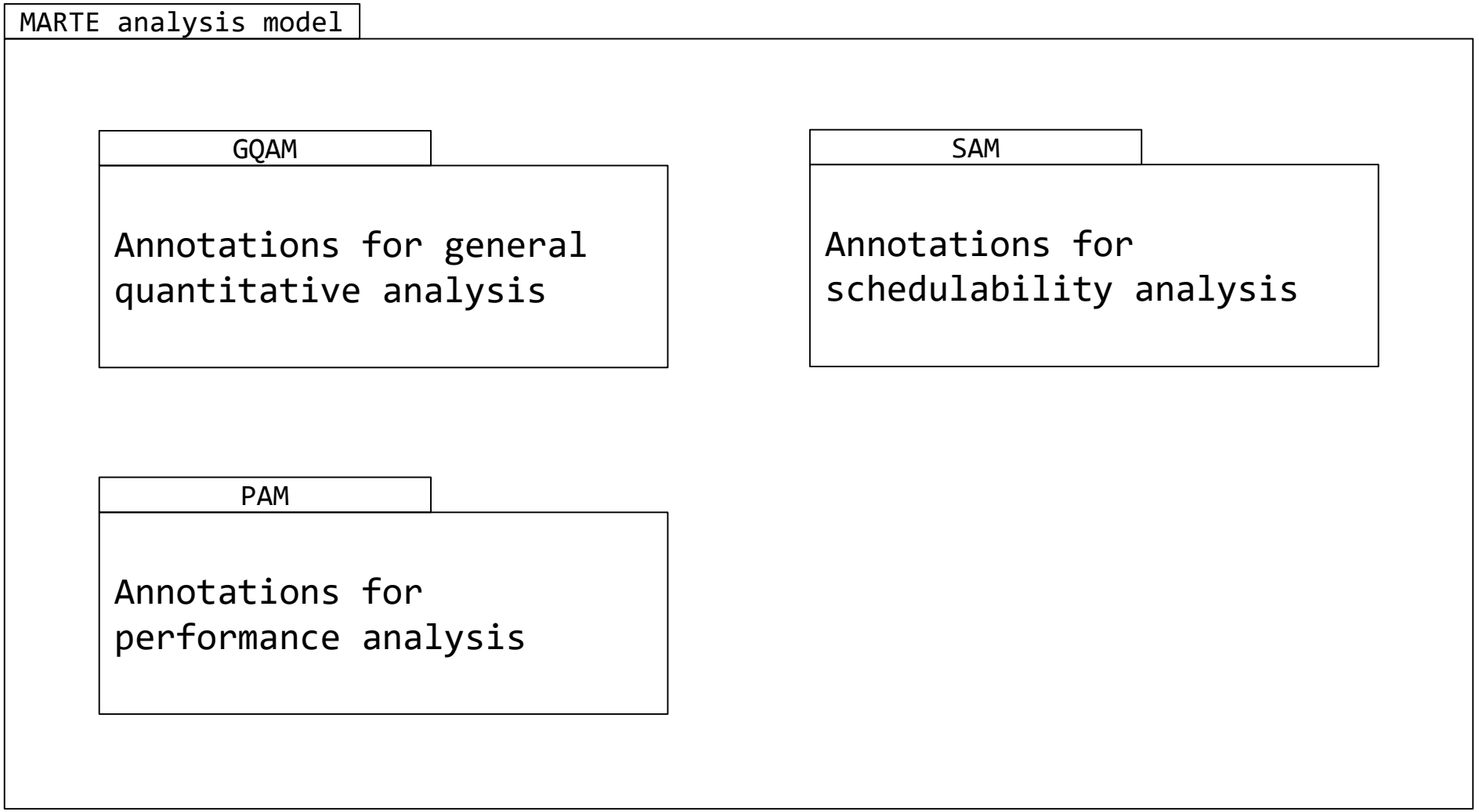
↘ expertise
↗ reliability
↗ scalability



MARTE packages

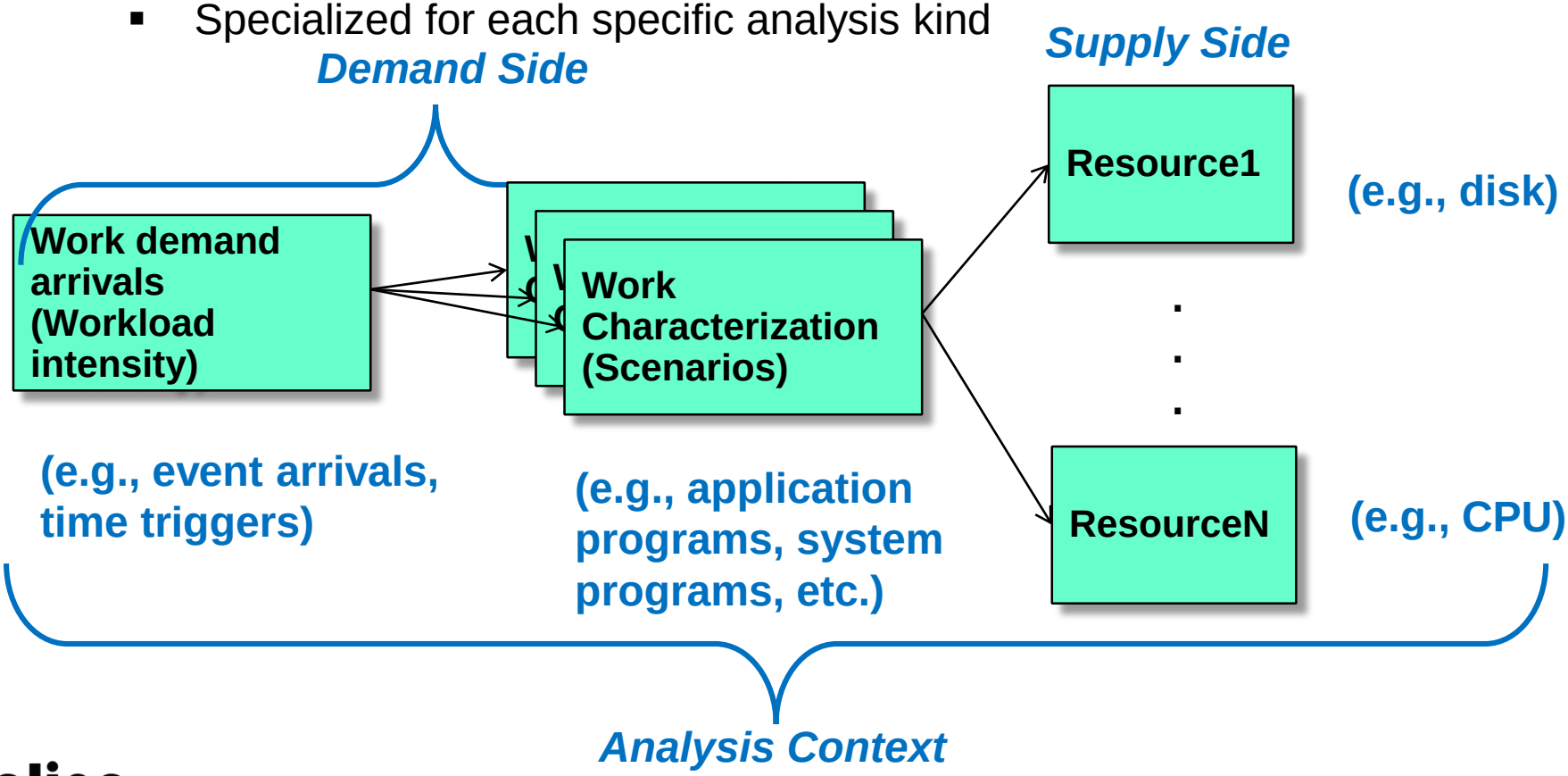


MARTE packages

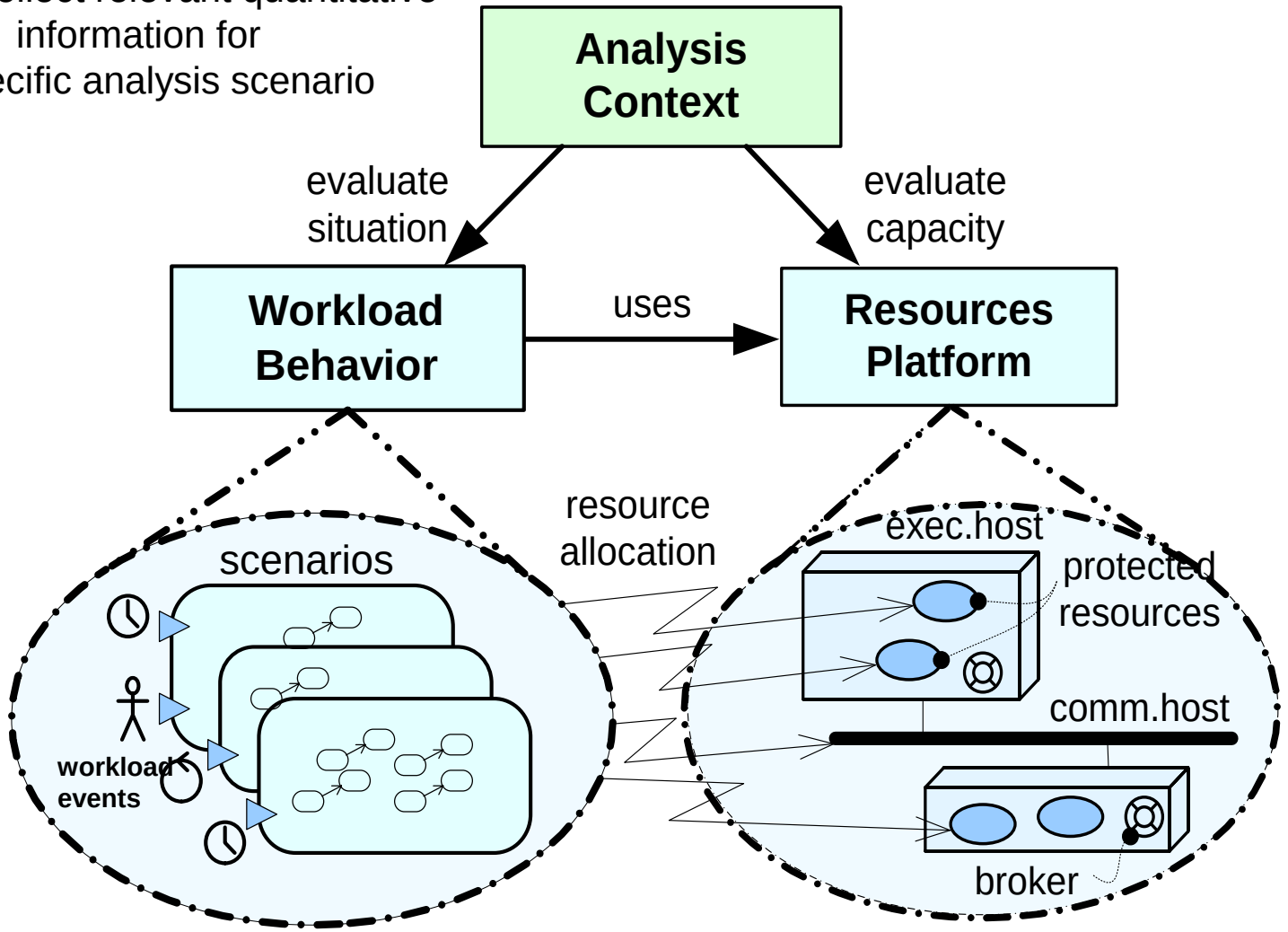


GENERIC QUANTITATIVE ANALYSIS MODEL (GQAM)

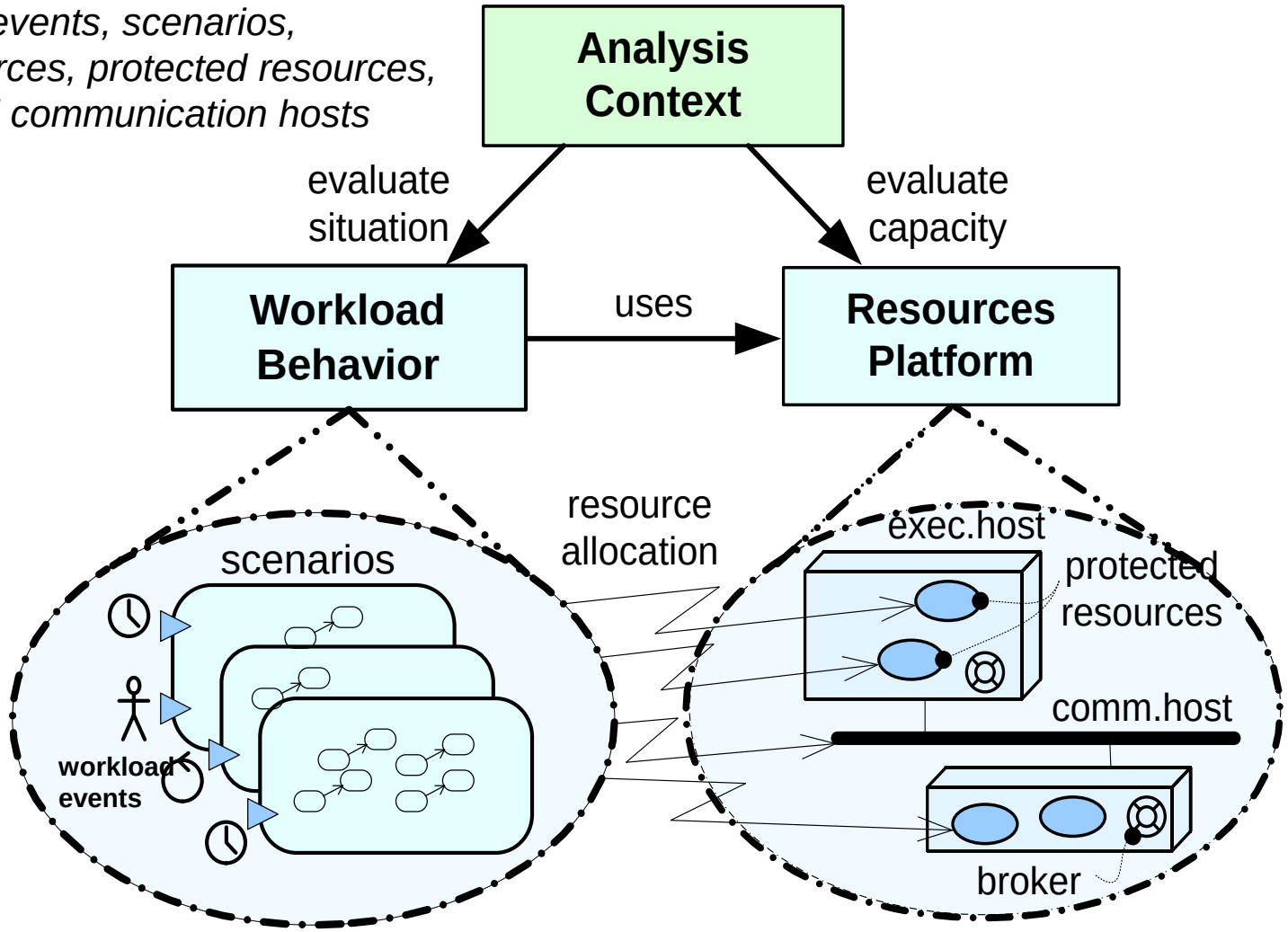
- **Captures the pattern common to many different kinds of quantitative analyses (using concepts from GRM)**
  - Specialized for each specific analysis kind



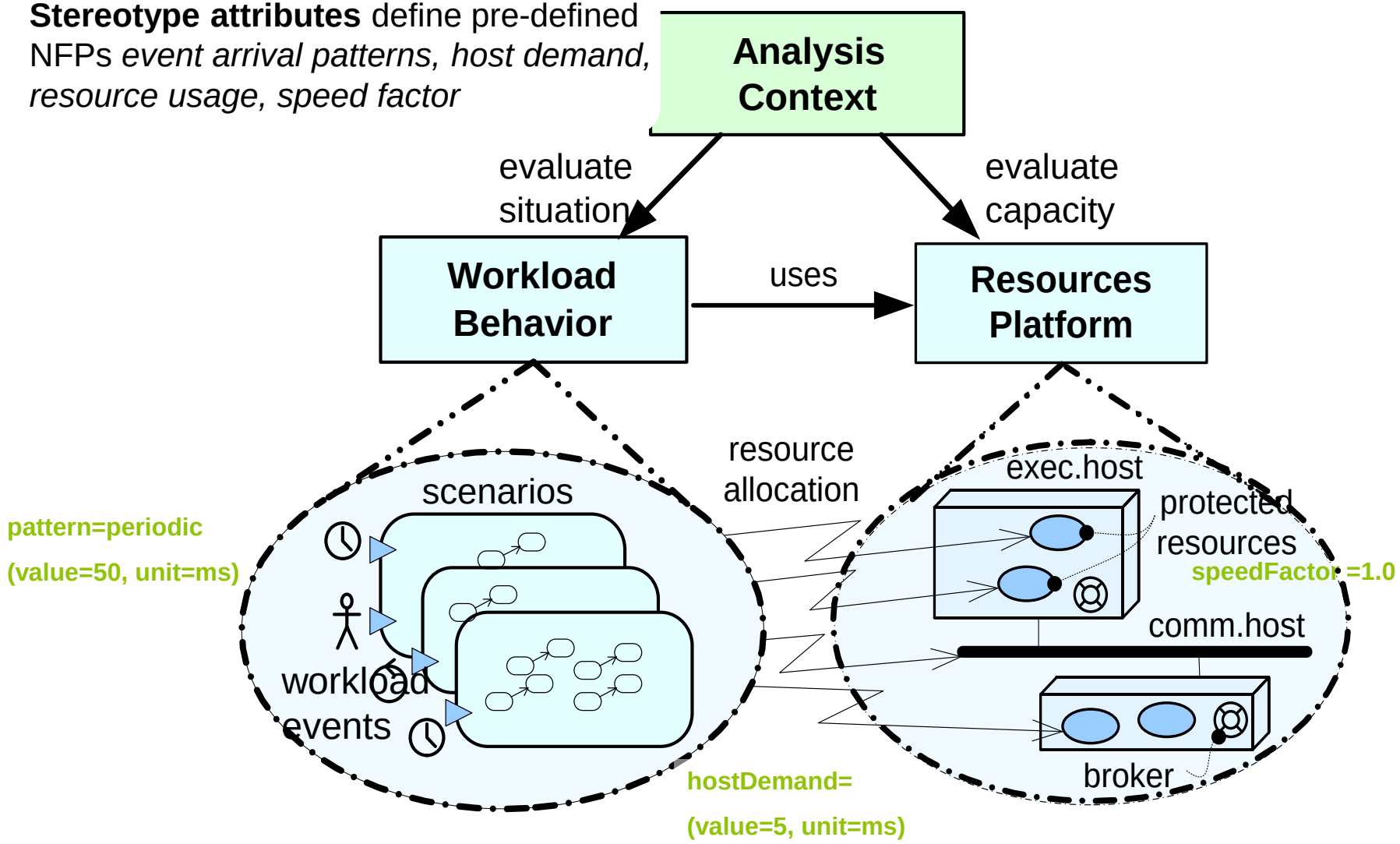
An **analysis context** is the root concept used to collect relevant quantitative information for a specific analysis scenario



Stereotypes define “analysis” abstractions
workload events, scenarios, schedulable resources, protected resources, execution and communication hosts

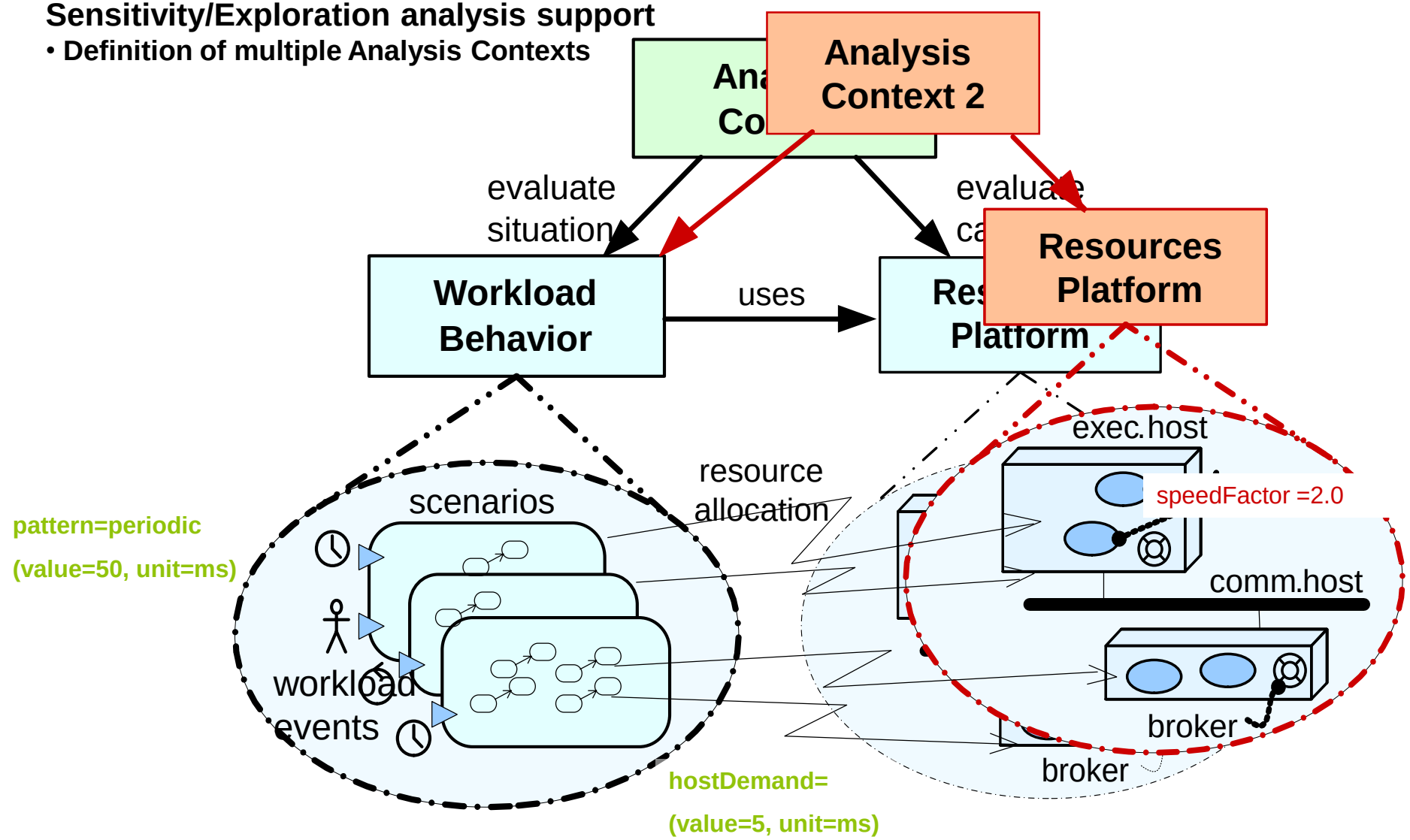


Stereotype attributes define pre-defined NFPs *event arrival patterns, host demand, resource usage, speed factor*



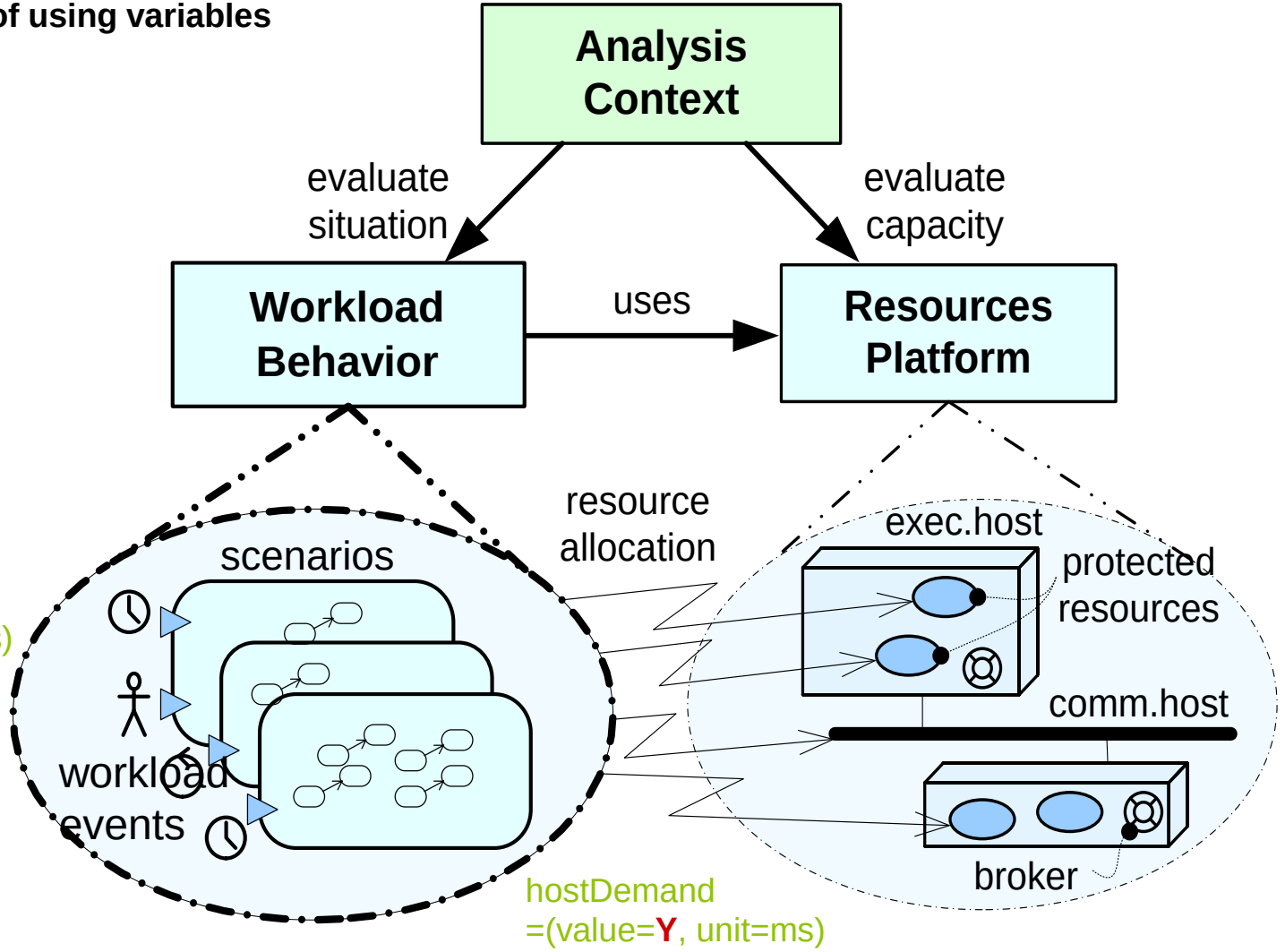
Sensitivity/Exploration analysis support

- **Definition of multiple Analysis Contexts**



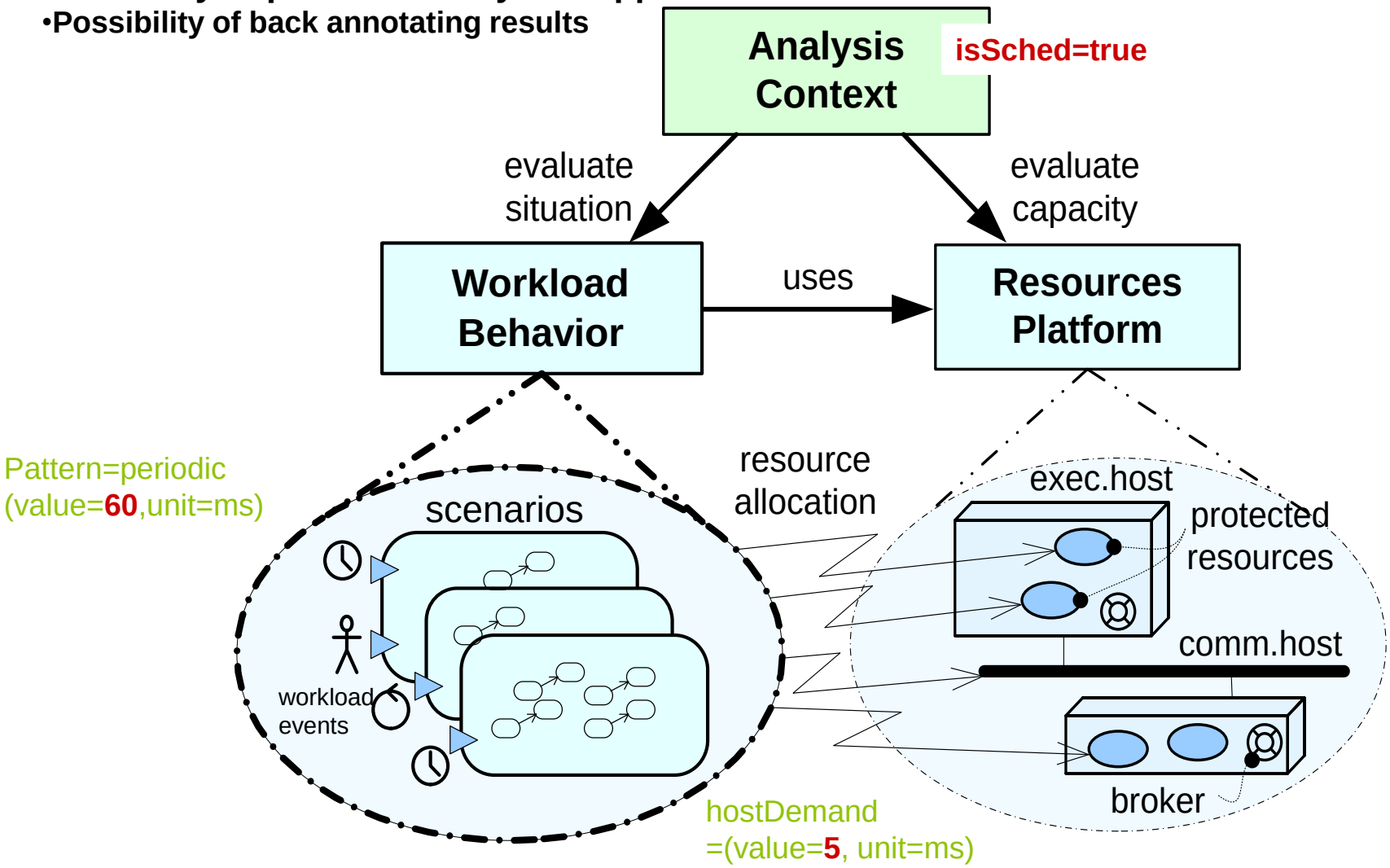
Sensitivity/Exploration analysis support

- **Possibility of using variables**



Sensitivity/Exploration analysis support

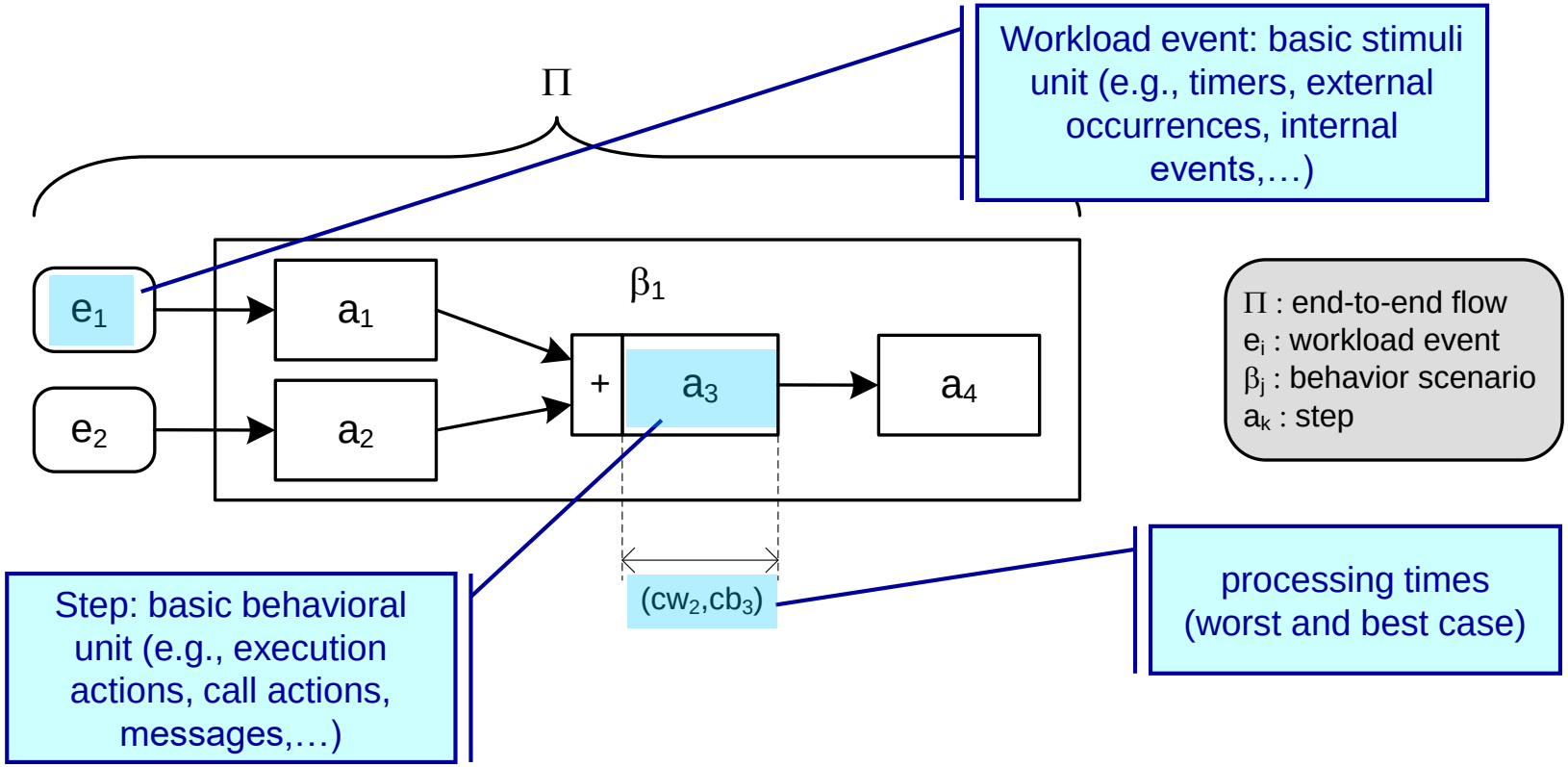
- **Possibility of back annotating results**



CONCEPT OF END-TO-END FLOW

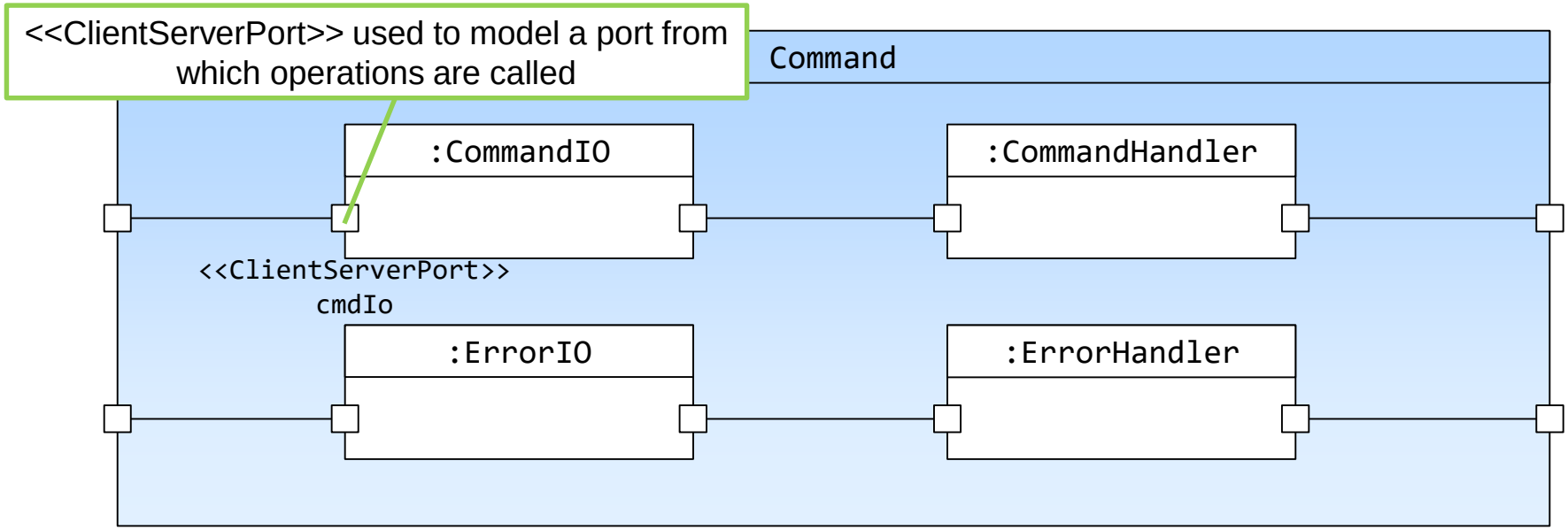
An "End-To-End Flow" is the basic workload unit to evaluate

→ An end-to-end flow refers to the entire causal set of steps triggered by one or more external workload events.

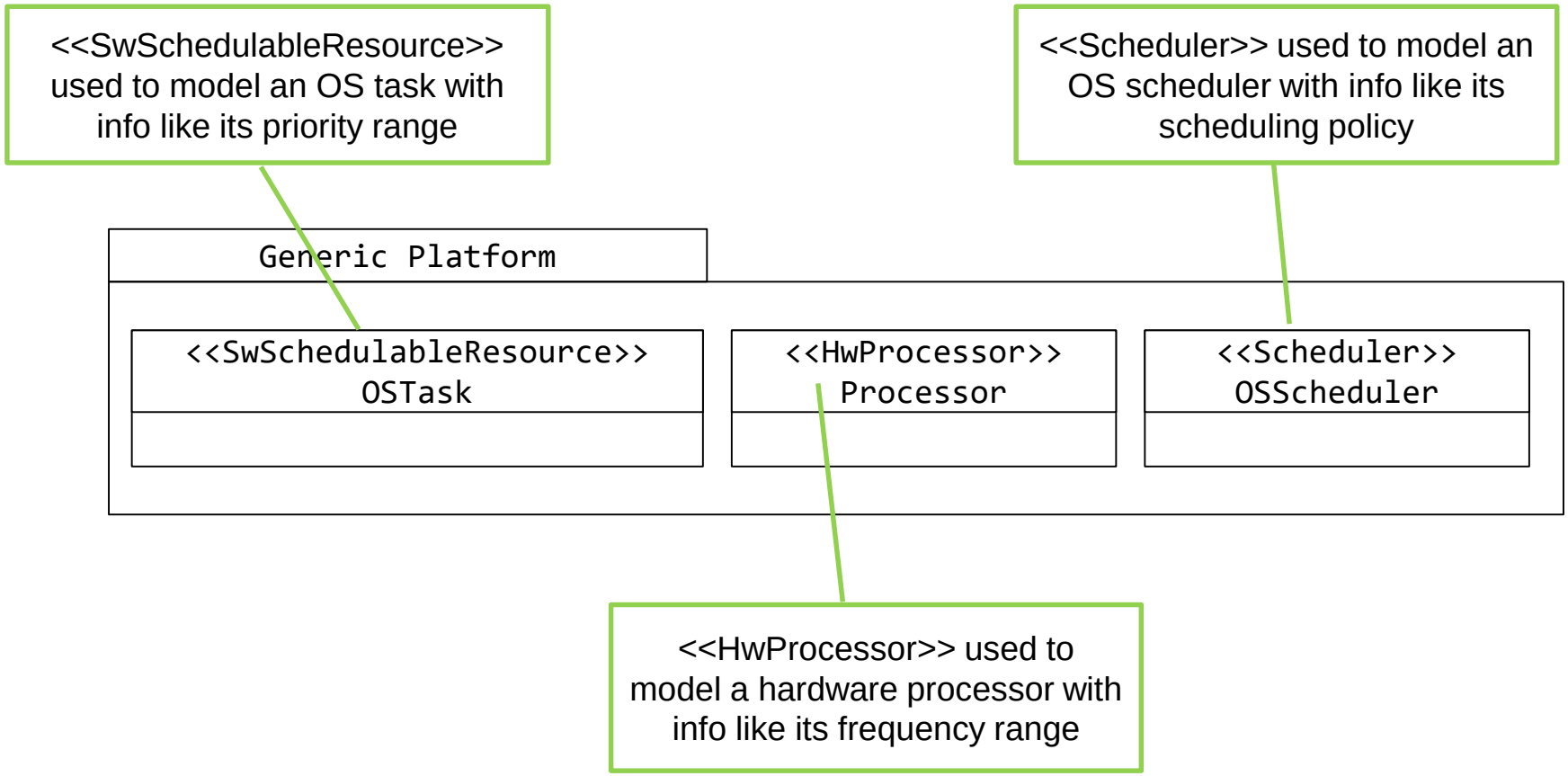


CONTENT

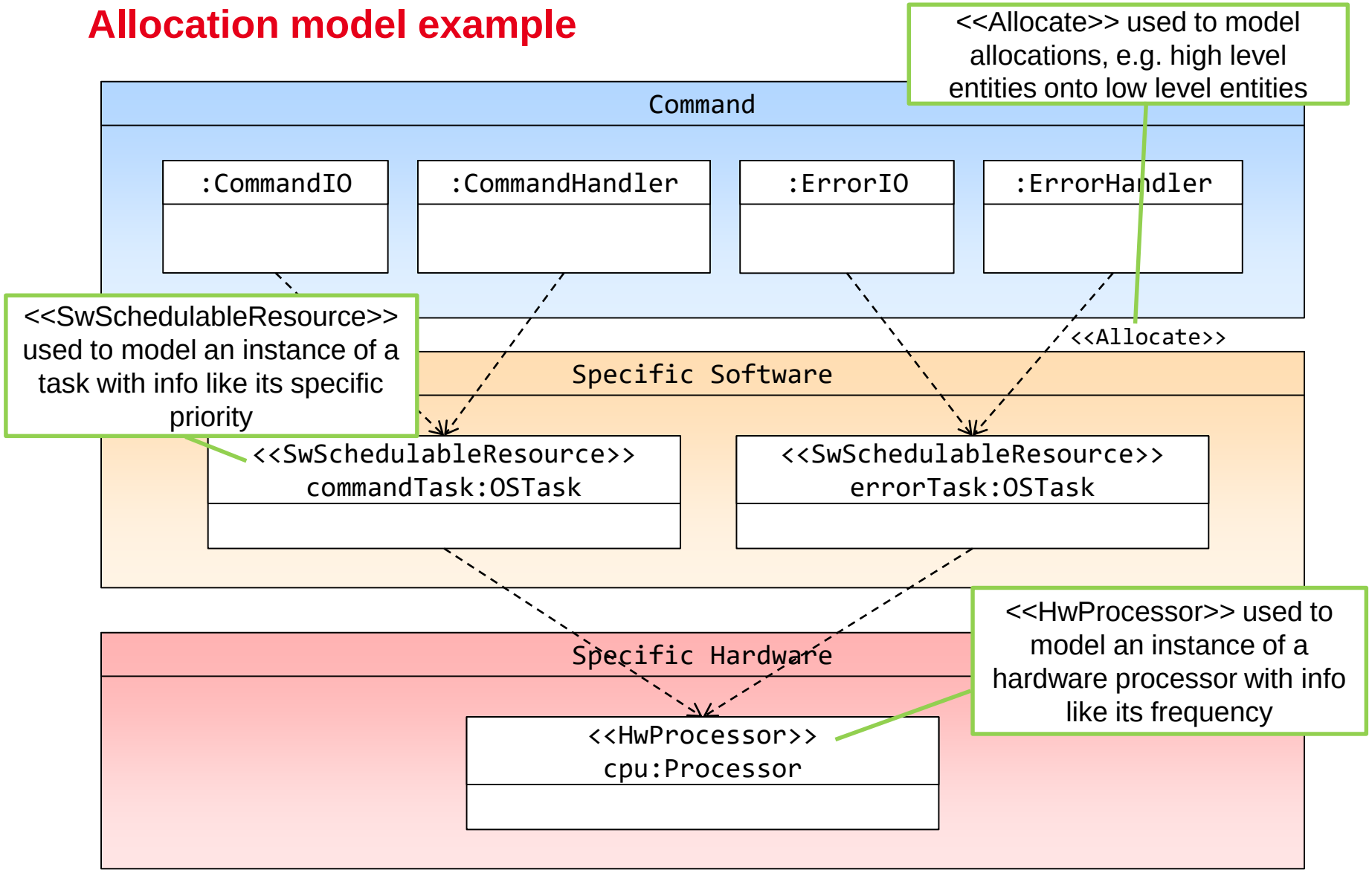
Application model example



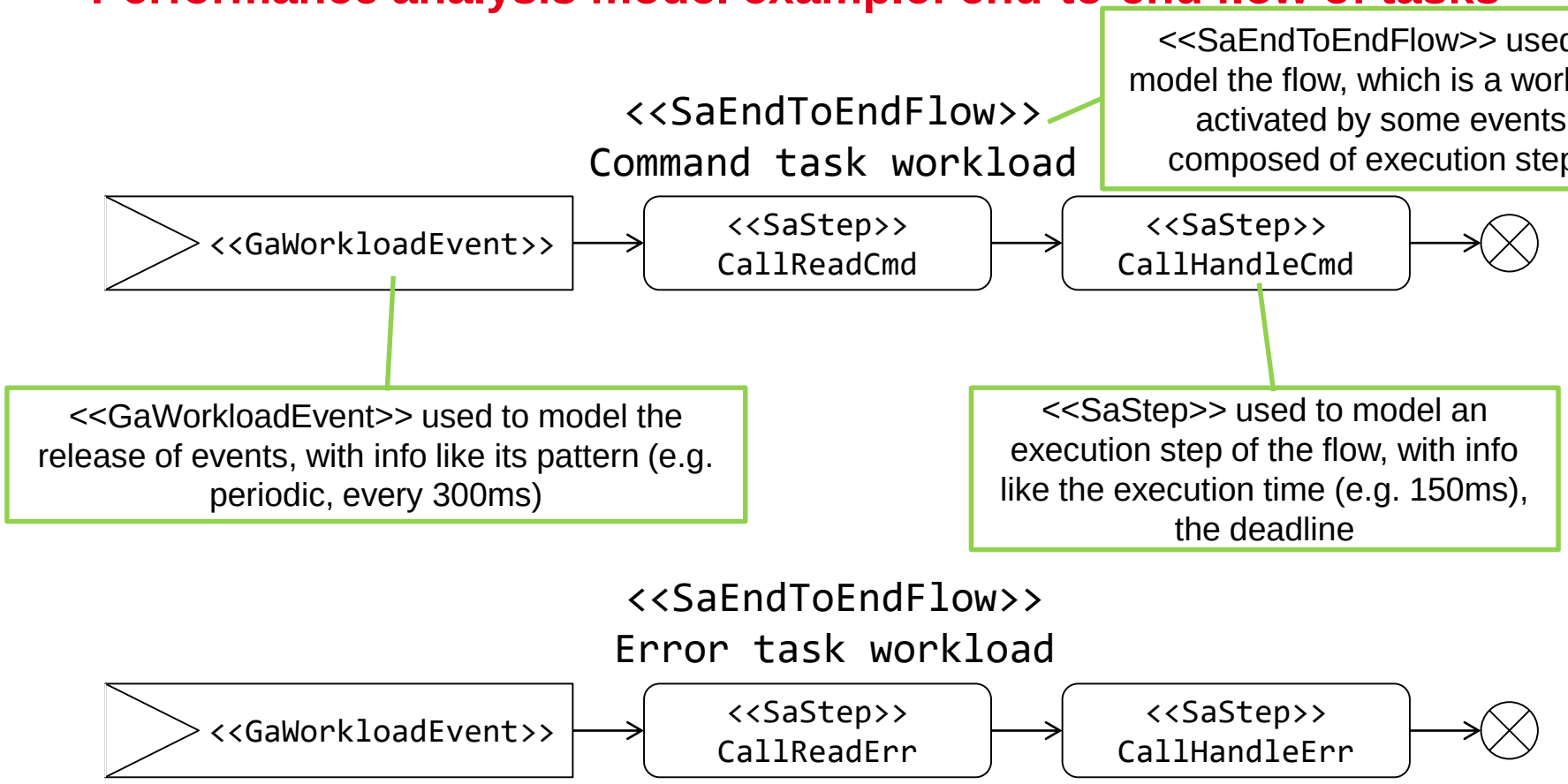
Platform model example



Allocation model example



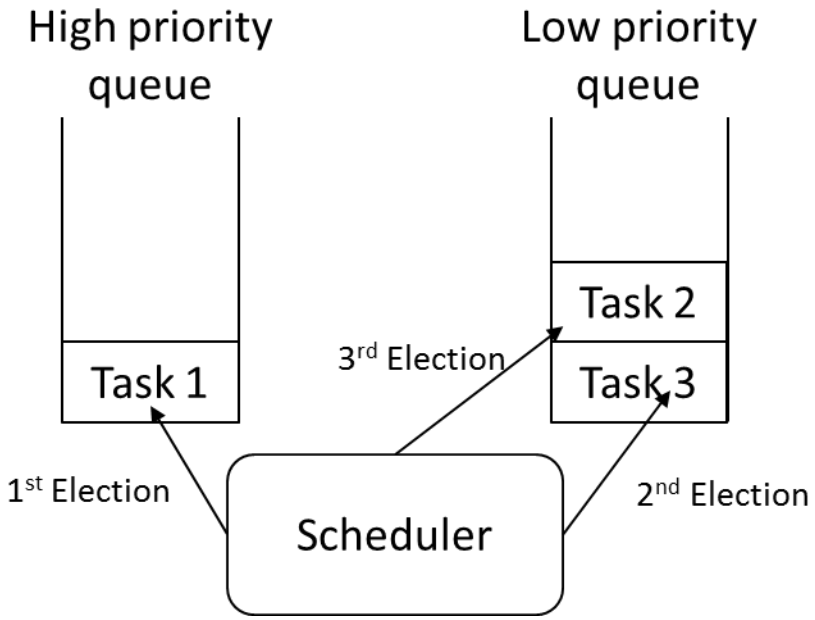
Performance analysis model example: end-to-end flow of tasks

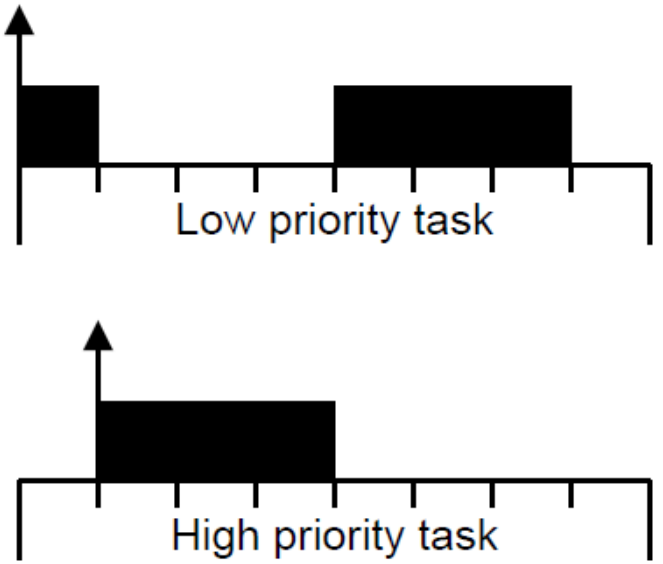


CONTENT

Tasks scheduling in RTES

- RTES (usually) implemented as a multi-tasking system; tasks are concurrent and must be scheduled
- Scheduling: method by which tasks are given access to processors, i.e. according to a scheduling policy; scheduling is performed by a scheduler in the OS

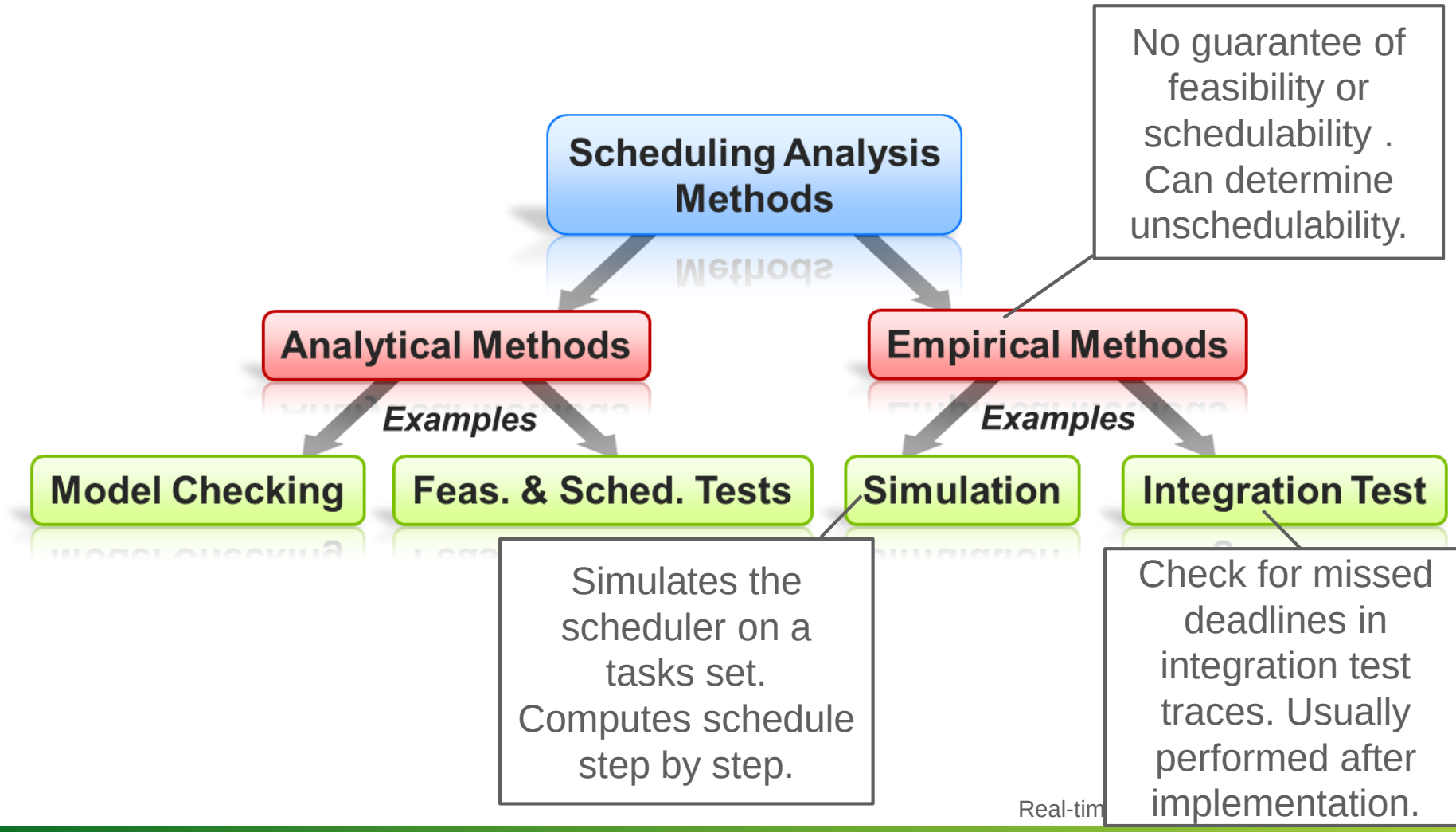




Scheduling analysis

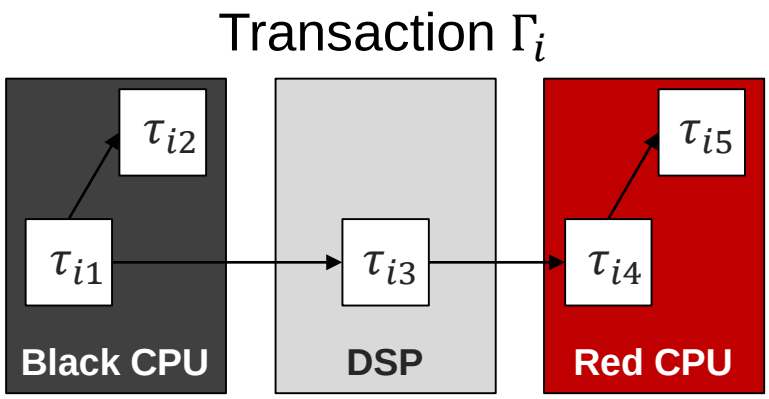
- Scheduling analysis either determines the feasibility or schedulability of a system.
- Schedulable: a task set is said schedulable if they are guaranteed to respect their deadline.
- Feasibility: for a given task set executing on a given set of processors, the tasks set is said to be feasible if the tasks set is schedulable on the processors by at least one existing scheduling policy.
- Schedulability: for a given task set executing on a given set of processors, and a given scheduling policy, schedulability is the assessment that the tasks set is schedulable by the scheduling policy on the processors.

Scheduling analysis



Offset and jitter-based sporadic task model in a transaction

- **Transaction: "Group of [sporadic] tasks, related through collectively performed functionality or timing attribute [like offset and jitter]" [Tindell94]**
- **Tasks related by precedence dependency [Palencia99], modeled as offset and jitter**



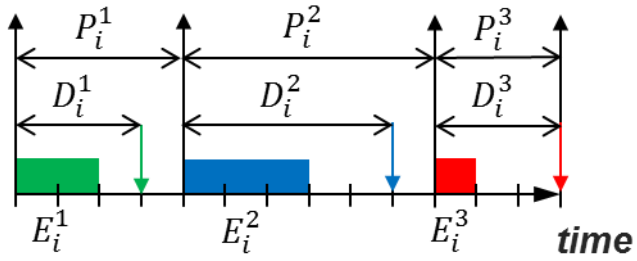
Task τ_{ij}

- C_{ij}: *WCET*
- D_{ij}: *Deadline*
- O_{ij}: *Offset*
- J_{ij}: *Jitter*
- B_{ij}: *WCBT*
- T_{ij}: *Period*
- $prio(\tau_{ij})$: *Priority*
- $proc(\tau_{ij})$: *Processor*

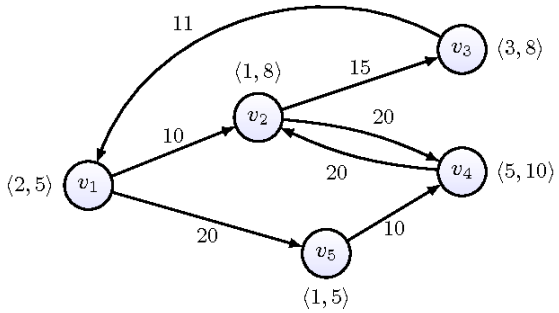
- **Partitioned multi-processor system (no task migration)**
- **Shared resources modeled as blocking time**

Offset and jitter-based sporadic task model in a transaction

- Many task models have been proposed to model jobs of a task. Motivation is to reduce pessimism when job behavior is known.
- General Multiframe task model



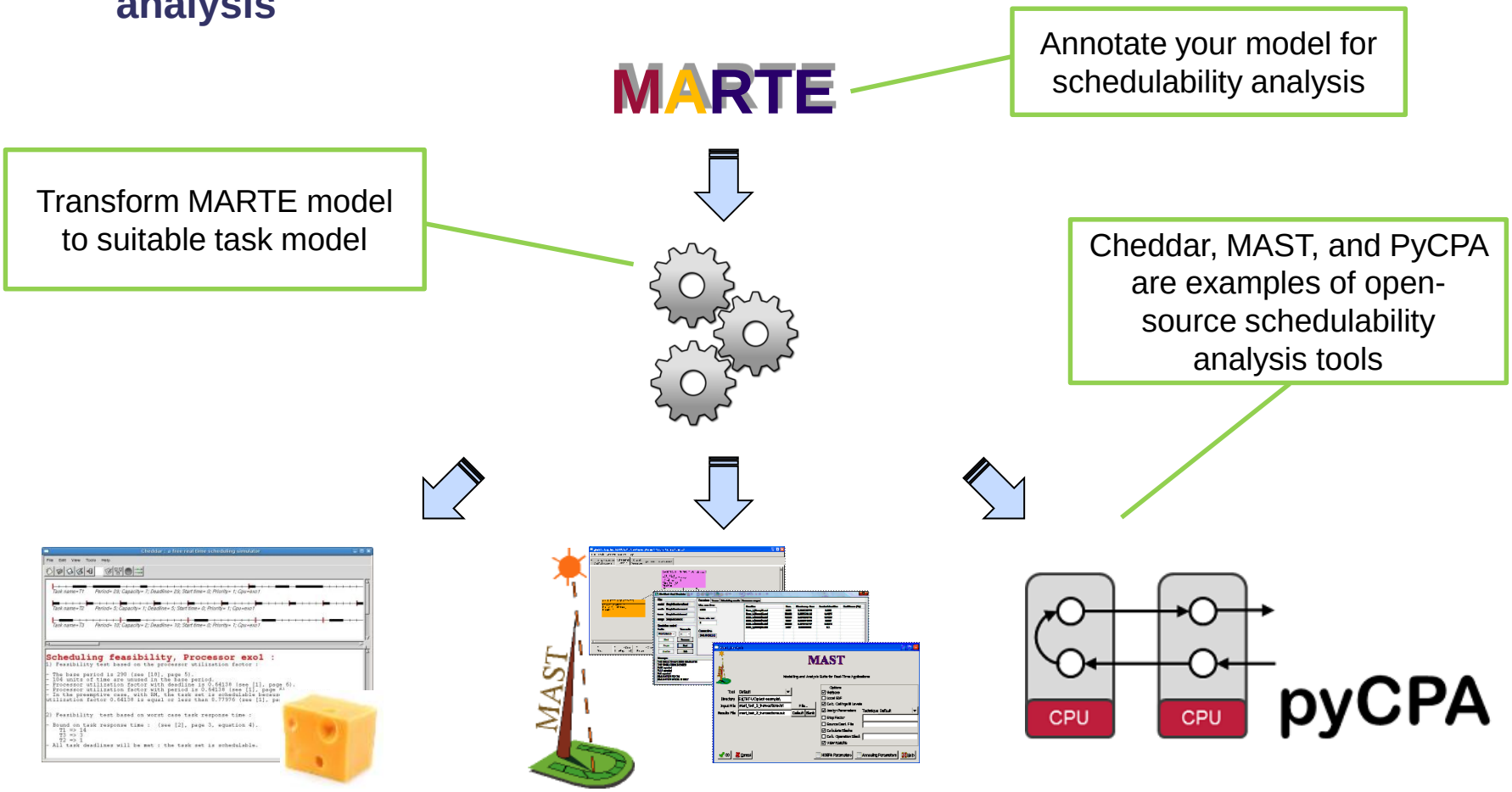
- Sporadic digraph task model



- Works have shown that such task models can be reduced to very low level sporadic task models, where tasks represent jobs

Schedulability analysis using MARTE

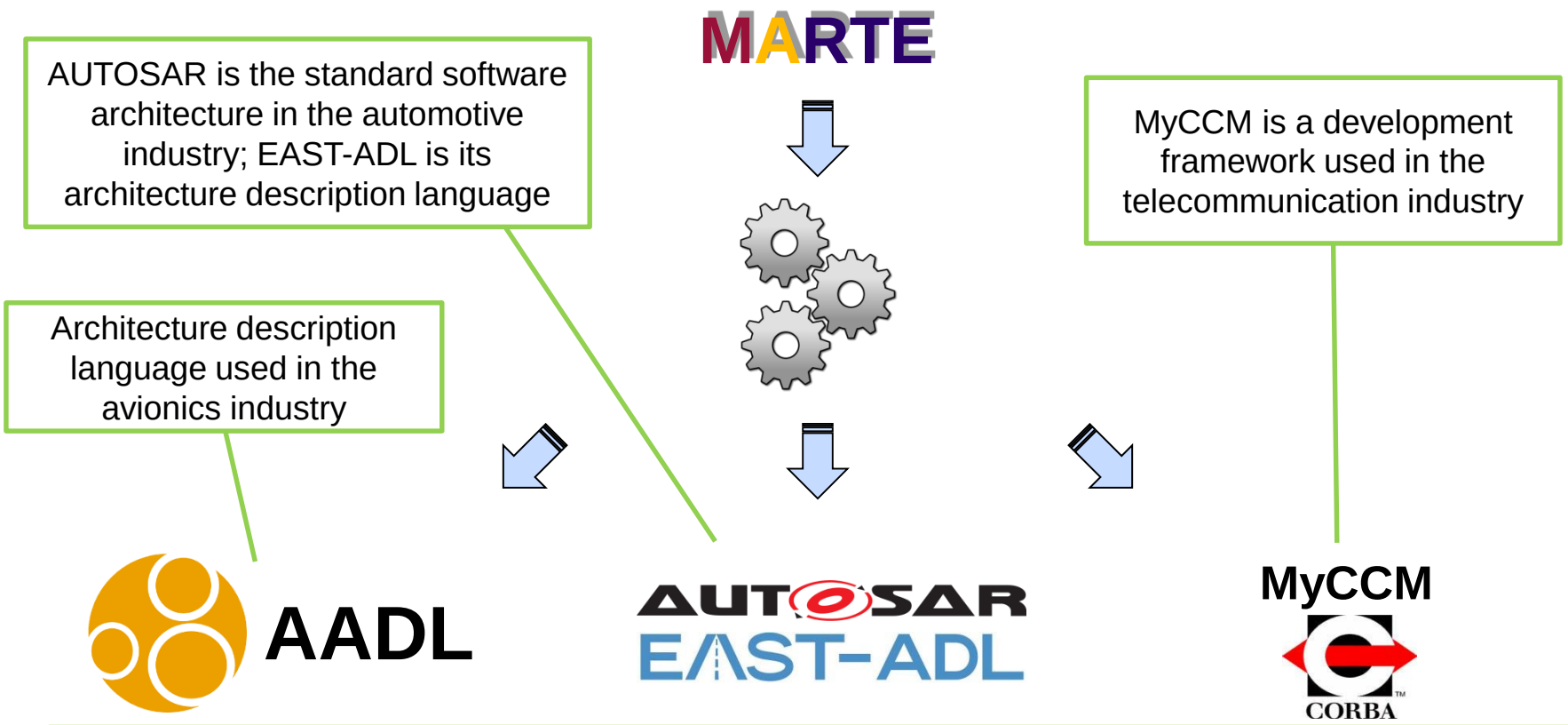
- MARTE is used to model the high level architecture model
- MARTE model is transformed to low level task models for the analysis



CONTENT

MARTE and the industry

- MARTE proposed by considering several industrial RTES domains



Modeling with MARTE depends on the development methodology, i.e., it can be used as a framework to build a DSML suitable to your company.

Tools

ProMARTE Team

Industrials
- Alcatel
- Lockheed Martin
- THALES
- Orange

Academics
- Carleton University
- CEA LIST
- ESEO
- ENSIETA
- INRIA
- INSA Lyon
- Software Engineering Institute (CMU)
- Universidad de Cantabria

Modeling tool vendors
- Atego
- IBM
- Mentor Graphical Corporation
- Softeam
- Tri-Pacific Software
- No Magic
- MathWorks

CONTENT

Summary

- MARTE proposed as a standard modeling language for RTES domain, based on several industrial systems and standards
- Adds ability to specify typical RTES concepts and quantitative information (e.g., QoS)
- Target both design and analysis of RTES
- MARTE implemented as a UML profile
- MARTE models can be analyzed through model transformation to analysis tool models
- Extensible and intended to be specialized further as a DSML for sub-domains of RTES (e.g., IoT Modeling Language)

Further reading

- B. Selic, S. Gérard, "*Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE: Developing Cyber-Physical Systems*", 1st edition, Burlington US-MA: Morgan Kaufmann, 2013