Model Driven Engineering and Real-Time Analysis of Embedded Systems: The UML MARTE Standard and its Challenges

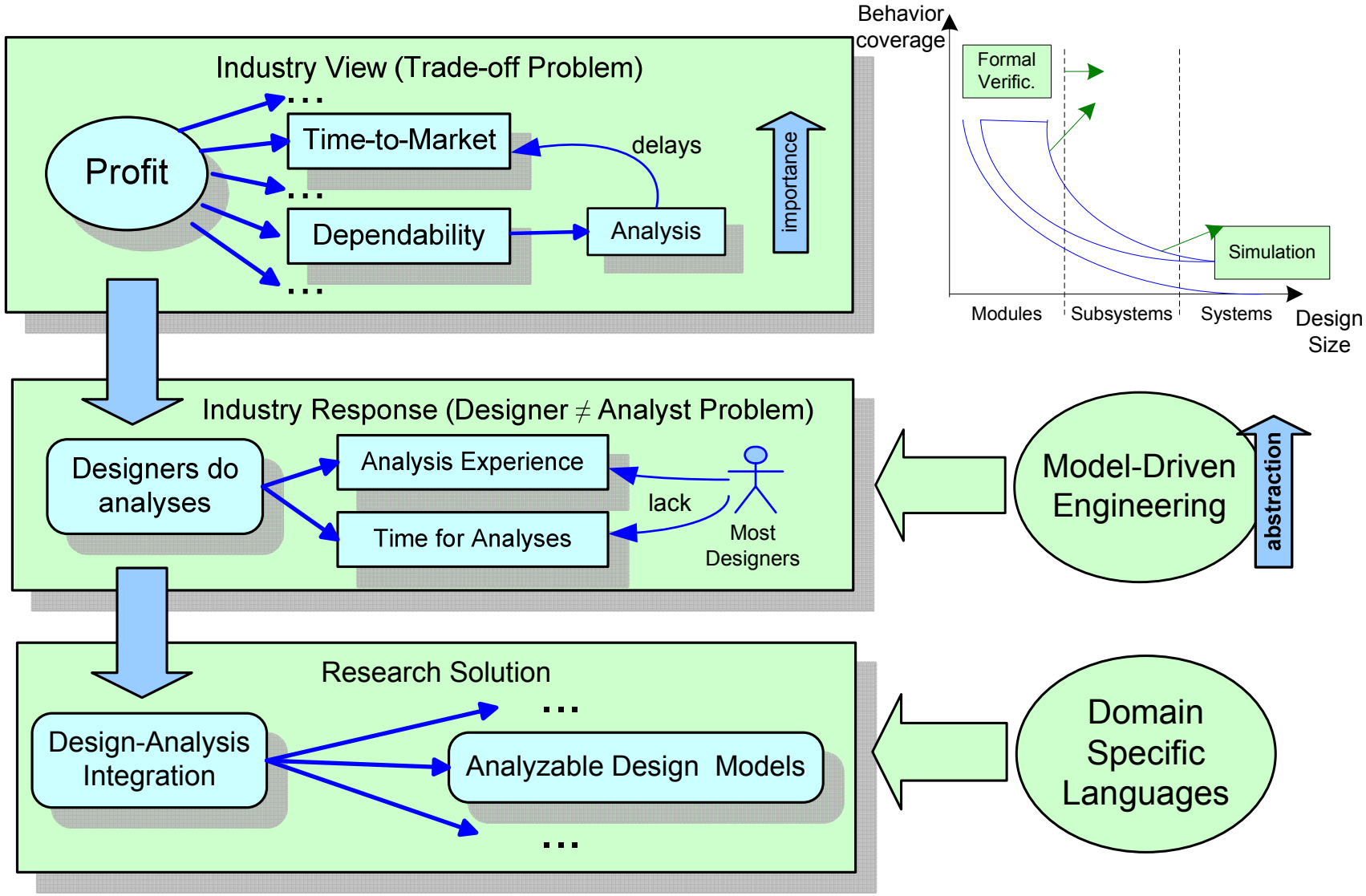
Tool Platforms for ES Modeling, Analysis and Validation
ARTIST Workshop at CAV 2007
June 1st, 2007
Berlin, Germany

Huascar Espinoza, François Terrier and Sébastien Gérard
CEA LIST / DTSI / SOL / L-LSP
Commissariat à l'Energie Atomique, FRANCE

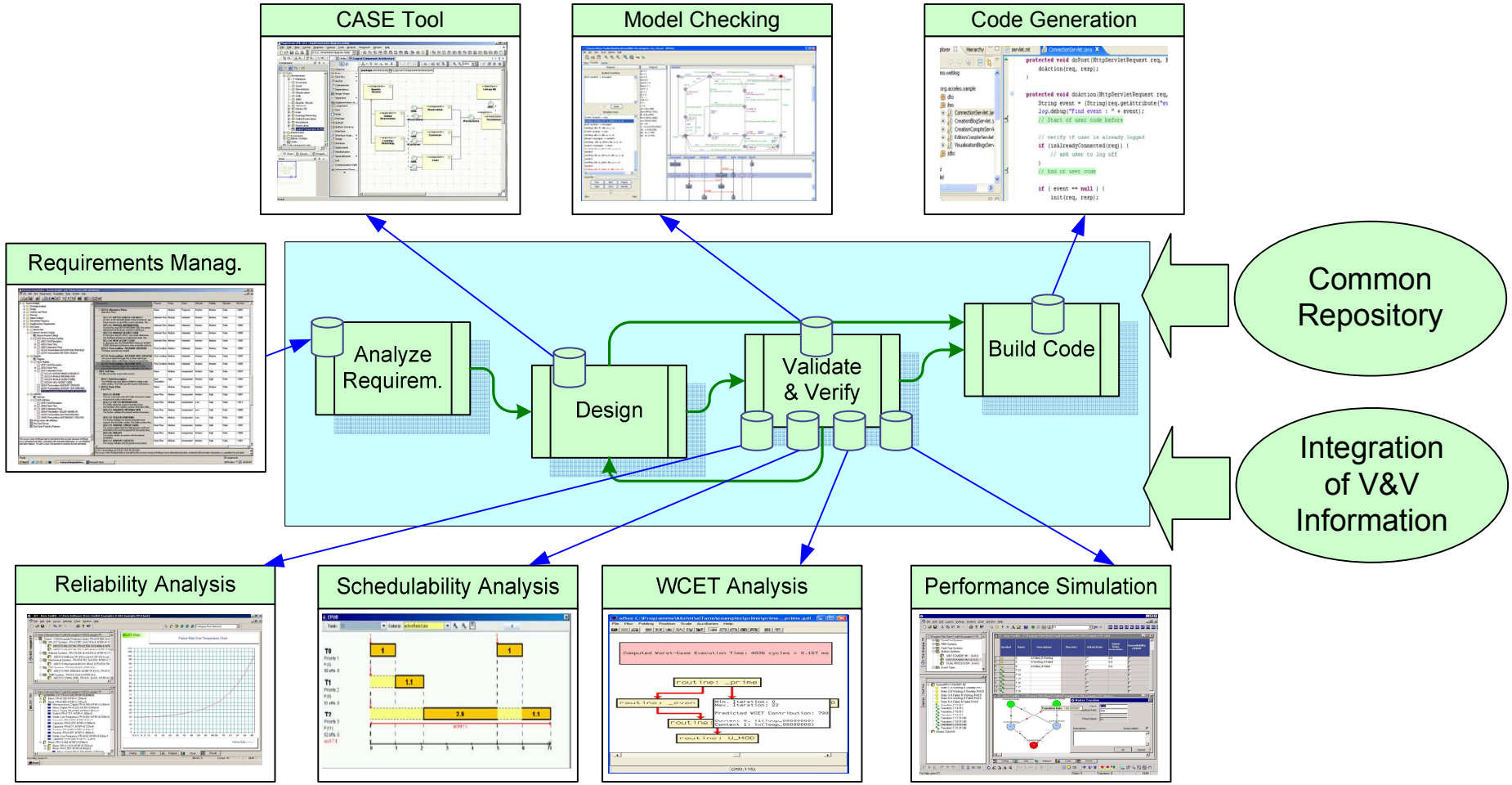
Agenda

- Background & Motivation
- UML MARTE
- MARTE for Quantitative Analysis
- Frontiers & Challenges
- Summary

Need for Design-Analysis Integration



Collaborative Environments



Model Driven Engineering & RTES

Integrate Models to…

DESIGN-BUILD-VALIDATE-VERIFY

- **Evaluate architectural design alternatives**
  - ➔ MoCCs, fault-tolerance configurations…
  - ➔ RTOS, SoC architectures, communication buses…
- **Multiple points of view with non-functional requirements**
  - ➔ memory usage, power consumption, real-time response…
- **Best software engineering practices**
  - ➔ refinement/abstraction, multi-perspective, layering…
  - ➔ metamodeling (metamodel = model repository = data structure)
    - Technology independent (mapping to XML, Java)

Domain Specific Modeling Languages

- **Use of domain concepts as language constructs**
  - ➔ E.g. concepts of “memory”, “task”, “bus”, instead of general ones “components”, “nodes”, “classes”…
- **Criteria for success of a DSML**
  - ➔ *Technical validity* and absence of ambiguities
  - ➔ *Domain appropriateness*: syntax close to RTES ontology
  - ➔ *Interoperability* with other technologies (verification, code generation)
  - ➔ *Support*: tools, documentation, capacity for evolution

But mainly… balance between *expressive power* and *simplicity*

- **Main approaches to defining a DSML**
  1. New language from scratch
  2. Extend an existing language (add new language constructs)
  3. Refining an existing language (specialize language constructs)

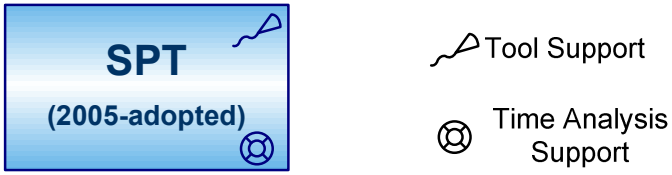
Profiling UML for a Domain

- **Advantages of UML Profiles:**
  - ➔ Reuse of language infrastructure (tools, specifications)
  - ➔ Requires less language design skills
  - ➔ Allows for new (graphical) notation of extending stereotypes
  - ➔ A profile can define model viewpoints
    - E.g. UML activity diagram extended to specify multitask behavior
- **Disadvantages:**
  - ➔ Syntactical constructs and semantics constrained by UML metamodel
- **How to face these limitations?**
  - ➔ Profile semantics: on-going work: 'Executable UML Foundation'
  - ➔ Create rich profiles (expressiveness) and complement with specific methodologies (precise semantics)

Agenda

- Background & Motivation
- UML MARTE
- MARTE for Quantitative Analysis
- Frontiers & Challenges
- Summary

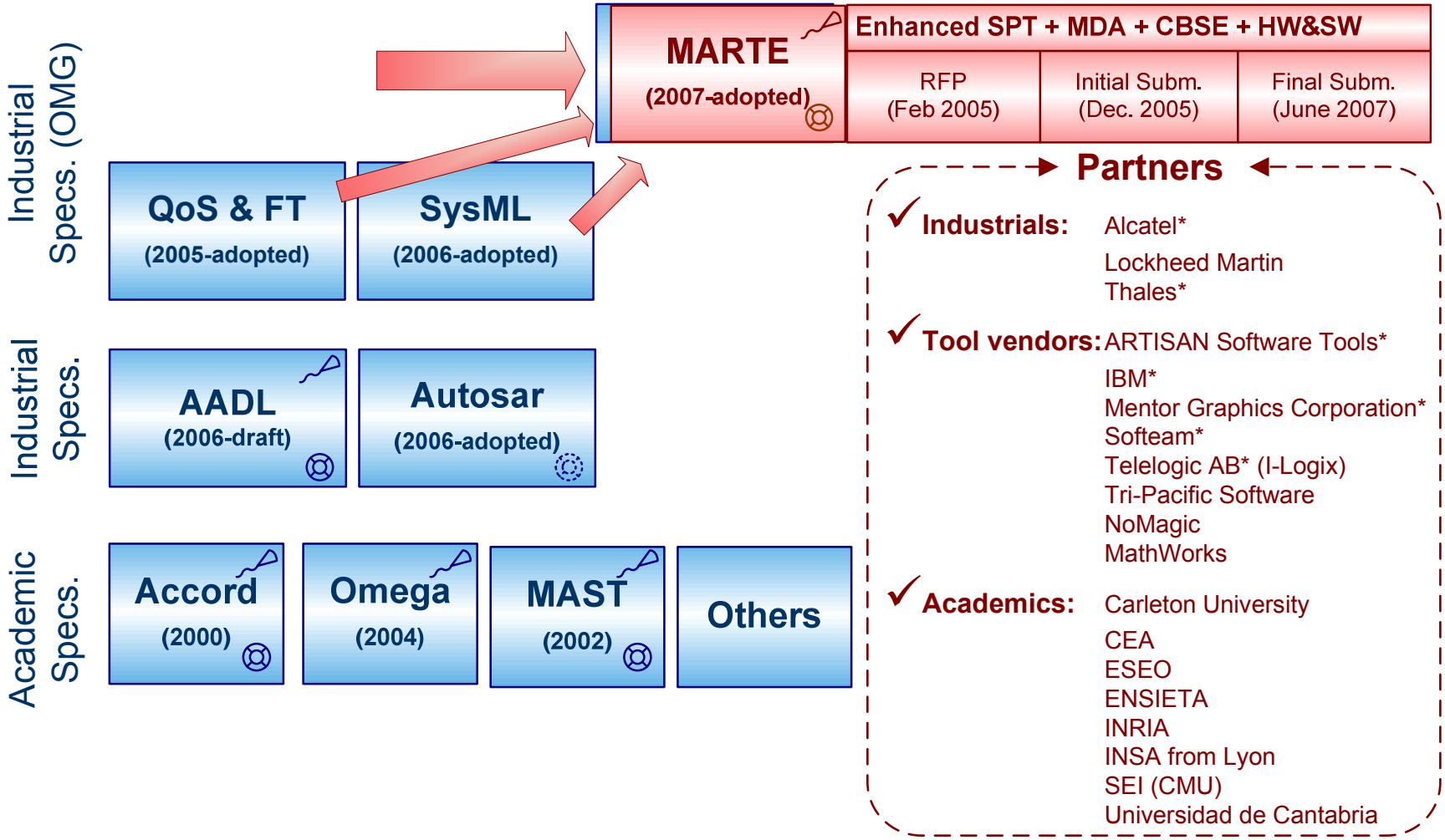
UML Profiles for RTES



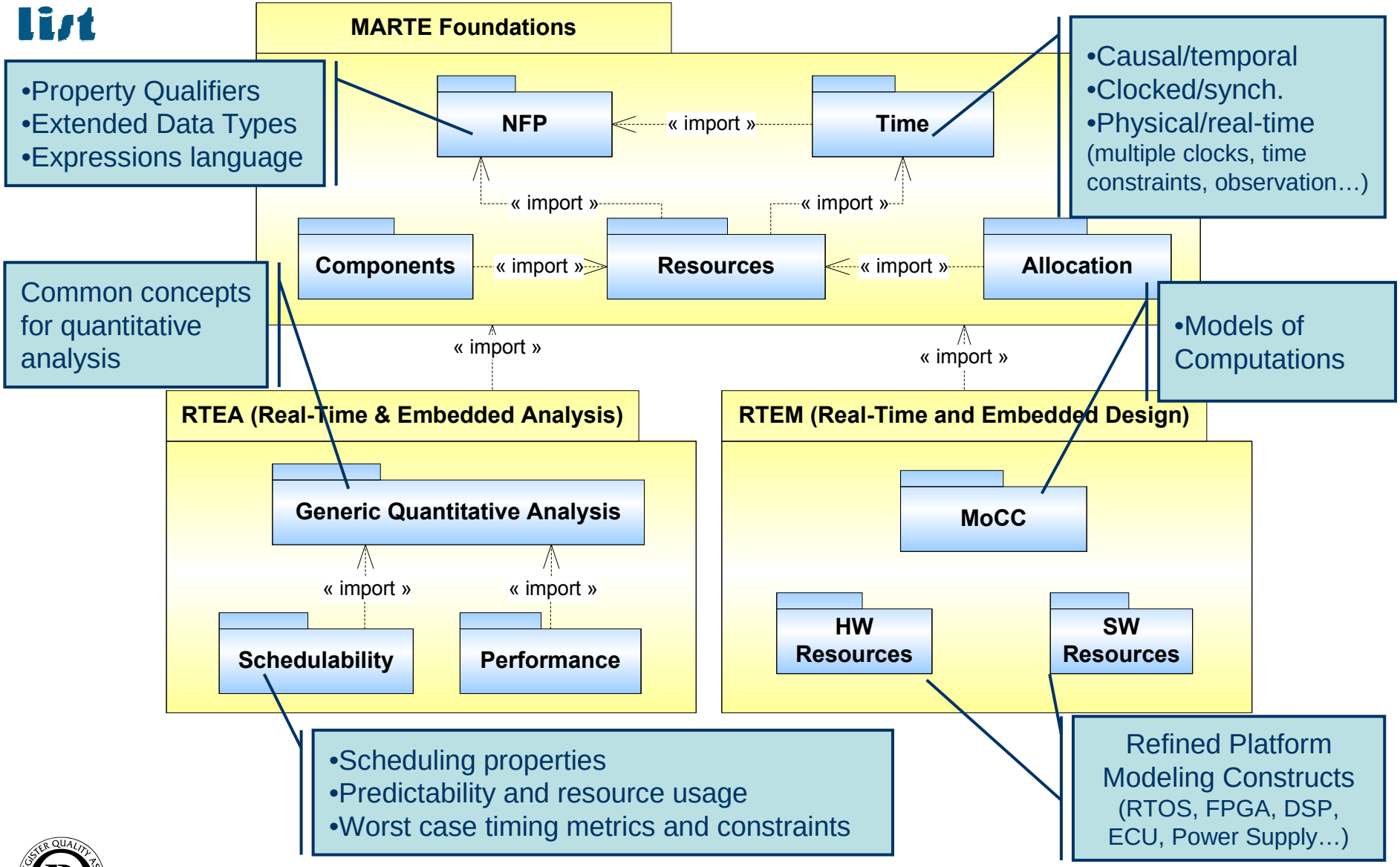
- **SPT was the first OMG's UML profile for Real-Time Systems:**
  - ➔ Support for Schedulability analysis with RMA-type techniques
  - ➔ Support for Performance analysis with Queuing theory and Petri nets
  - ➔ A rich model for "metric" Time and time mechanisms
- **Despite its extensive use, several improvements were required:**
  - ➔ Modeling HW and SW platforms, logical time, MoCCs, CBSE…
  - ➔ Alignment to UML2, QoS&FT, MDA,…
  - ➔ SPT's constructs were considered too abstract and hard to apply

A Request For Proposals for a new profile was issued

UML Profiles for RTES



MARTE High Level Architecture



Modeling Non-Functional Properties

Non-functional properties describe the "fitness" of systems behavior (e.g., performance, memory usage, power consumption…)

- **NPF's *nature*:**
  - ➔ Quantitative: magnitude + unit (e.g., energy, data size, duration)
  - ➔ Qualitative (e.g., "periodic" or "sporadic" event arrival patterns)
- **NFP values are *qualified* by:**
  - ➔ source, statistical measure, precision,…
- **NFPs are *parametric* and *derivable*:**
  - ➔ variables: placeholders for unknown values
  - ➔ expressions: math and time expressions
- **NFP annotation mechanism is *flexible*:**
  - ➔ predefined NFPs (e.g., end-to-end latency, processor utilization)
  - ➔ user-specific NFPs (but still unambiguously interpreted)

NFP Annotations in a Nutshell

1) NFP Type Declaration

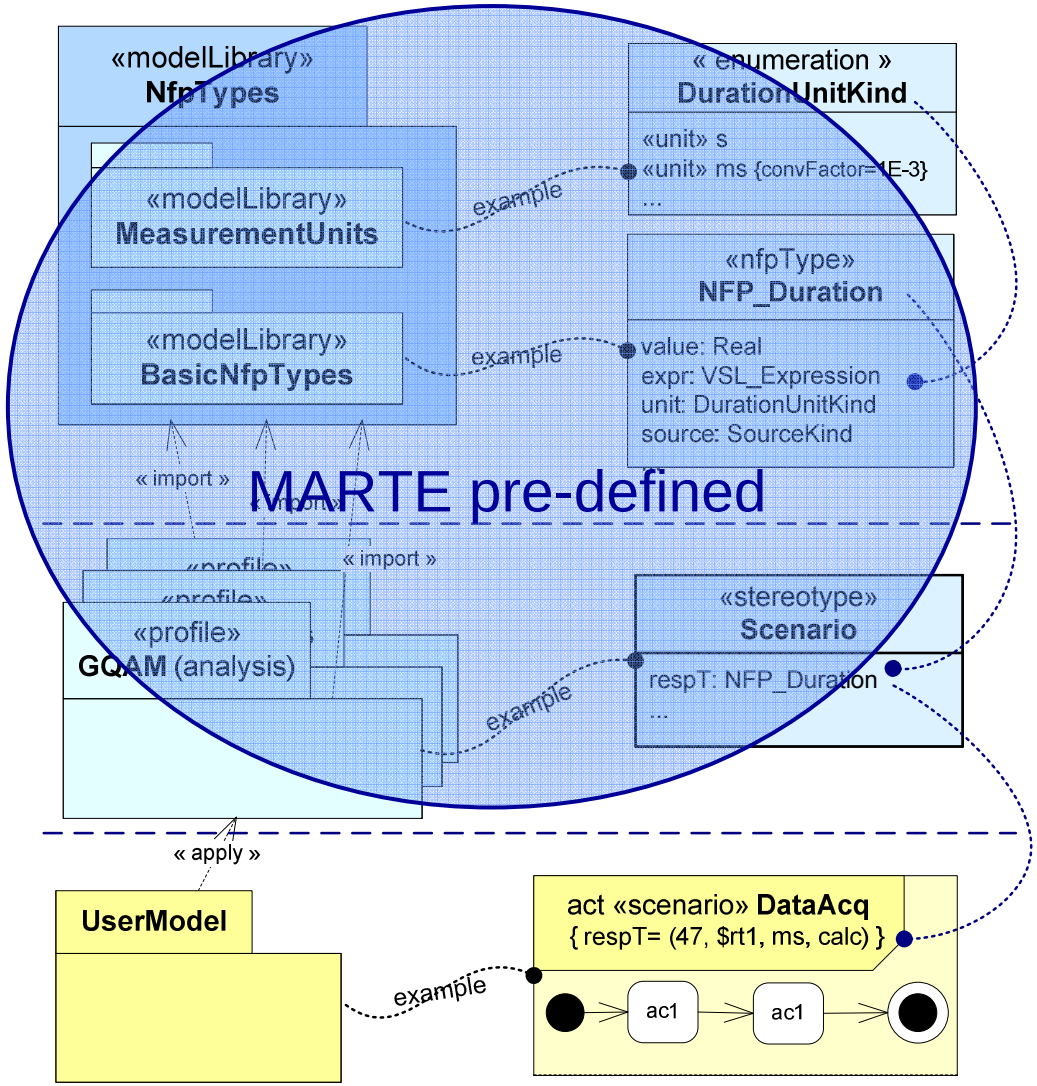
- Define measurement units and conversion parameters
- Define NFP types with qualifiers

2) NFP Declaration

- Define NFPs in stereotype attributes using NFP Types
- Modelers can also declare NFPs in UML Classifiers

3) NFP Value Specification

- Three mechanisms: ***tagged values***, ***constraints***, and ***slots***.
- Values and expressions use VSL (Value Specification Language)



Value Specification Language (VSL)

- **UML lacks of value specification mechanisms for NFPs**
  - ➔ No syntax for structured values
  - ➔ No syntax for algebraic and time expressions
  - ➔ Data type system is not sufficient for non-functional modeling
- **Foundations of VSL…**
  - ➔ Reuse OCL constructs: grammar for values and expressions
  - ➔ Generic data type system: (based on ISO's General-Purpose Datatypes)
  - ➔ Defined by abstract and concrete syntaxes (grammar).
  - ➔ Flexibility: expression operators in model-specific libraries
  - ➔ VSL extends UML Simple Time model (occurrence index, jitters,...)

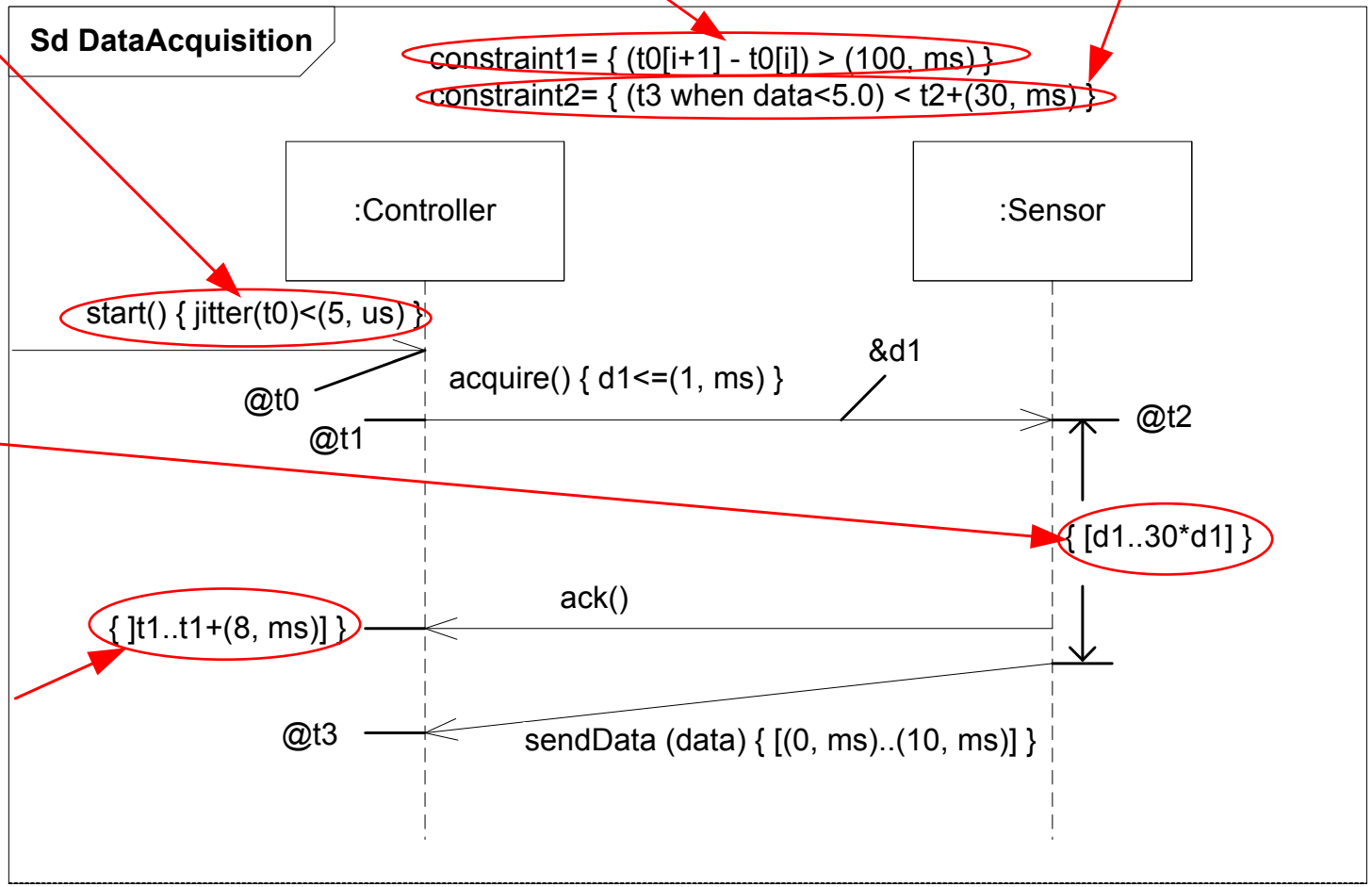
Time Expressions with VSL

Jitter constraint

Duration expression between two sucessive occurrences

Constraint in an observation with condition expression

Specification example in Sequence diagrams…



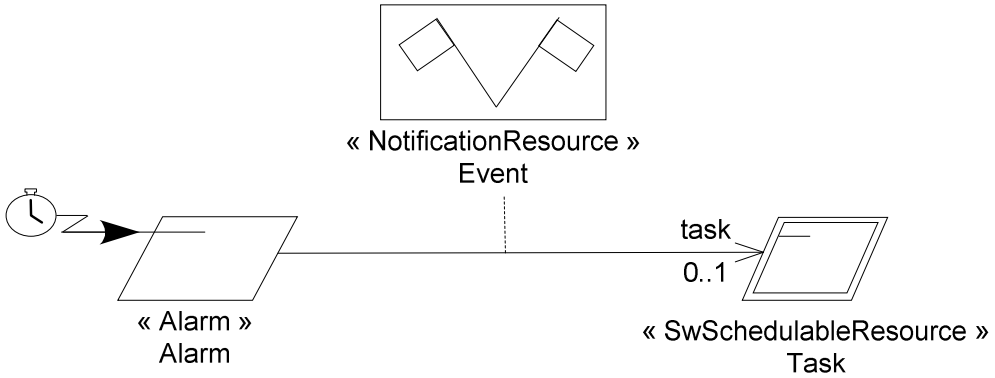
Extended duration intervals with bound « [ ] » specification

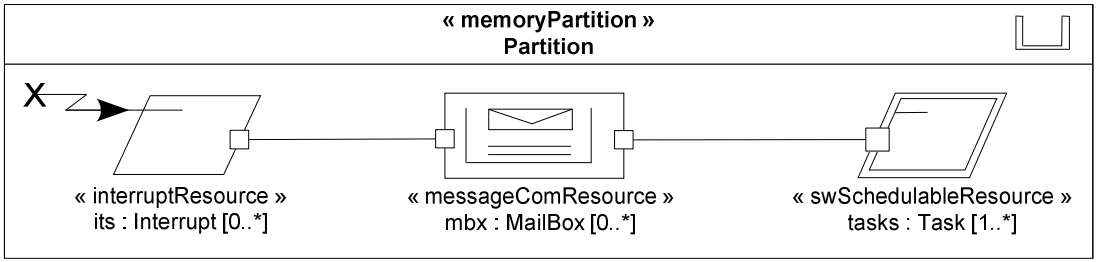
Instant Interval Constraint

Software Resource Modeling Profile

- **To describe API of software execution platforms**
  - ➔ Constructs for Multitasking models
  - ➔ Real Time Operating Systems (RTOS)
    - standard RTOS API (e.g. POSIX, OSEK/VDX and ARINC 653)
  - ➔ Language libraries (e.g. ADA)

- Concurrent execution mechanisms
  - ✓ Task, Interrupt, Alarm & Counter…
- Synchronization mechanisms
  - ✓ Events, Mutual Exclusion Access Mechanisms…

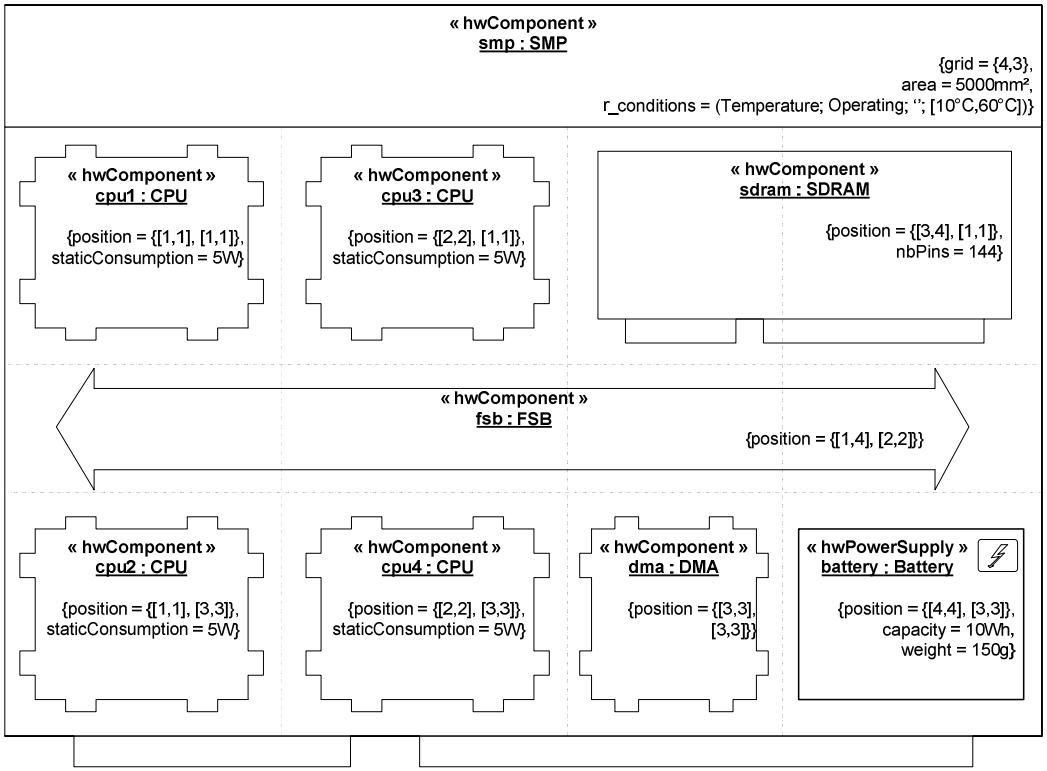




Hardware Resource Modeling Profile

- **To describe structure of Hardware platforms**
  - ➔ Different abstraction levels for: processor simulation, power consumption calculation, WCET analysis, block diagrams
  - ➔ Two sub-views: logical (functionality) and physical (layouts)

- HW Logical constructs
  - ✓ RAMs, ASICs, I/O Devices, Clocks, Buses…
- HW Physical constructs
  - ✓ Cards, Chips, Power Supplies,…



Agenda

- Background & Motivation
- UML MARTE
- MARTE for Quantitative Analysis
- Frontiers & Challenges
- Summary

MARTE Features for Quantitative Analysis

Quantitative analysis calculate mathematically NFPs of interest based on other available NFPs and the system behavior

- **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
- **New features w.r.t. SPT**
  - ➔ Support for sensitivity analysis
  - ➔ Improve modeling reuse and component-based design.
  - ➔ Support of the MDA approach

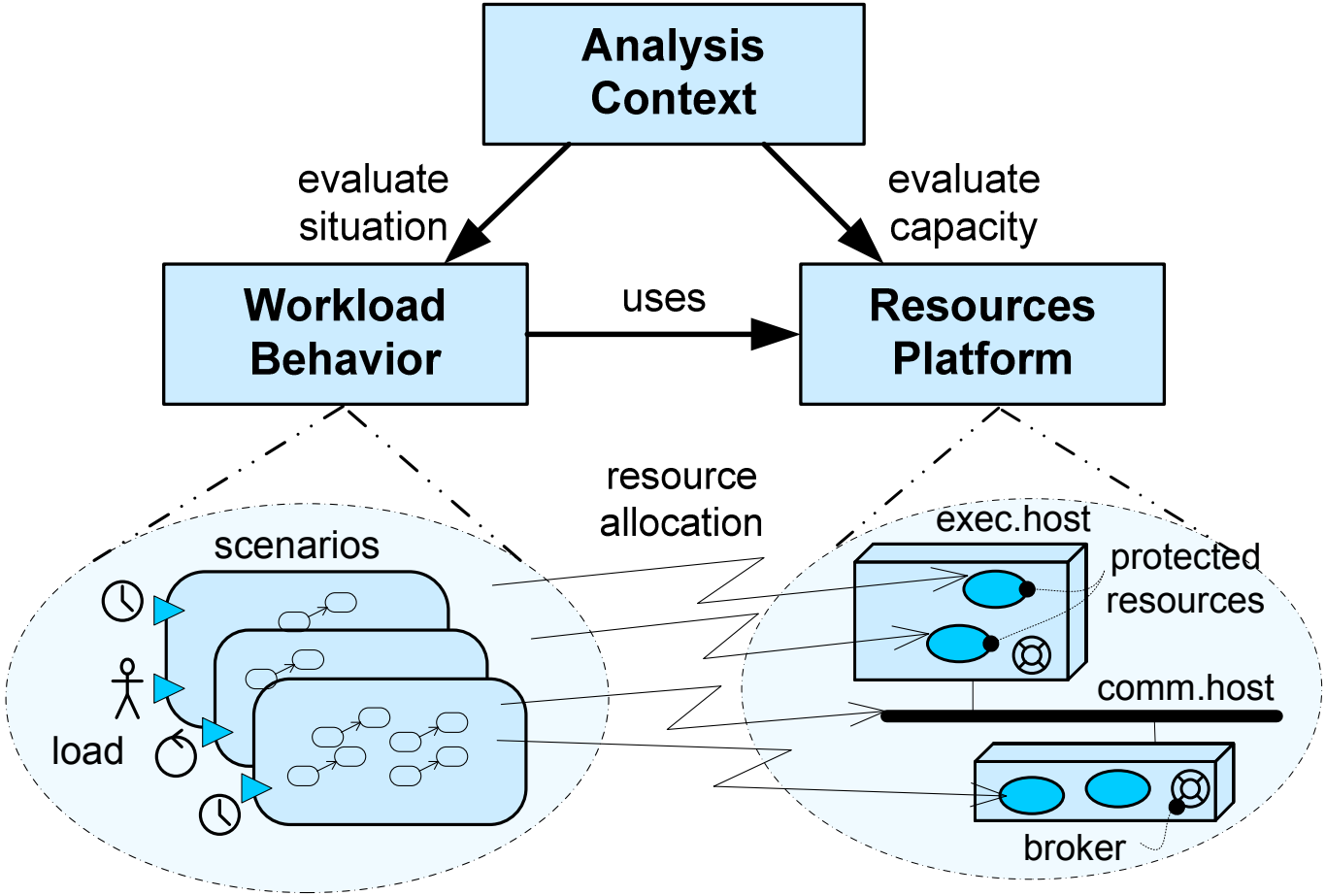
UML-Based Analysis Foundations

- **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

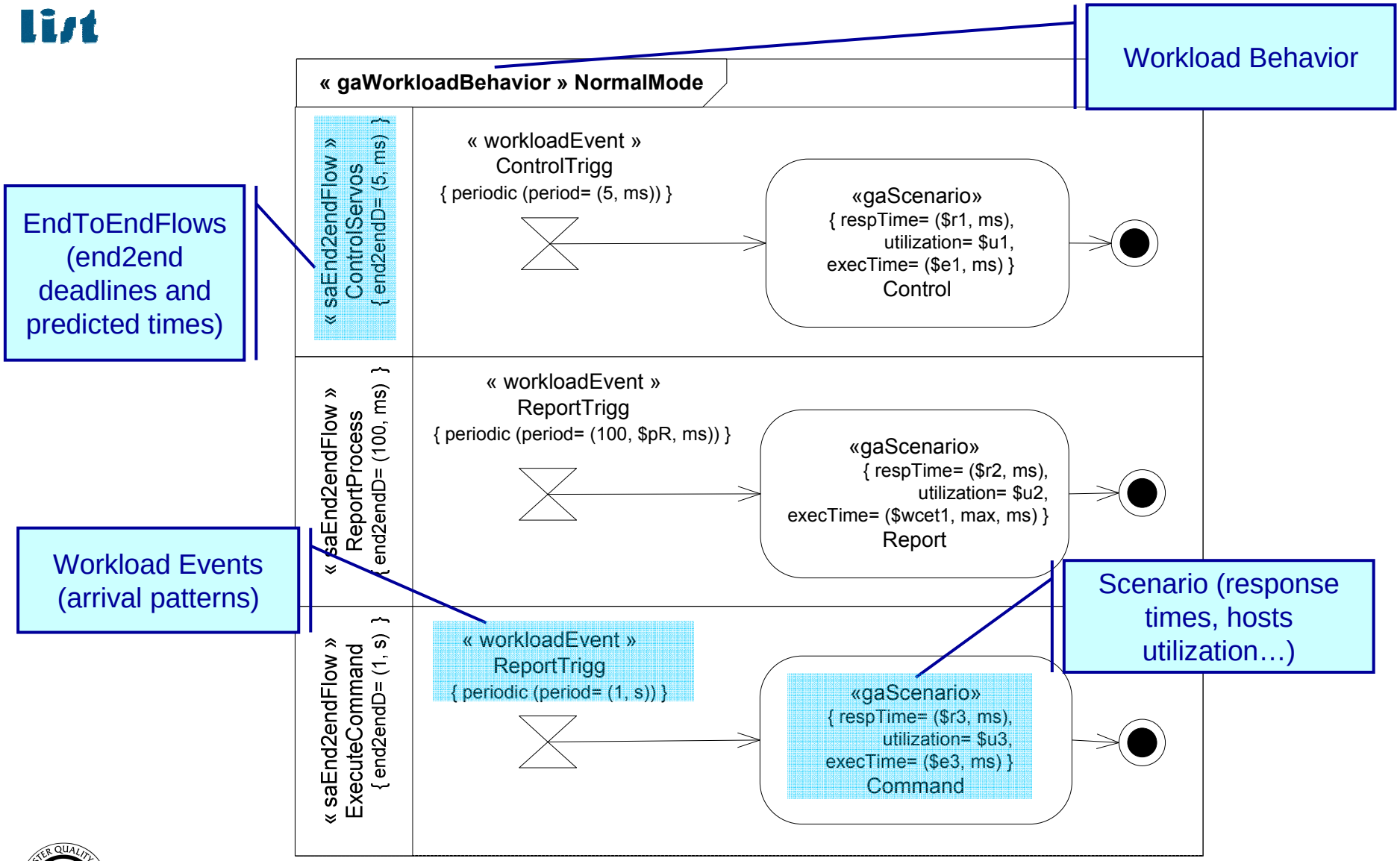
The MARTE analysis sub-profiles provide standard constructs to map UML models on <u>well-established analysis techniques</u>

➔MARTE "Foundations" and "GQAM" allow for extending to further techniques

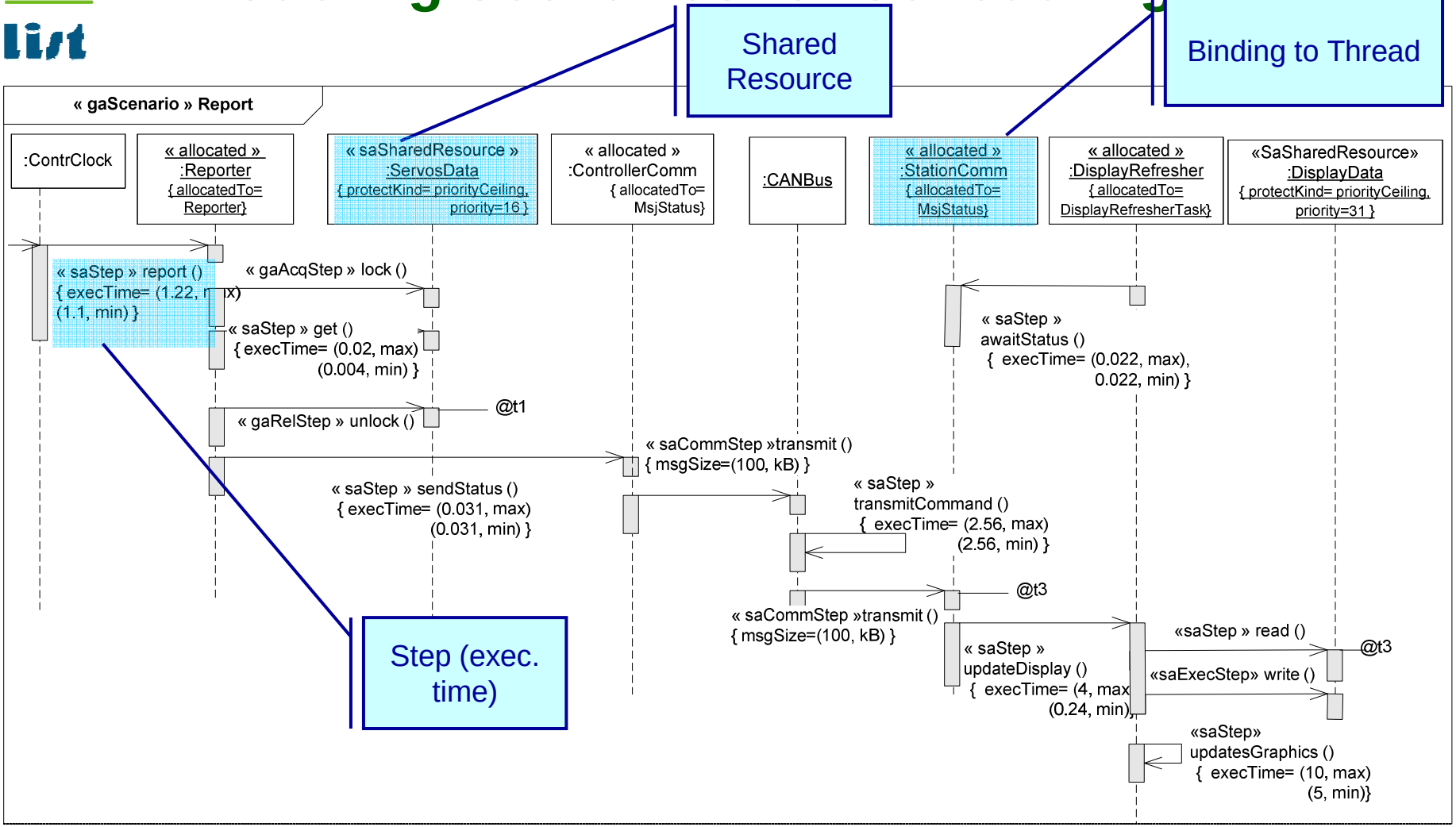
Analysis Modeling Structure



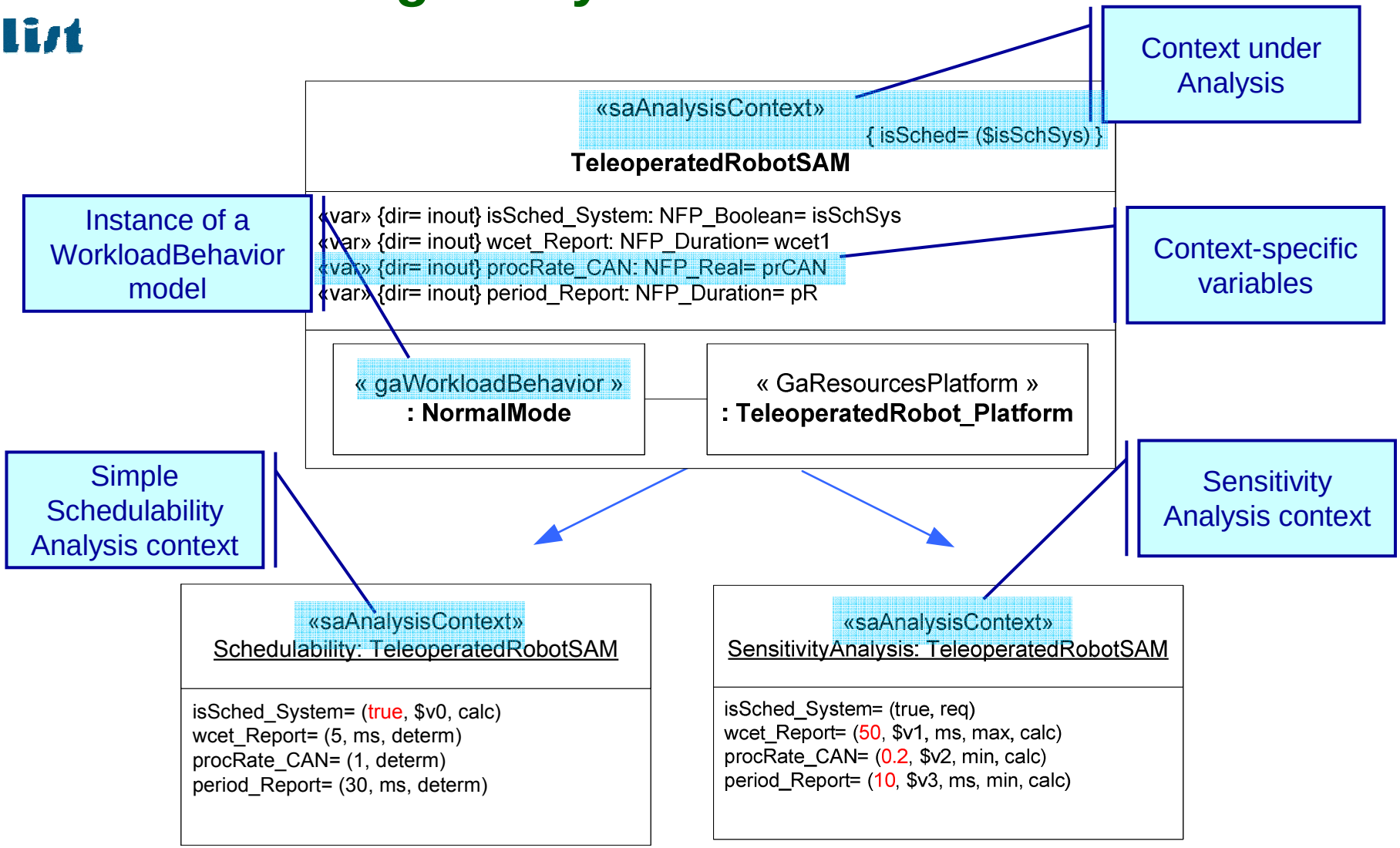
Modeling Workloads for Scheduling Analysis



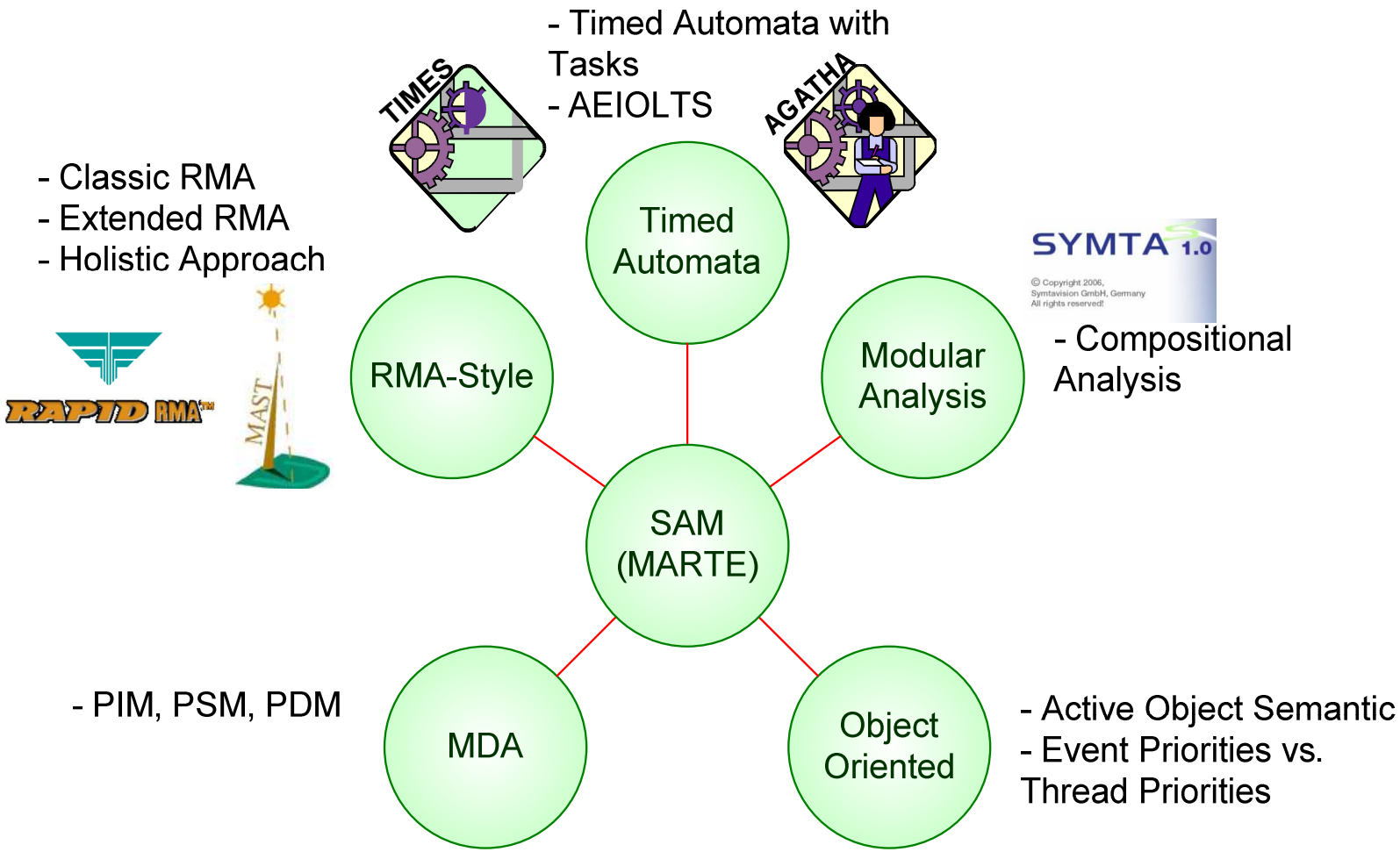
Modeling Scenarios for Scheduling Analysis



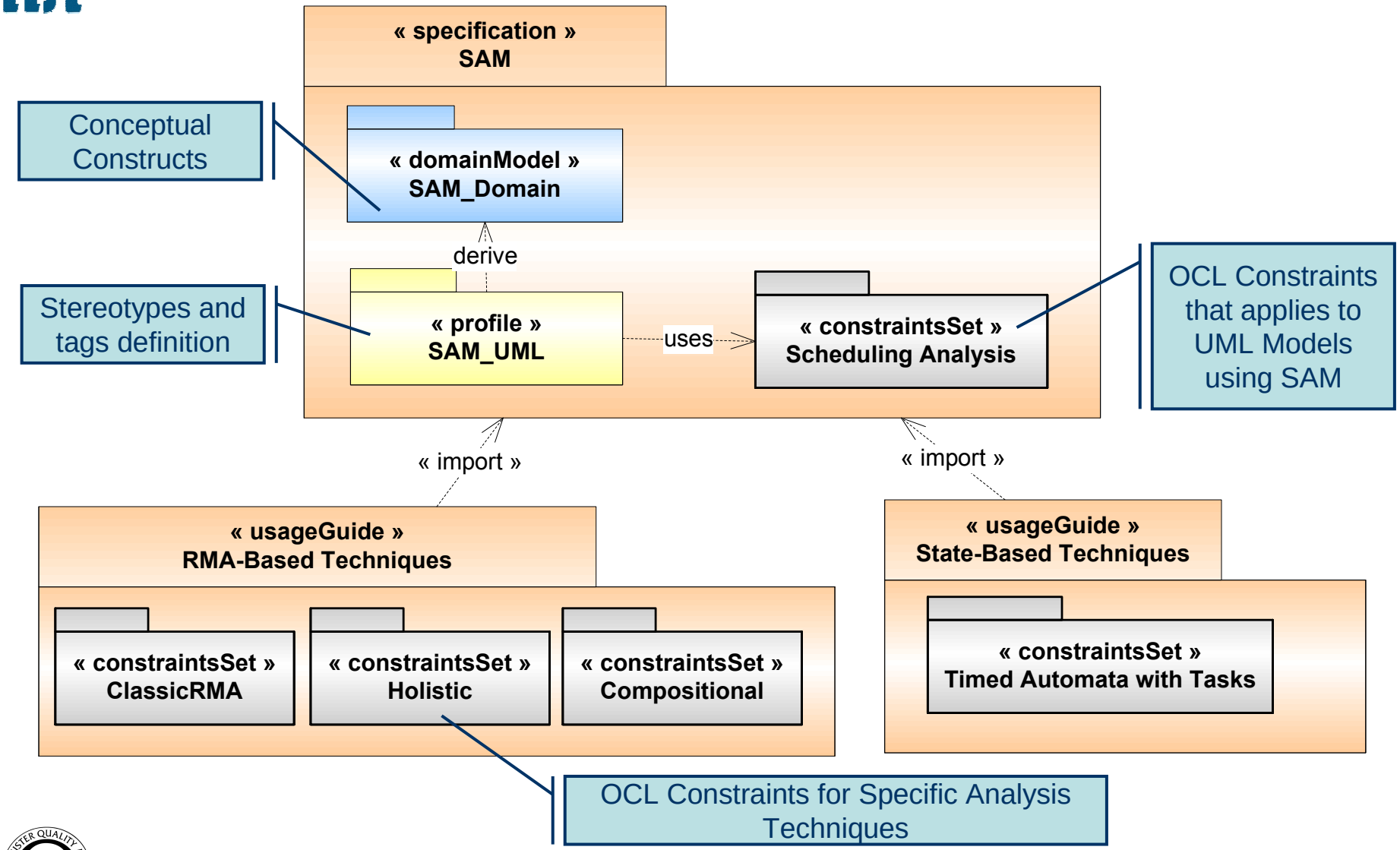
Modeling Analysis Contexts



Schedulability Analysis Modeling Profile



Constraining SAM for Specific Techniques



MARTE Tooling

- **Current Implementations supporting MARTE**
  1. Full MARTE Profile & Libraries for Eclipse UML2
  2. VSL edition assistant and type checker as a Eclipse plug-in for the UML Papyrus tool
  3. Generator of OIL (OSEK Implementation Language) files for OSEK from UML Models annotated with the SRM profile
- **On-going work:**
  1. Eclipse plug-ins to transform UML models annotated with the SAM profile to input files of MAST and SymTA/S tools
  2. Eclipse plug-ins to transform UML models annotated with the HRM profile to processor simulators data files
  3. …

Open Source Implementation in UML Papyrus:
www.papyrusuml.org

Agenda

- Background & Motivation
- UML MARTE
- MARTE for System Analysis
- Frontiers & Challenges
- Summary

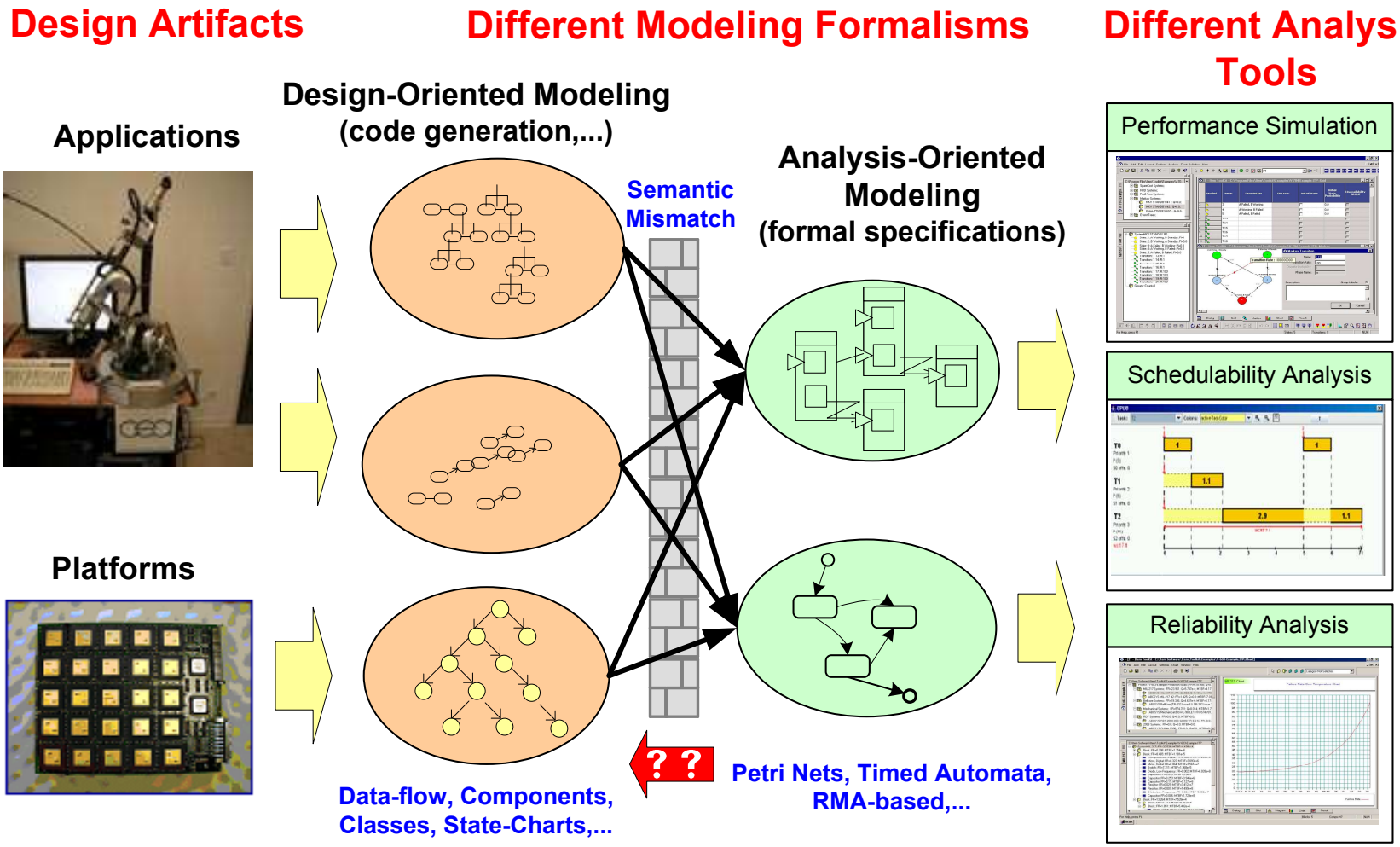
MARTE Frontiers

- **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 large and open specification formalisms!

Revisiting the A&D Integration Problem:

Trends: Dedicated & Monolithic Approaches

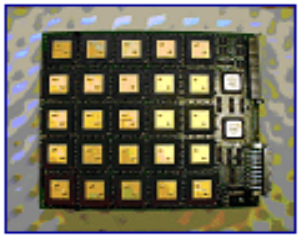


Revisiting the A&D Integration Problem:
Challenge: Standardized & Compositional Approach

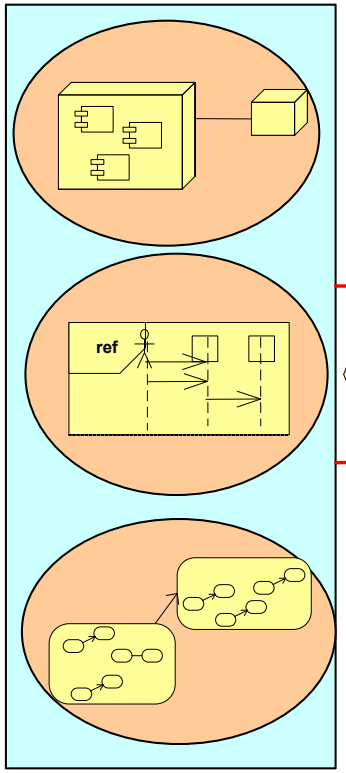
Design Artifacts | **Standard Modeling Formalisms** | **Different Analysis Tools**

Applications

Platforms

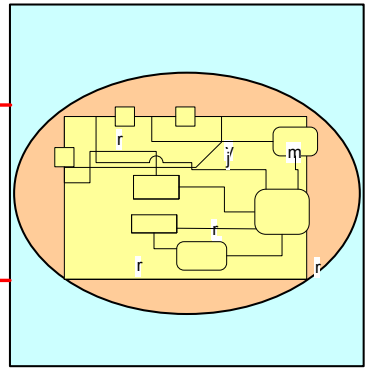


Analyzable Design Models



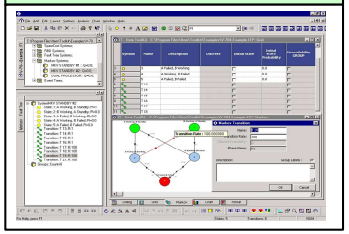
UML + non-functional annotations

Context-Based Analysis Models (composition of analysis building blocks)

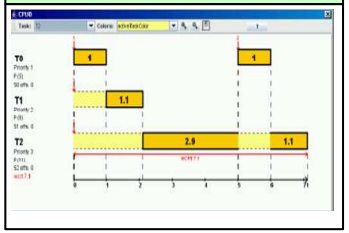


Extended UML Composite Structures

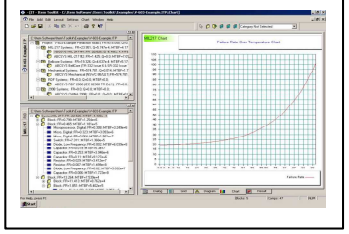
Performance Simulation



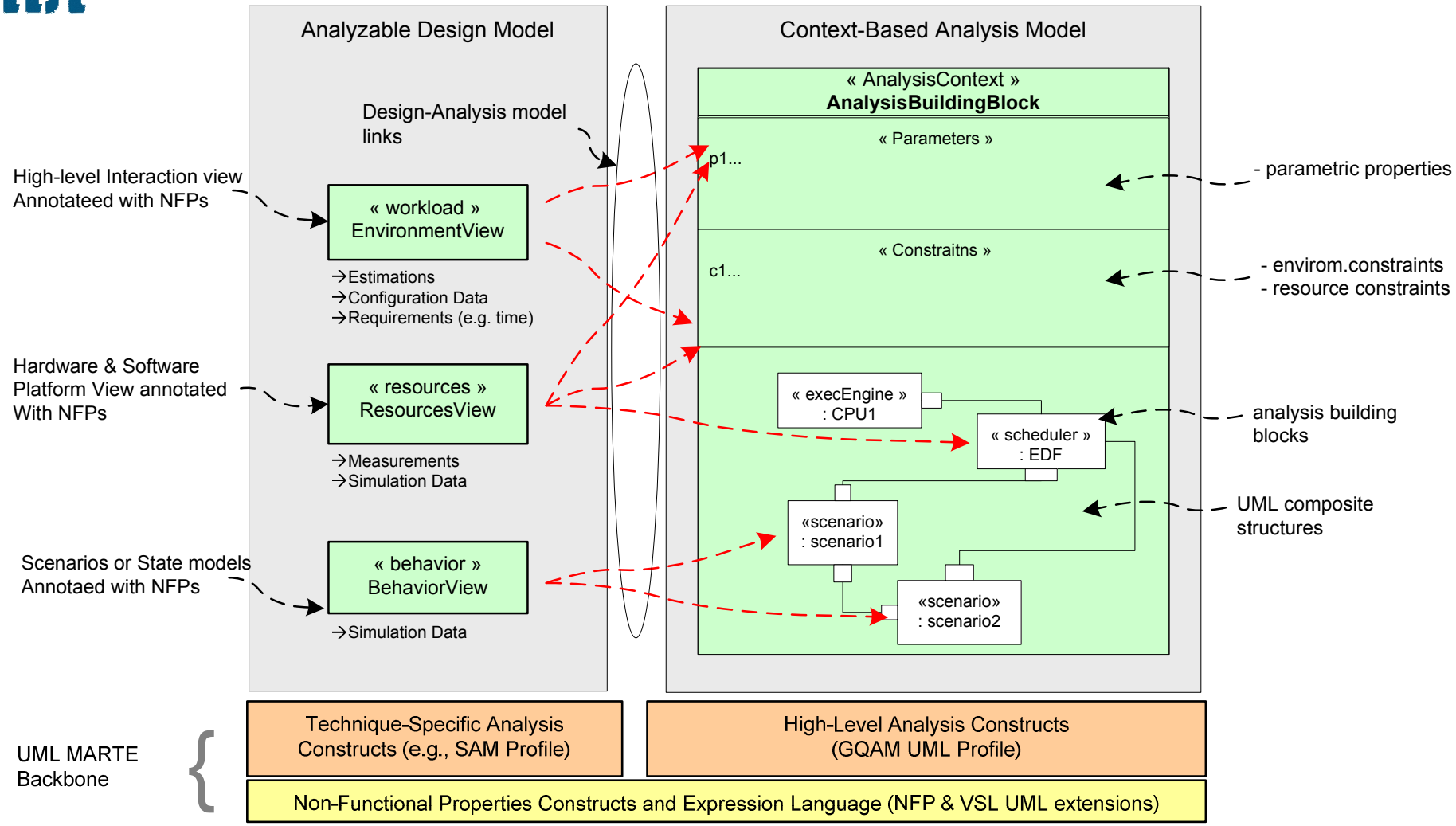
Schedulability Analysis



Reliability Analysis



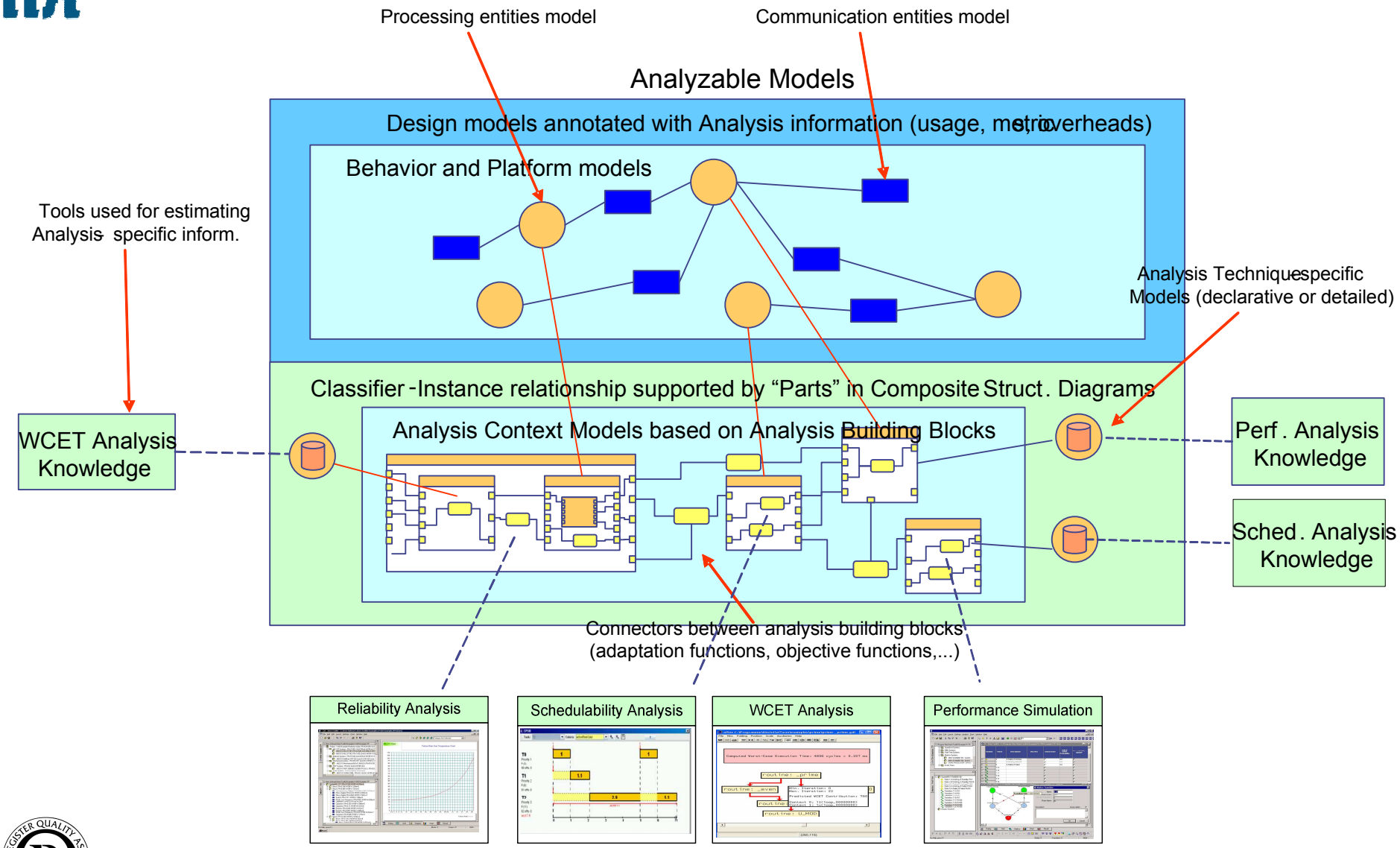
How Does MARTE Support it?



Needs for the Compositional Approach

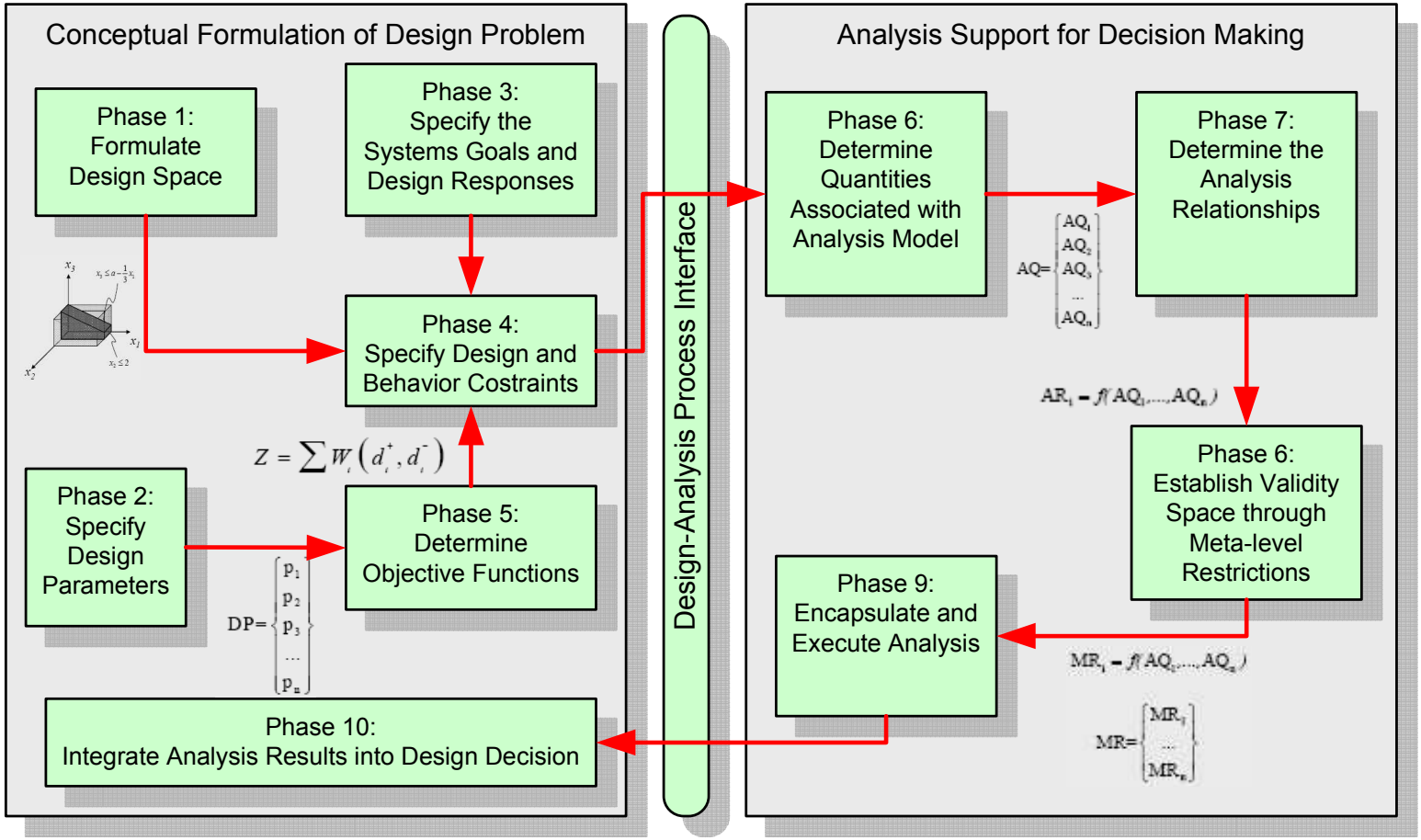
- **Mechanisms to attach analysis annotations in design models**
  - ➔ Consistent analysis views over design views
    - e.g. fragment of activity model annotated with scenario NFPs to extract analysis model
  - ➔ Analysis annotations that not disturb design model
    - e.g. observer and observation mechanism
- **Mechanisms to manage multiple "NFP value views":**
  - ➔ NFP already allow to define different versions of NFP!
    - Use NFP qualifiers to indicate the context of validity of NFP values (refinement, platform binding, analysis context)
  - ➔ NFP already support VSL variables!
    - Use VSL variables to define parametric NFP values
  - ➔ Provide support to manage transformation and consistency checking between different NFP value refinement levels
- **Different analysis techniques in (composite) analysis contexts**
  - ➔ To define adaptation interfaces to transform heterogeneous information

Integration of Multiple Analysis Information



Analysis-Based Design Decision

To cohesively link the design and analysis processes in a well-structured, traceable and constrained decision-making process



Summary

- **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
- **MARTE: a rich MDE-oriented specification language**
  - ➔ Common patterns, base building blocks, standard NFP annotations
  - ➔ Generic constraints that doesn't force specific execution models, analysis techniques or implementation technologies
- **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