Using the MARTE UML-profile to engineer predictable real-time systems

Dr. Julio Medina
Universidad de Cantabria, Spain
julio.medina[at]unican.es)

(Based on extracts from the MARTE Tutorial:
http://www.omgmarte.org/sites/default/files/TutorialMARTE-Final_version1_1_0.pdf

Agenda

- Introduction to MDD for RT/E systems
- The UML profile for MARTE
- Introduction to model-based RT/E analysis
- Model-based schedulability analysis
- MARTE Frontiers and Challenges

Models in Traditional Engineering

- Probably as old as engineering

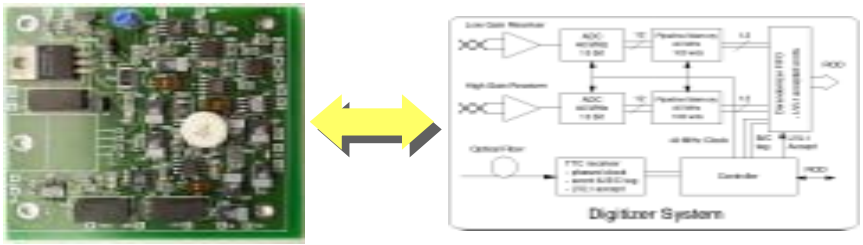
Extracted from B. Selic presentation during Summer School
MDD For DRES 2004 (Brest, September 2004)

What is a Model in MDD

Inspired from B. Selic presentation during Summer School MDD For DRES 2004 (Brest, September 2004)

- Phil Bernstein, "A Vision for Management of Complex Systems".
  A model is a complex structure that represents a design artifact such as a relational schema, an interface definition (API), an XML schema, a semantic network, a UML model or a hypermedia document.
- OMG, "UML Superstructure".
  A model captures a view of a physical system. It is an abstraction of the physical system, with a certain purpose. This purpose determines what is included in the model and what is relevant. Thus the model completely describes those aspects of the physical system that are relevant to the purpose of the model, at the appropriate level of detail.
- OMG, "MDA Guide".
  A formal specification of the function, structure and/or behavior of an application or system.
- Steve Mellor, et al., "UML Distilled"
  A model is a simplification of something so we can view, manipulate, and reason about it, and so help us understand the complexity inherent in the subject under study.
- Anneke Kleppe, et. al. "MDA Explained"
  A model is a description of (part of) a system written in a well-defined language. A well-defined language is a language with well-defined form (syntax), and meaning (semantics), which is suitable for automated interpretation by a computer.
- Chris Raistrick et al., "Model Driven Architecture with Executable UML"
  A formal representation of the function, behavior, and structure of the system we are considering, expressed in an unambiguous language.
- J. Bézivin & O. Gerbé, "Towards a Precise Definition of the OMG/MDA Framework"
  A simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system.

- One definition
  - A reduced/abstract representation of some system that highlights the properties of interest from a given viewpoint.
  - The viewpoint defines concern, scope and detail level of the model.



Modeled system **Functional model**

Why Model Driven Engineering is Needed?

- To deal with complexity of systems development
  - Abstract a problem to focus on some particular points of interest
    ➔ improve understandability of a problem
  - Possible set of nearly independent views of a model
    - Separation of concerns (e.g. “Aspect Oriented Modeling”)
  - Iterative modeling may be expressed at different level of fidelity

- To minimize development risks
  - Through analysis and experimentation performed earlier in the design cycle
  - Enable to investigate and compare alternative solutions

- To improve communication ...
  ... to foster information sharing and reuse!
  - A model is often best suited than a long speech !

- To focus on specific domain expertise while developing software system
  - Domain Specific Language

Characteristics of Useful Models

- Abstract
  - Emphasize important aspects while removing irrelevant ones
- Understandable
  - Expressed in a form that is readily understood by observers
- Accurate
  - Faithfully represents the modeled system
- Predictive
  - Can be used to answer questions about the modeled system
- Inexpensive
  - Much cheaper to construct and study than the modeled system

To be useful, engineering models must satisfy all of these characteristics!

(Extracted from B. Selic presentation during Summer School MDD For DRES 2004 (Brest, September 2004)

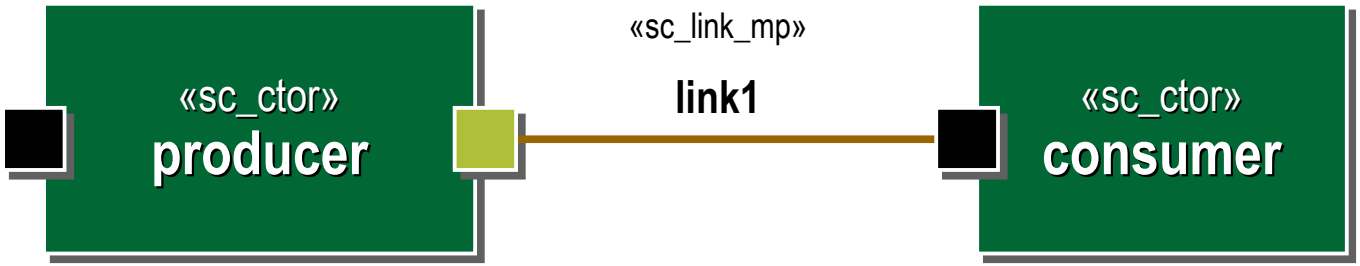
A Bit of Modern Software…

```
SC_MODULE(producer)
{
sc_outmaster<int> out1;
sc_in<bool> start; // kick-start
void generate_data ()
{
for(int i =0; i <10; i++) {
out1 =i ; //to invoke slave;}
}
SC_CTOR(producer)
{
SC_METHOD(generate_data);
sensitive << start;}};
SC_MODULE(consumer)
{
sc_inslave<int> in1;
int sum; // state variable
void accumulate (){
sum += in1;
cout << "Sum = " << sum << endl;}
SC_CTOR(consumer)
{
SC_SLAVE(accumulate, in1);
sum = 0; // initialize
};
SC_MODULE(top) // container
{
producer *A1;
consumer *B1;
sc_link_mp<int> link1;
SC_CTOR(top)
{
A1 = new producer("A1");
A1.out1(link1);
B1 = new consumer("B1");
B1.in1(link1);}};
```

Can you spot the architecture?

(Extracted from B. Selic presentation during Summer School MDD For DRES 2004 (Brest, September 2004)

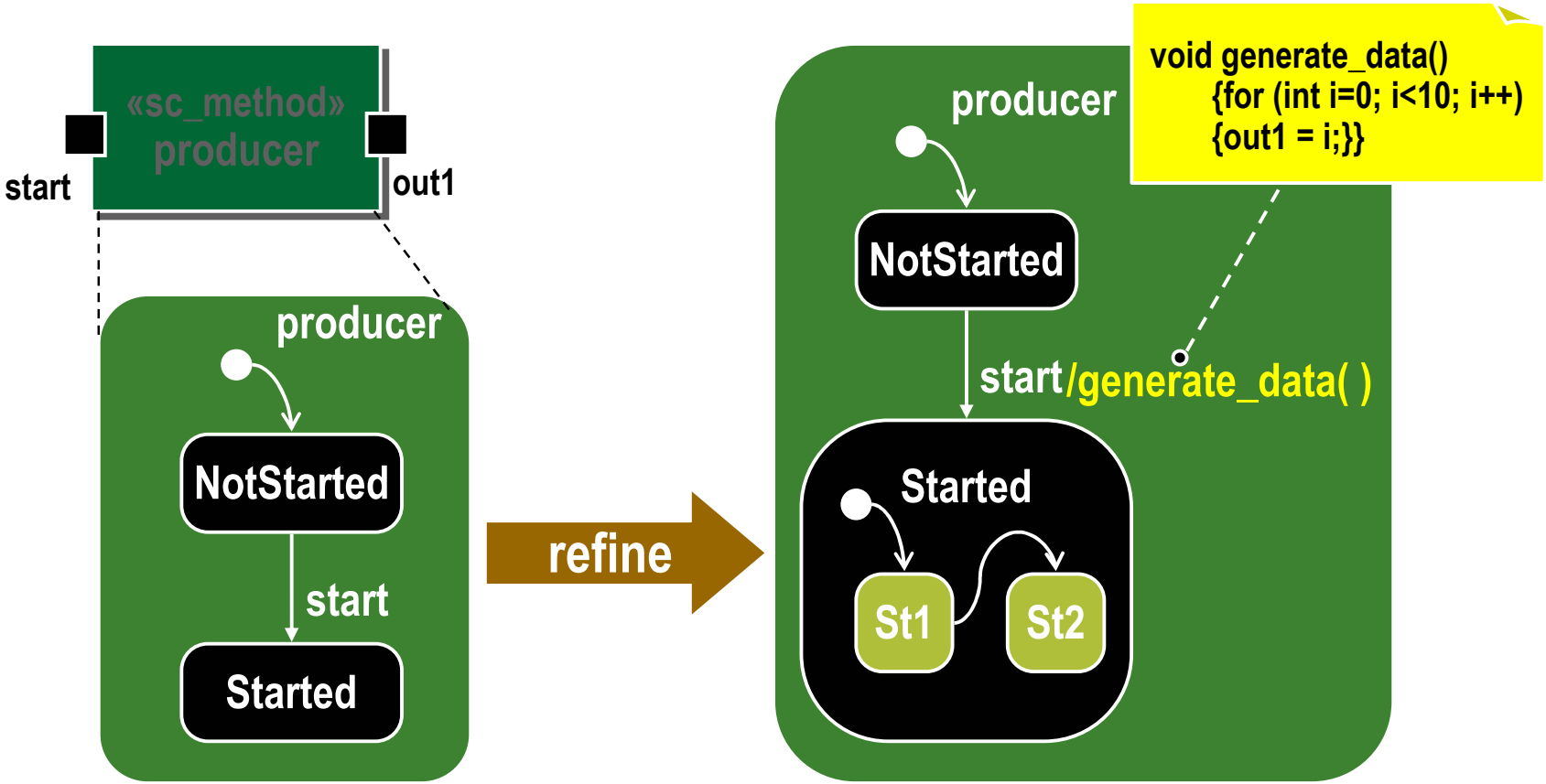
…and its UML Model



Can you spot the architecture?

(Extracted from B. Selic presentation during Summer School MDD For DRES 2004 (Brest, September 2004)

Model Evolution: Refinement

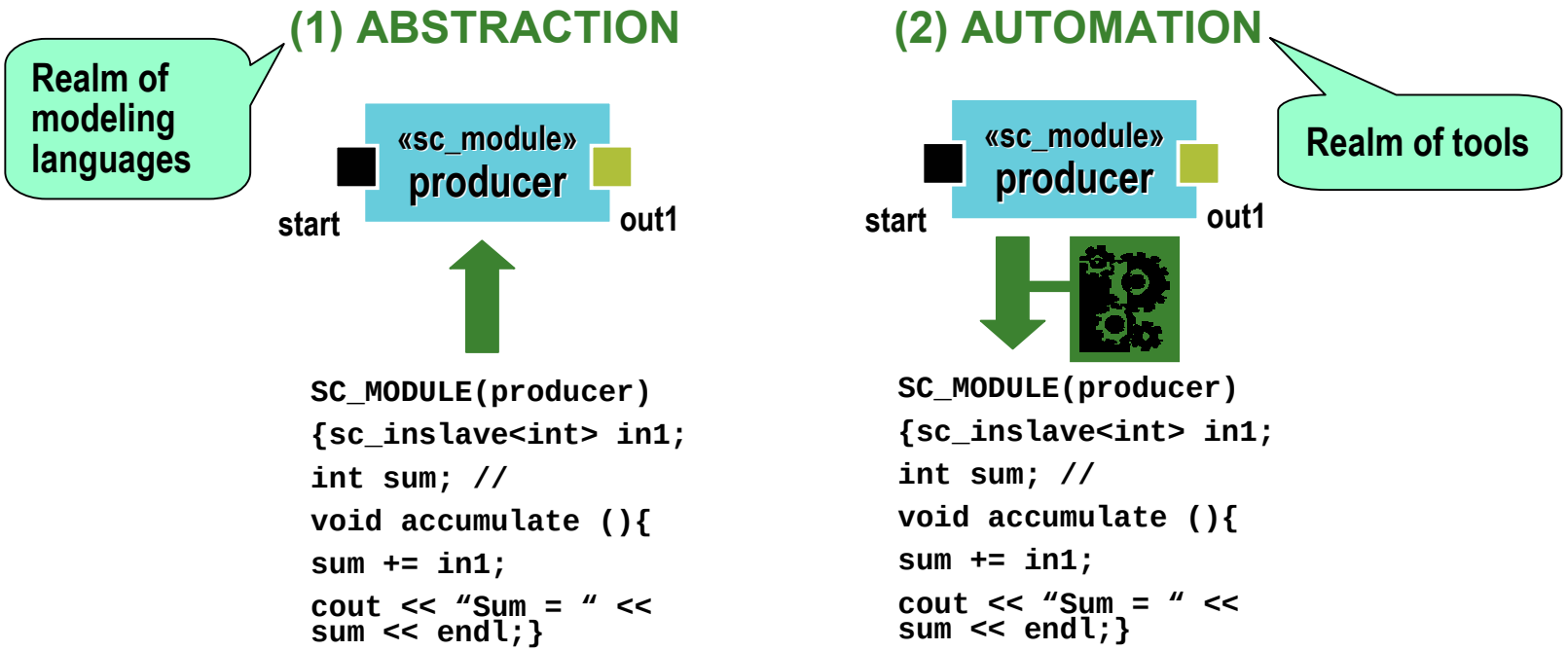


- Models can be refined continuously until the specification is complete

(Extracted from B. Selic presentation during Summer School MDD For DRES 2004 (Brest, September 2004)

Model-Driven Style of Development (MDD)

- An approach to software development in which the focus and primary artifacts of development are models (as opposed to programs)
- Based on two time-proven methods



(Extracted from B. Selic presentation during Summer School MDD For DRES 2004 (Brest, September 2004)

The Remarkable Thing About Software

Software has the rare property that it allows us to directly evolve models into fully-fledged implementations without changing the engineering medium, tools, or methods!

⇒ This allows perfect accuracy of software models, since the model and the system that it models are the same thing.

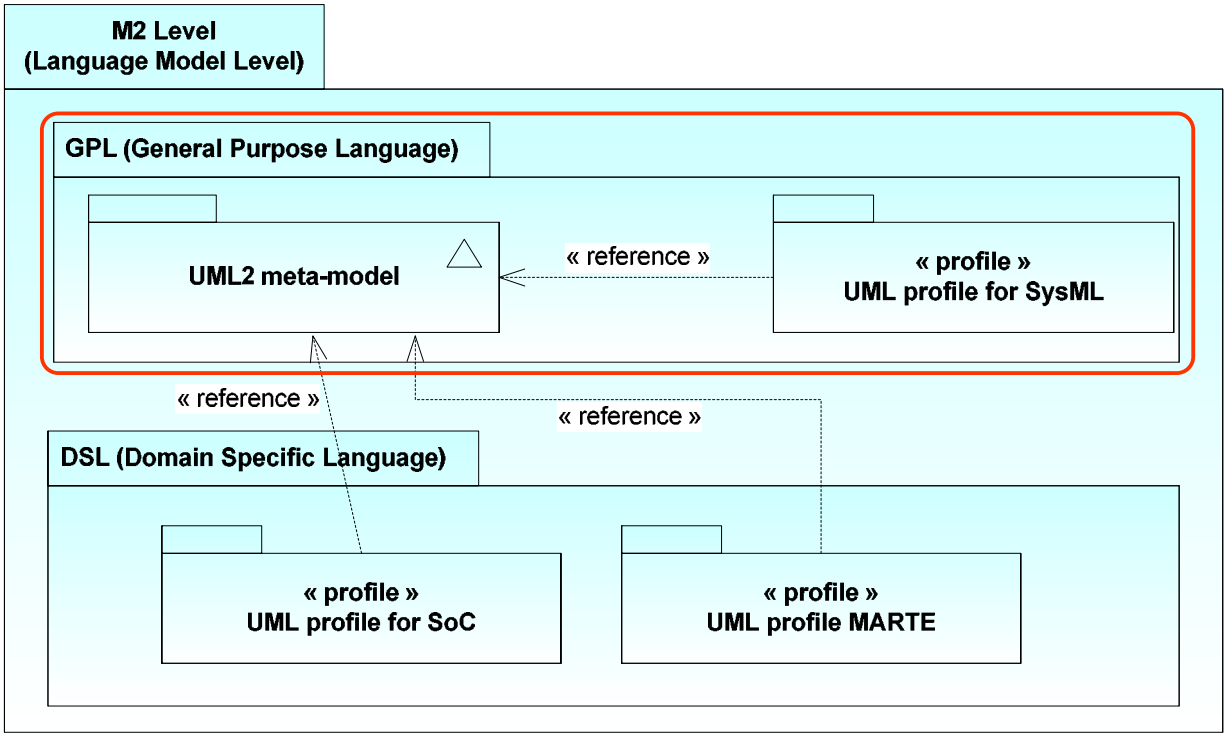
The software model may evolve into the system it was modeling in a seamless process.

(Extracted from B. Selic presentation during Summer School MDD For DRES 2004 (Brest, September 2004)

The UML profile for MARTE

Two Kinds of OMG Modeling Languages

- General Purpose Language (GPL)
  - E.g., UML2, SysML and CCM
- Domain Specific Language
  - E.g. LwCCM, UML profile for SoC and MARTE



General Requirements of the MARTE RFP

- Proposals shall support Modeling and Analysis of Real-Time and Embedded (in short MARTE) systems including its software and hardware aspects.
  - The Proposals will define a metamodel and its underlying UML profile.
  - It shall be possible to use independently software and hardware parts of the profile.
  - It shall comply with existing standards
  - It shall update the SPT profile 1.1

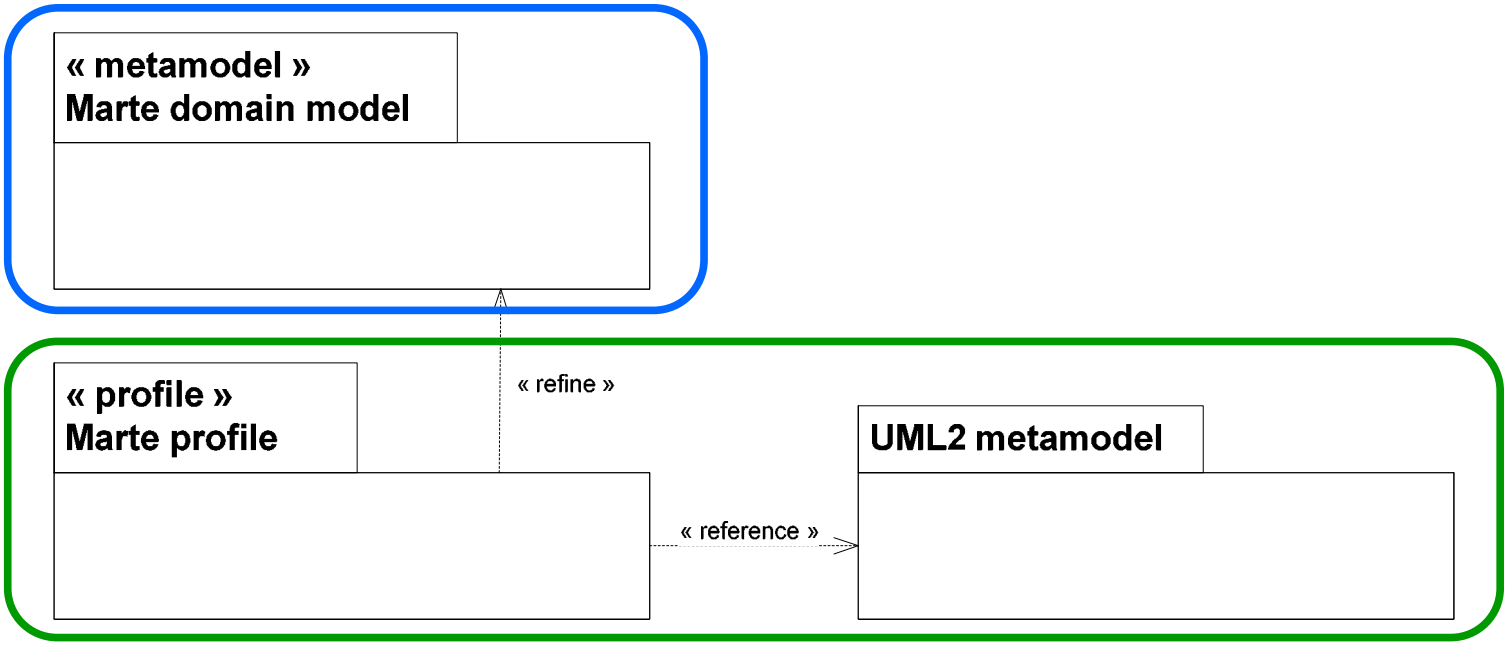
Ref. of MARTE RFP (realtime/05-02-06):
http://www.omg.org/cgi-bin/doc?realtime/05-02-06

The ProMARTE Team

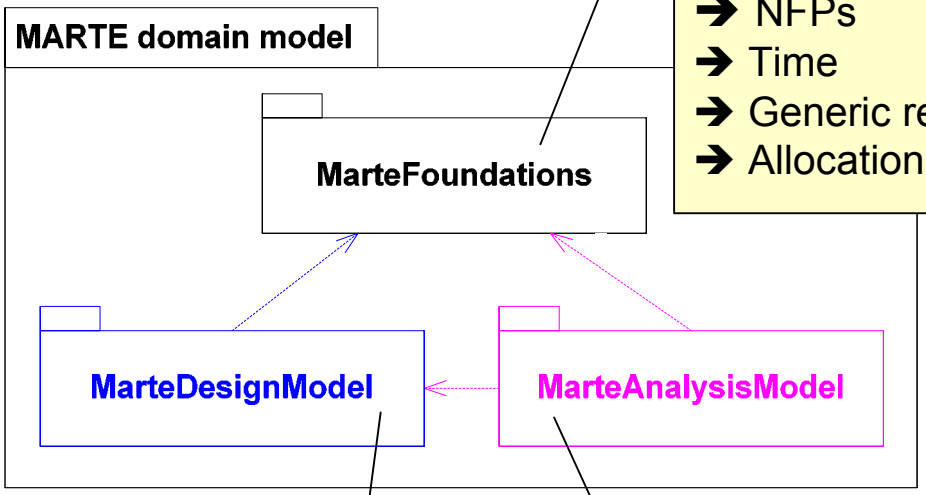
- Public web site => **www.promarte.org**
- Partners
  - Industrials
    - Alcatel*
    - Lockheed Martin*
    - Thales*
  - Tool vendors
    - ARTISAN Software Tools*
    - International Business Machines*
    - Mentor Graphics Corporation*
    - Softeam*
    - Telelogic AB (I-Logix*)
    - Tri-Pacific Software
    - France Telecom
    - No Magic
    - Mathworks
  - Academics
    - Carleton University
    - Commissariat à l'Energie Atomique
    - ESEO
    - ENSIETA
    - INRIA
    - INSA from Lyon
    - Software Engineering Institute (Carnegie Mellon University)
    - Universidad de Cantabria

Design Pattern Adopted for the MARTE Profile

- Stage 1 → Description of MARTE domain models
  - Purpose: Formal description of the concepts required for MARTE
  - Techniques: Meta-modeling
- Stage 2 → Mapping of MARTE domain models towards UML2
  - Purpose: MARTE domain models design as a UML2 extensions
  - Techniques: UML2 profile



MARTE Overview



Foundations for RT/E systems modeling and analysis:
- ➔ CoreElements
- ➔ NFPs
- ➔ Time
- ➔ Generic resource modeling
- ➔ Allocation

Specialization of MARTE foundations for modeling purpose (specification, design…):
- ➔ Generic component model
- ➔ High-level application modeling
- ➔ Software resource modeling
- ➔ Hardware resource modeling

Specialization of foundations for annotating model for analysis purpose:
- ➔ Generic quantitative analysis
- ➔ Schedulability analysis
- ➔ Performance analysis

How to read, use, & implement MARTE

(see Section 2)

Extension Units

NFP	Non-Functional Properties	Section 8
Time	Enhanced Time Modeling	Section 9
GRM	Generic Resource Modeling	Section 10
Alloc	Allocation Modeling	Section 11
GCM	Generic Component Model	Section 12
HLAM	High-Level Application Modeling	Section 13
SRM	Software Resource Modeling	Section 14.1
HRM	Hardware Resource Modeling	Section 14.2
RTM	Real-Time objects Modeling	Section 13
GQAM	Generic quantitative Analysis	Section 15
SAM	Schedulability Analysis	Section 16
PAM	Performance Analysis	Section 17
VSL	Value Specification Language	Annex B
CHF	Clock Handling Facilities	Annex C
RSM	Repetitive Structure Modeling	Annex E
AADL	AADL models with UML	Section A.2

Usage & Compliance Cases vs. Extension Units

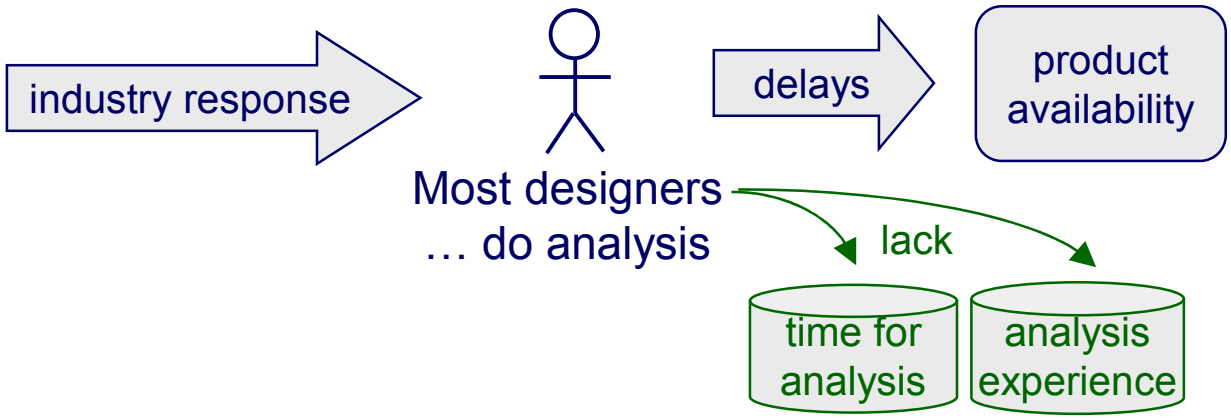
Table 7.2 - Extension Units that must be supported in each Compliance Case

CASE	Level	GRM	NFP	VSL	Time	CHF	SRM	HRM	GCM	Alloc	HLAM	GQAM	PAM	SAM	RSM
Software	Base	X	X		X						X				
	Full			X		X	X		X						
Hardware	Base	X	X		X			X							
	Full			X		X			X	X					X
System	Base	X	X		X					X	X				
	Full			X		X	X	X	X						X
Performance	Base	X	X		X							X	X		
	Full			X		X									
Schedulability	Base	X	X		X							X		X	
	Full			X		X									
Infrastructure	Base	X	X		X		X								
	Full			X		X				X	X				
Methodologist	Base	X	X		X			X			X	X			
	Full			X		X	X		X	X			X	X	X

Introduction to Model-based RT/E analysis

Industry Landscape of Analysis Practice

- Why “non-functional analysis” is important?
  - Early: avoid design mistakes, asses design trade-offs…
  - Later: evaluate the impact of modifications,…
- But,… doing analysis is hard and time-consuming!...
  - From 40% to 50% of development costs [Hum02], …

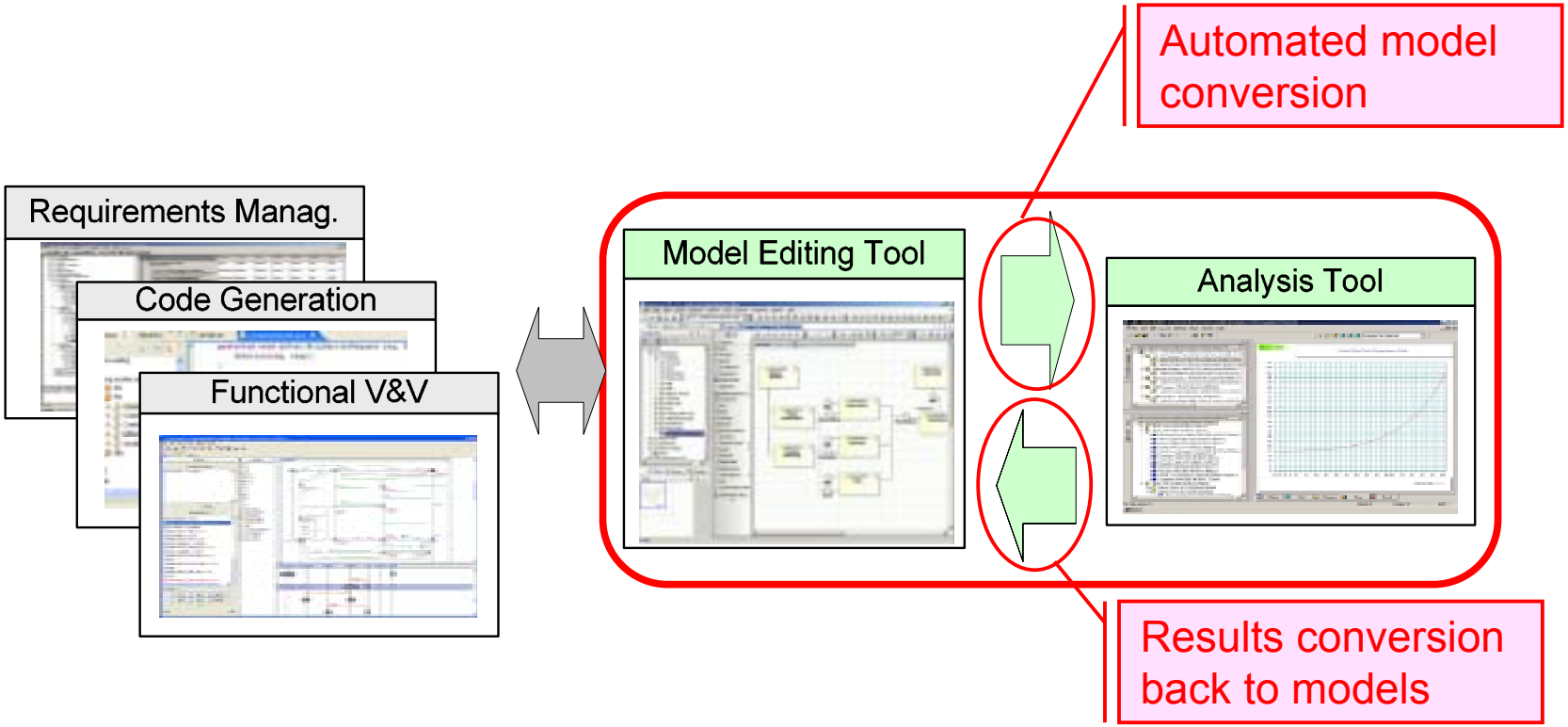


We need new approaches to cope with analysis complexity!

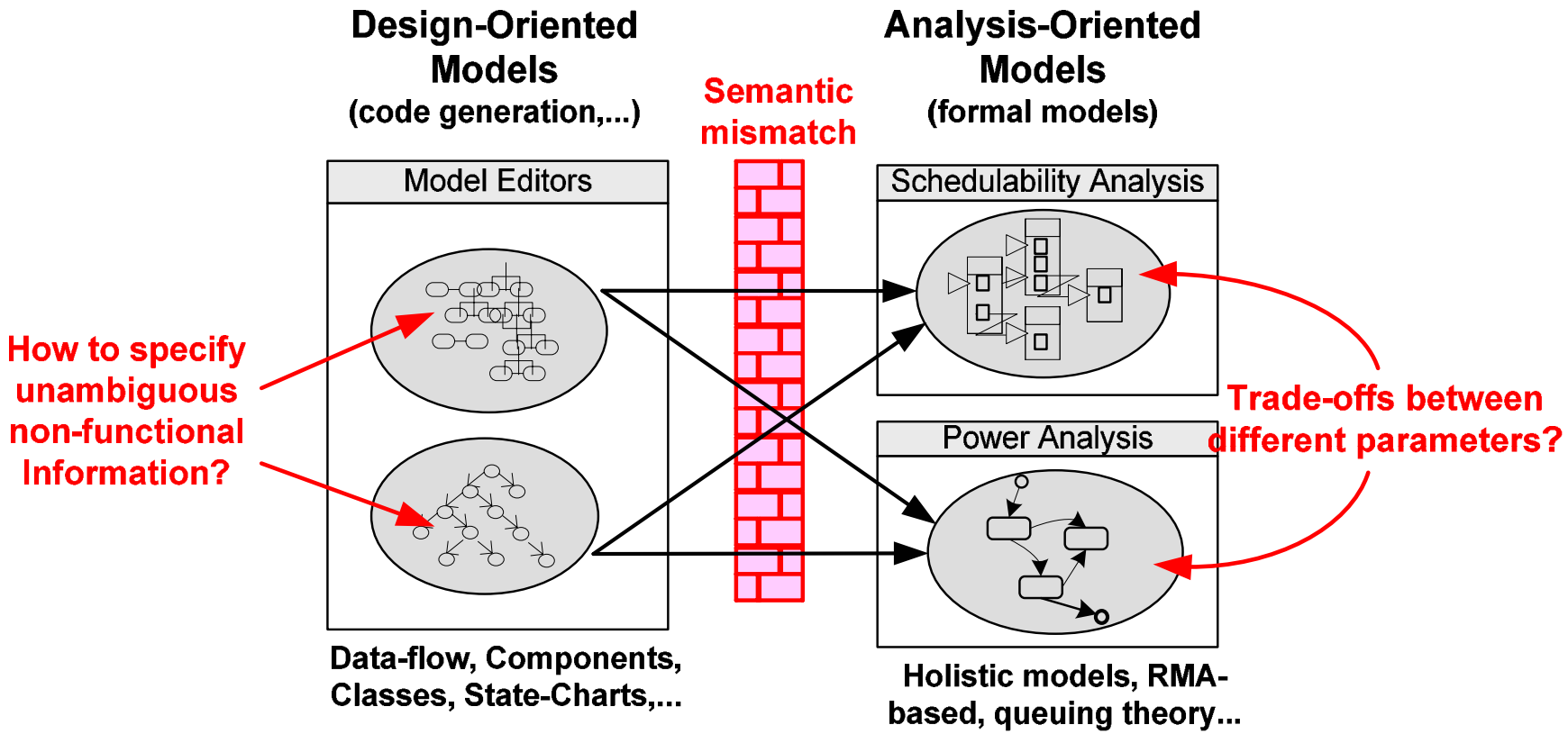
[Hum02] Humphrey, W. S., "The Payoff from Software Quality", In Computerworld, 2002

Goals for Model-Based RTES Analysis

- Ability to specify non-functional information in models
  - ➔ Automate the generation of analysis models (time savings!)
  - ➔ Not deep knowledge of mathematical aspects.



Needs for Model-Based RTES Analysis



Goals in Non-Functional (or Quantitative) Analysis

It offers a mathematically-sound way to calculate NFPs of interest based on other available NFPs and the system behavior

- Different Goals for Evaluate & Verify System Architectures
  - Point evaluation of the output NFPs for a given operating point defined by input NFPs
  - Search over the parameter space for feasible or optimal solutions
  - Sensitivity analysis of some output results to some input parameters
  - Scalability analysis: how the system performs when the problem size or the system size grow.

MARTE Features for Quantitative Analysis

- Improvements w.r.t. SPT
  - Extend implementation and scheduling models
    - e.g. distributed systems, hierarchical scheduling
  - Extend the set of analysis techniques supported
    - e.g. offset-based techniques
  - Extend timing annotations expressiveness
    - Overheads (e.g. messages passing)
    - Response times (e.g. BCET & ACET)
    - Timing requirements (e.g. miss ratios and max. jitters)
- New features w.r.t. SPT
  - Support for sensitivity analysis
  - Improve modeling reuse and component-based design.
  - Support of the “Y-chart” approach: application vs. platform models

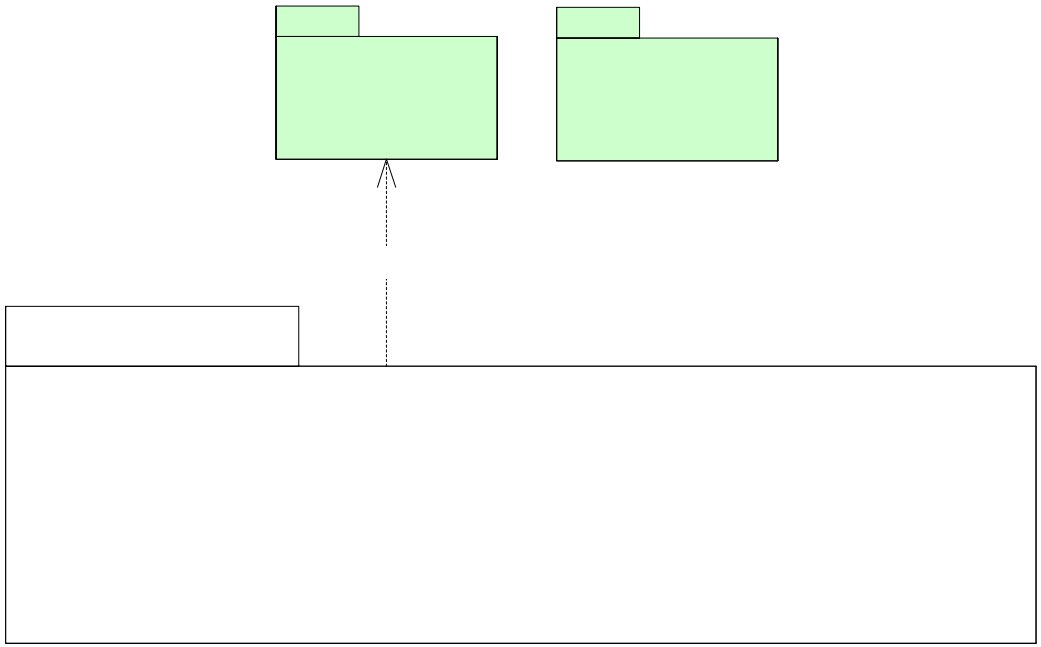
UML-Based Analysis Foundations

- GQAM Profile factorizes common constructs and NFPs
  - Stereotypes define “analysis” abstractions
    - workload events, scenarios,…
    - schedulable entities, shared resources, processing nodes, schedulers…
  - Stereotype attributes define pre-defined NFPs
    - e.g. event arrival patterns, end-to-end deadlines, wcet-bcet-acet,…
- The analysis sub-profiles define model well-formedness rules
  - It includes “constraints” to construct “analyzable” models, w.r.t…
  - ”Analysis Model Viewpoints” (e.g., schedulability analysis viewpoint)
  - Specialized constraints must be refined by technique-specific approaches

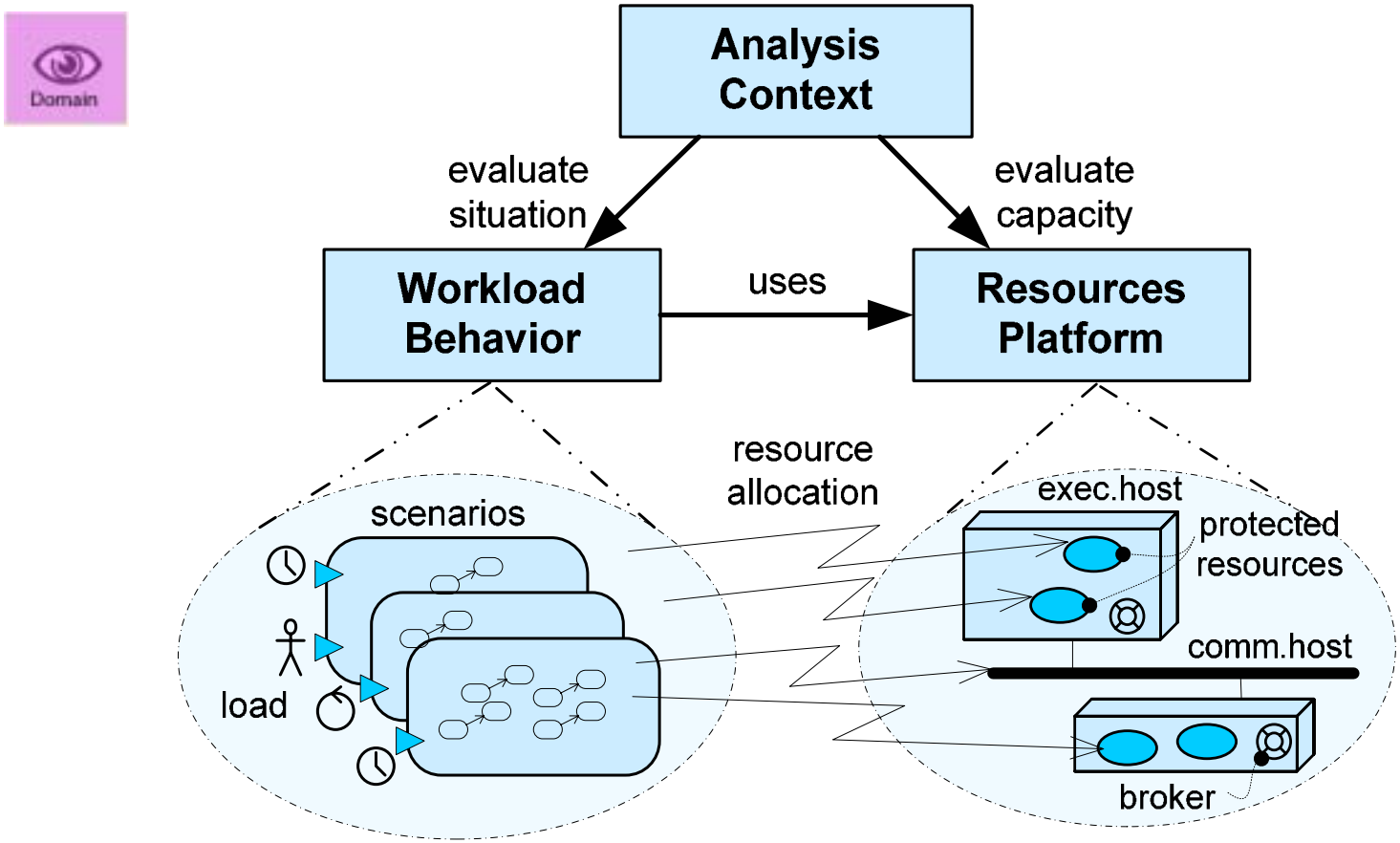
The MARTE analysis sub-profiles provide standard constructs to map UML models on well-established analysis techniques

➔MARTE “Foundations” and “GQAM” allow for extending to further techniques

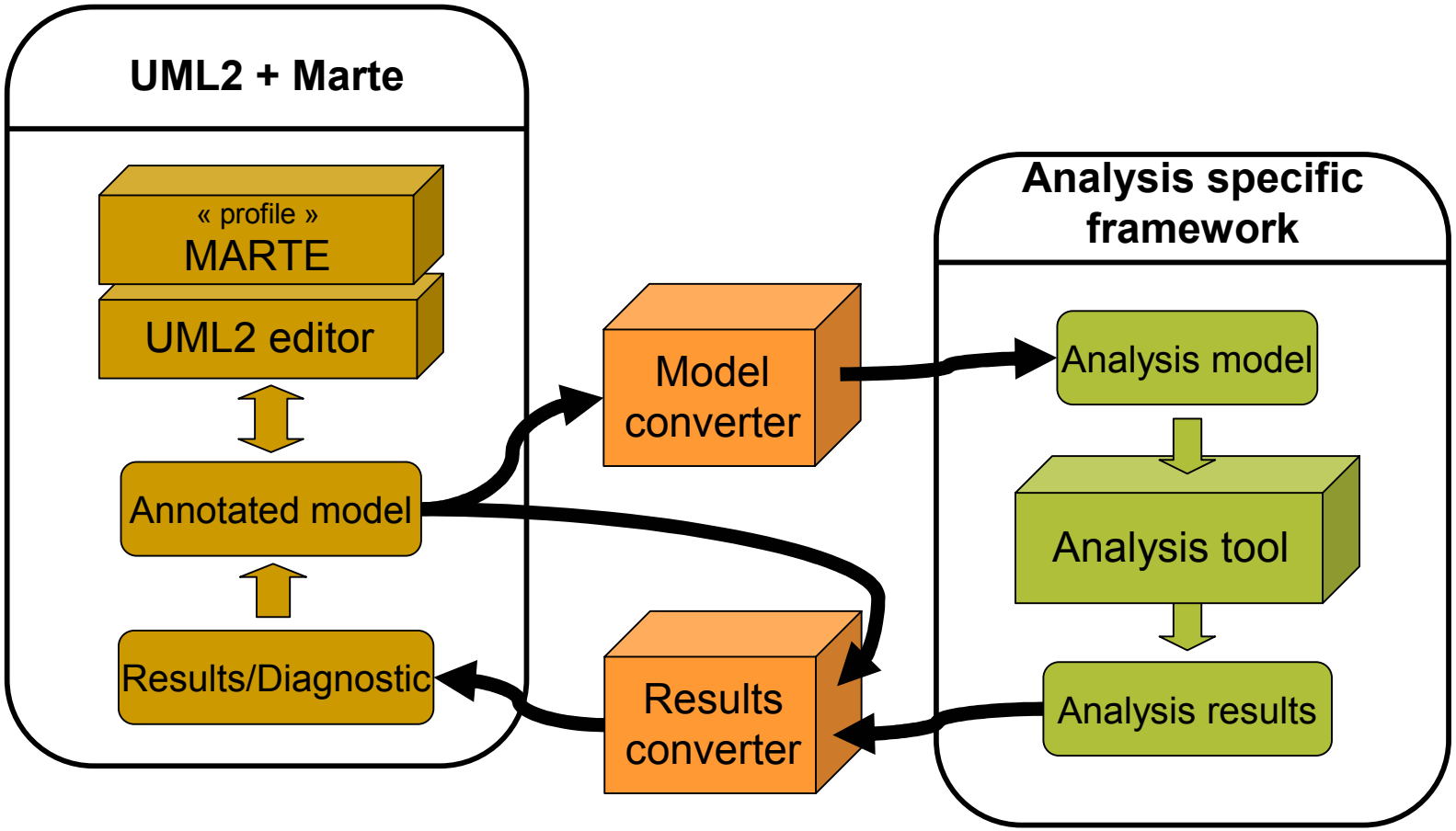
GQAM: Dependencies and Architecture



GQAM: Analysis Modeling Structure



Processing schema for model-based analysis



Model-based schedulability analysis

Key reasoning for the use of UML in the scheduling analysis of RTE Systems

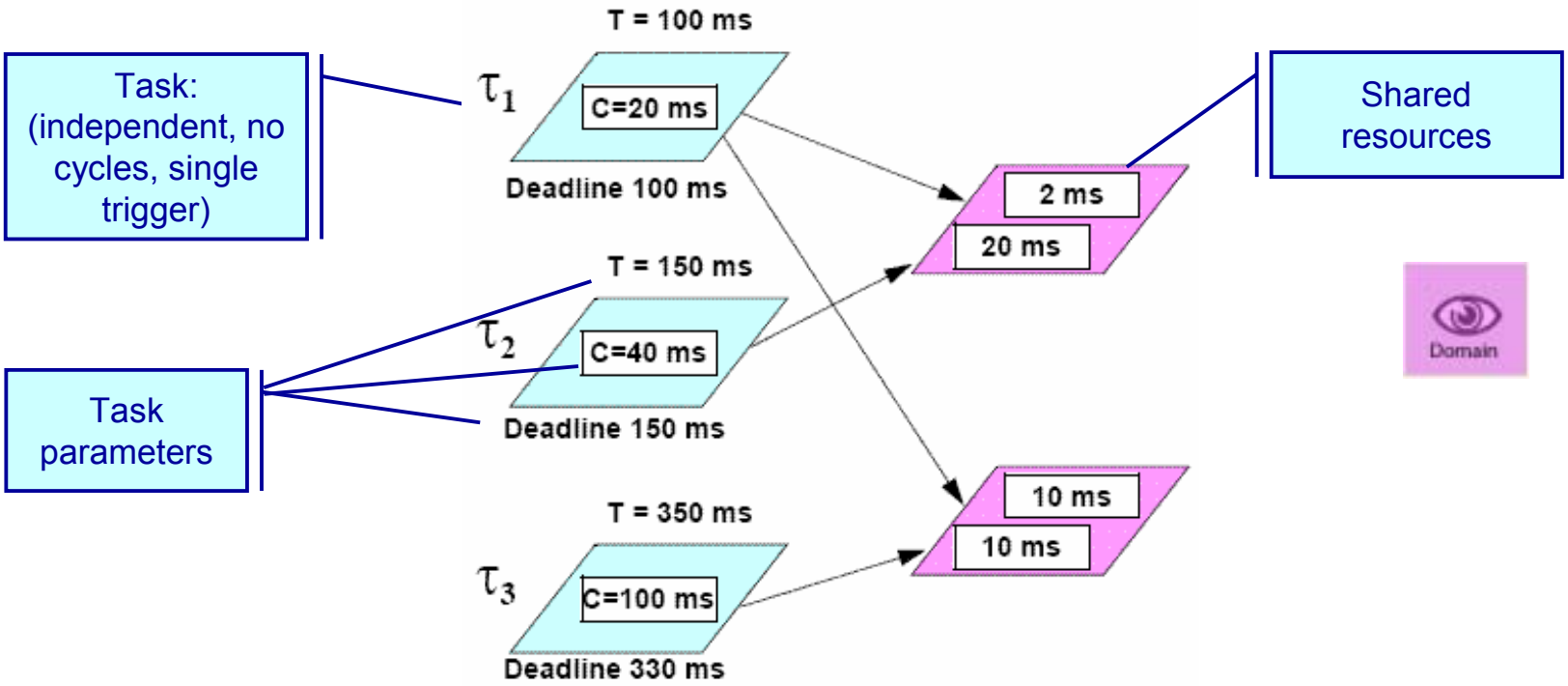
- UML is a standard semi-visual language for conceptual modeling that enables the usage of a Model Based approach for software and system engineering.
- MDD and UML have been broadly introduced and used in principle by the software engineering community, and have reached a significant number of practitioners and tool support.
- To take benefit of this in the RTE domain, they need to be capable of supporting the necessary (at least timing) verifications.
- This leads to the necessity of model based scheduling analysis techniques.
- As well as the necessity to have the modeling elements to describe the platform, the interacting environment and the timing requirements.

Schedulability Analysis

Provides the ability to evaluate time constraints and guarantee worst-case behavior of a system or particular piece of software

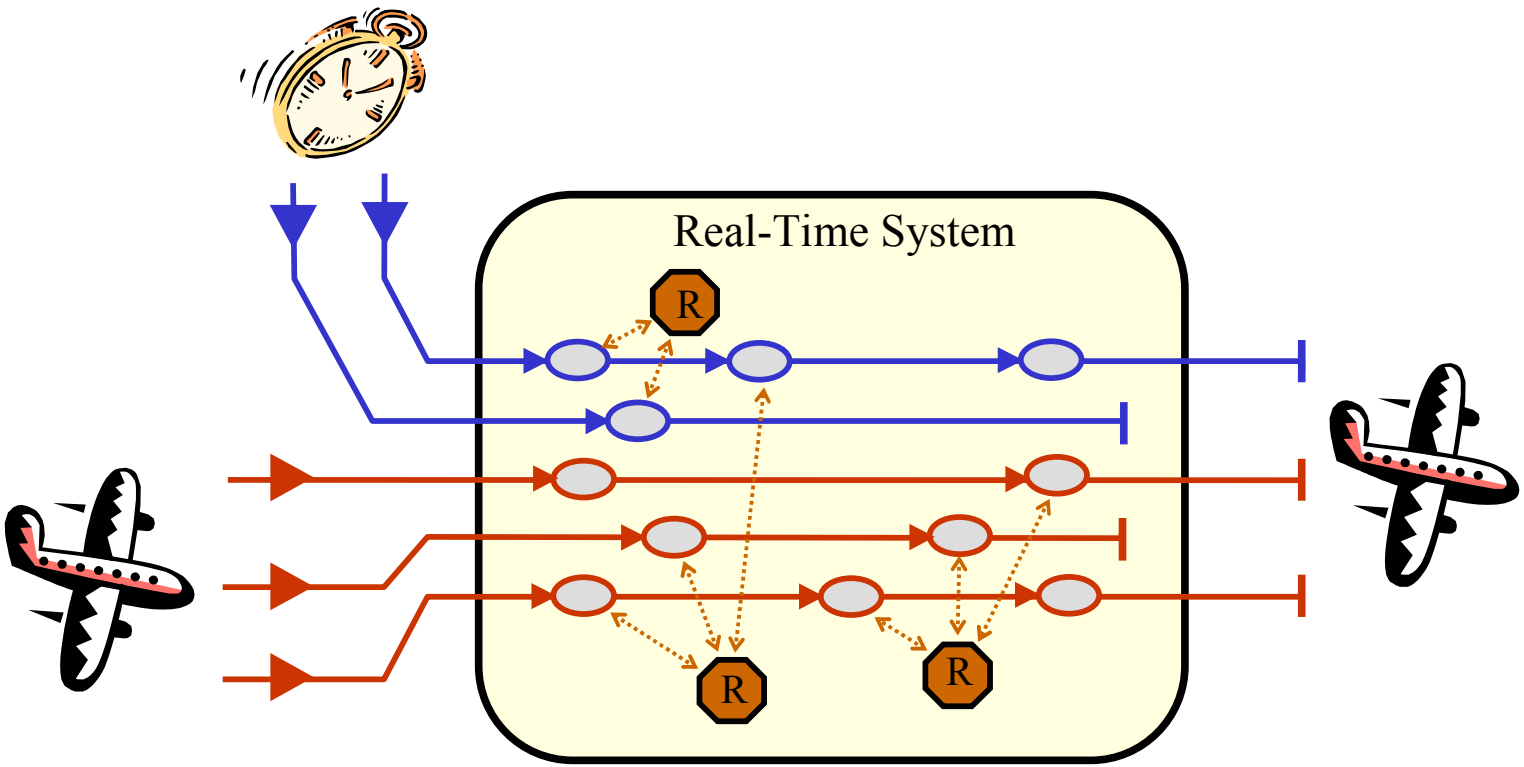
- Schedulability analysis offers:
  - Offline guarantees. E.g., worst-case latencies and worst-case resource usage.
  - At different development stages.
    - Early analysis: to detect potentially unfeasible real-time architectures.
    - Later analysis: to discover temporal-related faults, or to evaluate the impact of migrations (e.g., scheduling strategies).
- Provide answer to questions such as for example...
  - Will we miss any deadline if we switch a processor from a normal operation mode to a lower-consumption mode?
  - If yes, how can we modify task workloads for allowing our system to still work?

A Simple Example (Classical Scheduling Theory)



- Typically analyzed with RMA:
  - Critical instant calculation
  - Utilization bound test or Response time calculation for the first deadline.

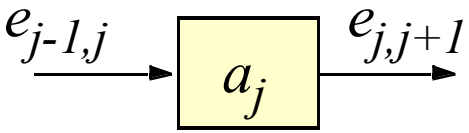
More general approach



Scenario(instance) based, distributed, control-flow dependencies

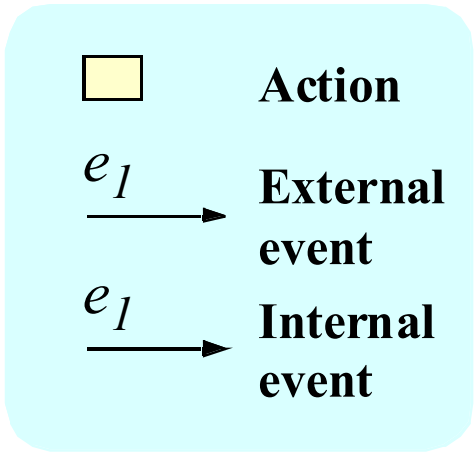
Distributed system model

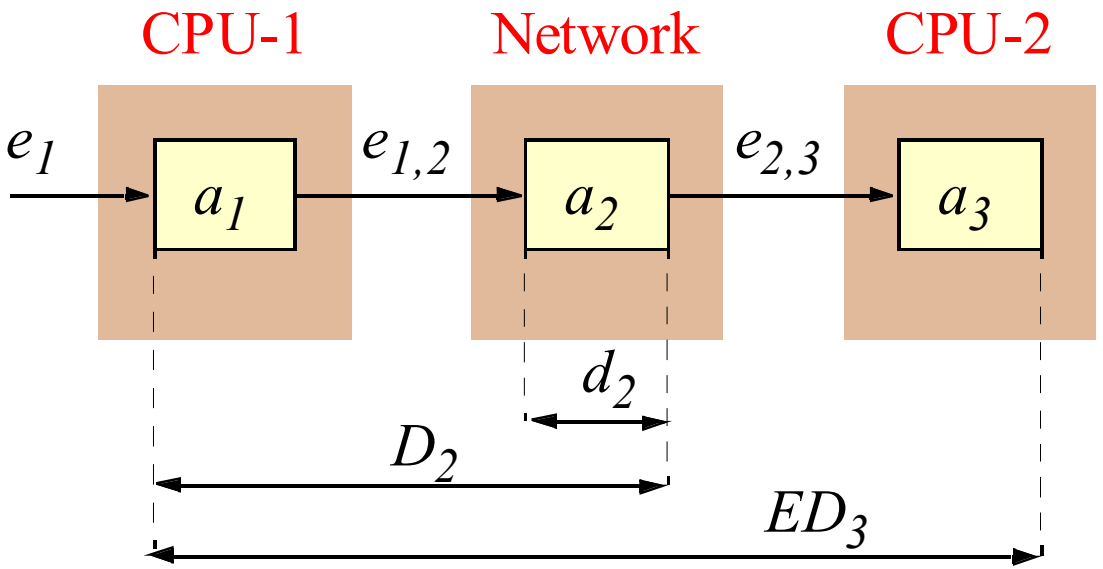
Linear Action:



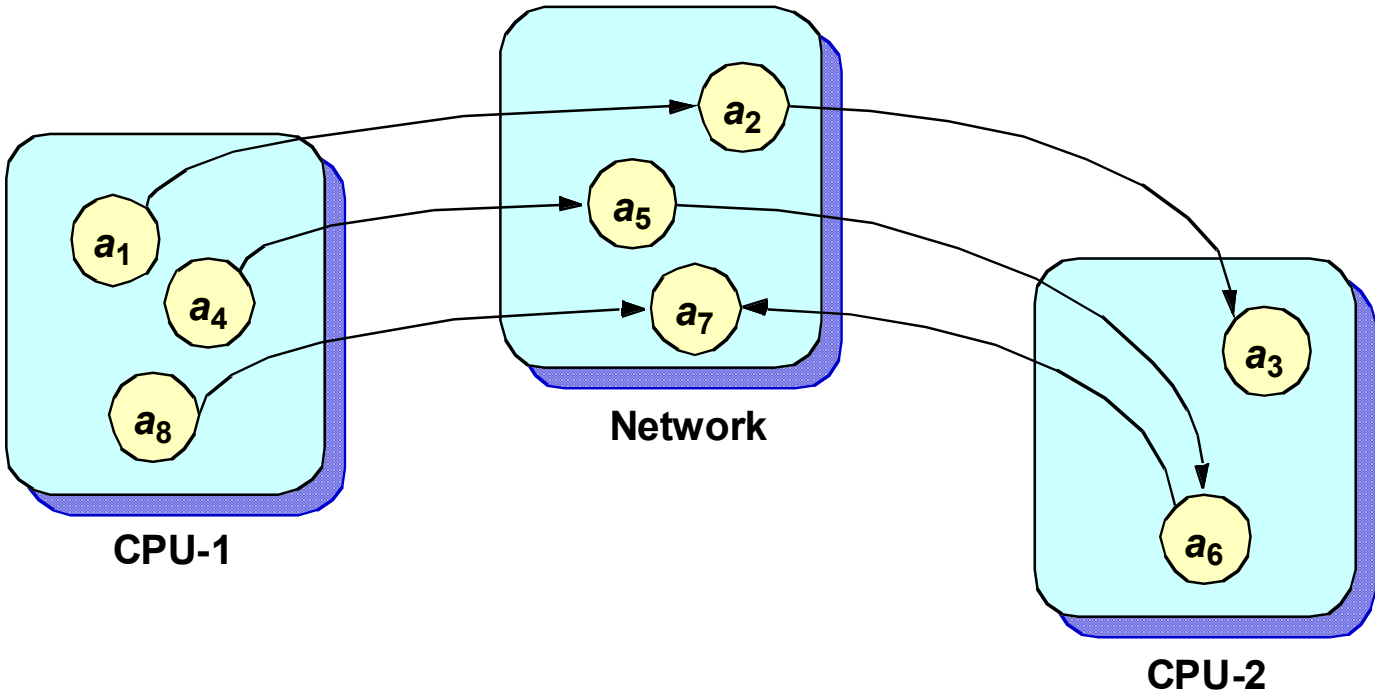
$$T_{j-1,j} = T_j = T_{j,j+1}$$

Linear Response to an Event:



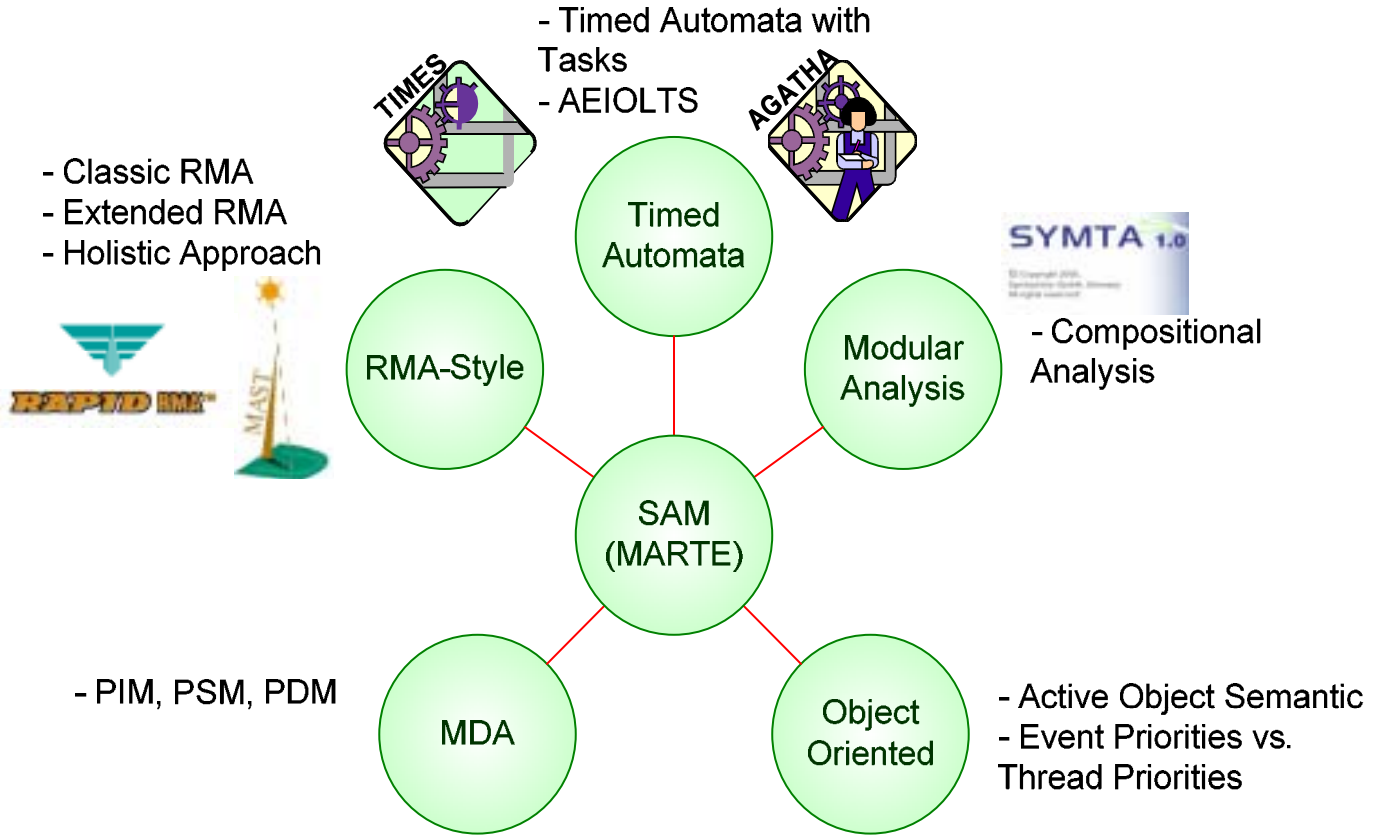


Jitter in distributed systems



- Jitter in one processing resource depends on the response times in other processing resources
- Response times depend on jitters

SAM: Integration Different Approaches

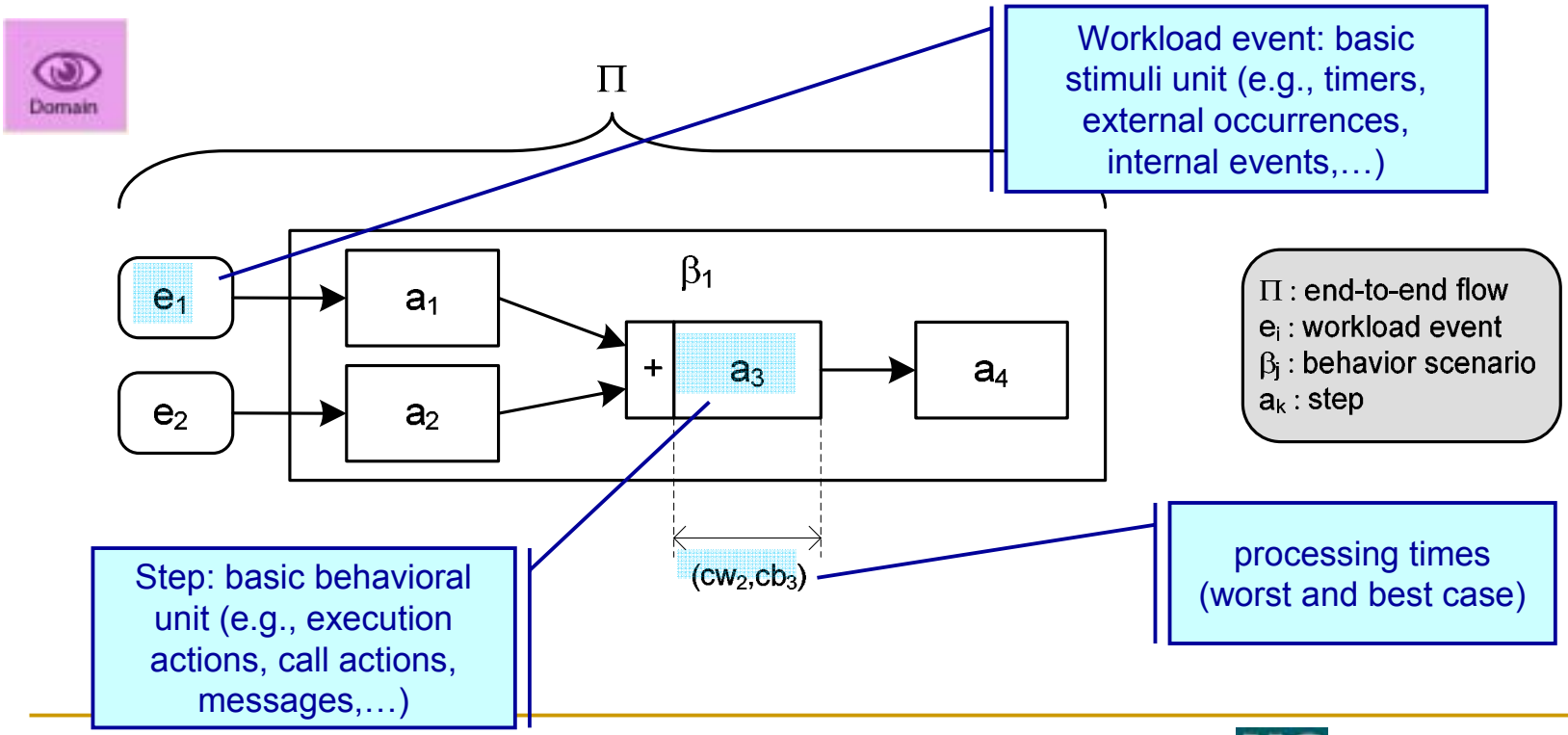


Other Sched. Analysis tools: Livedevices' Real-Time Architect, CoMET from VaST, Vector's CANAlyzer…

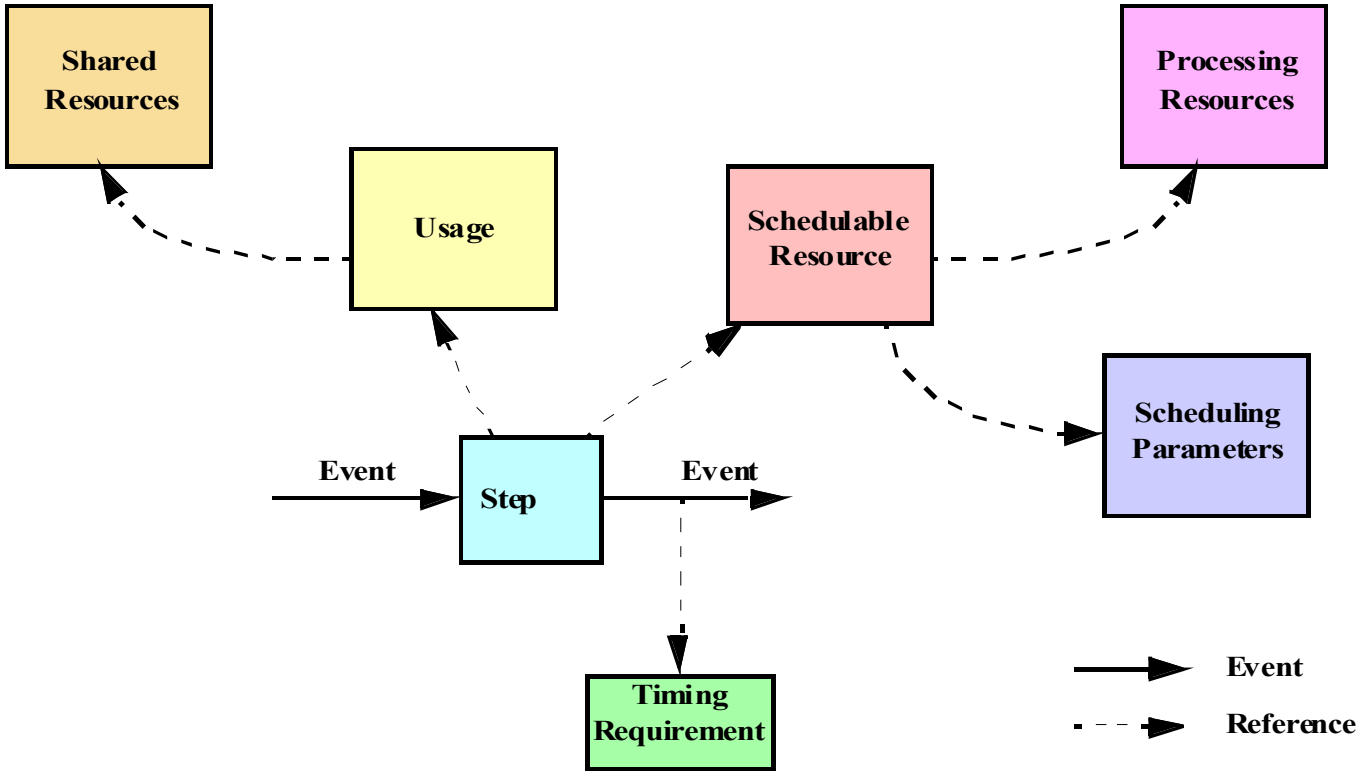
SAM: The Notion of End-To-End Flow

An "End-To-End Flow" is the basic workload unit to be evaluated by schedulability analysis tools.

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

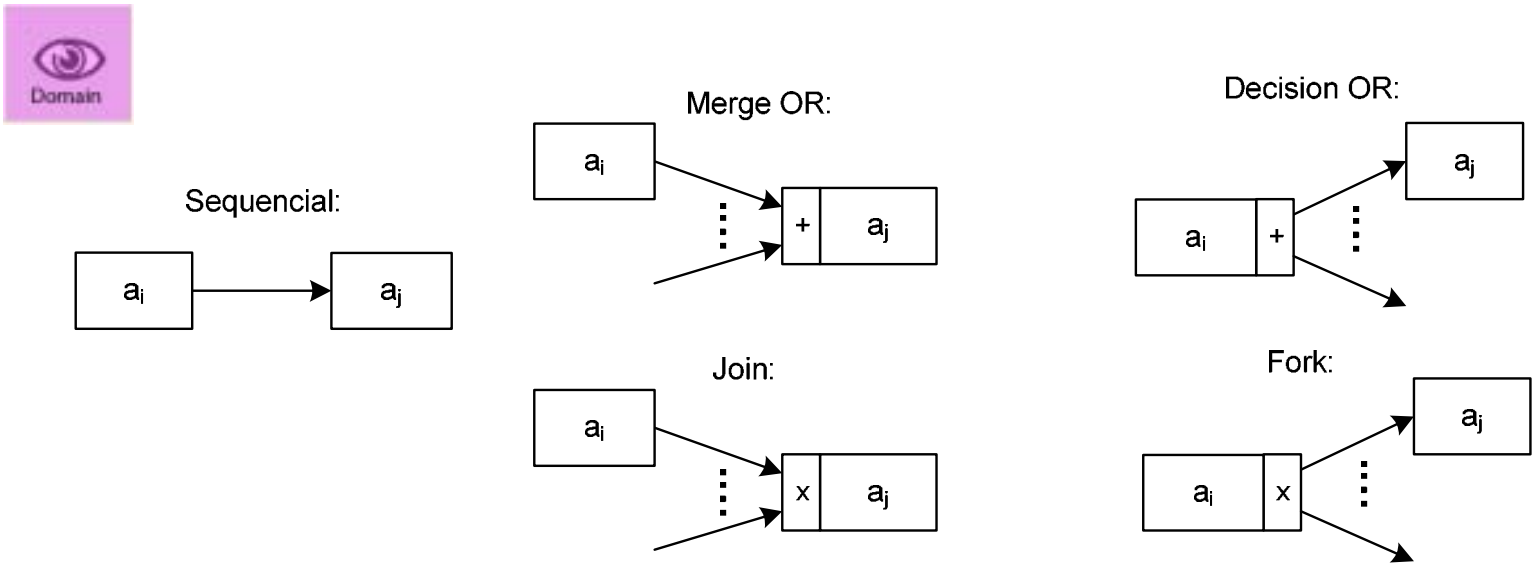


End-to-end flows (from MAST)

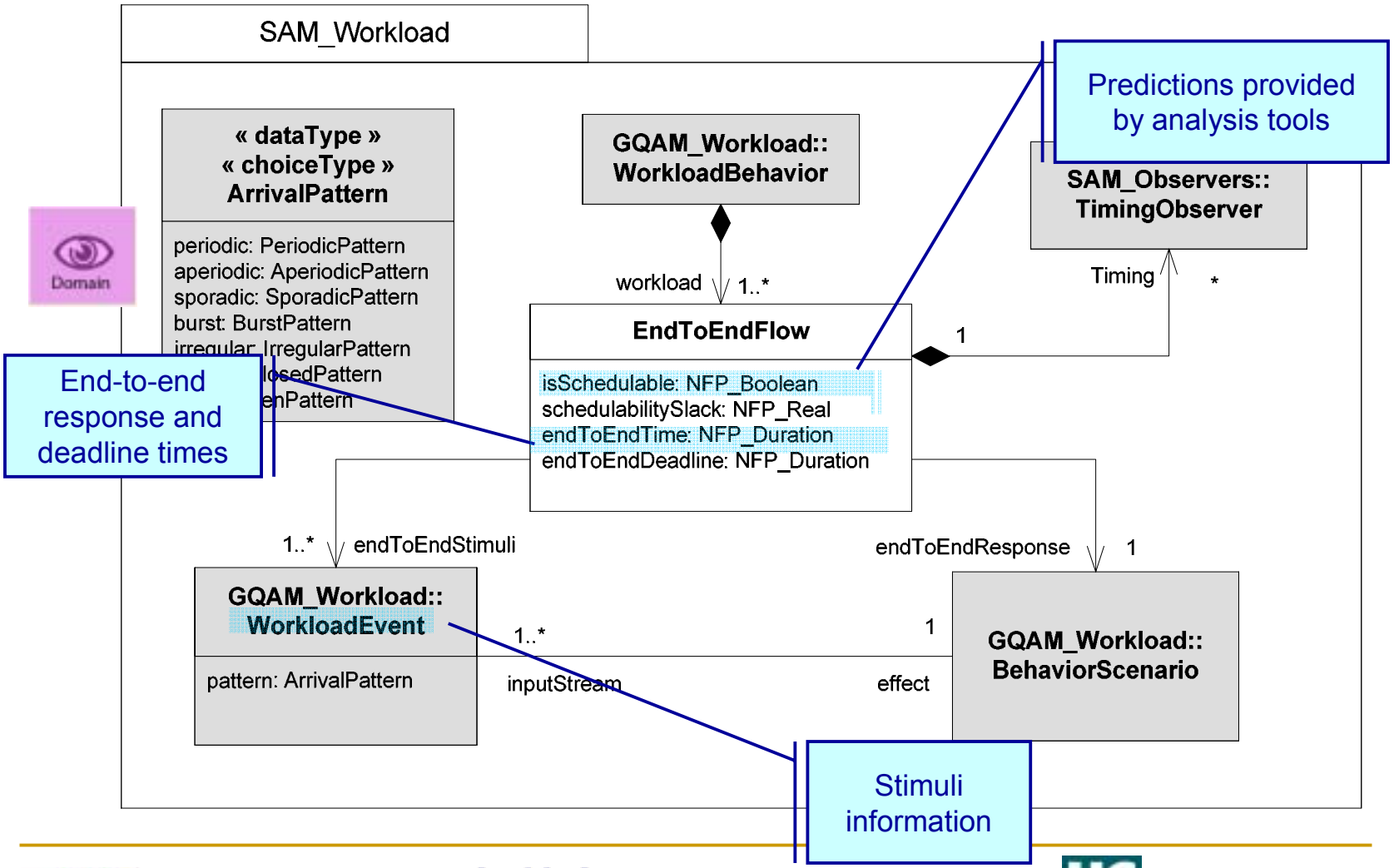


SAM: Precedence Relations

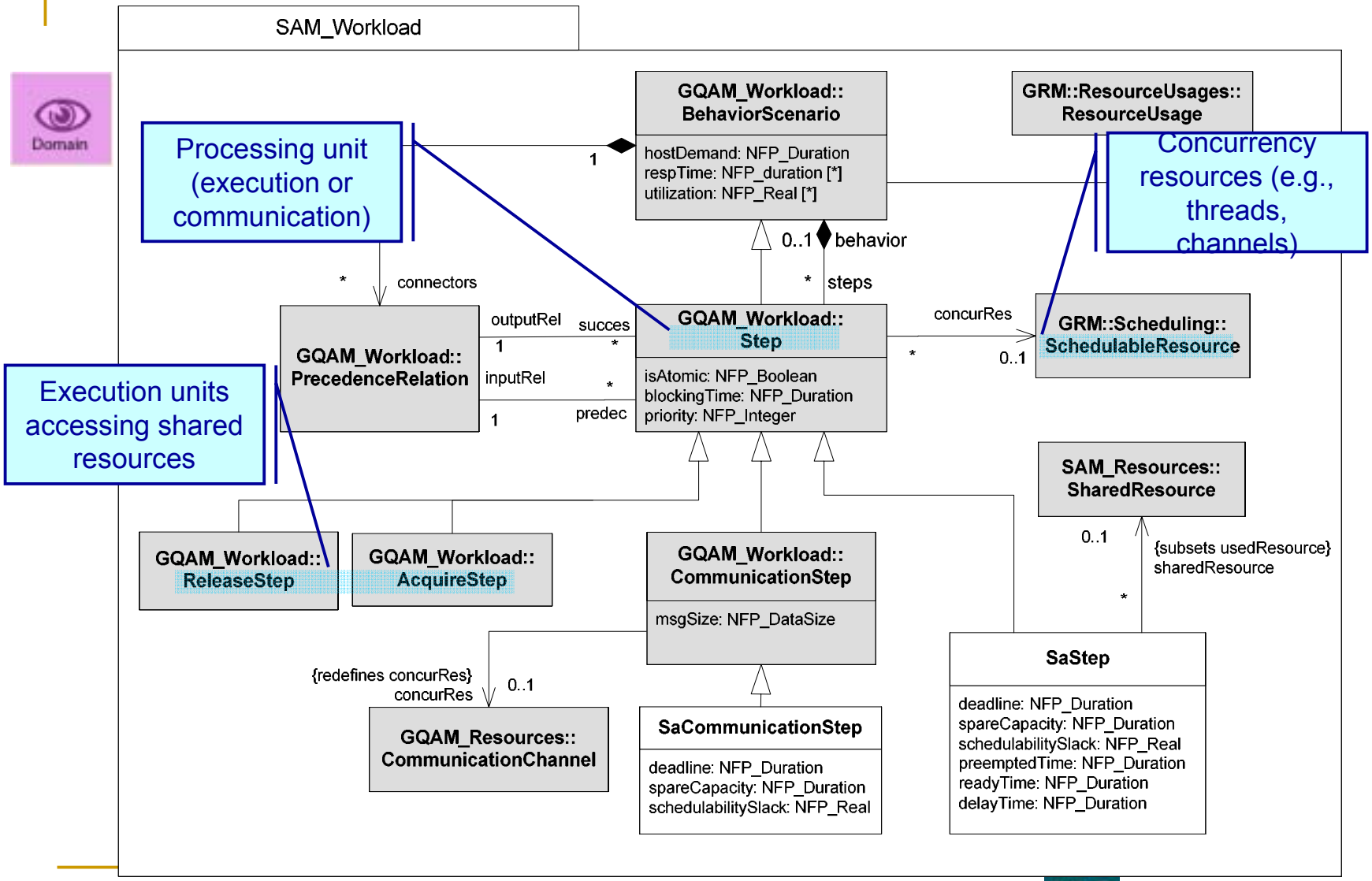
Execution and communication steps may be causally related by one of the following precedence relations:



SAM: Workload Domain Metamodel (end-to-end)



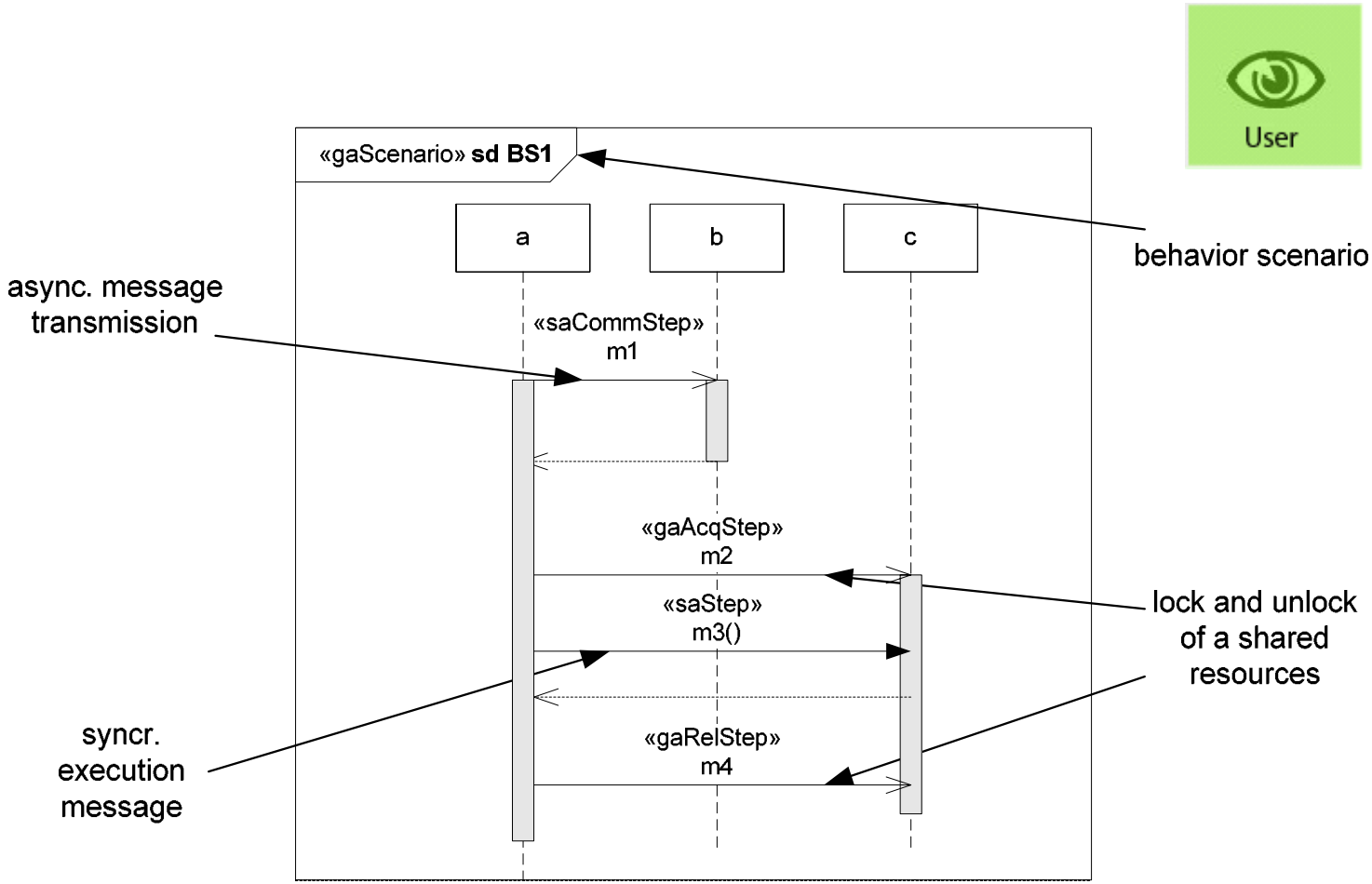
SAM: Workload Domain Metamodel (detailed behavior.)



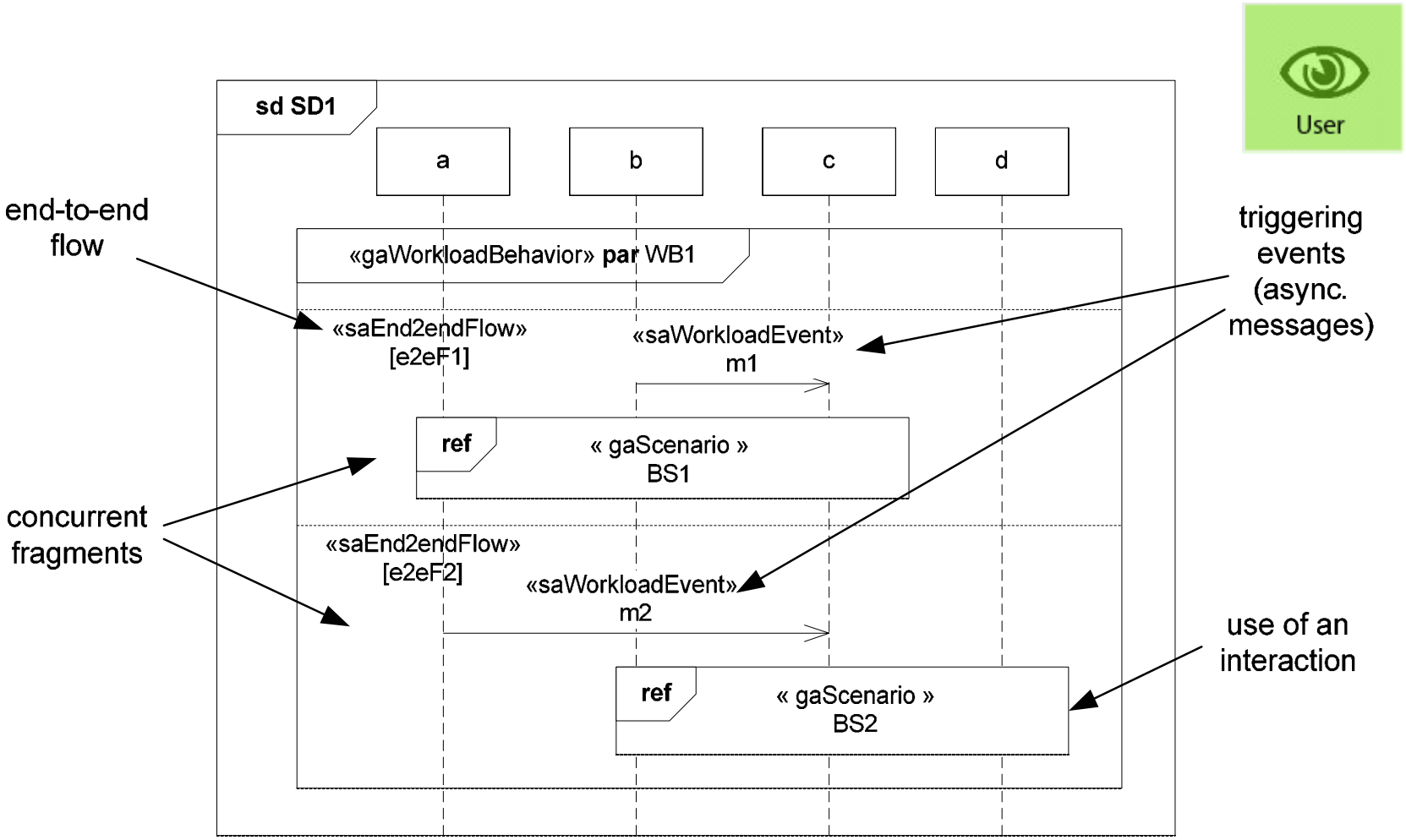
SAM: Example of Stereotype Extensions Usage

SAM Domain Model	SAM Stereotype	UML Metaclasses	Context
WorkloadBehavior	GaWorkloadBehavior	UML::Interactions::Fragments:: **CombinedFragments**	Modeled in a high-level interaction
EndToEndFlow	SaEnd2EndFlow	UML::Interactions::Fragments:: **InteractionOperand**	Modeled in a high-level interaction
WorkloadEvent	GaWorkloadEvent	UML::Interactions::BasicInteractions:: **Message**	Modeled in a high-level interaction
BehaviorScenario	GaScenario	UML::Interactions::BasicInteractions:: **Interaction**	Modeled as a low-level interaction nested within a higher-level interaction
Step **CommunicationStep** **ReleaseStep** **AcquireStep**	SaStep SaCommStep GaRelStep GaAcqStep	UML::Interactions::BasicInteractions:: **Message**	Messages in low-level interactions

SAM: Examples of Behavior Annotations

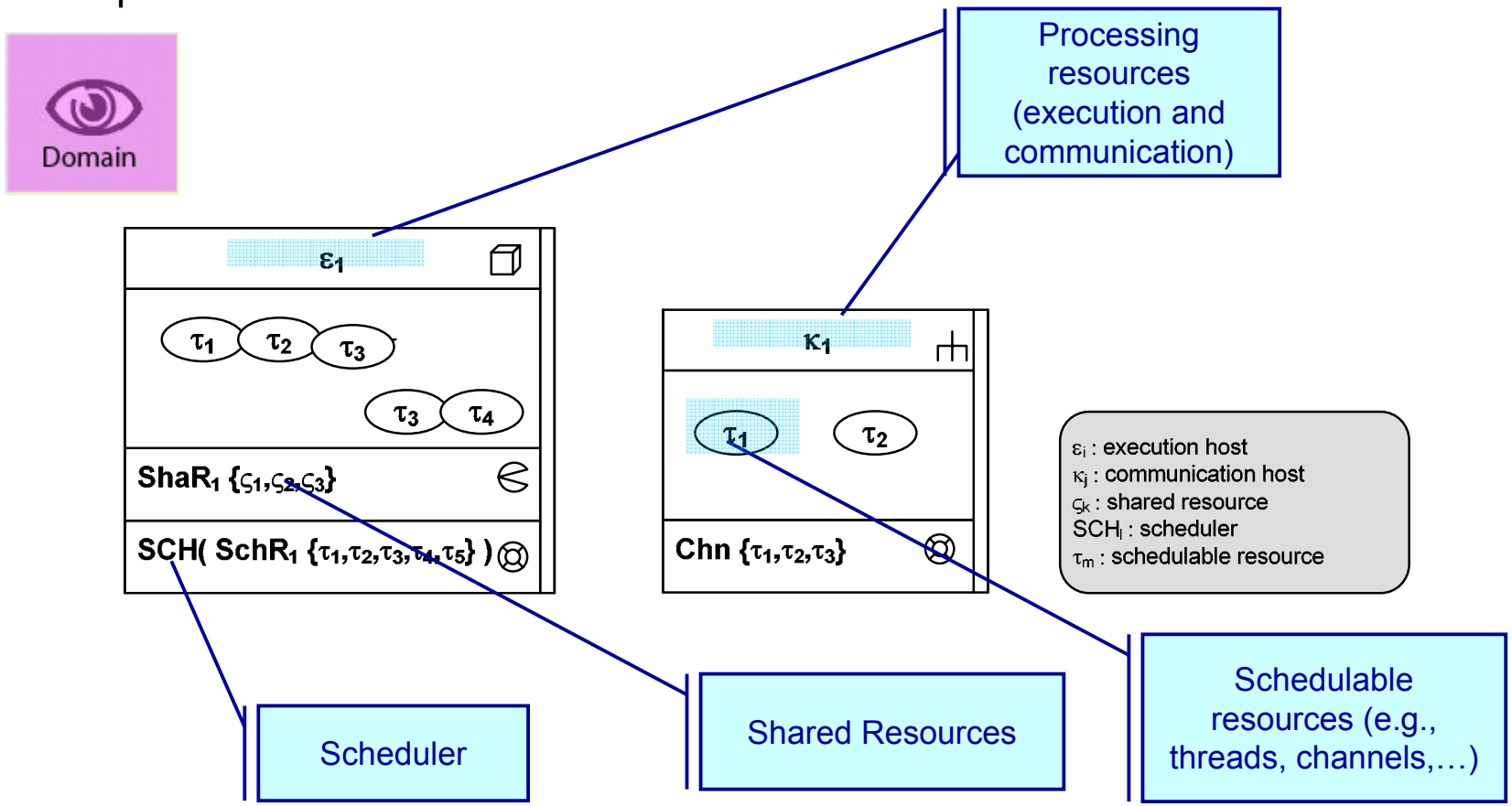


SAM: Example of Workload Annotations

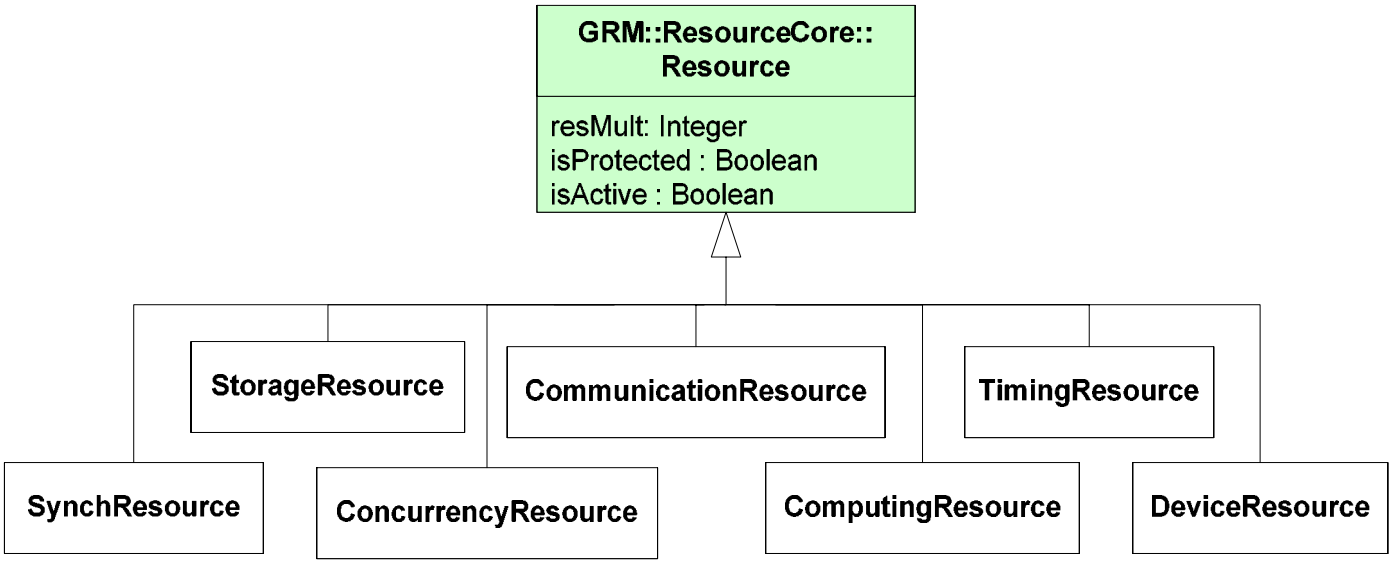


SAM: Resources Concepts

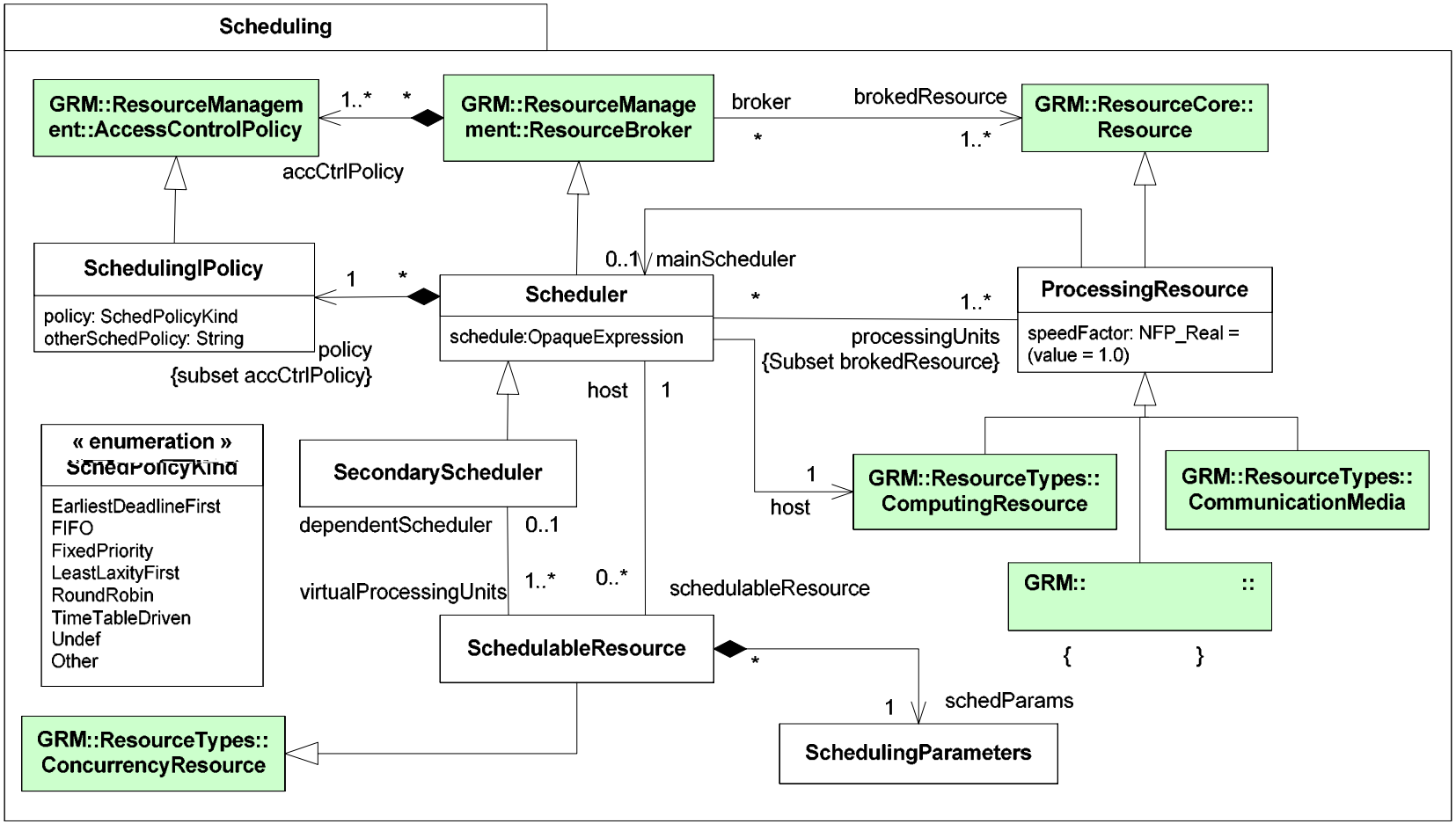
- Provide additional (analysis-specific) annotations to annotate resources platform models



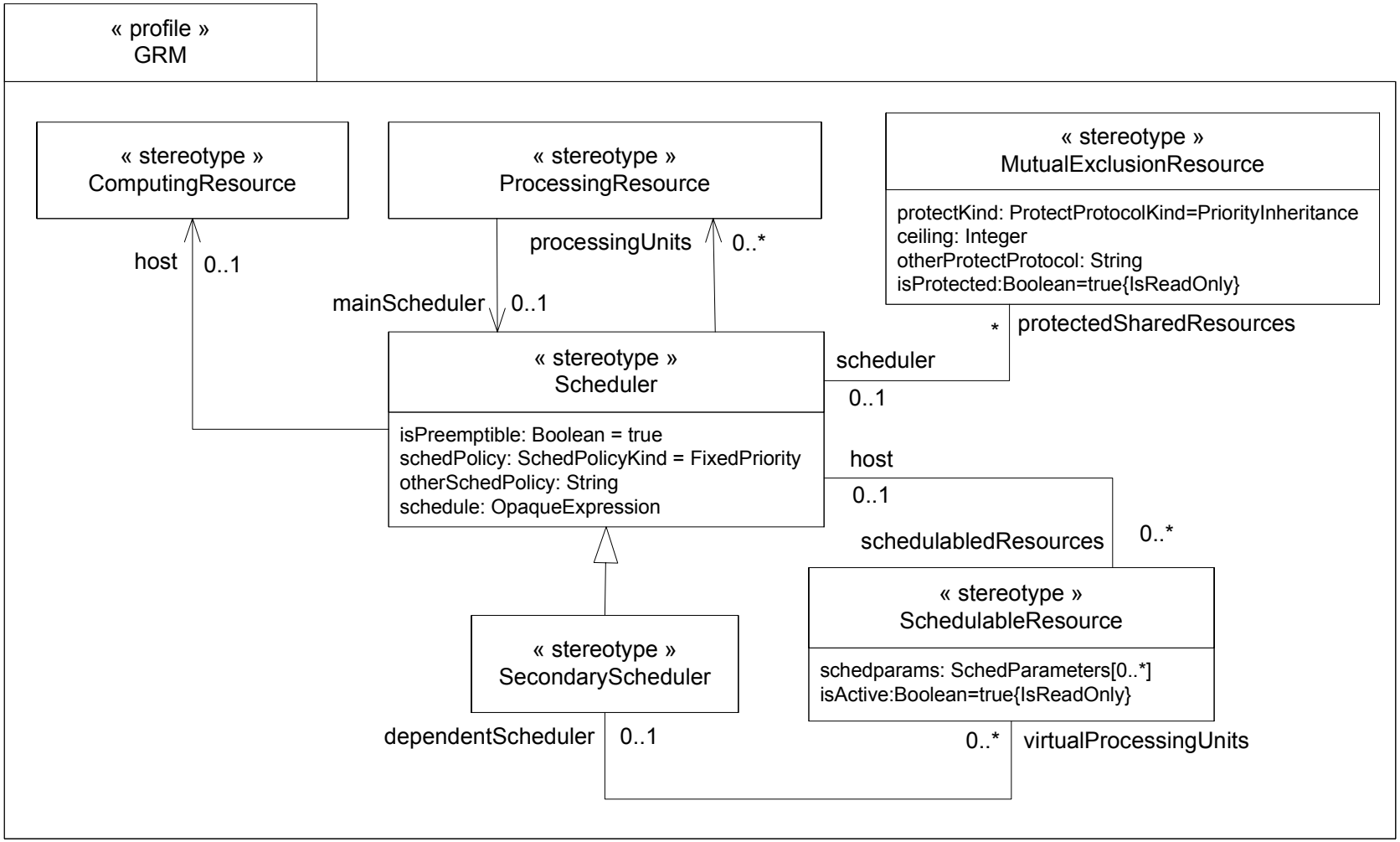
Resources (from GRM)



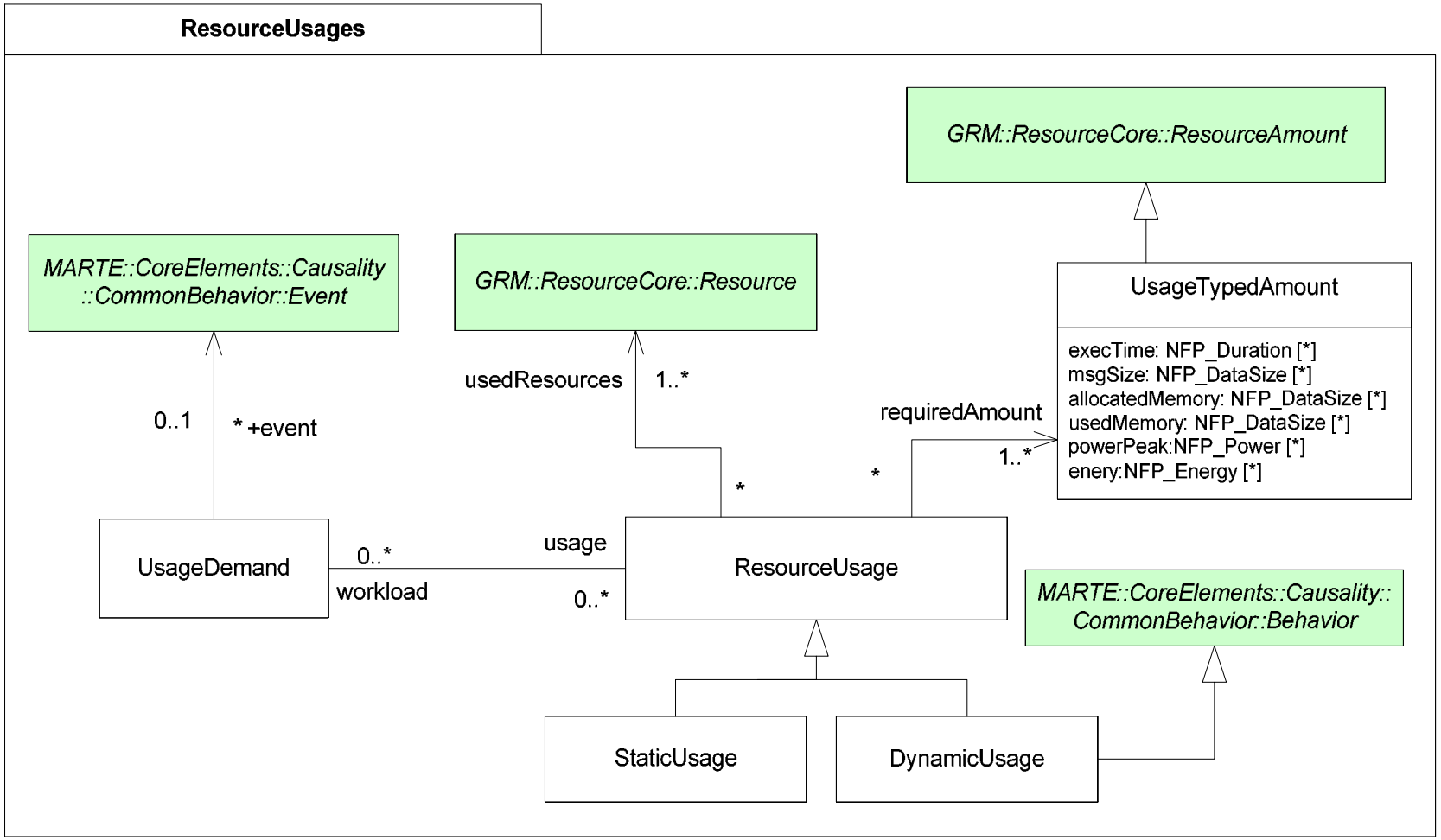
Domain model for Schedulable resources

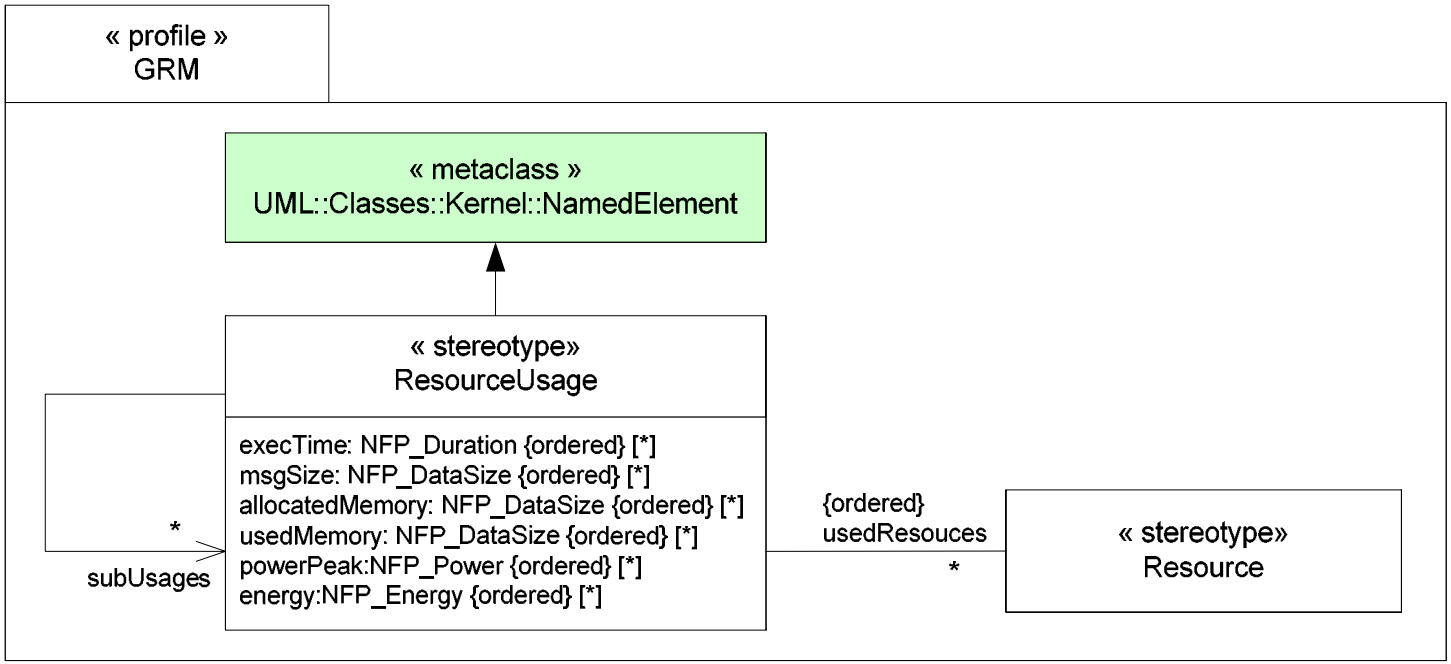


Extensions for scheduling



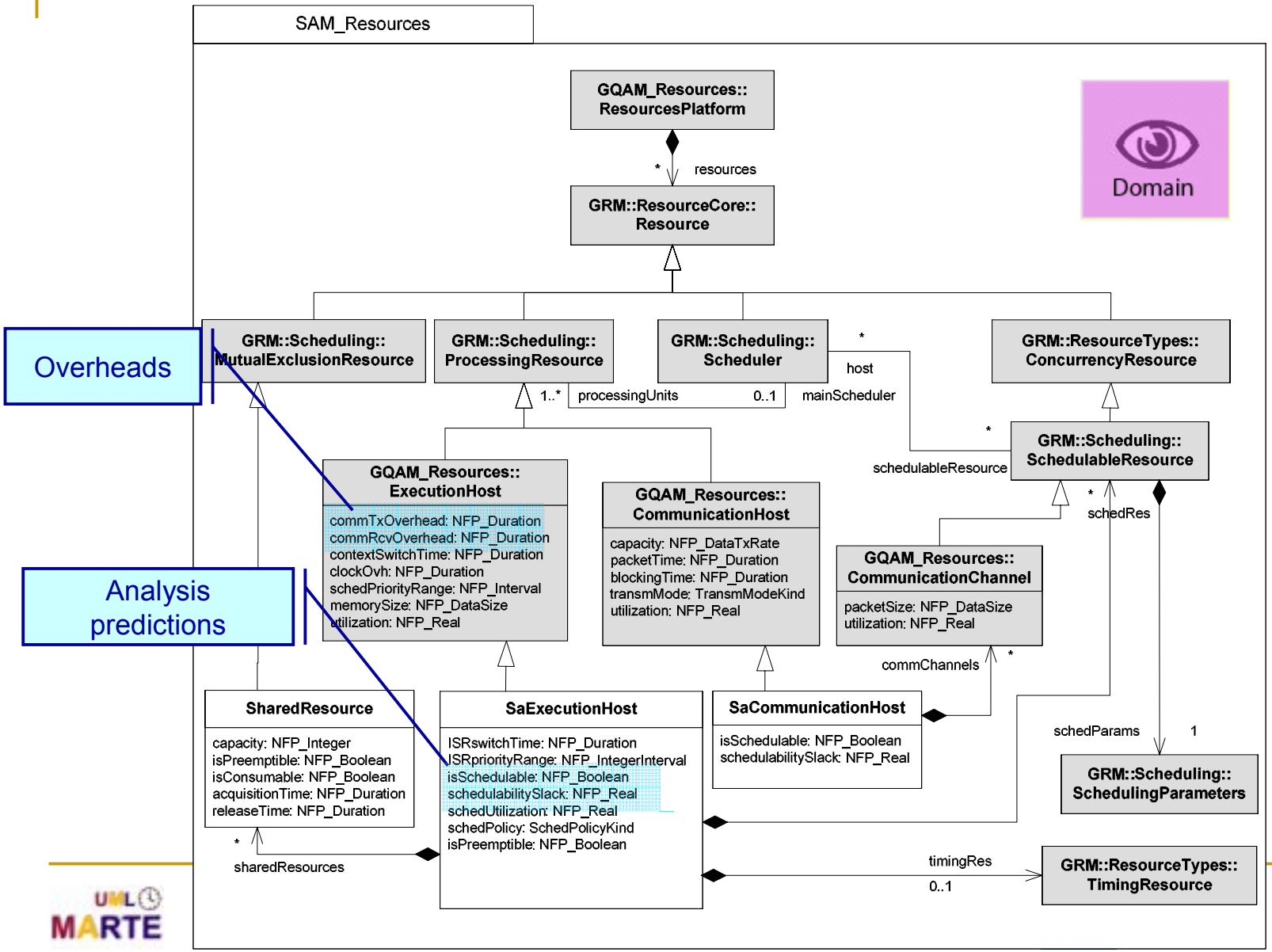
Resource Usage





1. If the list usedResources is empty the list subUsages should not be empty and viceversa..
2. If the list usedResources has only one element, all the optional lists of attributes refer to this unique Resource and at least one of them must be present.
3. If the list usedResources has more than one element, all of the optional lists of attributes that are present, must have that number of elements, and they will be considered to match one to one.
4. If the list subUsages is not empty, and any of the optional lists of attributes is present, then more than one annotation for the same resource and kind of usage may be expressed. In this case, if the annotations have also the same source and statistical qualifiers they will be considered in conflict, and hence the ResourceUsage inconsistent.

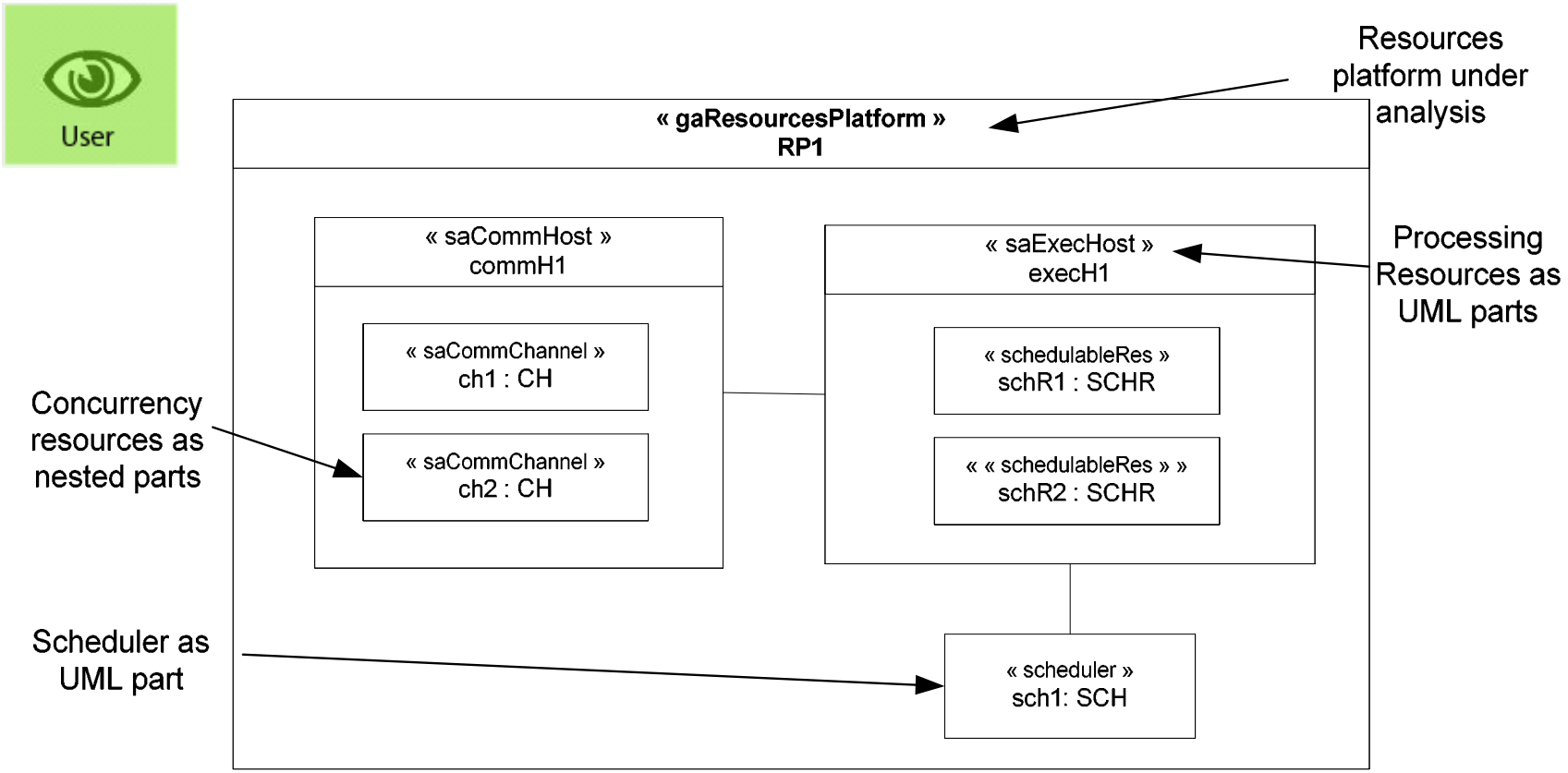
SAM: Resources Domain Metamodel



SAM: Examples of the Stereotypes Usage

SAM Domain Model	SAM Stereotype	UML Metaclasses	Context
ResourcesPlatform	GaResourcesPlatform	UML::StructuredClasses:: **SctructuredClass**	Main container of resources
SaExecutionHost **SaCommunicationHost** **GRM::Scheduler**	SaExecHost SaCommHost Scheduler	UML:: StructuredClasses:: **Property**	Parts of the resources platform
GRM::SchedulableResource **SaCommChannel**	SchedulableRes SaCommChannel	UML:: StructuredClasses:: **Property**	Parts of processing resources

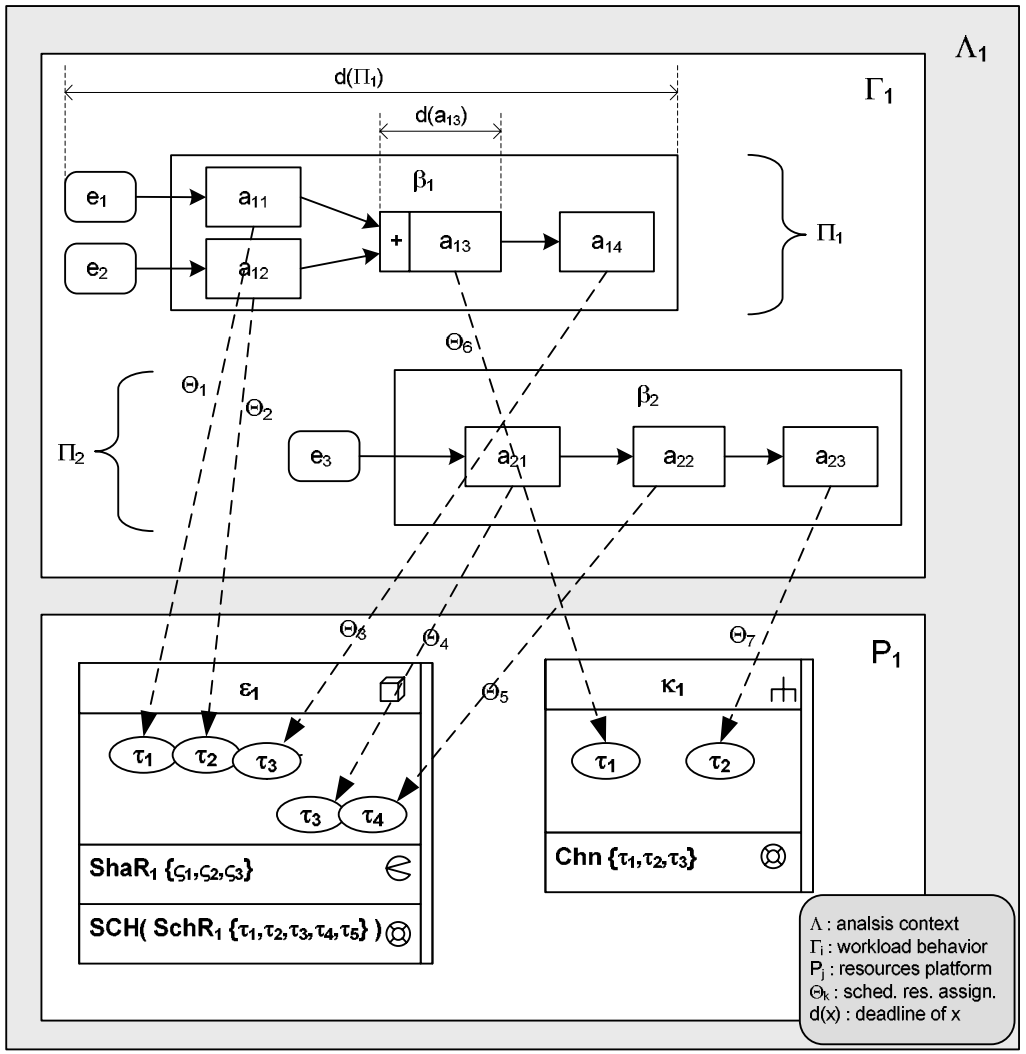
SAM: Example of Resources Stereotype Usage



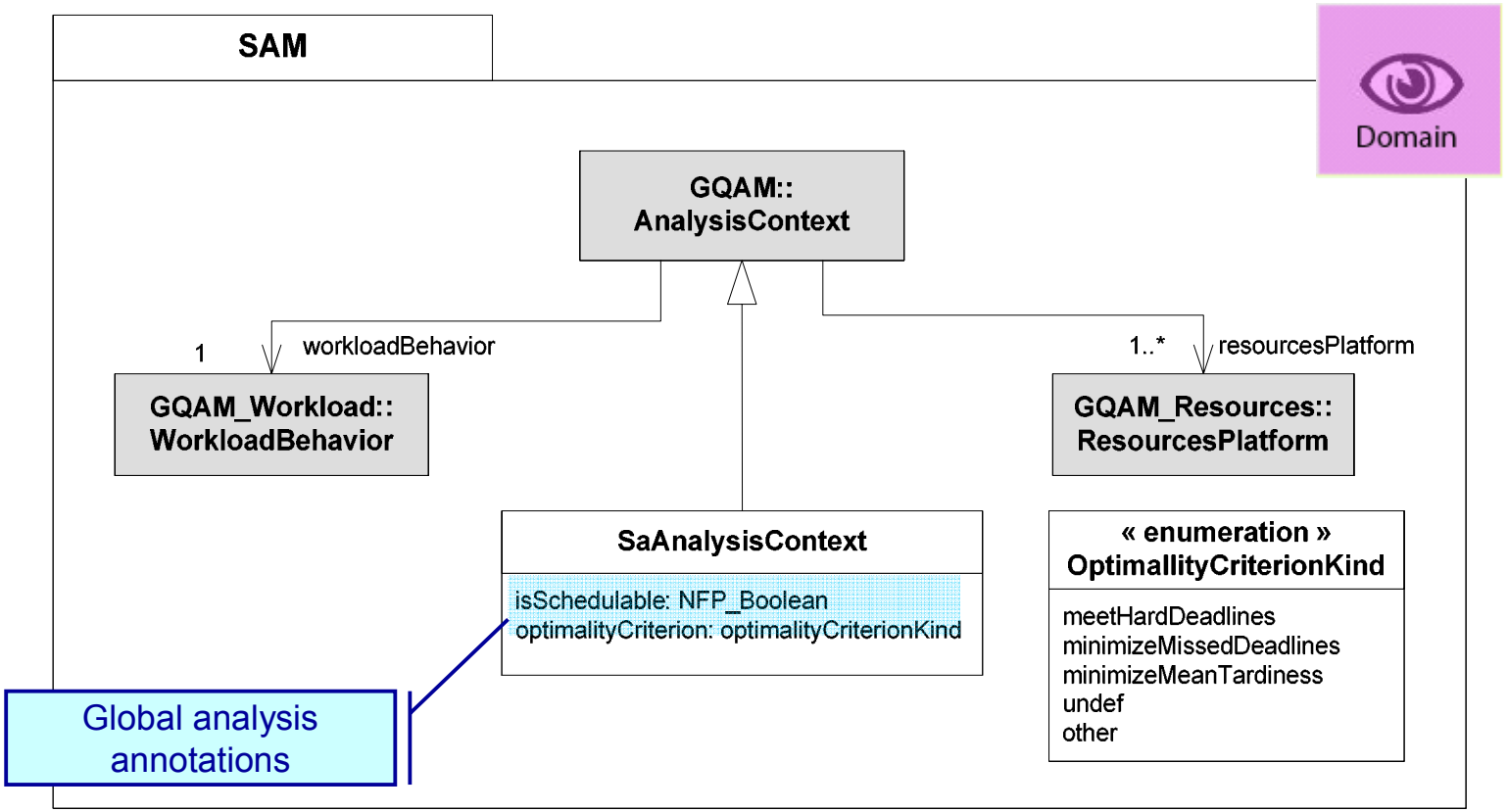
SAM: Analysis Context concepts

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

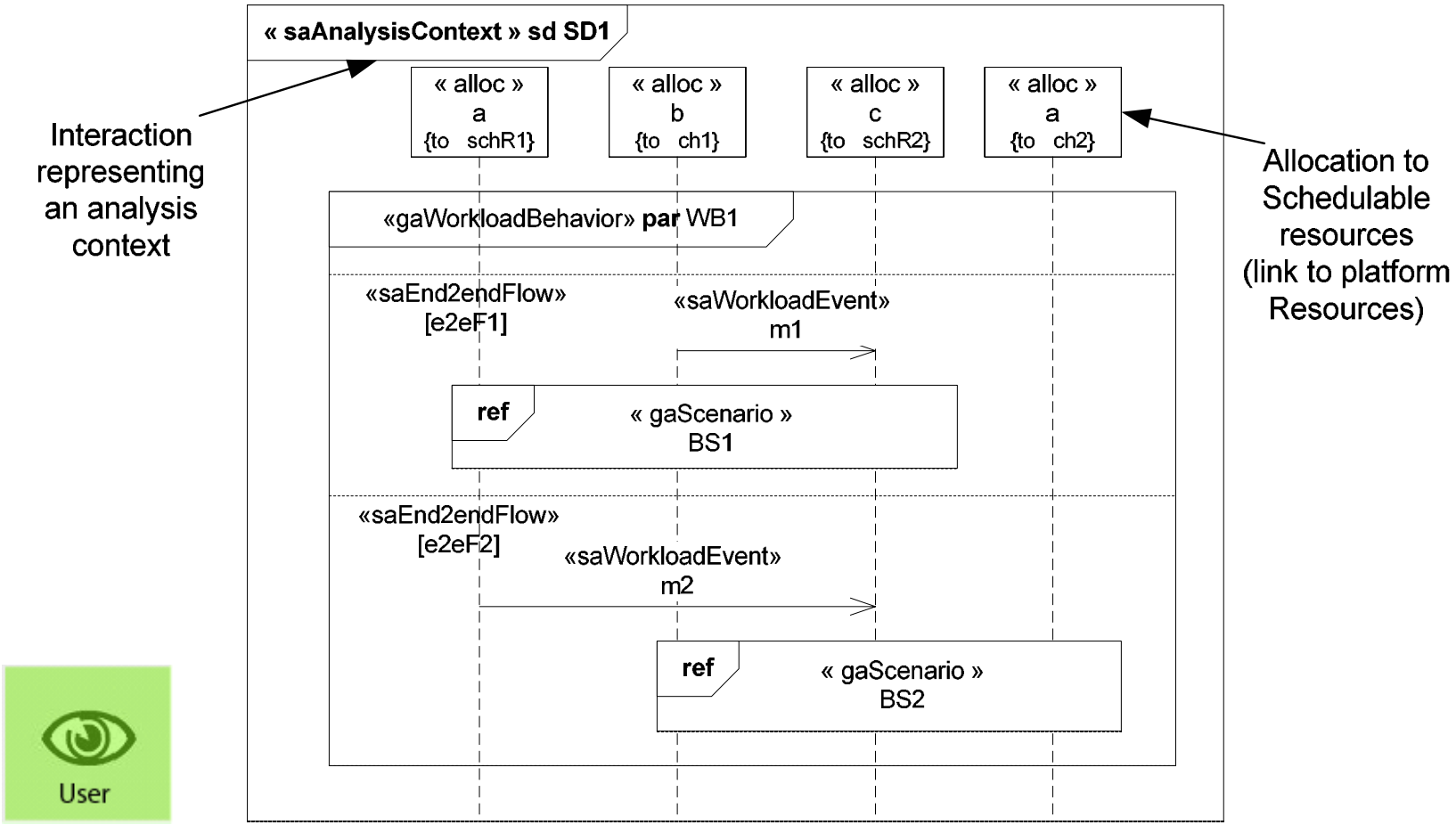
- **An** analysis context **integrates workload behavior models and resources platform models.**



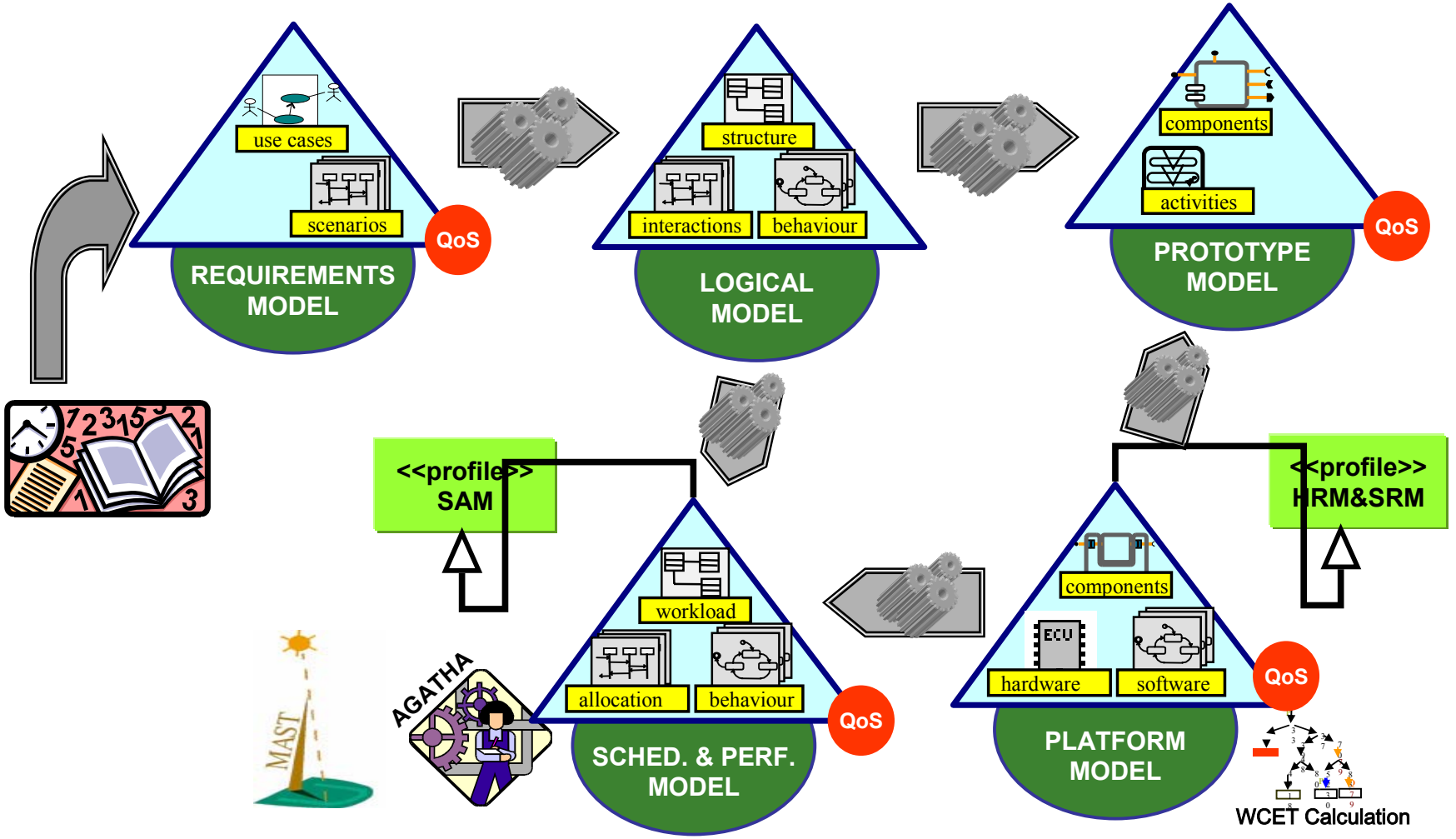
SAM: Analysis Context Domain Metamodel



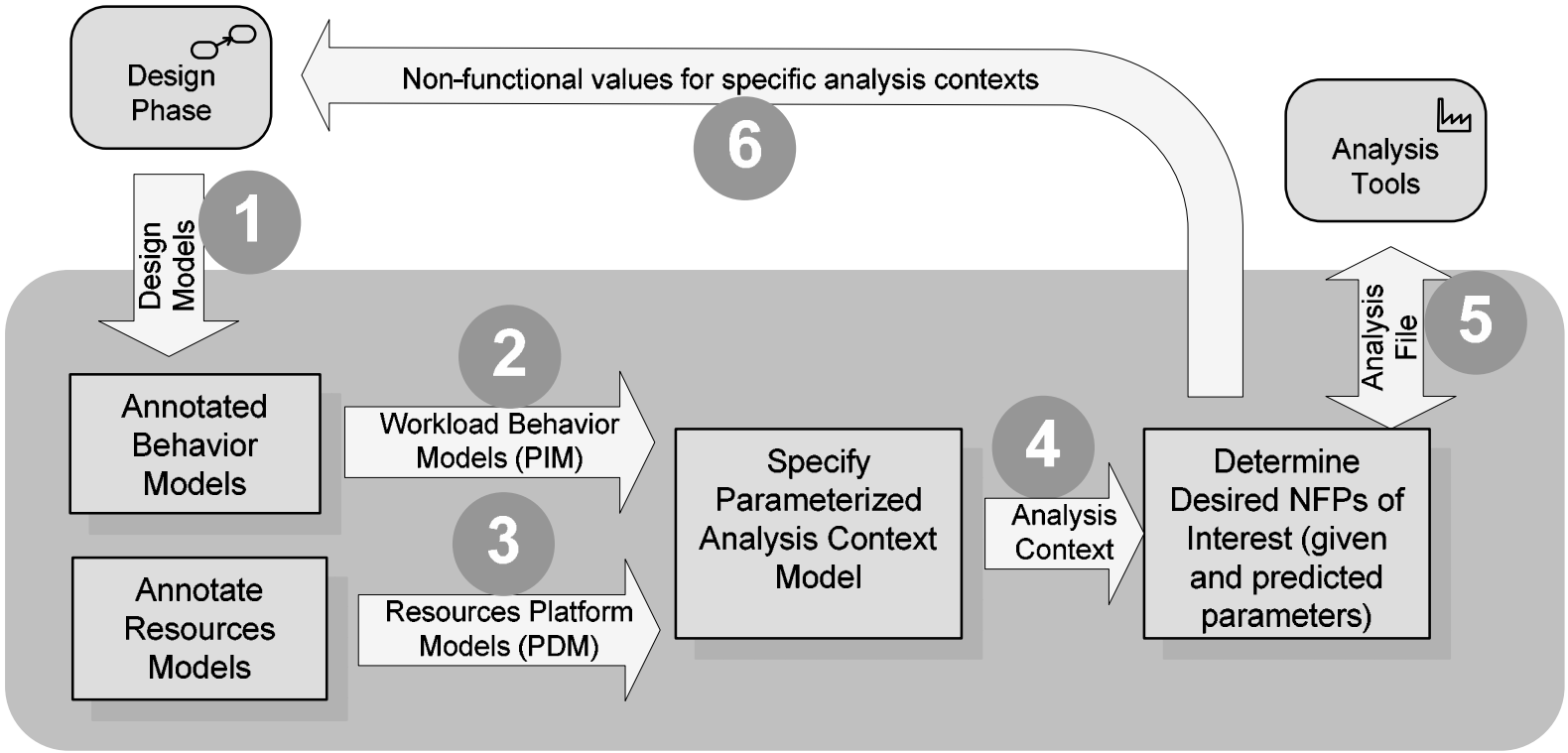
SAM: Example of Analysis Context Stereotype Applic.



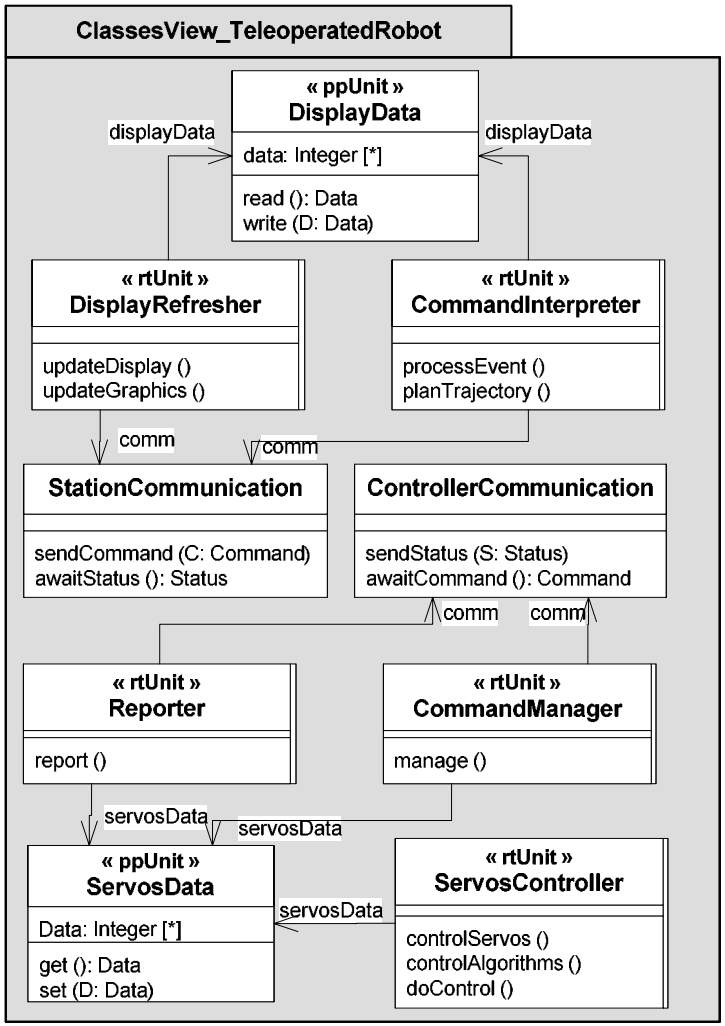
Example of Global Development Process

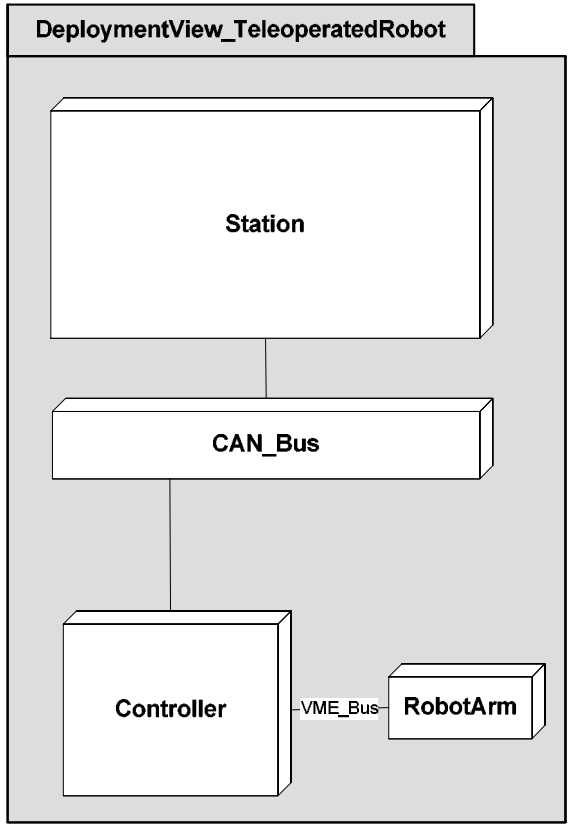


General Procedure to Use the SAM Profile

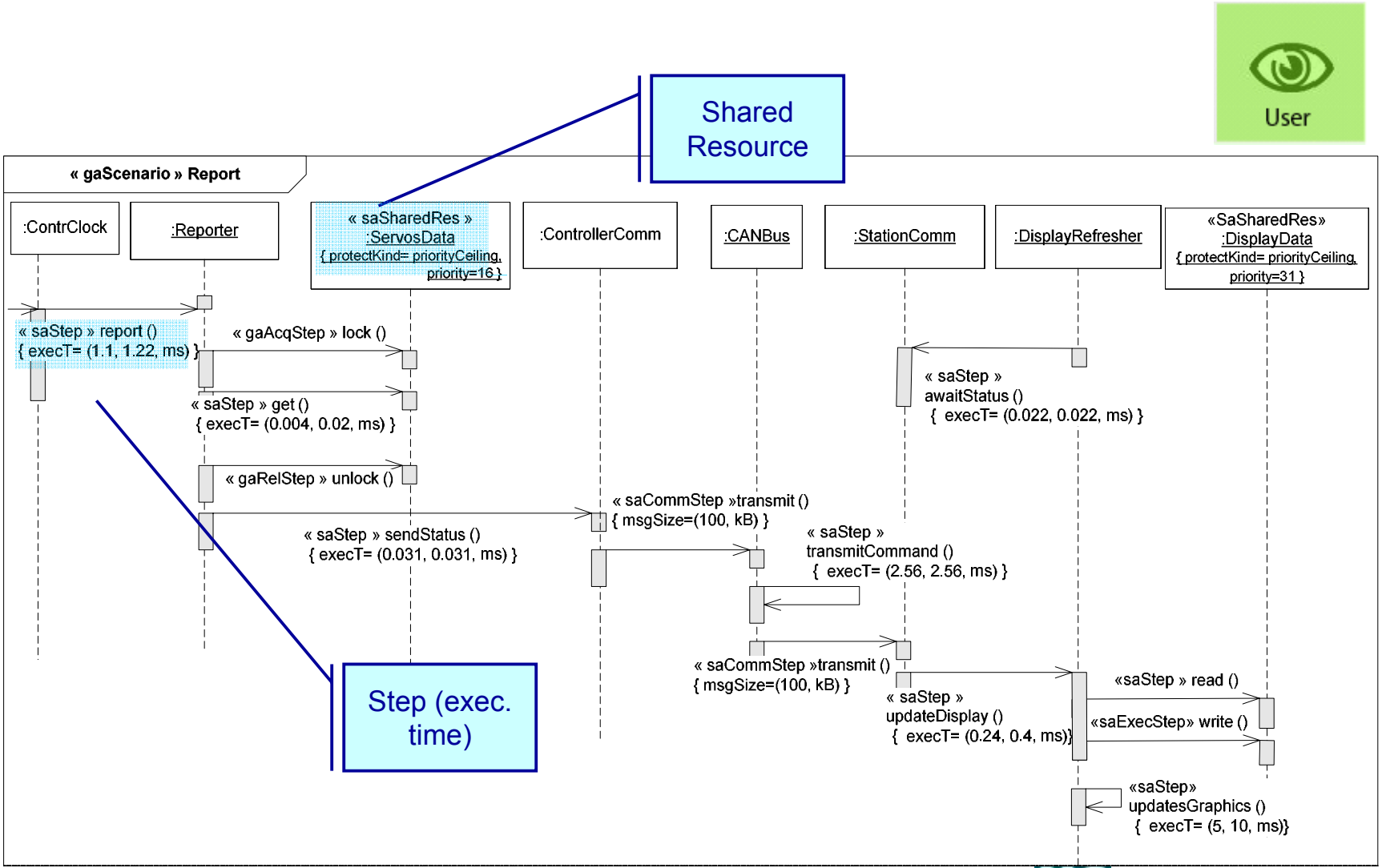


Example: A Teleoperated Robot

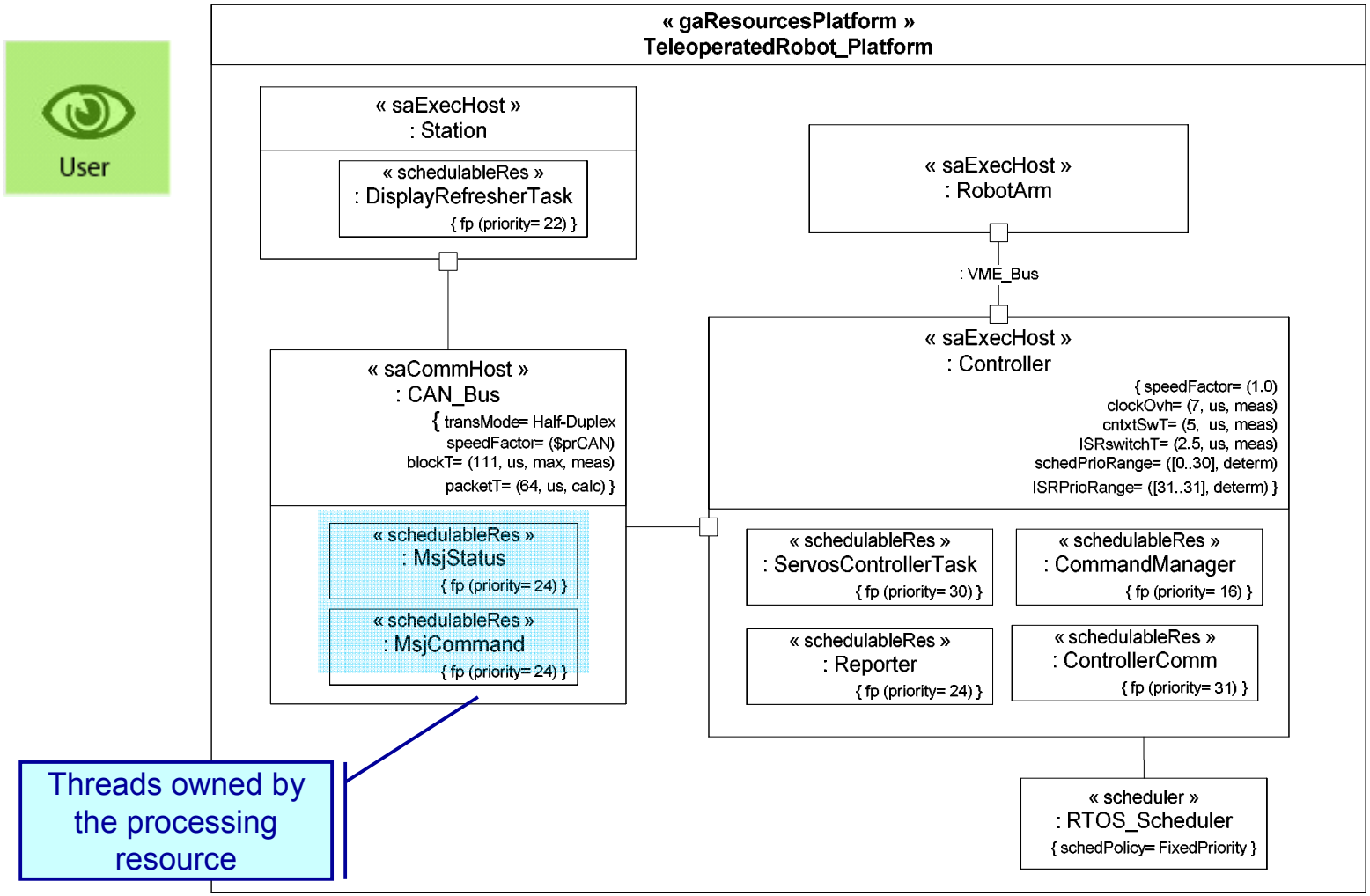




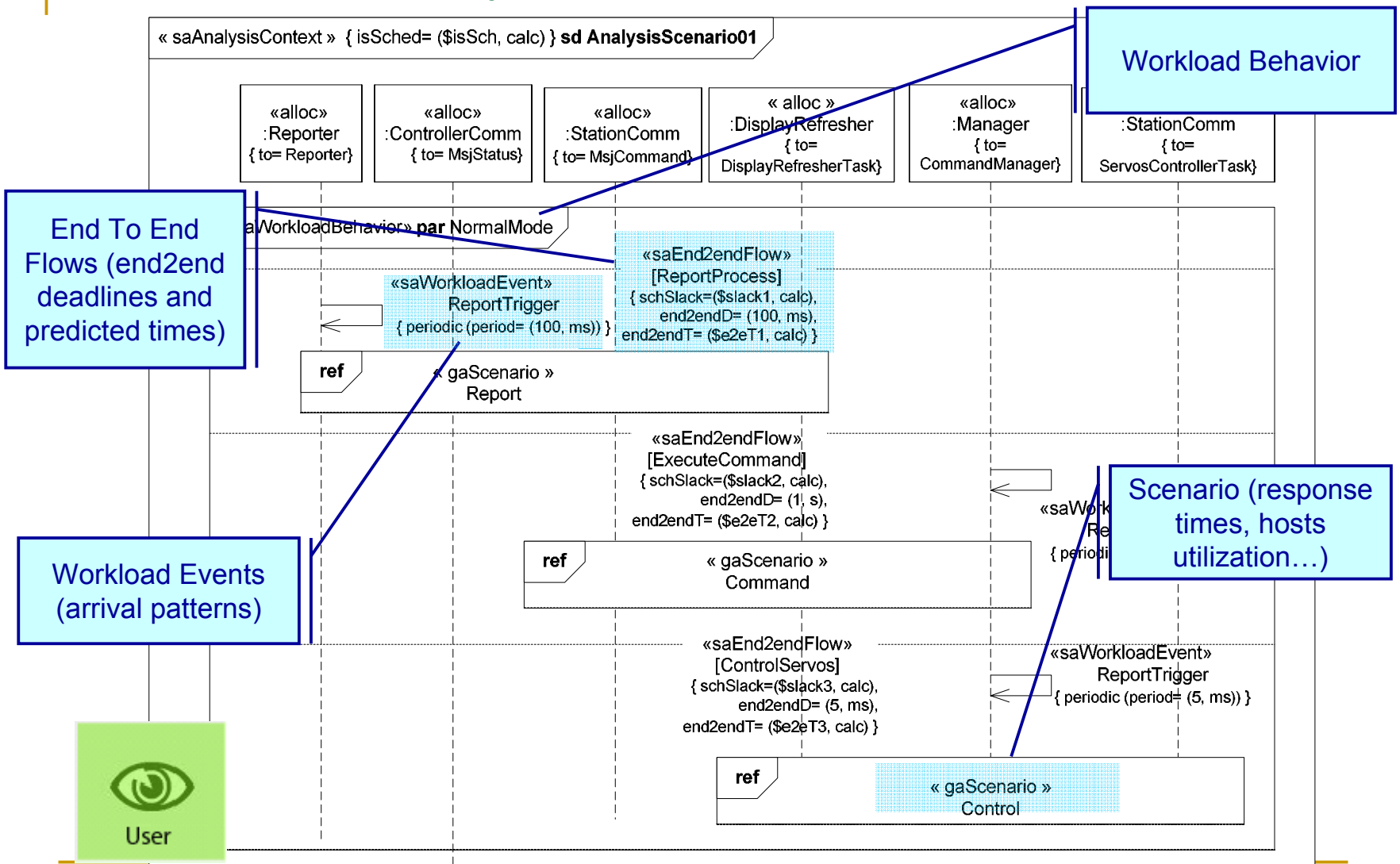
Example of Annotated Scenario with SAM



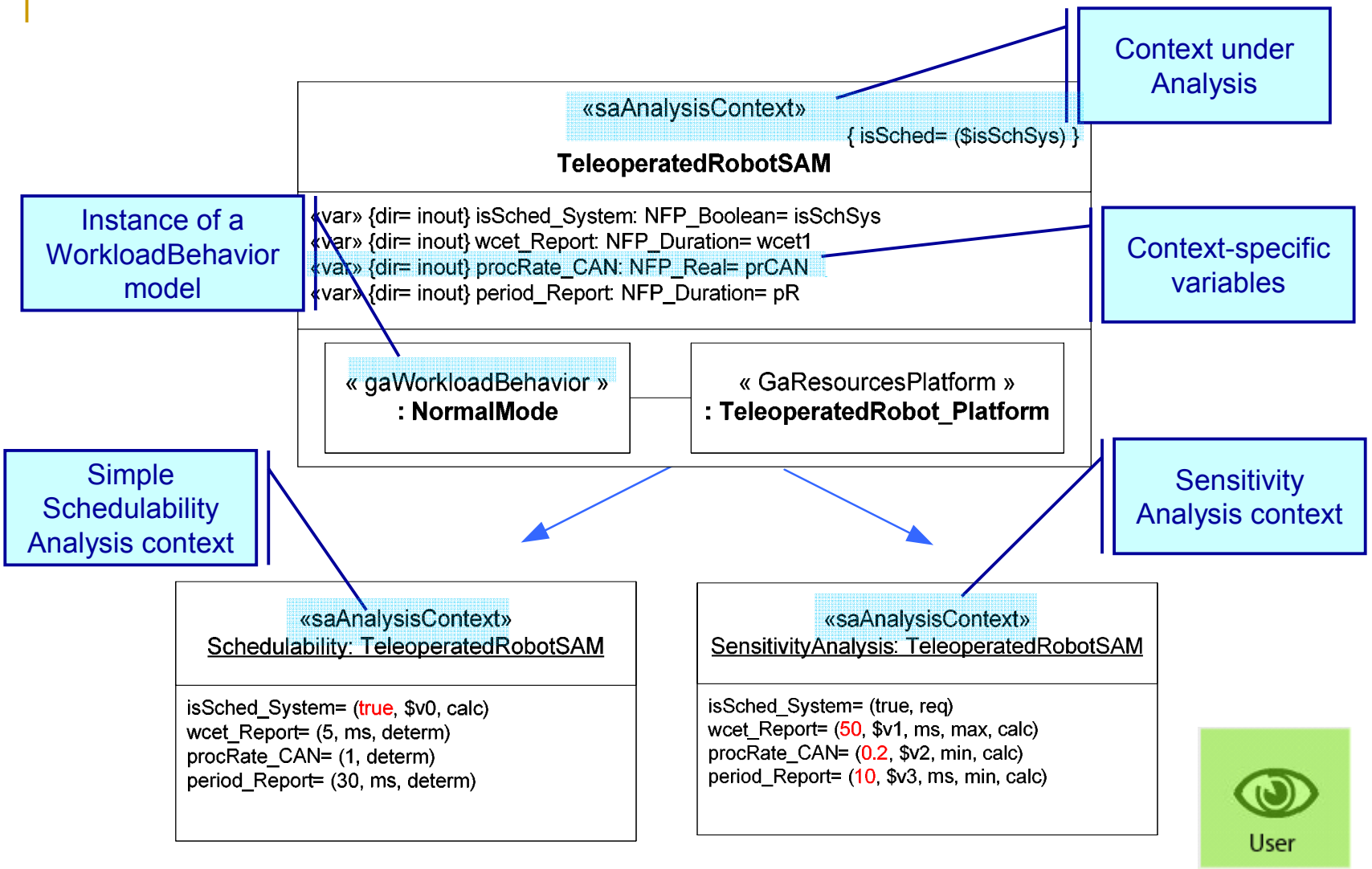
Example of Annotated Resources Model with SAM



Example of Analysis Context Model



Example of Parametric Analysis Context



MARTE Tooling

- Current Implementations supporting MARTE
  - Full MARTE Profile & Libraries for Eclipse UML2
  - VSL edition assistant and type checker as a Eclipse plug-in for the UML Papyrus tool and RSA 7.0

- On-going work:
  - Eclipse plug-ins to transform UML models annotated with the SAM profile to input files of MAST, SymTA/S, Cheddar and RapidRMA tools

MARTE Open Source Implementation in

UML Papyrus: **www.papyrusuml.org**
IBM RSA: **www.omgmarte.org**

Conclusions on analysis capabilities of MARTE

- Industrial Use of V&V can benefits from MDE
  - Analysis task must be cohesively integrated with Design tasks
  - Application of individual analysis techniques should be regarded as an essential part of an integrated V&V methodology

- Methodological support is still under way:
  - Complex analysis scenarios for Interface-Based Design, Multiobjective Design Space Exploration…
  - Means to manage NFP measurement models
  - Methods to map/transform MoCCs into analysis models

MARTE Frontiers and Challenges

- MARTE define the language constructs only!
  - Common patterns, base building blocks, standard NFP annotations
  - Generic constraints that do not force specific execution models, analysis techniques or implementation technologies
- It does not cover methodologies aspects:
  - Interface-Based Design, Design Space Exploration
  - Means to manage refinement of NFP measurement models
  - Concrete processes to storage, bind, and display NFP context models
  - Mapping to transform MoCCs into analysis models

MARTE is to the RTES domain as UML to the System & Software domain: a family of well known and open specification formalisms!

Thanks for your attention..!

Questions ?

- www.omgmarte.org
  - The MARTE web site
- www.papyrusuml.org
  - On open source Eclipse plug-in for UML2 graphical modeling
  - With a MARTE implementation available on:
    - http://www.eclipse.org/papyrus