UML MARTE

www.omgmarte.org

13th SDL Forum, Sep 2007, Paris

MARTE Tutorial

An OMG UML profile to develop Real-Time and Embedded systems

Sébastien Demathieu

(sebastien.demathieu@thalesgroup.com)

Thales Research & Technology

cea list

THALES

INRIA

www.omgmarte.org

Acknowledgment

- This presentation reuses and extends material prepared by the ProMARTE partners for the OMG RTESS PTF meeting in San Diego, on March 28th 2007

- The initial presentation (realtime/07-03-14) is available to OMG members

Modeling Real-Time and Embedded systems in UML

- **UML is emerging as a possible solution to address the Real-Time and Embedded domain**
  - A large audience in the Software Engineering community
  - Steady semantic foundations
  - Extension capabilities through UML profiles (e.g. SysML)
  - But lacks key notions to fully address RTE specifics (time, resource, scheduling)
- **Previous attempts to adapt UML to the RTE domain**
  - Academic initiatives (e.g. ACCORD, GASPARD)
  - Commerical Tools: ARTiSAN, ROSE RT (ROOM), Rhapsody (Real-Time UML)
  - UML profile for Scheduling, Performance and Time (SPT)
    - The first OMG adopted specification in this domain
    - Defines annotation mechanisms to perform quantitative analysis
    - Required major improvements over time

In 2005, OMG called for a new UML profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE)

Introducing MARTE

- **"The UML profile for MARTE addresses modeling and analysis of real-time and embedded systems, including their software and hardware aspects"**

- **Key features**
  - Provides support for non-functional property modeling
  - Adds rich time and resource models to UML
  - Defines concepts for software and hardware platform modeling
  - Defines concepts for allocation of applications on platforms
  - Provides support for quantitative analysis (e.g. scheduling, performance)
  - Complies with UML 2.1 and other existing OMG standards
  - Replaces the UML SPT profile 1.1

- **MARTE specification adopted in June 2007**
  - Alpha document available: http://www.omg.org/cgi-bin/doc?ptc/2007-08-04
  - Finalization Task Force comment deadline: December 22nd 2007

The ProMARTE partners

Tool vendors

- ARTiSAN Software Tools*
- International Business Machines*
- Mentor Graphics Corporation*
- Softeam*
- Telelogic AB (I-Logix*)
- Tri-Pacific Software
- No Magic
- The Mathworks

Industrial companies

- Alcatel*
- France Telecom
- Lockheed Martin*
- Thales*

Academics

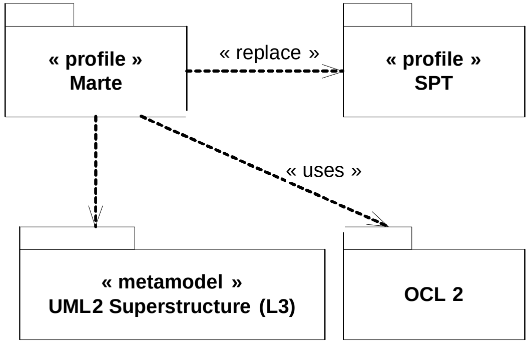
- Carleton University
- Commissariat à l'Energie Atomique
- ESEO
- ENSIETA
- INRIA
- INSA from Lyon
- Software Engineering Institute (Carnegie Mellon University)
- Universidad de Cantabria

* Submitters to the OMG UML for MARTE RFP

www.omgmarte.org

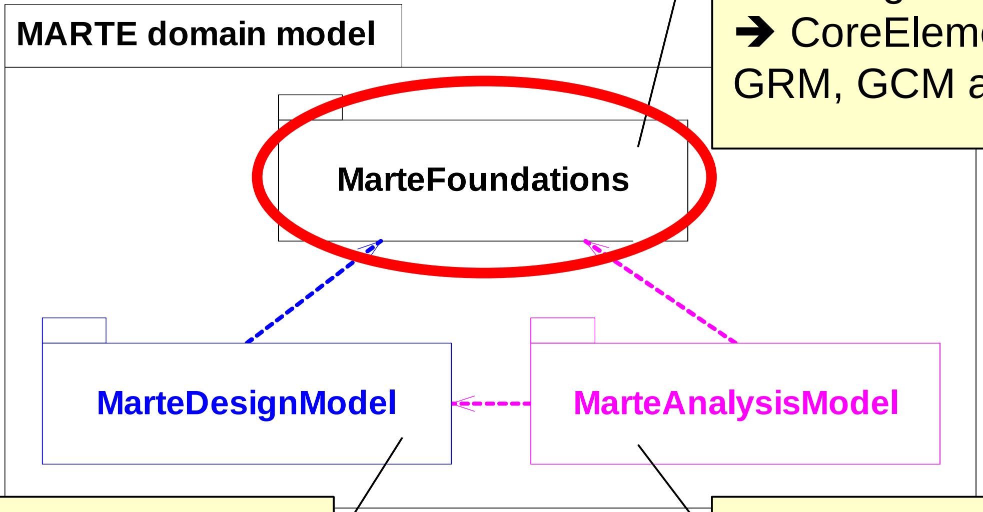
Relationships with other OMG standards

- **Relationships with generic OMG standards**
  - Profiles the UML 2 superstructure meta-model
  - Uses OCL 2 for description of domain constraints



- **Relationships with RTE specific OMG standards**
  - The UML profile for Modeling QoS and FT Characteristics and Mechanisms
    - Addressed through MARTE NFP package
  - The UML profile for SoC
    - More specific than MARTE purpose
  - The Real-Time CORBA profile
    - Real-Time CORBA based architecture can be annotated for analysis with MARTE
  - The UML profile for Systems Engineering (SysML)
    - Specialization of SysML allocation concepts and reuse of flow-related concepts
    - Ongoing discussion to include VSL in next SysML version
    - Overlap of team members

Architecture of the MARTE specification



Foundations for RT/E systems modeling and analysis: 6 pkgs
➔ CoreElements, NFP, Time, GRM, GCM and Alloc

Specialization of Foundations for modeling purpose (specification, design…): 3 pkgs
➔ RTEMoCC, SRM and HRM

Specialization of Foundations for annotating model for analysis purpose: 3 pkgs
➔ GQAM, SAM and PAM

Non-Functional Properties (NFP)

- **Formalize a number of ideas existing in SPT and QoS&FT**
  - From the SPT profile
    - e.g. Tag Value Language (variables, math. expressions) and time-related values
  - From the QoS&FT profile
    - e.g. Property Qualifiers
- **Add new modeling constructs required for MARTE**
  - e.g. tuple and choice values, time expressions and unit measurements conversion
- **NFP modeling required general extensions to UML tools**
  - e.g. value expressions editing and data type checking

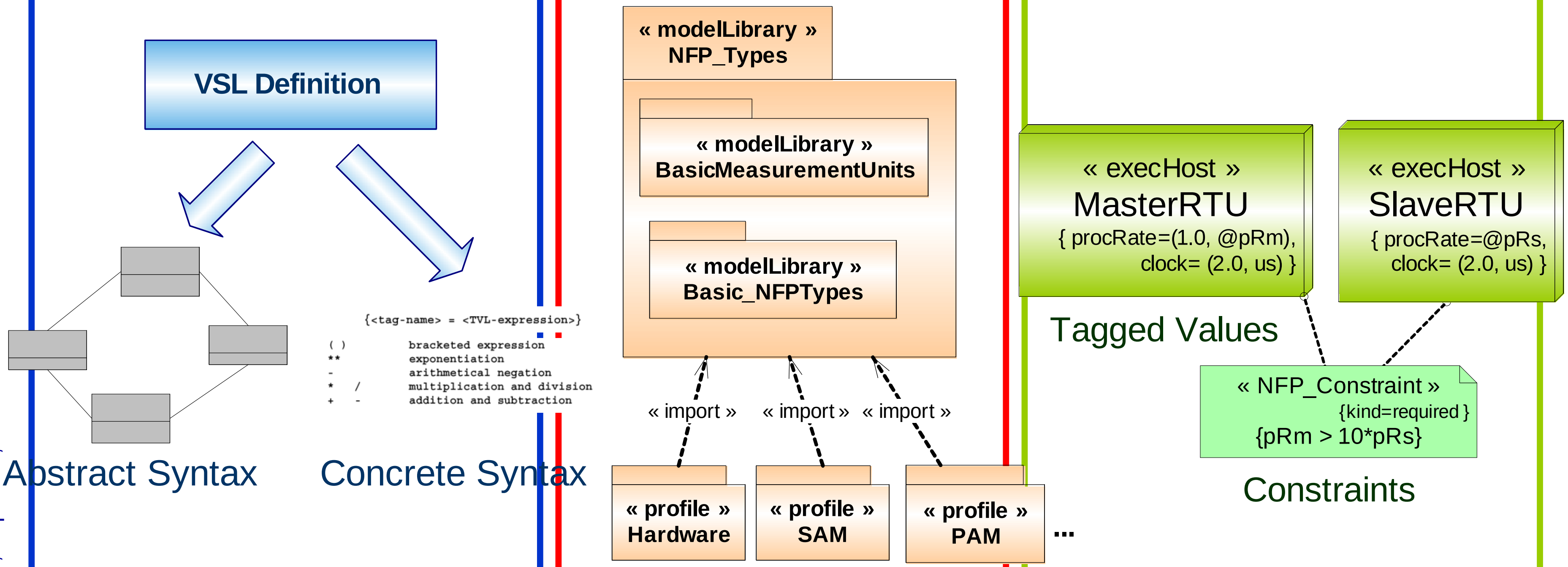
  ➔ This is a key feature in DRES modeling that UML lacks

Organization of NFP constructs

1) Value Spec. Lang. (VSL)

2) Declaration of NFPs

3) Annotation Mechanism



```
{<tag-name> = <TVL-expression>}

( )          bracketed expression
**           exponentiation
-            arithmetical negation
*   /        multiplication and division
+   -        addition and subtraction
```

{ procRate=(1.0, @pRm), clock= (2.0, us) }

{ procRate=@pRs, clock= (2.0, us) }

{kind=required }
{pRm > 10*pRs}

- Exact notation for values: extended Literals, Intervals, Tuples, Choices, Variables, Complex and Time Expressions

- To define, qualify measures (measurement units, source, statistical nature…) and organize NFPs

1. Tagged values
2. Constraints
3. Direct specification in UML models by using NFP types library

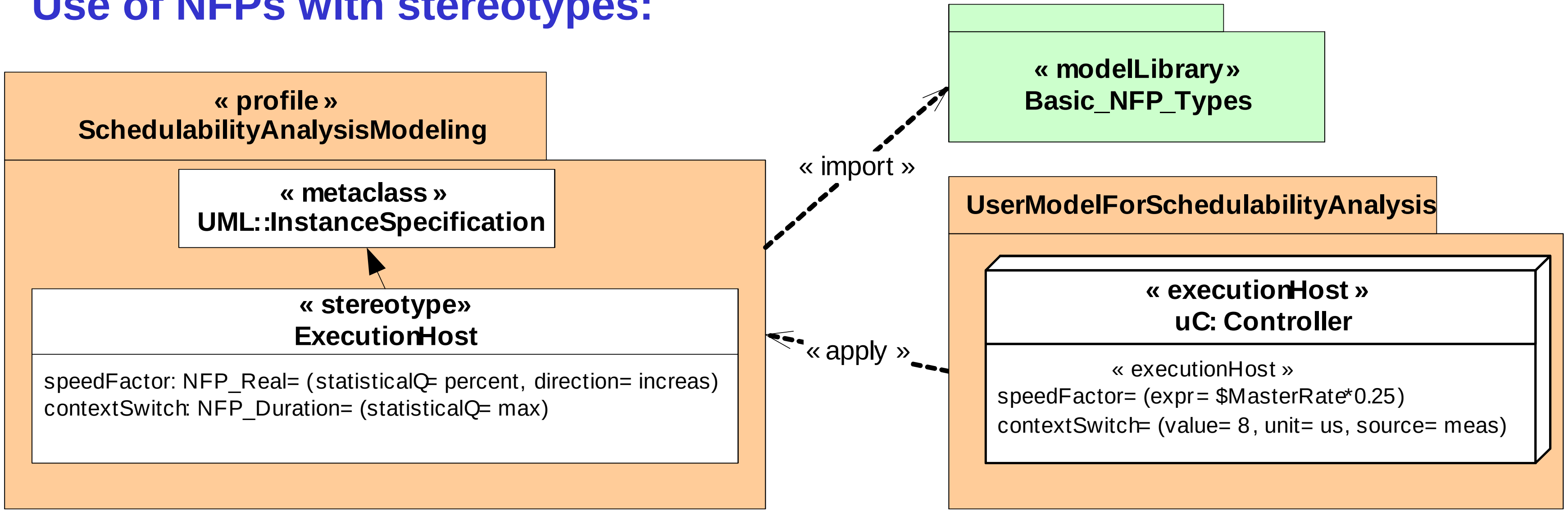
Examples of textual expressions

Value Spec.	Examples
Real Number	1.2E-3 //scientific notation
DateTime	#12/01/06 12:00:00# //calendar date time
Collection	{1, 2, 88, 5, 2} //sequence, bag, ordered set.. {{1,2,3}, {3,2}} //collection of collections
Tuple and choice	(value=2.0, unit= ms) //duration tuple value periodic(period=2.0, jitter=3.3) //arrival pattern
Interval	[1..251[//upper closed interval between integers [@A1..@A2] //interval between variables
Variable declaration & Call	io@var1 //input/output variable declaration var1 //variable call expression.
Arithmetic Operation Call	+(5.0,var1) //”add” operation on Real datatypes 5.0+var1 //infix operator notation
Conditional Expression	(($var1<6.0)?(10^6):1) //if true return 10 exp 6,else 1

+ additional constructs to reference UML properties and time observations

Examples of NFP annotations

Use of NFPs with stereotypes:



« profile »
SchedulabilityAnalysisModeling

« metaclass »
UML::InstanceSpecification

« stereotype»
ExecutionHost

speedFactor: NFP_Real= (statisticalQ= percent, direction= increas)
contextSwitch: NFP_Duration= (statisticalQ= max)

« modelLibrary»
Basic_NFP_Types

« import »

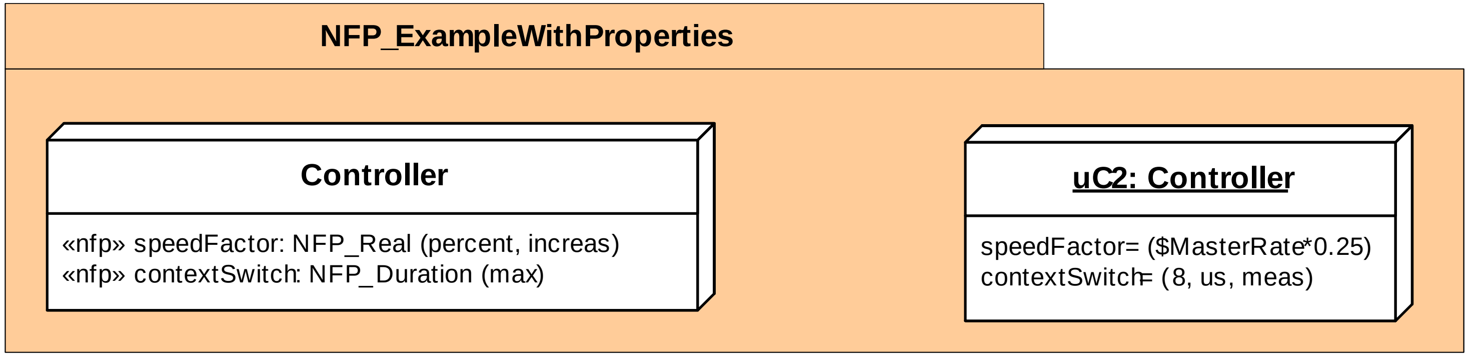
« apply »

UserModelForSchedulabilityAnalysis

« executionHost »
uC: Controller

« executionHost »
speedFactor= (expr= $MasterRate*0.25)
contextSwitch= (value= 8, unit= us, source= meas)

Use of NFPs as M1 level properties:



NFP_ExampleWithProperties

Controller

«nfp» speedFactor: NFP_Real (percent, increas)
«nfp» contextSwitch: NFP_Duration (max)

uC2: Controller

speedFactor= ($MasterRate*0.25)
contextSwitch= (8, us, meas)

Time modeling

- **The Time model introduced in MARTE completes the features provided by the SimpleTime package of UML 2**

- **Basic ideas**
  - Any thing related to time may explicitly refer to a clock
  - Time is multiform (not limited to “physical” time)
  - Support distribution, clock uncertainties
  - Design vs. Runtime clocks

- **What are the domain concepts?**
  - Events → TimedEvent
  - Behaviors and Actions → TimedProcessing
  - Constraints → TimedConstraint
  - Observations → TimedObservation

Time modeling (cont'd)

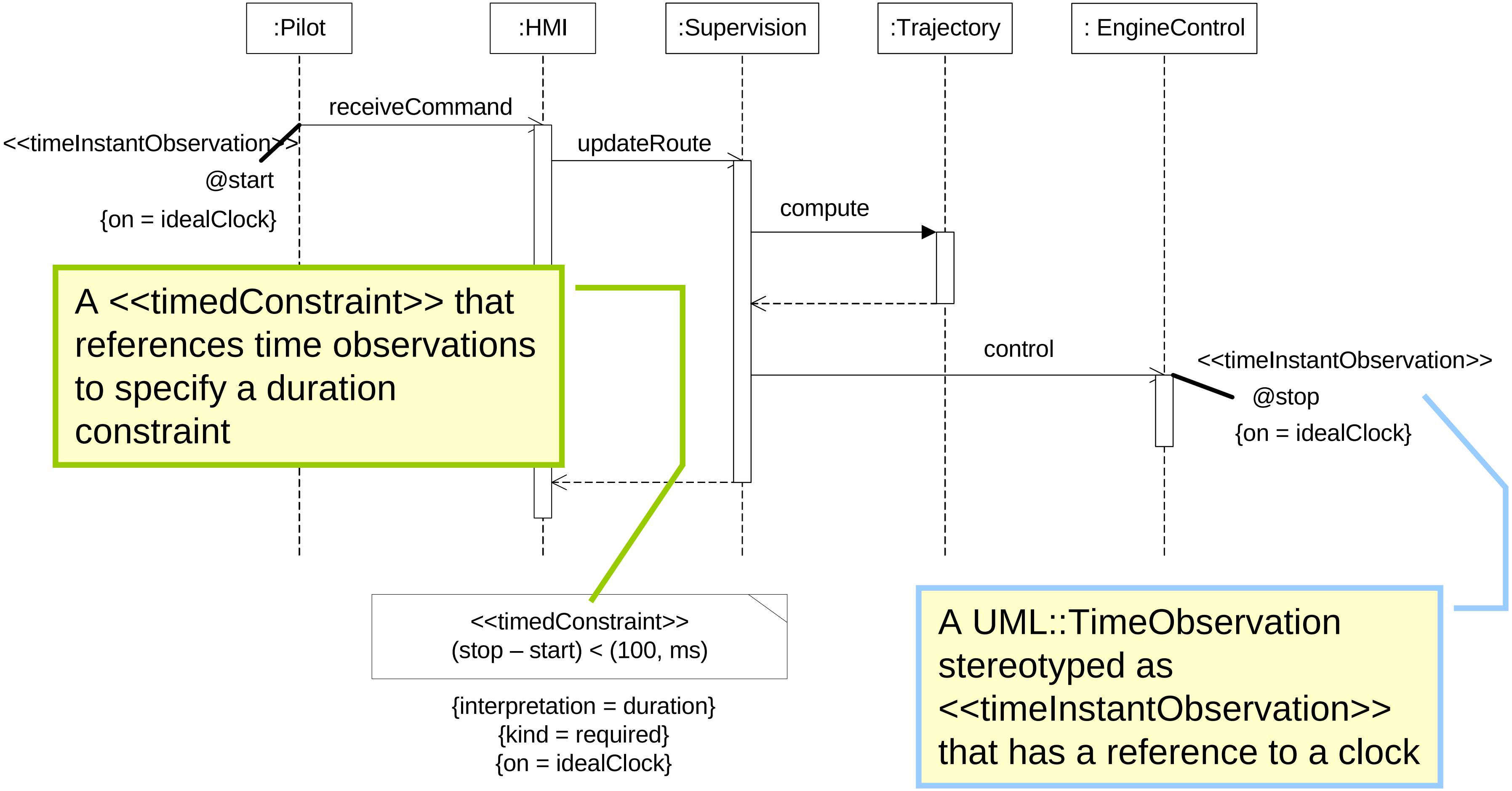
- **Time Structure**
  - Made of several clocks
- **Clock**
  - A totally ordered set of instants
  - Access to instant value and duration with units
- **Relations on Clocks**
  - Expression → ClockConstraint
  - Reflect causality (from algorithm and allocations)

nature / isLogical	discrete	dense
true	Logical clock	Not used
false	Chronometric clock	
	discrete	dense

Stereotype properties :
Special semantics

+ optional
- set of properties
- set of operations

Example of a time constraint

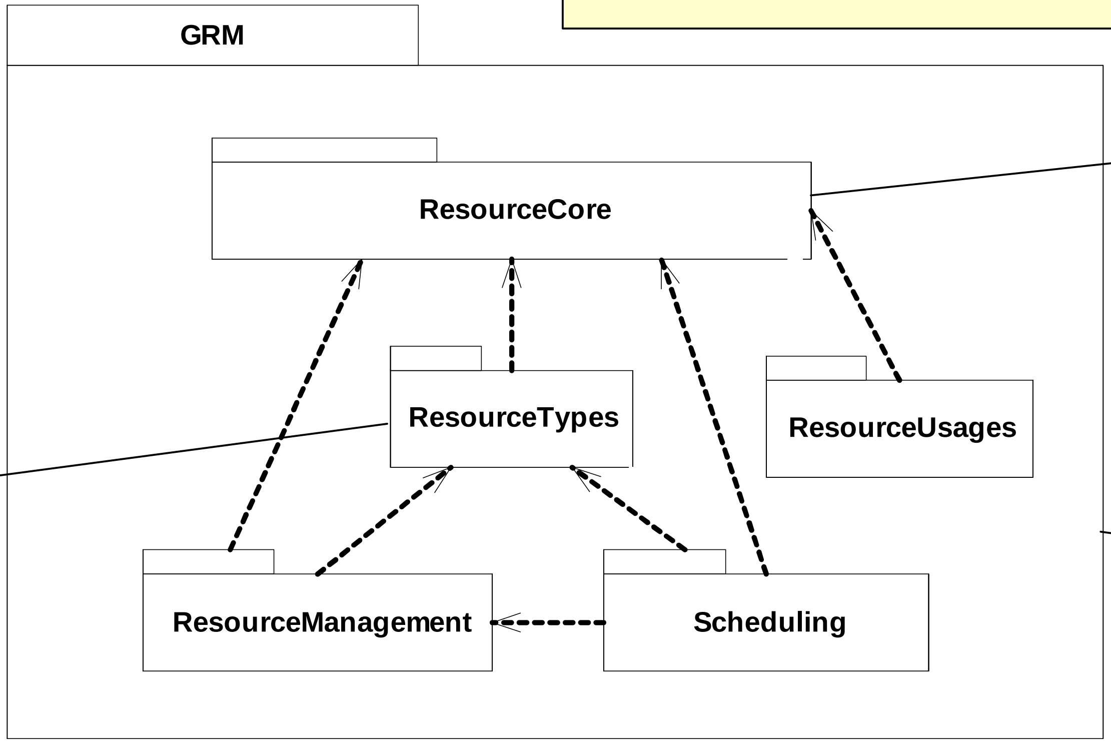


A <<timedConstraint>> that references time observations to specify a duration constraint

A UML::TimeObservation stereotyped as <<timeInstantObservation>> that has a reference to a clock

General Resource Modeling

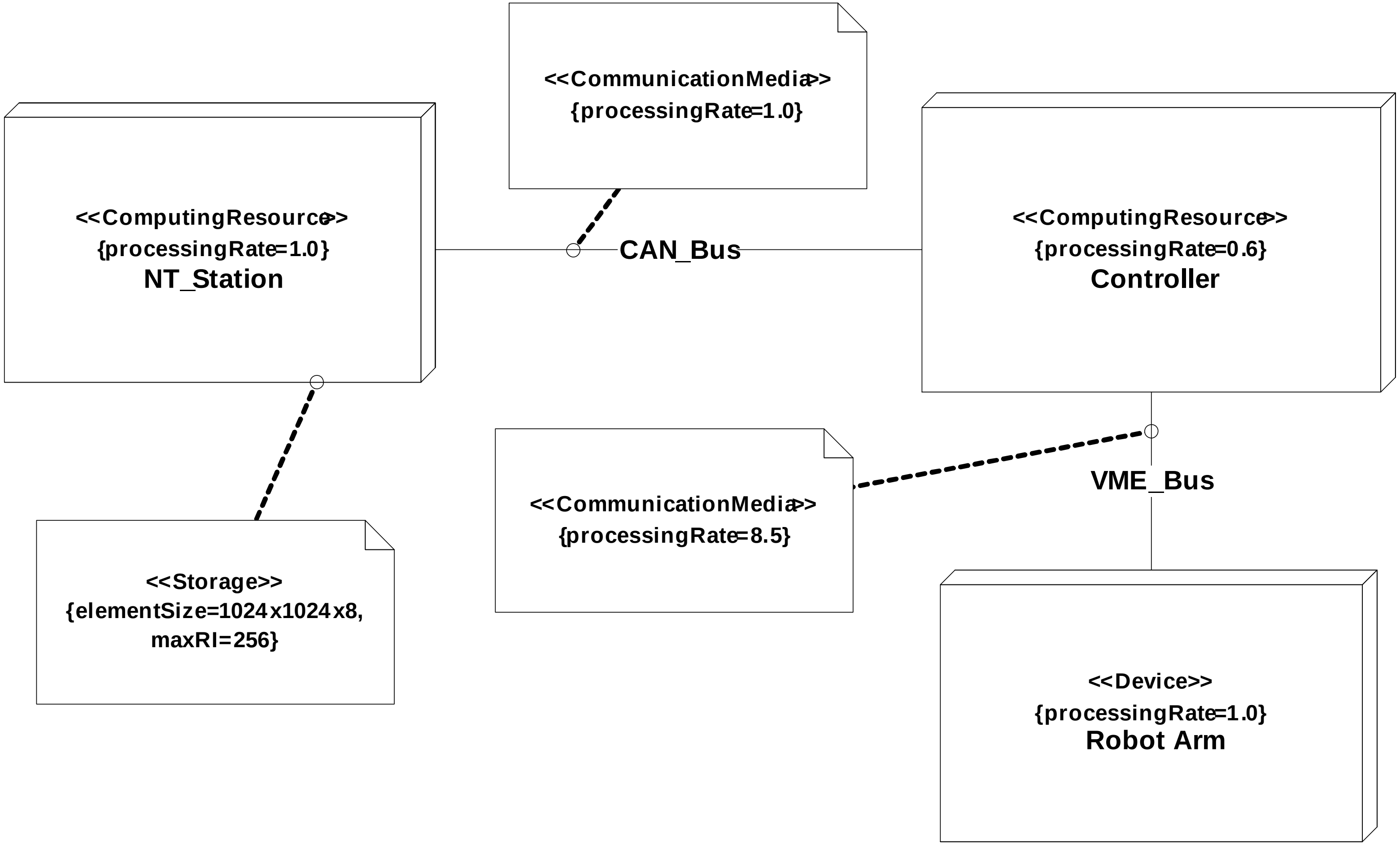
Resource offers Services and may have NFPs for its definition and usage



A rich categorization is provided: Storage, Synchronization, Concurrency, Communication, Timing, Computing, and Device Resources may be defined.

Shared resources, scheduling strategies and specific usages of resources (like memory consumption, computing time and energy) may be annotated.

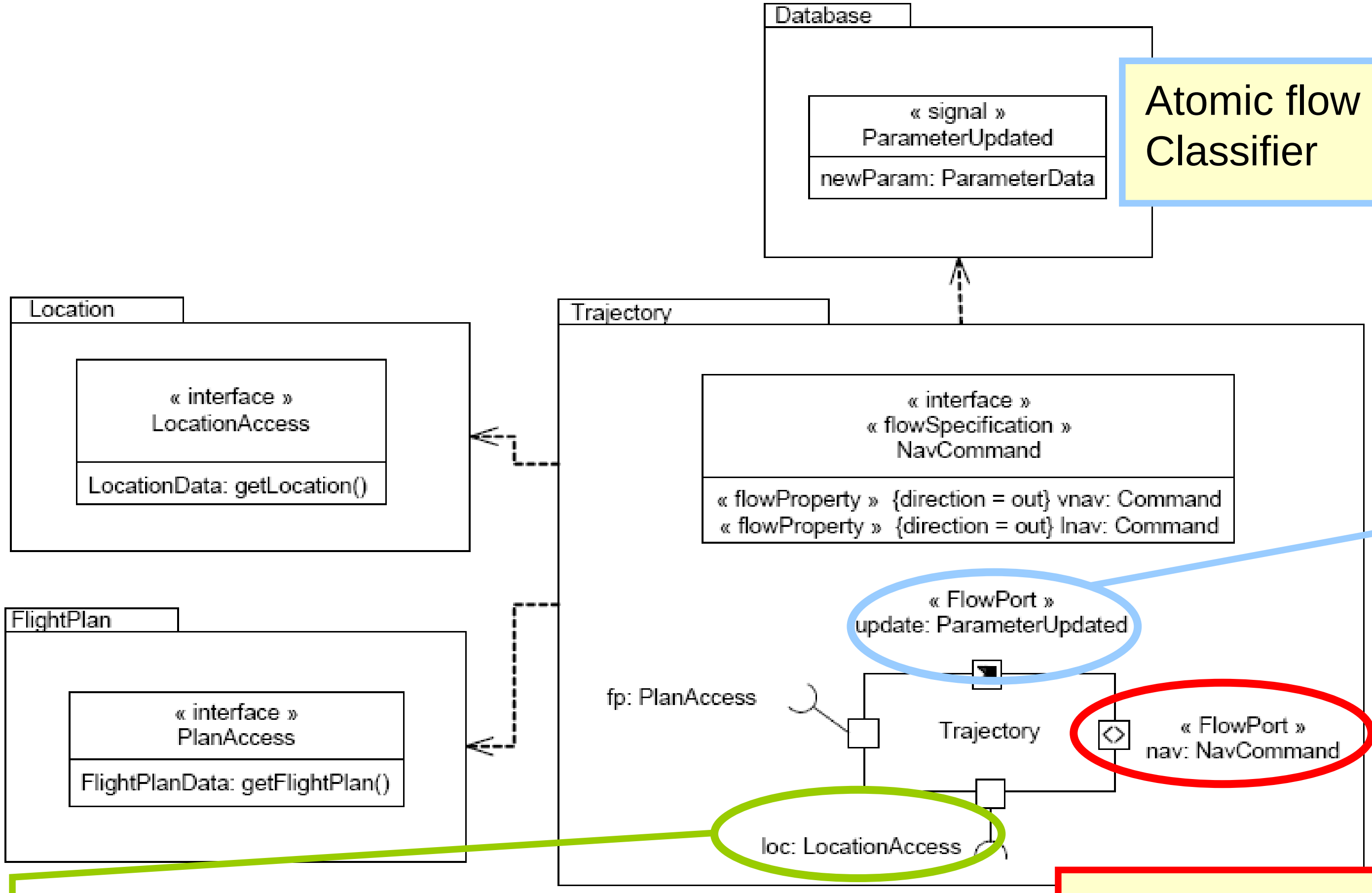
Example of resource modeling



General Component Model

- **Introduced to cope with various component-based models**
  - UML2, SysML, Spirit, AADL, Lightweight-CCM, EAST-ADL2, Autosar…
- **Relies mainly on UML structured classes, on top of which a support for SysML blocks has been added**
  - Atomic and non-atomic flow ports
  - Flow properties and flow specifications
- **But also providing a support for Lightweight-CCM, AADL and EAST-ADL2, Spirit and Autosar**

Example of component definition



Atomic flow port typed by a Classifier

Complex flow port typed by a flow specification

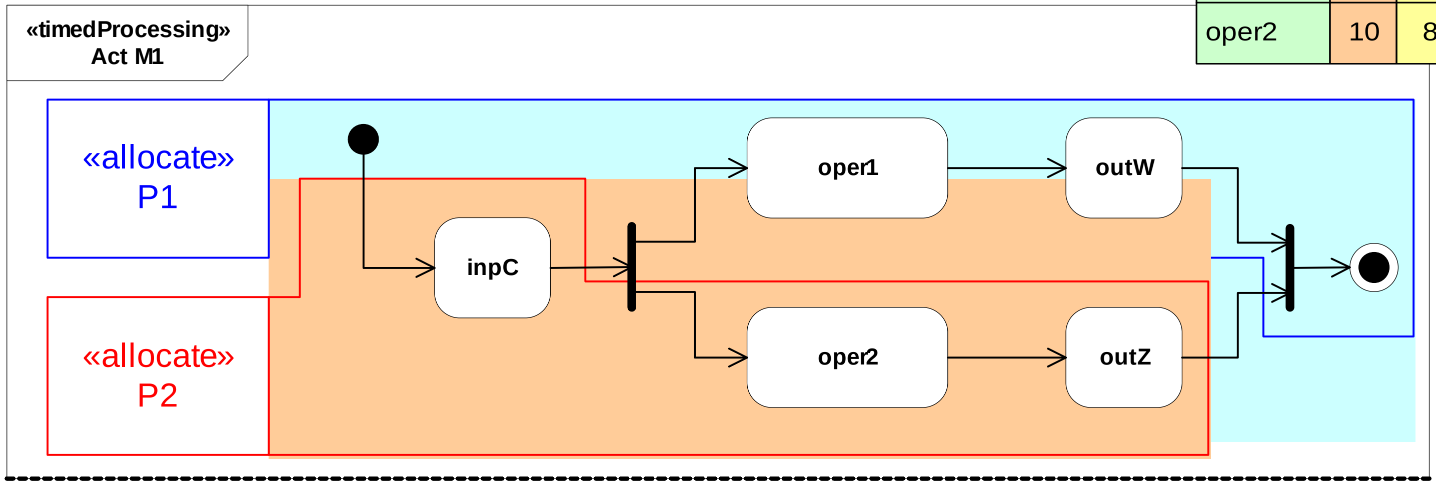
Standard UML port typed by a class that uses the LocationAccess interface

Allocation modeling

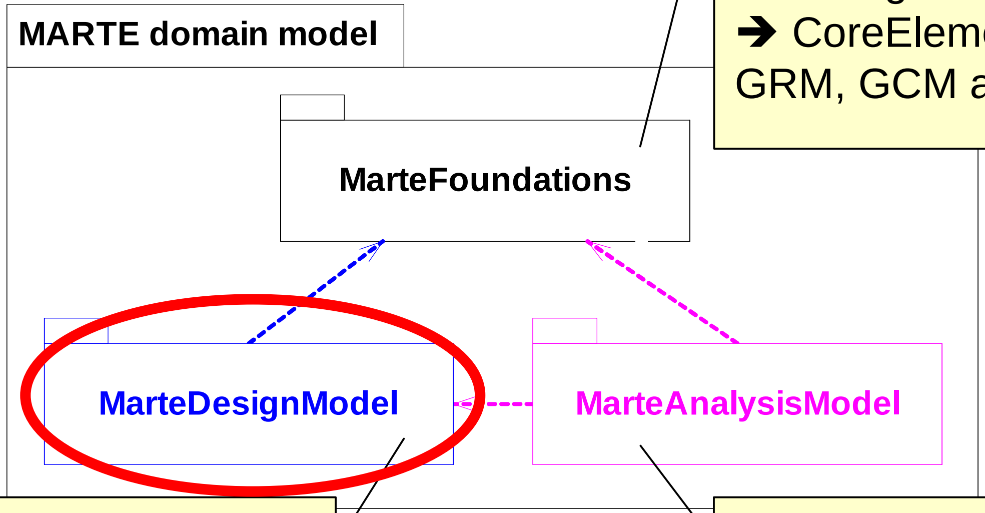
- **Basic ideas**
  - Allocate an application element to an execution platform element
  - Refine a general element into specific elements
  - Inspired by the SysML allocation
    - Can only allocate application to execution platform
    - Can attach NFP constraints to the allocation

- **Example of allocation**

	P1	P2	Unique Alloc
inpC	4	6	true
outpW	4		true
outpZ		6	true
oper1	10		true
oper2	10	8	true



Architecture of the MARTE specification



Foundations for RT/E systems modeling and analysis: 6 pkgs
➔ CoreElements, NFP, Time, GRM, GCM and Alloc

Specialization of Foundations for modeling purpose (specification, design…): 3 pkgs
➔ RTEMoCC, SRM and HRM

Specialization of Foundations for annotating model for analysis purpose: 3 pkgs
➔ GQAM, SAM and PAM

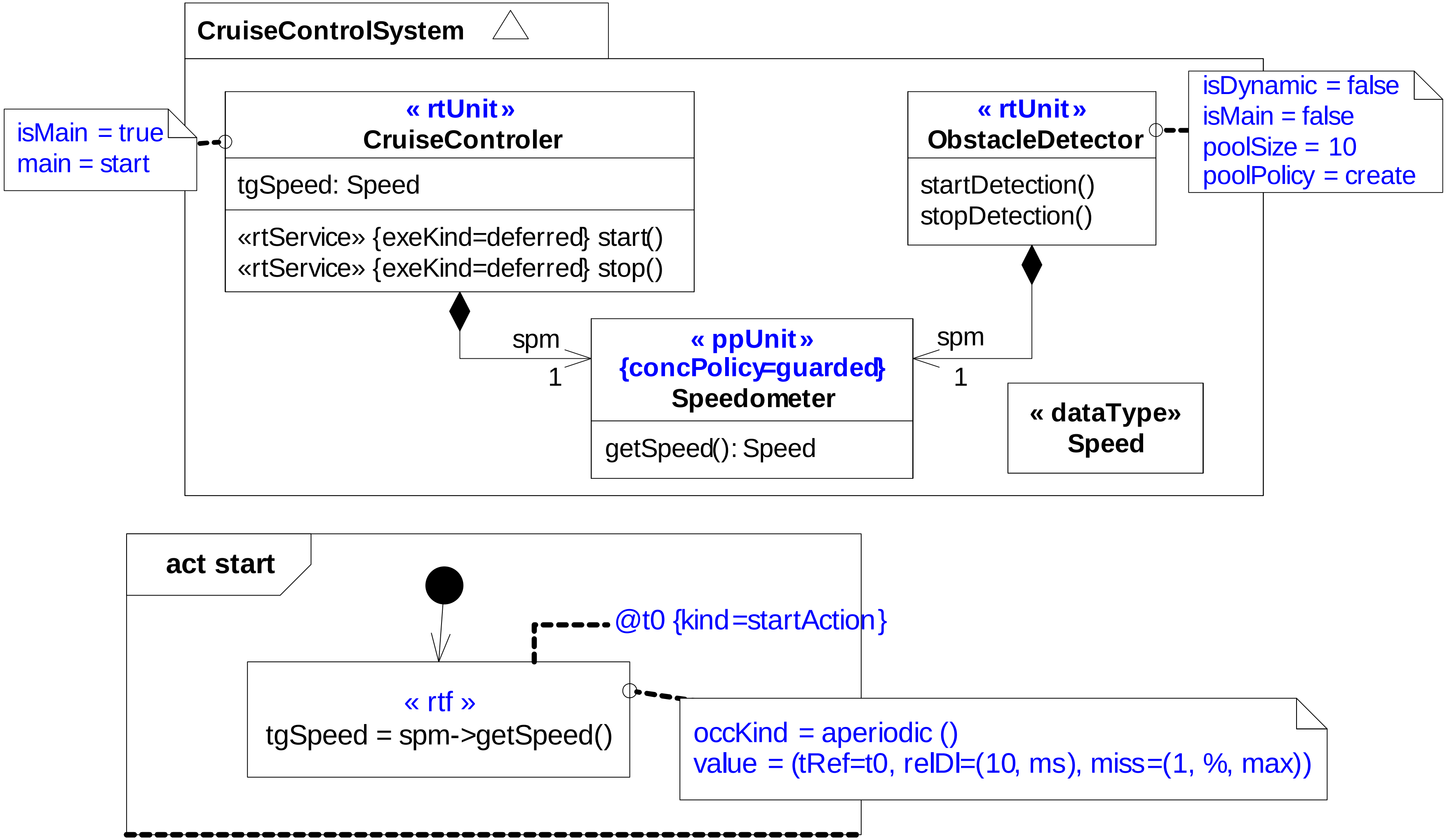
RTE Model of Computation and Communication

- **Provides high-level concepts for modeling qualitative real-time features**
  - Real-Time Unit (RTUnit)
    - Generalization of the Active Objects of the UML 2
    - Owns at last one schedulable resource
    - Resources are managed either statically (pool) or dynamically
    - May have operational mode description (similar to AADL concept)
  - 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 (cont’d)

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

Usage examples of the RTEMoCC extensions



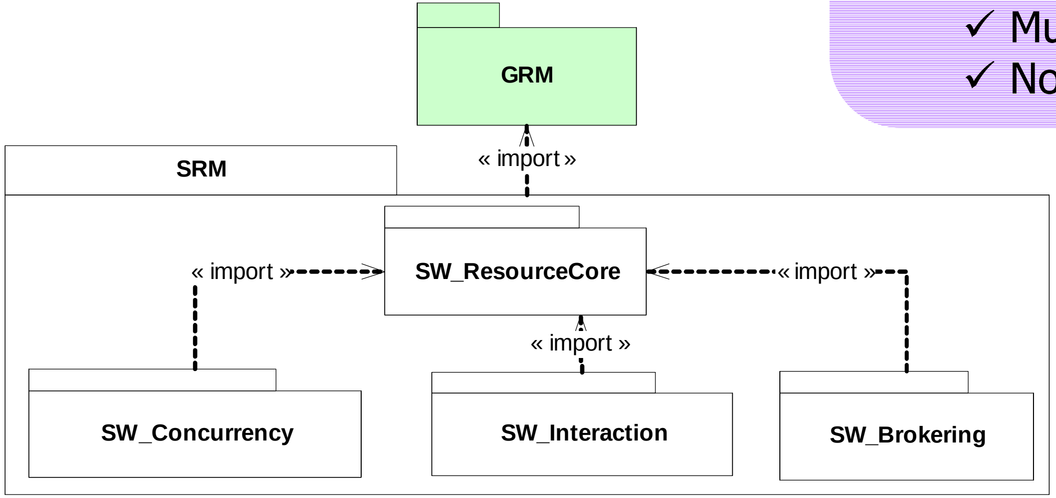
Outline of the Software Resource Model

Concurrent execution contexts:

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

Interactions between concurrent contexts:

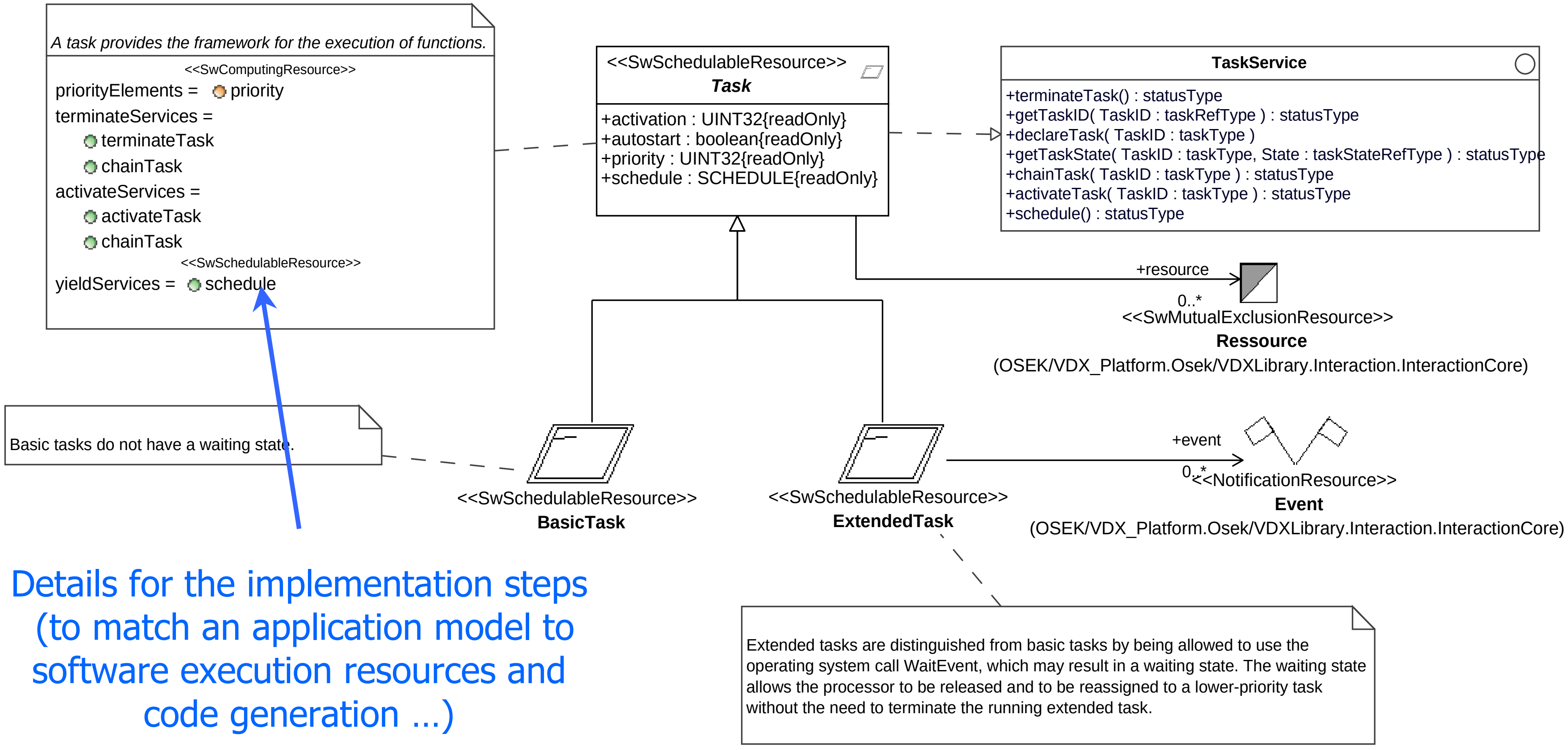
- Communication (Data exchange)
  - ✓ Shared data
  - ✓ Message (Message queue)
- Synchronization
  - ✓ Mutual Exclusion (Semaphore)
  - ✓ Notification (Event mechanism)



Hardware and software resources brokering:

- Drivers
- Memory management

OSEK/VDX modeled with SRM



A task provides the framework for the execution of functions.

<<SwComputingResource>>
priorityElements = priority
terminateServices =
terminateTask
chainTask
activateServices =
activateTask
chainTask
<<SwSchedulableResource>>
yieldServices = schedule

<<SwSchedulableResource>>
Task
+activation : UINT32{readOnly}
+autostart : boolean{readOnly}
+priority : UINT32{readOnly}
+schedule : SCHEDULE{readOnly}

TaskService
+terminateTask() : statusType
+getTaskID(TaskID : taskRefType) : statusType
+declareTask(TaskID : taskType)
+getTaskState(TaskID : taskType, State : taskStateRefType) : statusType
+chainTask(TaskID : taskType) : statusType
+activateTask(TaskID : taskType) : statusType
+schedule() : statusType

<<SwMutualExclusionResource>>
Ressource
(OSEK/VDX_Platform.Osek/VDXLibrary.Interaction.InteractionCore)

Basic tasks do not have a waiting state.

<<SwSchedulableResource>>
BasicTask

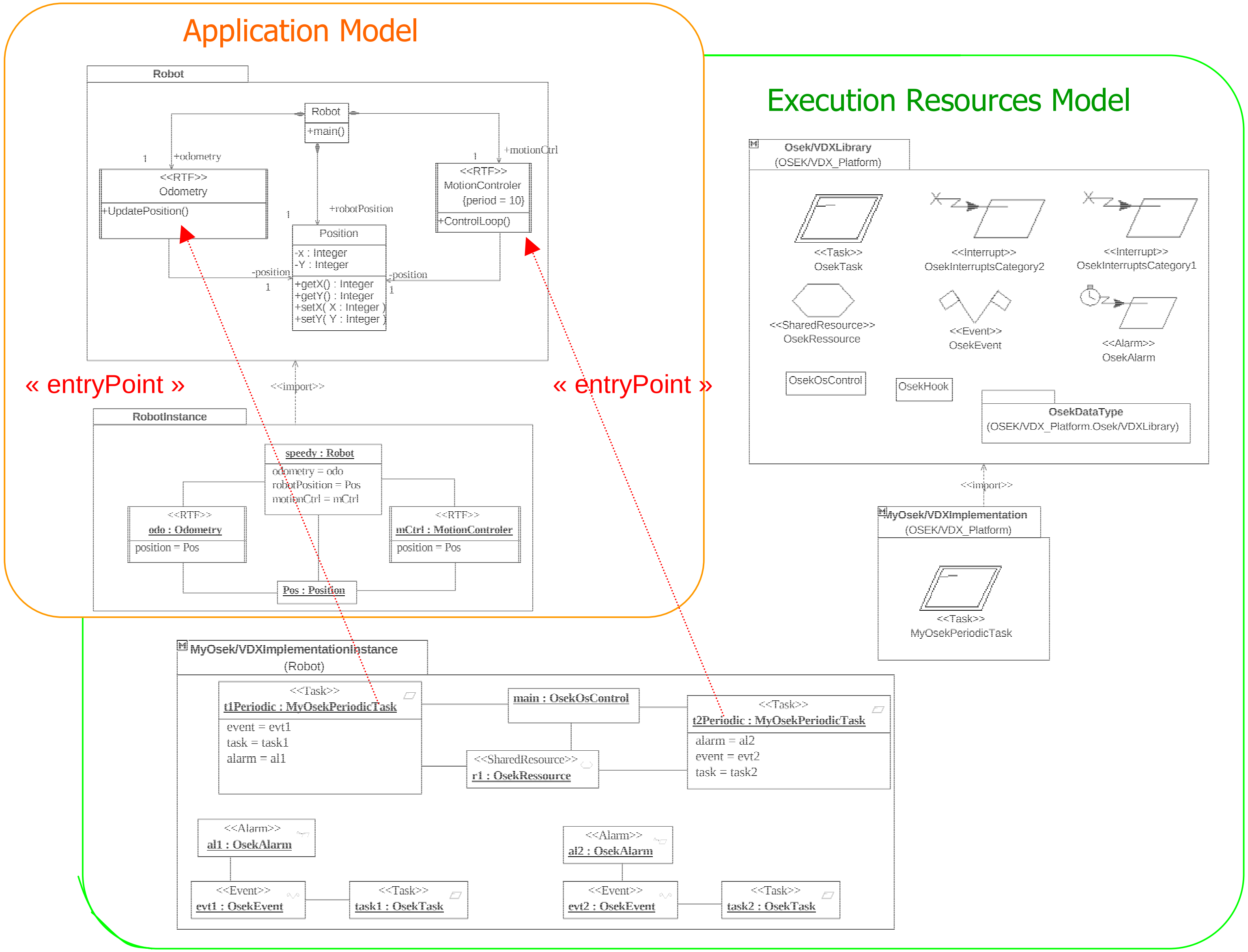
<<SwSchedulableResource>>
ExtendedTask

<<NotificationResource>>
Event
(OSEK/VDX_Platform.Osek/VDXLibrary.Interaction.InteractionCore)

Details for the implementation steps (to match an application model to software execution resources and code generation …)

Extended tasks are distinguished from basic tasks by being allowed to use the operating system call WaitEvent, which may result in a waiting state. The waiting state allows the processor to be released and to be reassigned to a lower-priority task without the need to terminate the running extended task.

SRM Usage example

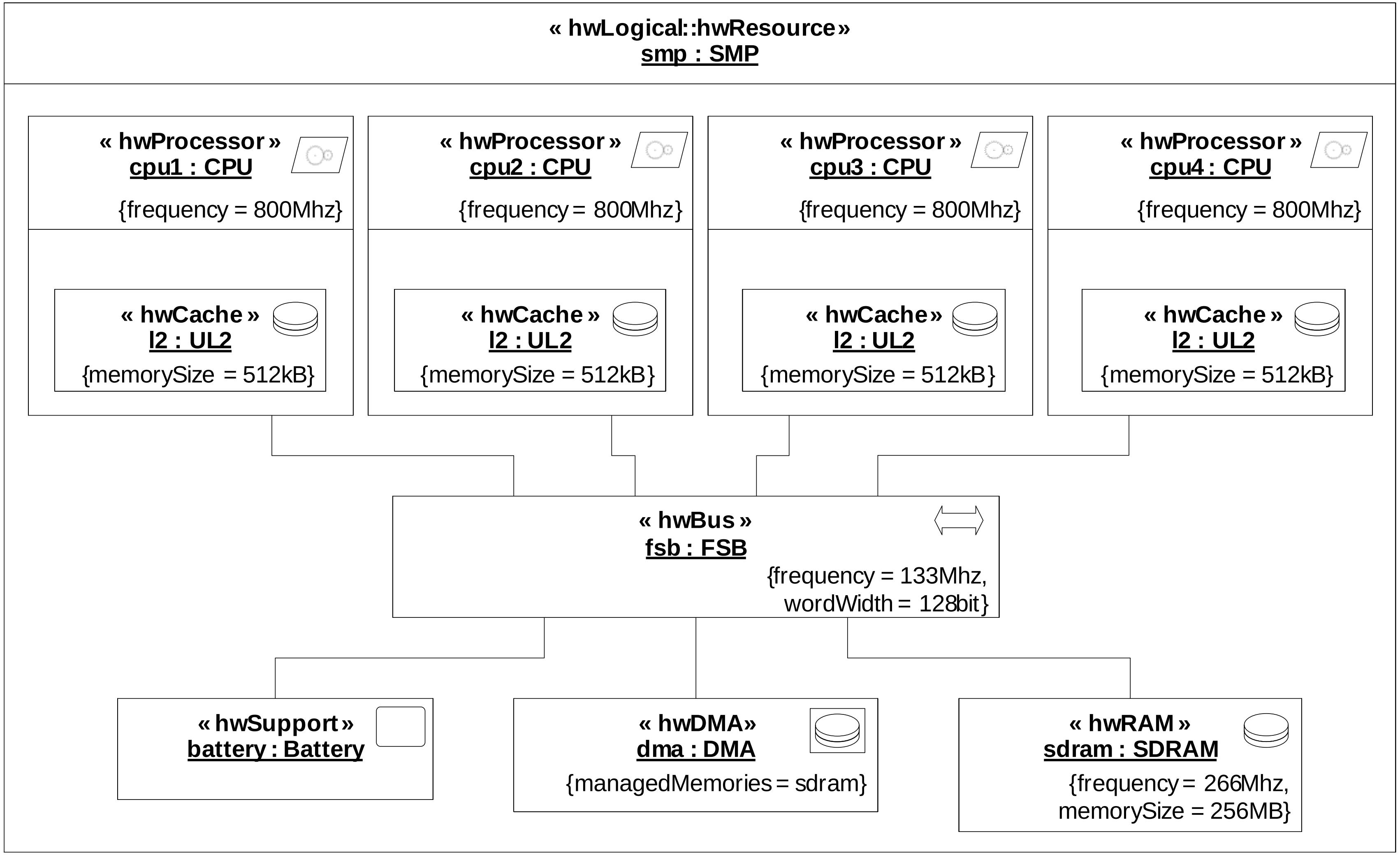


Hardware Resource Model

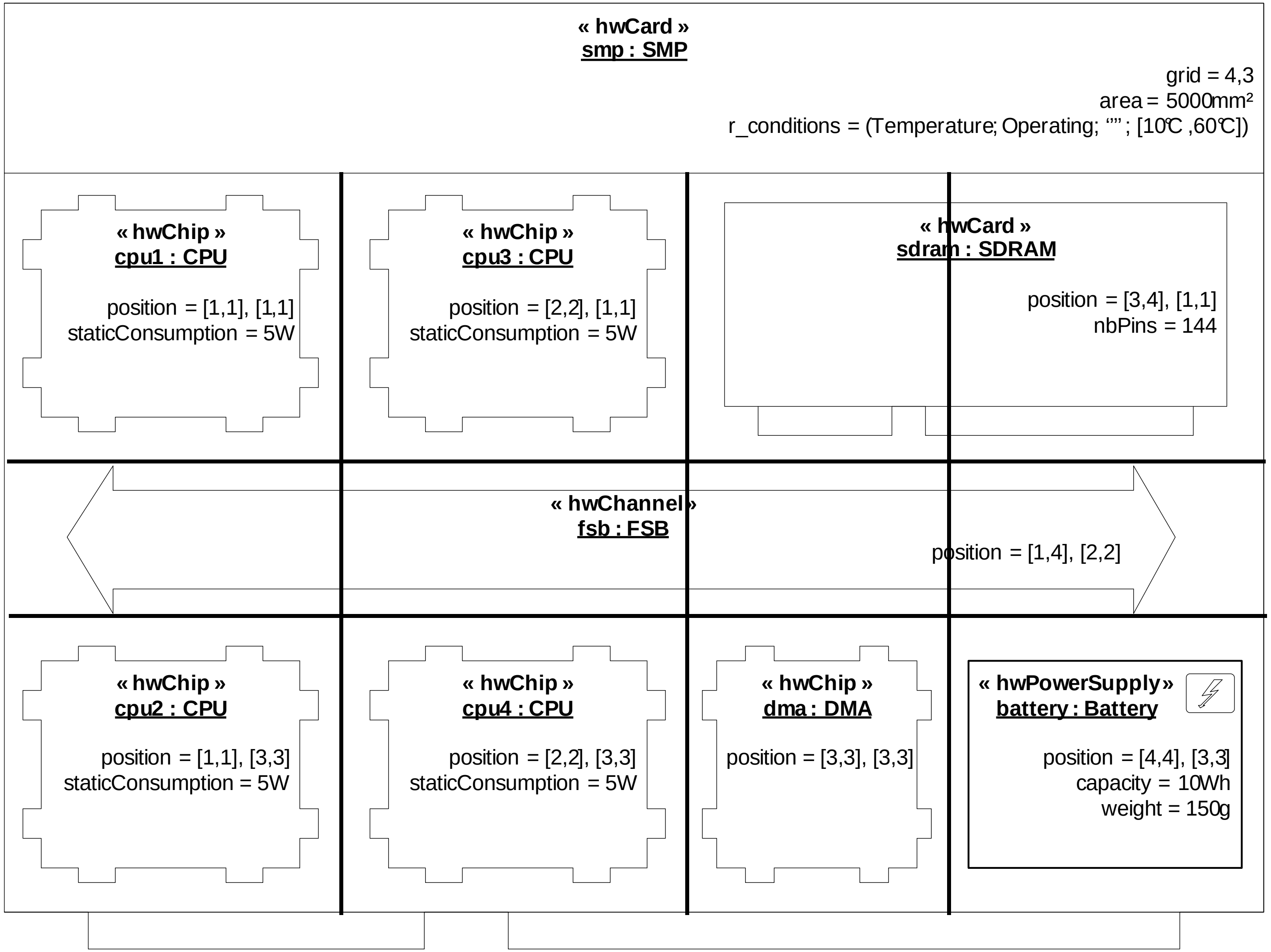
- **Logical view (functional modeling)**
  - Provides a description of functional properties
  - Based on a functional classification of hardware resources:
    - HwComputing resources
    - HwStorage resources
    - HwCommunication resources
    - HwTiming resources
    - HwDevice resources

- **Physical view**
  - Provides a description of physical properties
  - Based on both following packages:
    - HwLayout
    - HwPower

HRM usage example: Logical View



HRM usage example: Physical View


« hwCard »
smp : SMP

grid = 4,3
area = 5000mm²
r_conditions = (Temperature; Operating; “” ; [10℃ ,60℃])

« hwChip »
cpu1 : CPU

position = [1,1], [1,1]
staticConsumption = 5W

« hwChip »
cpu3 : CPU

position = [2,2], [1,1]
staticConsumption = 5W

« hwCard »
sdram : SDRAM

position = [3,4], [1,1]
nbPins = 144

« hwChannel »
fsb : FSB

position = [1,4], [2,2]

« hwChip »
cpu2 : CPU

position = [1,1], [3,3]
staticConsumption = 5W

« hwChip »
cpu4 : CPU

position = [2,2], [3,3]
staticConsumption = 5W

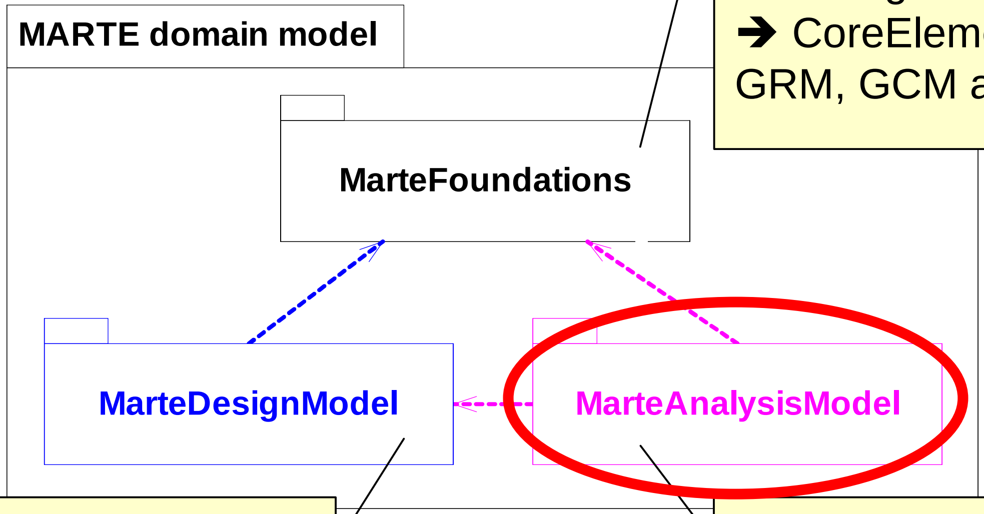
« hwChip »
dma : DMA

position = [3,3], [3,3]

« hwPowerSupply »
battery : Battery

position = [4,4], [3,3]
capacity = 10Wh
weight = 150g

Architecture of the profile



Foundations for RT/E systems modeling and analysis: 6 pkgs
➔ CoreElements, NFP, Time, GRM, GCM and Alloc

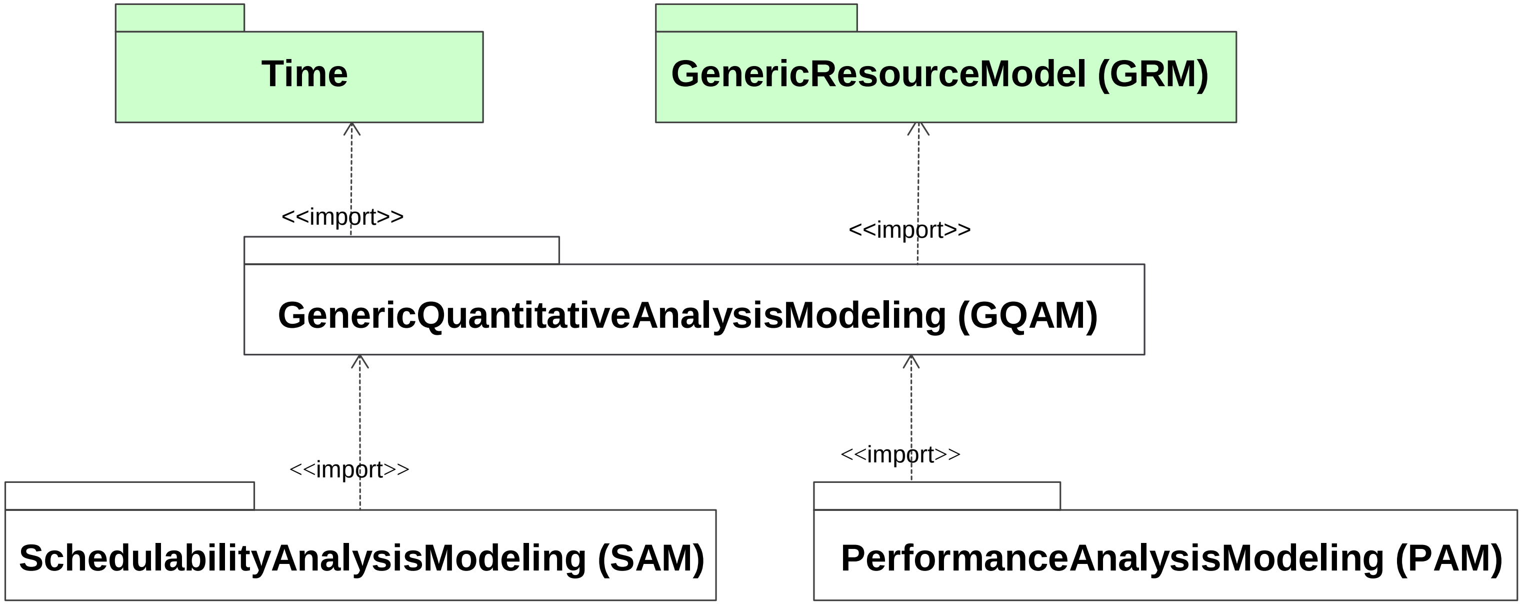
Specialization of Foundations for modeling purpose (specification, design…): 3 pkgs
➔ RTEMoCC, SRM and HRM

Specialization of Foundations for annotating model for analysis purpose: 3 pkgs
➔ GQAM, SAM and PAM

General Quantitative Analysis Model

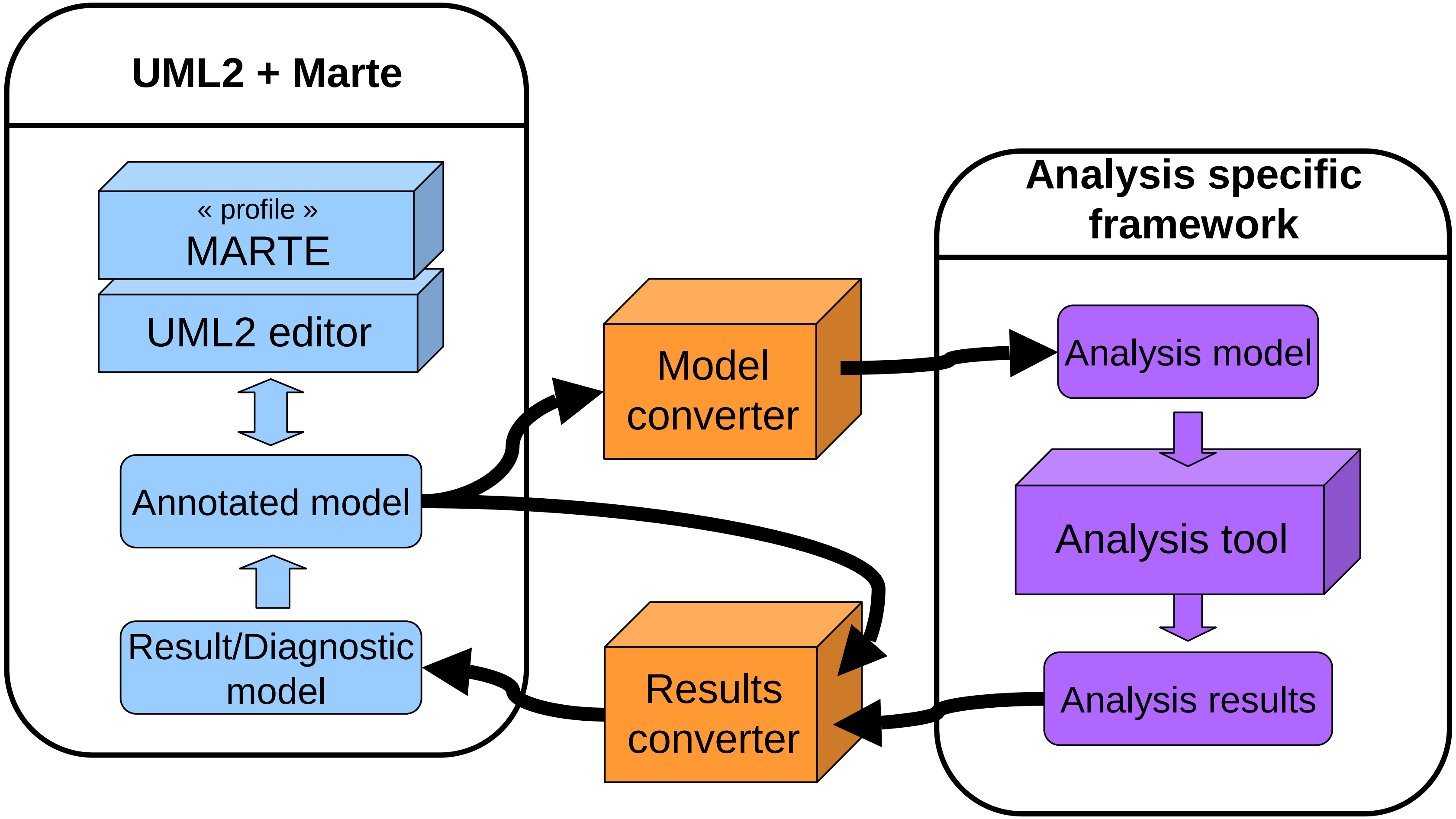
- **GQAM updates SPT**
  - Alignment on UML2
  - Harmonization between two SPT subprofiles: sched. and perf.
  - Extension of timing annotations expressiveness
    - Overheads (e.g. messages passing)
    - Response times (e.g. BCET & ACET)
    - Timing requirements (e.g. miss ratios and max. jitters)
- **Main concepts common for quantitative analysis**
  - Resources
  - Behavior
  - Workload
  - All embedded in an analysis context (may have analysis parameters)

Dependencies and architecture of GQAM


Time
GenericResourceModel (GRM)
<<import>>
<<import>>
GenericQuantitativeAnalysisModeling (GQAM)
<<import>>
<<import>>
SchedulabilityAnalysisModeling (SAM)
PerformanceAnalysisModeling (PAM)

- **GQAM**
  - Common concepts to be used by analysis sub-profiles
- **SAM**
  - Modeling support for schedulability analysis techniques.
- **PAM**
  - Modeling support for performance analysis techniques.

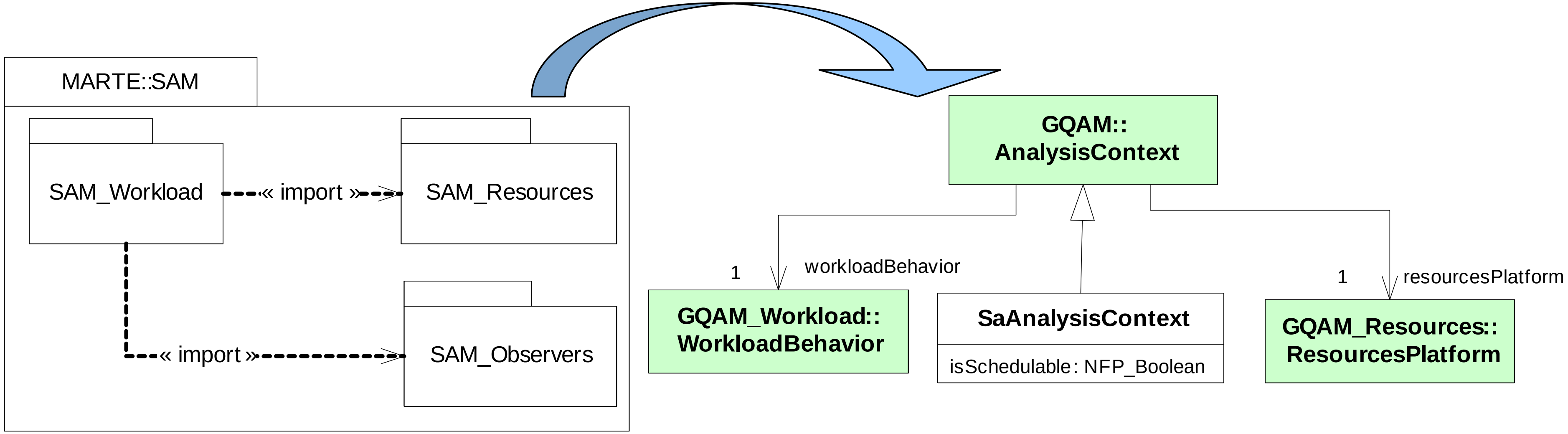
Processing schema for model-based analysis



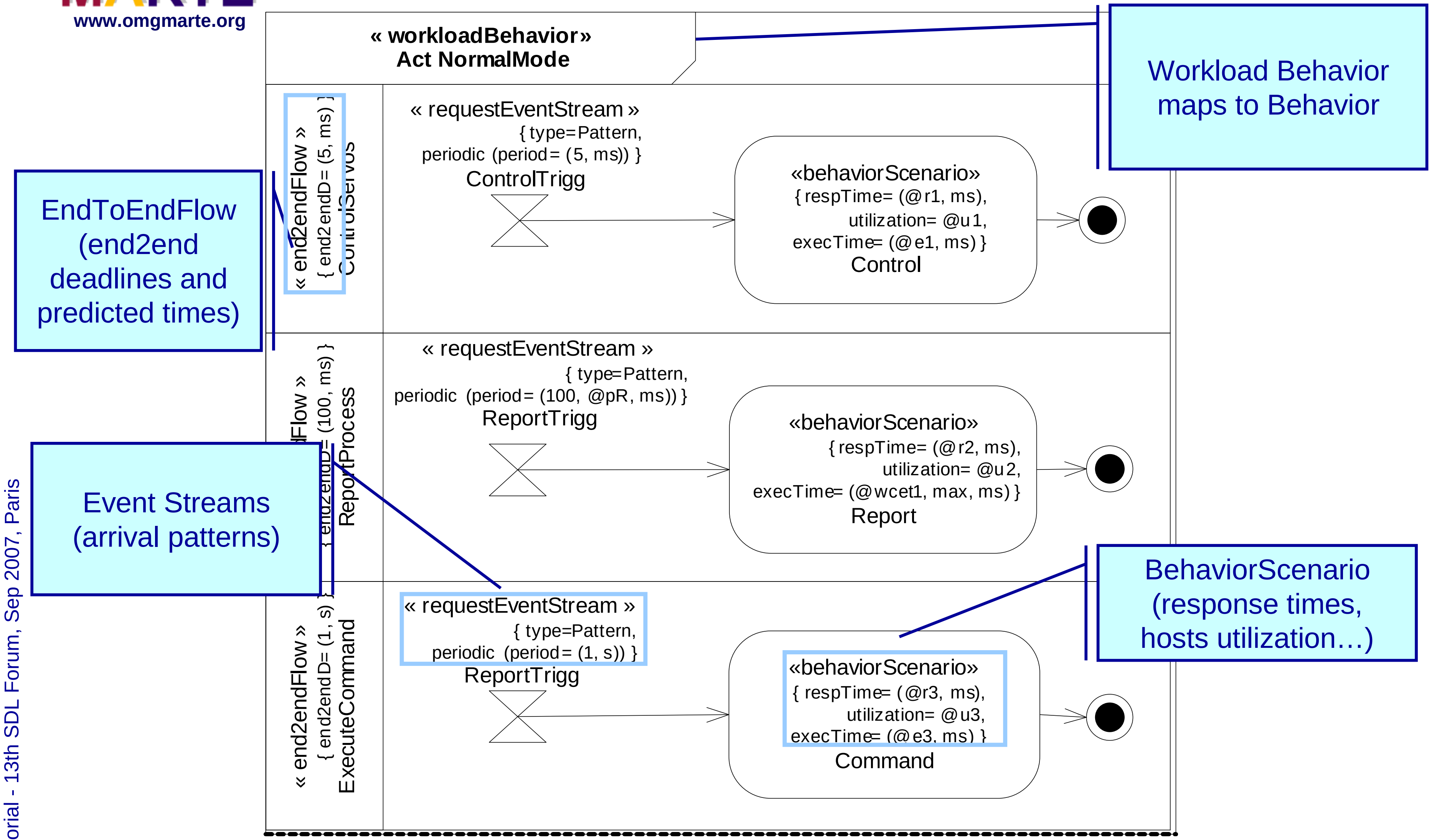
Schedulability Analysis Model

www.omgmarte.org

- **Modeling for analysis techniques taking into account scheduling aspects**
- **Provides high-level analysis constructs**
  - Sensitivity analysis, parametric analysis
  - Observers for time constraints and time predictions at analysis context level
- **Supports most common sched. analysis techniques**
  - RMA-based, holistic techniques and modular techniques



Workload Modeling Example



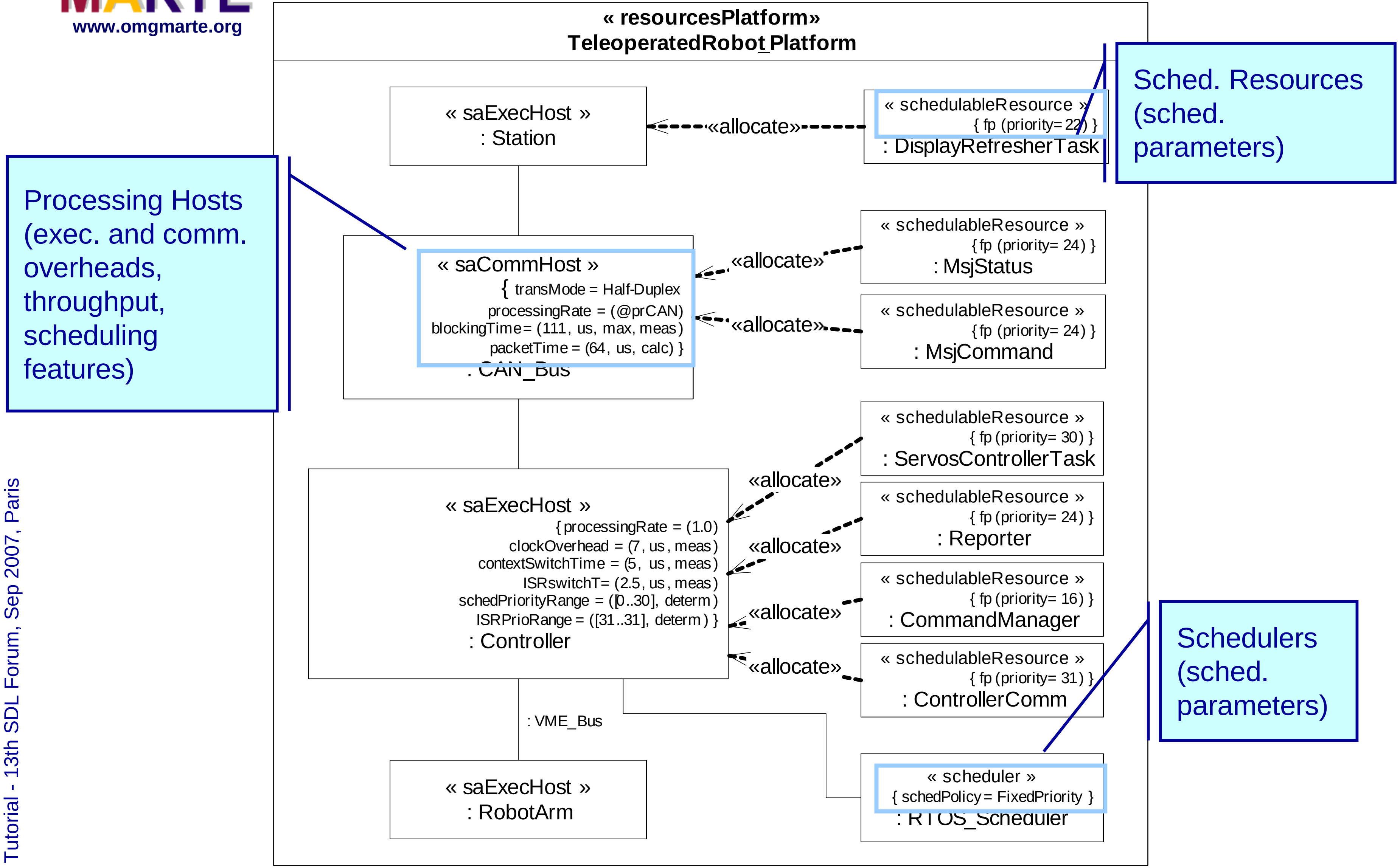
Workload Behavior maps to Behavior

EndToEndFlow (end2end deadlines and predicted times)

Event Streams (arrival patterns)

BehaviorScenario (response times, hosts utilization…)

Resources Platform Modeling Example



« resourcesPlatform»
TeleoperatedRobot_Platform

« saExecHost »
: Station

« schedulableResource »
{ fp (priority= 22) }
: DisplayRefresherTask

«allocate»

« saCommHost »
{ transMode = Half-Duplex
processingRate = (@prCAN)
blockingTime= (111, us, max, meas)
packetTime = (64, us, calc) }
: CAN_Bus

« schedulableResource »
{ fp (priority= 24) }
: MsjStatus

« schedulableResource »
{ fp (priority= 24) }
: MsjCommand

«allocate»

«allocate»

« saExecHost »
{ processingRate = (1.0)
clockOverhead = (7, us, meas)
contextSwitchTime = (5, us, meas)
ISRswitchT= (2.5, us, meas)
schedPriorityRange = ([0..30], determ)
ISRPrioRange = ([31..31], determ) }
: Controller

« schedulableResource »
{ fp (priority= 30) }
: ServosControllerTask

« schedulableResource »
{ fp (priority= 24) }
: Reporter

« schedulableResource »
{ fp (priority= 16) }
: CommandManager

« schedulableResource »
{ fp (priority= 31) }
: ControllerComm

«allocate»

«allocate»

«allocate»

«allocate»

: VME_Bus

« saExecHost »
: RobotArm

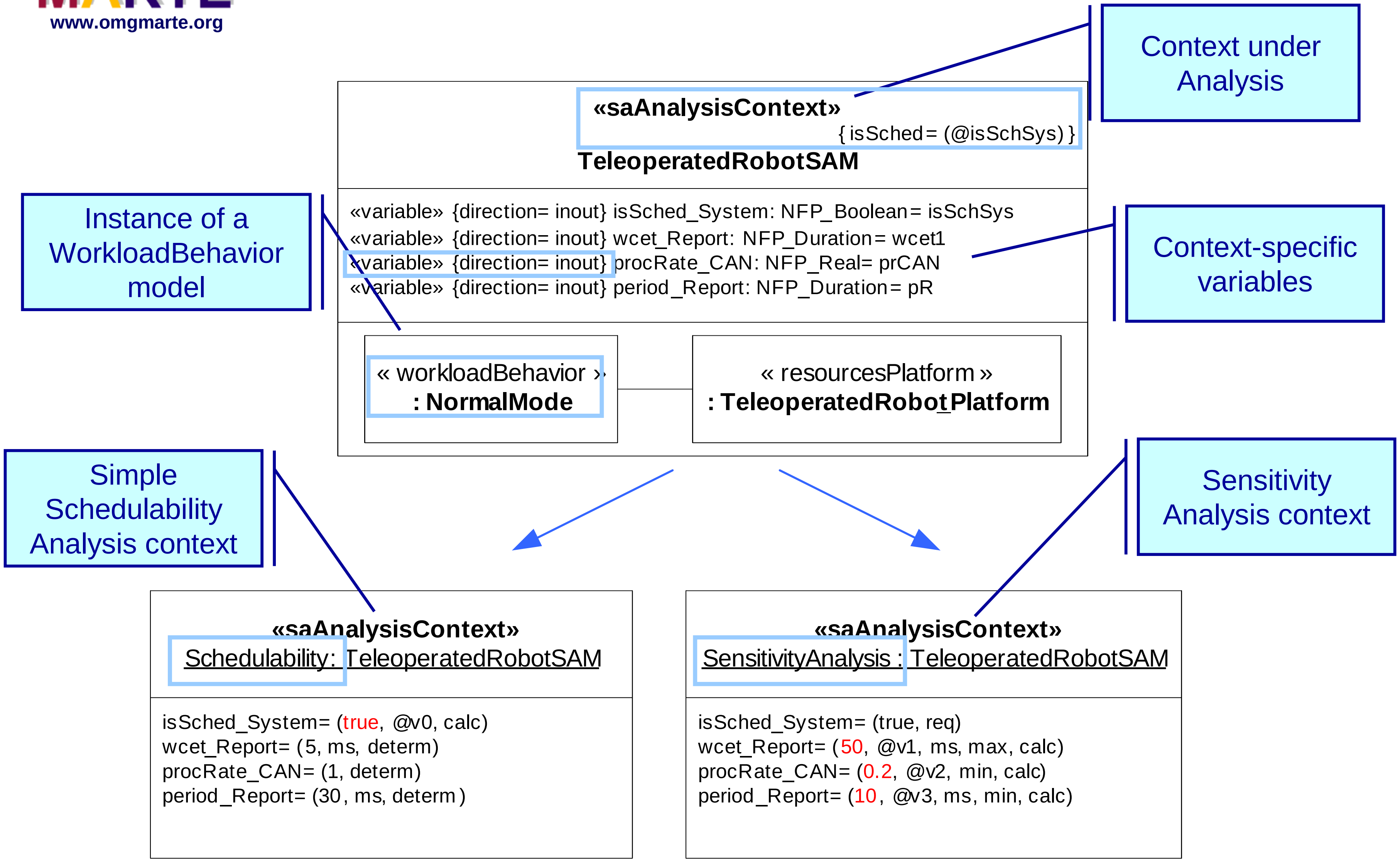
« scheduler »
{ schedPolicy = FixedPriority }
: RTOS_Scheduler

Processing Hosts (exec. and comm. overheads, throughput, scheduling features)

Sched. Resources (sched. parameters)

Schedulers (sched. parameters)

Sched. Analysis Context Example



Context under Analysis

«saAnalysisContext»
{ isSched= (@isSchSys) }
TeleoperatedRobotSAM

«variable» {direction= inout} isSched_System: NFP_Boolean= isSchSys
«variable» {direction= inout} wcet_Report: NFP_Duration= wcet1
«variable» {direction= inout} procRate_CAN: NFP_Real= prCAN
«variable» {direction= inout} period_Report: NFP_Duration= pR

Context-specific variables

Instance of a WorkloadBehavior model

« workloadBehavior »
: NormalMode

« resourcesPlatform »
: TeleoperatedRobotPlatform

Simple Schedulability Analysis context

Sensitivity Analysis context

«saAnalysisContext»
Schedulability: TeleoperatedRobotSAM

isSched_System= (true, @v0, calc)
wcet_Report= (5, ms, determ)
procRate_CAN= (1, determ)
period_Report= (30, ms, determ)

«saAnalysisContext»
SensitivityAnalysis : TeleoperatedRobotSAM

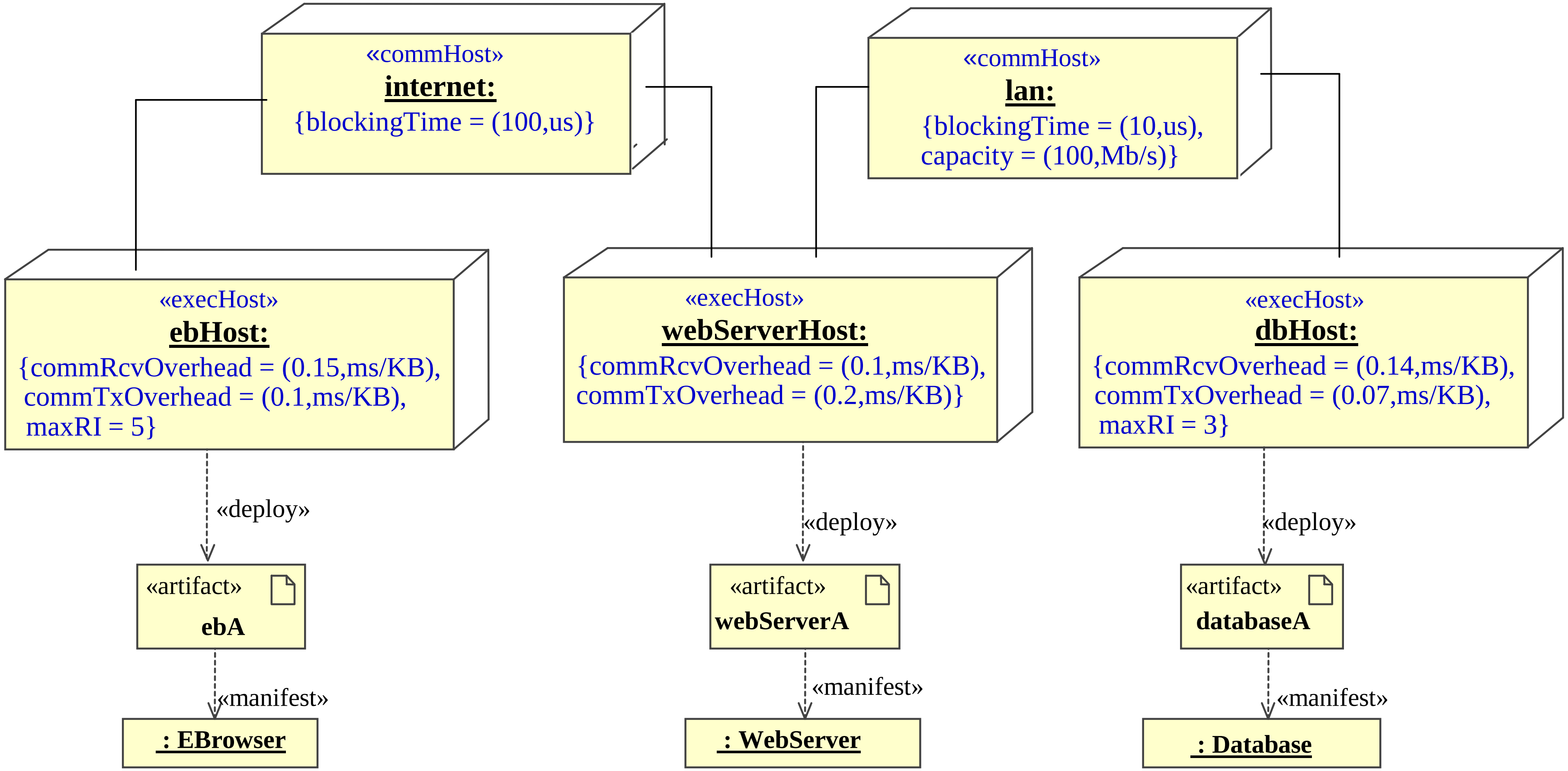
isSched_System= (true, req)
wcet_Report= (50, @v1, ms, max, calc)
procRate_CAN= (0.2, @v2, min, calc)
period_Report= (10, @v3, ms, min, calc)

Performance Analysis Model

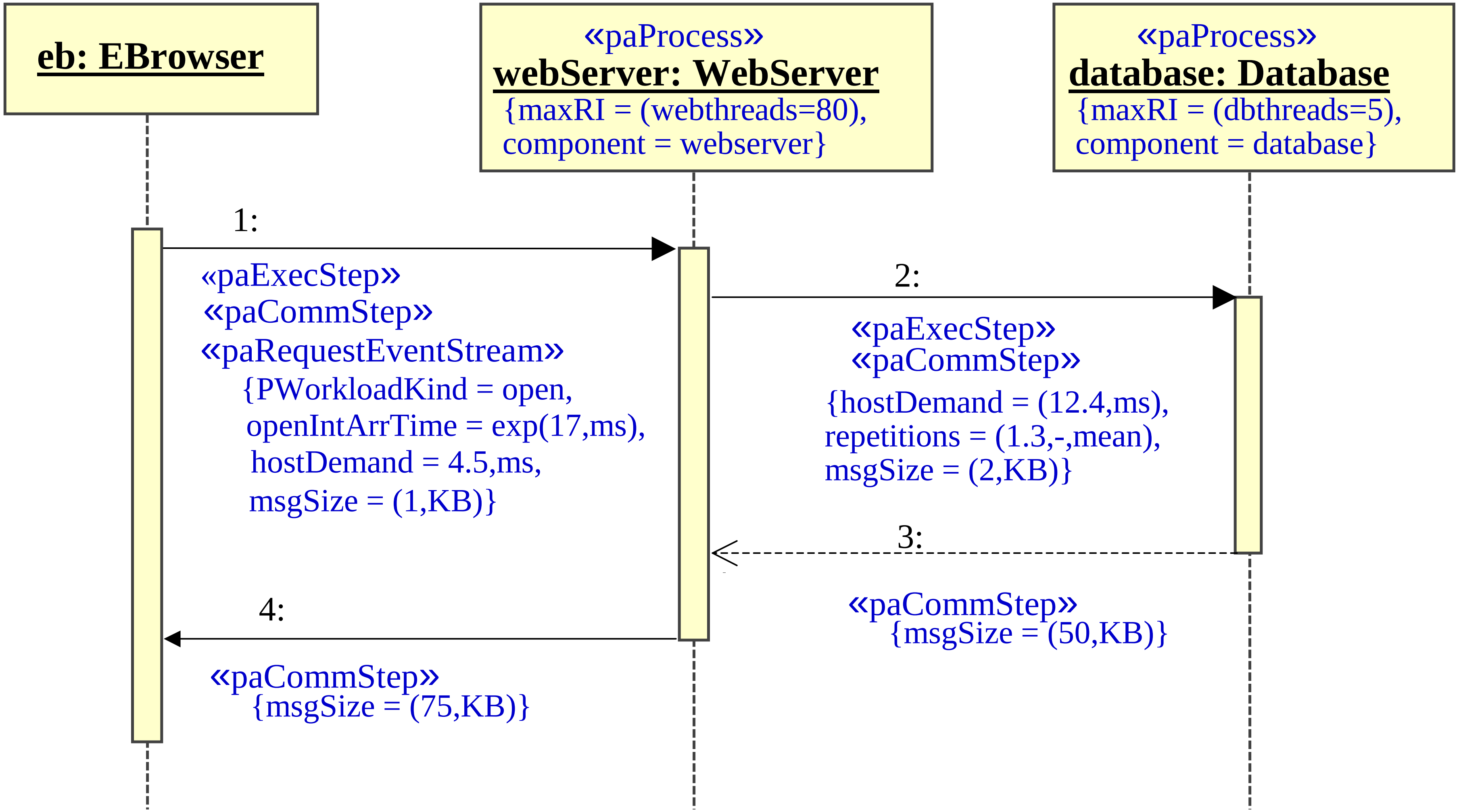
- **Specializes some GQAM stereotypes and reuses others**
  - Workload
    - specialized: PaRequestEventStream, PaWorkloadGenerator, PaEventTrace
  - Behaviour
    - reused: BehaviorScenario, AcqStep, RelStep
    - specialized: PaStep, PaExecStep, PaCommStep, ResPassStep, RequestedService
  - Resources
    - Reused: ExecHost, CommHost, CommChannel
    - Specialized: PaProcess
- **Supports most common performance analysis techniques**
  - Queueing Networks and extensions, Petri Nets, simulation
- **UML + MARTE models should contain**
  - Structural view: software architecture and deployment
  - Behavioral view: key performance scenarios

UML MARTE
www.omgmarte.org

Example: deployment



Example: simple scenario



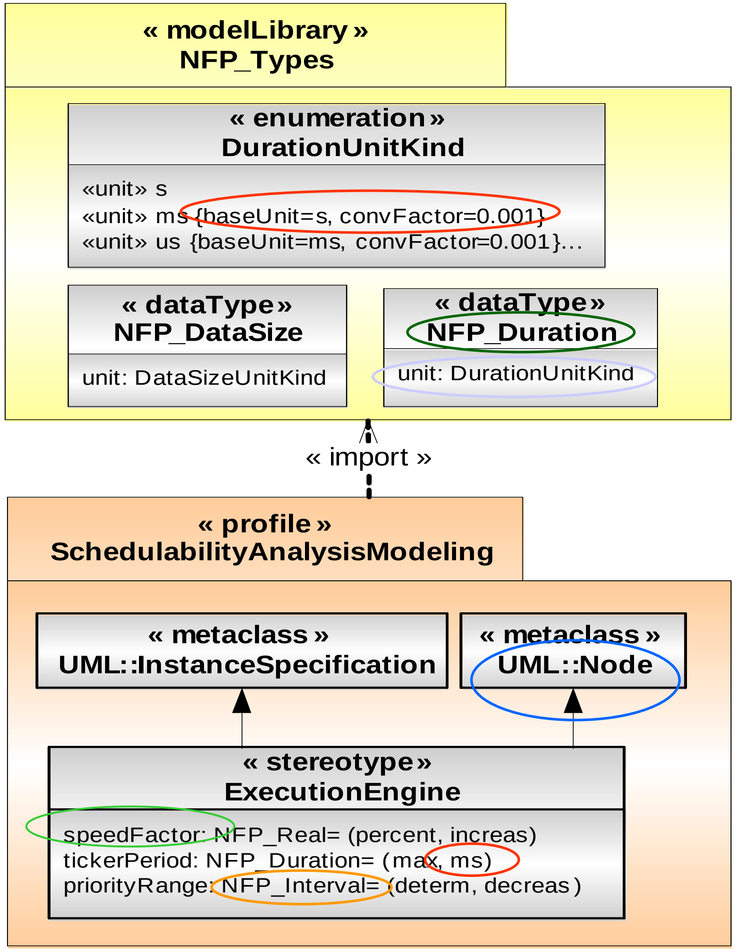
MARTE Annexes

- **Repetitive Structure Modeling**
- **Guidance for use of MARTE**
  - e.g. AADL-like models with MARTE
- **Value Specification Language (VSL)**
- **Clock Handling Facilities**
  - Clock Value Specification Language (CVSL)
  - Clock Constraint Specification Language (CCSL)
- **MARTE Library**

UML MARTE
www.omgmarte.org

AADL-like models with MARTE

UML + MARTE



« modelLibrary »
NFP_Types

« enumeration »
DurationUnitKind

«unit» s
«unit» ms {baseUnit=s, convFactor=0.001}
«unit» us {baseUnit=ms, convFactor=0.001}...

« dataType »
NFP_DataSize

unit: DataSizeUnitKind

« dataType »
NFP_Duration

unit: DurationUnitKind

« import »

« profile »
SchedulabilityAnalysisModeling

« metaclass »
UML::InstanceSpecification

« metaclass »
UML::Node

« stereotype »
ExecutionEngine

speedFactor: NFP_Real= (percent, increas)
tickerPeriod: NFP_Duration= (max, ms)
priorityRange: NFP_Interval= (determ, decreas)

AADL

```
Length_Unit : type units ( mm, cm => mm
* 10, m => cm * 100, km => m * 1000 );

OnOff : type aadlboolean;

Speed_Range : type range of aadlreal 0
.. 250 units ( kph );

mass_t: type aadlreal units mass_u;

mass_u: type units (g, kg => g*1000, t
=> kg*1000);
```

```
Wheel_speed : aadlinteger 0 rpm .. 5000
rpm units ( rpm ) applies to (system);

allowed_mass: mass_range_t applies to
memory, processor, bus, device, system);

actual_mass: mass_t applies to (memory,
processor, bus, device, system);
```

Conclusion

- **MARTE is the new OMG specification for Modeling and Analysis Real-Time and Embedded systems**
  - Specification adopted in June 2007
- **MARTE provides extensions to UML for modeling non-functional properties, time and resource, software and hardware execution platforms and allocations**
- **MARTE enables model-based quantitative analysis, including schedulability and performance**
- **A first Eclipse-based open-source implementation is available**
  - Papyrus for MARTE (http://www.papyrusuml.org)
- **Ongoing discussions to align parts of MARTE and SysML**

References

- OMG MARTE web site
  - http://www.omgmarte.org

- UML profile for MARTE (alpha)
  - http://www.omg.org/cgi-bin/doc?ptc/2007-08-04
- UML profile for MARTE RFP
  - http://www.omg.org/cgi-bin/doc?realtime/2005-2-6
- UML 2 Superstructure
  - http://www.omg.org/cgi-bin/doc?formal/07-02-05