Model based schedulability analysis with MAST and the UML Profile for MARTE

Julio Medina
Universidad de Cantabria
SPAIN

WATERS 2010.
Focusing

“...creating a common ground and a community to collect methodologies, software tools, best practices, data sets, application models, benchmarks and any other way to improve comparability of results in the current practice of research in real-time and embedded systems.”

- Various Schedulability Analysis Techniques
- Assumptions about the platform scheduling capabilities
- The model of computation and the semantic link to the design intent
- Characterization of the environment
- Expressing the constraints and expected results

→ Analysis Model ←
Outline

• Basic ideas in (RMA based) schedulability analysis
• The MAST suite: model and tools
• Modeling for schedulability analysis with MARTE
• Discussion:
  – Where we are, what do we need?
  – What are the real possibilities?
Key Ideas

• Real-time goals are: predictability, guaranteed deadlines, and stability in overload.

• Rate monotonic analysis
  – based on rate monotonic scheduling theory
  – analytic formulas to determine schedulability
  – framework for reasoning about system timing behavior
  – separation of timing and functional concerns

• Provides an engineering basis for designing real-time systems
Key Ideas (cont.)

Two concepts help to build the worst-case condition:

- **Critical instant.** The worst-case response time for all tasks in the task set is obtained when all tasks are activated at the same time.

- **Checking the first deadline.** When all tasks are activated at the same time, if a task meets its first deadline, it will always meet all of its deadlines.
Transactional approach for analysis and design

Instance based (Classic RMA, MAST, SPT, MARTE,...)
The MAST suite: model and tools

Brief view of MAST
MAST:
A Timing Behavior Model for Embedded Systems Design/Verification Processes

By:
Michael González Harbour, José Javier Gutiérrez, José Carlos Palencia, José María Drake, Julio Medina, Patricia López

Universidad de Cantabria, Spain
http://mast.unican.es/

1. Introduction: Background

Many real-time systems are now distributed

- Cyclic executives being replaced by run-time schedulers
- Fixed priority and EDF scheduling are most popular among the run-time scheduling policies

Schedulability analysis techniques have evolved a lot in the last decade

- Originally RM and DM priority assignment techniques, together with response-time analysis
- Extended to distributed systems (holistic analysis, HOPA)
- Offset-based analysis introduced (FP and EDF)
- Multiple-event synchronization handled
Motivation

The latest schedulability analysis techniques are difficult to apply by hand

Need for a rich and flexible model of the real-time system:
- distributed, multiprocessor, or single processor
- composable software modules
- separation of architecture, platform, and software modules
- rich set of event-driven patterns; e.g.:
Objectives

• Develop a model for describing the timing behavior of event-driven distributed real-time systems

• Open model that may evolve to include new characteristics or points of view of the system

• Develop a set of tools for analyzing the timing behavior of the application:
  - Schedulability analysis (hard real-time requirements)
  - Synchronization blocking calculation
  - Discrete-event simulation (soft real-time)
  - Priority assignment
  - Sensitivity analysis
MAST: A timing behavior model

1. Introduction: Background, Motivation and Objectives

2. Overview of the Real-Time Model

3. Elements of the MAST Model

4. Integration into design processes

5. The MAST tool suite

6. Conclusions and Future Work
2. Real-Time Model: Overview

Software Modules

- Shared Resources
- Operation
- Event Handler
- Activity
- Timing Requirement

Platform

- Processing Resource
- Scheduler
- Scheduling Parameters
- Scheduling Server

Real-time situation

- Event
- Reference
Real-Time Situation

Transaction

External Event

Activity

Event Handler

Internal Event

Activity

Event Handler

Multicast

Event Handler

Timing Requirement

Event Handlers

Transaction

...
3. Elements of the MAST model: Example

**RMT:** Teleoperated Robot

Remote Station

- GUI
- Trajectory Planner
- Reporter

1sec

50ms

Local Controller

- Command Manager
- Data Sender
- Servo Control

5ms

Ethernet Network

- Command Message
- Status Message
Processing resources, schedulers, drivers, and timers
Scheduling servers

priority

reporter

trajectory planner

gui

data sender

command manager

servo control

ethernet driver1

remote_station

primary scheduler

message stream

ethernet

local_controller

primary scheduler

ethernet driver2

priority

remote_station

primary scheduler

message stream

local_controller

primary scheduler
Shared resources

- **Shared_Resource**
  - Name
- **Immediate_Ceiling_Resource**
  - Ceiling
  - Preassigned
- **Priority_Inheritance_Resource**
- **SRP_Resource**
  - Preemption_Level
  - Preassigned
Logical Operations Model

- **Operation**
  - **Name**
  - **Composite_Operations_List** (1..*)

- **Overridden_Sched_Parameters**
  - 0..1

- **Simple_Operation**
  - **Execution_Times**
    - 0..*
    - To_Lock
    - To_Unlock

- **Shared_Resource**

- **Composite_Operation**

- **Enclosing_Operation**
  - **Execution_Times**

- **Message_Transmission**
  - **Transmission_Times**
GUI operation in the example

- Gui
  - Enclosing operation
    - Set_command
      - Simple operation
        - Command
          - Immediate ceiling resource
    - Read_status
      - Simple operation
        - Status
          - Immediate ceiling resource
Transactions: Distributed transaction in the example

1. **e1**
   - Timed activity
   - Op: tr_planning
   - Srv: tr_planning

2. **e2**
   - Activity
   - Op: command
   - Srv: msg_stream

3. **e3**
   - Activity
   - Op: Comm_mgmt
   - Srv: Comm_mgmt

4. **e4**
   - Activity

5. **e5**
   - Activity
   - Op: data_sender
   - Srv: data_sender

6. **e6**
   - Activity
   - Op: status
   - Srv: msg_stream

Event pattern and timing reqs.
External Events

- **External_Event**
  - **Periodic_Event**
    - Period
    - Max_Jitter
    - Phase
  - **Aperiodic_Event**
    - Avg_Interarrival
    - Distribution
  - **Singular_Event**
    - Phase
  - **Sporadic_Event**
    - Min_Interarrival
  - **Unbounded_Event**
  - **Bursty_Event**
    - Bound_Interval
    - Max_Arrivals
Event Handlers

Activity / Rate Divisor / Delay / Offset

Concentrator (Merge)

Barrier (Join)

Delivery Server (Branch)

Multicast (Fork)
Timing requirements

- **Timing Requirement**
- **Composite Timing Requirement**
- **Simple Timing Requirement**
- **Deadline Timing Requirement**
- **Deadline**
- **Max Output Jitter Req**
- **Event**
- **Max Output Jitter**
- **Referenced Event**
- **Requirements List**
4. Integration into the design process

Components built with their own timing behavior model
- passive components: operations and shared resources
- active components: single or multithreaded, distributed, ...

The model is parameterized
- i.e., actual data for WCETS

Deployment tool
- instantiates the parameterized component models
- provides the platform model
- integrates them with the real time situation model

Automatic schedulability analysis is then made
Integration into the design process

Translation
- Unit Testing
- Integration and Test

Testing
- Validation
- Priority Assignment
- Sensitivity analysis

Design
- Detailed Design
- Mechanism Design
- Architectural Design

Analysis
- Identification of real-time situations:
  - Transactions
  - Timing requirements
  - Workloads

Architecture
- Real-time models
- Scheduling policies

Concurrency patterns
- Synchronization patterns

Mapping real-time properties to subsystems
- High-level real-time analysis

WCET evaluation
- Generation of detailed real-time models

Schedulability analysis
5. The MAST Environment

Model Building Tools
- Component Profile
- Ada Profile
- UML Profile

Data handling Tools
- Results viewer
- Graphical editor
- XML converter

RT Model Data
- Model description
- Results description
- Trace log

Tool launcher
- Simulator
- Sensitivity
- Prio. assignment
- Schedulability analysis
6. Conclusions

MAST defines a model for describing real-time systems

- distributed and multiprocessor
- complex synchronization and event-driven schemes
- composable software modules
- independence of architecture, platform and modules

MAST provides an open set of tools

- hard and soft real-time analysis, FP and EDF
- automatic blocking times, priority assignment, sensitivity analysis, graphical editor...

XML specification and the conceptual modelling approach allows easy integration with other tools and formalisms like UML
**Future Work: MAST 2.0**

- Align MAST with MARTE. The elements in MARTE can be described using MAST, but the names are different. Since MARTE is an industry standard the names in both models should be harmonized.

- Add partitioned scheduling in processors and networks, and resource reservations such as FRESCOR contracts.

- Enhance the modelling capabilities of MAST, by adding two new elements: modelling the effects of mutual exclusion due to the use of a thread, even if it suspends itself, for instance in a synchronous remote procedure call (RPC); and modelling communication switches, such as AFDX switches.
MAST 2.0 (cont.)

- Enhance the timer overhead models.
- Enhance the expressiveness of average-case performance parameters, used by the simulator.
- Add new timing requirements, in particular analysis for maximum queue sizes.
http://mast.unican.es
Key reasoning behind the chances of a standard language for conceptual modeling to be useful in the scheduling analysis of RTE Systems

- Consider the framework of a Model Based approach (MDD).
- 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.
- These 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.
Modeling for schedulability analysis with MARTE

Tutorial on the analysis capabilities of the UML Profile for MARTE
Agenda
(extract from the MARTE Tutorial)

- Part 1
  - Introduction to MDD for RT/E systems & MARTE in a nutshell
- Part 2
  - Non-functional properties modeling
  - Outline of the Value Specification Language (VSL)
- Part 3
  - The timing model
- Part 4
  - A component model for RT/E
- Part 5
  - Platform modeling
- Part 6
  - Repetitive structure modeling
- Part 7
  - Model-based analysis for RT/E
- Part 8
  - MARTE and AADL
- Part 9
  - Conclusions
 Industry Landscape of Analysis Practice

- Why “non-functional analysis” is important?
  - Early: avoid design mistakes, assess 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], …

![Diagram showing industry response, delays, product availability, most designers] 

We need new approaches to cope with analysis complexity!

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.

![Diagram of model-based analysis process]

Automated model conversion

Results conversion back to models
Needs for Model-Based RTES Analysis

Design-Oriented Models
(code generation,...)

Semantic mismatch

Analysis-Oriented Models
(formal models)

Model Editors

Schedulability Analysis

Power Analysis

Data-flow, Components, Classes, State-Charts,...

Holistic models, RMA-based, queuing theory...

How to specify unambiguous non-functional information?

Trade-offs between different parameters?
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
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

- General NFP types
- Processing & Scheduling model
- Performance analysis (non-deterministic performance)
- Schedulability analysis (timeliness)
- Timed processing model

Diagram:
- NFPs
- Time
- GRM
- « modelLibrary » MARTE_Library
- GQAM
- GQAM_Workload
- GQAM_Resources
- GQAM_Observers
- SAM
- PAM
Processing Schema for Analysis

Tech. space for UML modeling

- « profile » MARTE
- UML2 editor
- Annotated model
- Result/Diagnostic model

Tech. space for analysis

- Domain model
- Analysis tool
- Analysis results

Model converter

Results converter

© OMGMARTE.org in WATERS 2010
Copyright THALES – Copyright CEA – Copyright INRIA – Copyright UC
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)

- Three main analysis approaches for verify timeliness:
  - Critical instant calculation
  - Utilization bound test
  - Response time calculation
SAM: Integration Different Approaches

- Classic RMA
- Extended RMA
- Holistic Approach
- Timed Automata with Tasks
- AEIOLTS

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

Step: basic behavioral unit (e.g., execution actions, call actions, messages, …)

Workload event: basic stimuli unit (e.g., timers, external occurrences, internal events, …)

$\Pi$: end-to-end flow
$e_i$: workload event
$\beta_i$: behavior scenario
$\alpha_k$: step

Processing times (worst and best case)
SAM: Precedence Relations

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

- **Sequencial:**
  
- **Merge OR:**
  
- **Join:**
  
- **Decision OR:**
  
- **Fork:**
SAM: Workload Domain Metamodel (end-to-end)

**SAM_Workload**

- **« dataType »**
- **« choiceType »**
  - ArrivalPattern

**Predictions provided by analysis tools**

**SAM_Observers:: TimingObserver**

**EndToEndFlow**

- **workload**

**GQAM_Workload:: WorkloadBehavior**

**GQAM_Workload:: WorkloadEvent**

- **pattern**: ArrivalPattern
- **inputStream**: effect
- **1..* endToEndStimuli**

**GQAM_Workload:: BehaviorScenario**

- **endToEndResponse**: 1

- **endToEndTime**: NFP_Duration
  - **endToEndDeadline**: NFP_Duration

**Stimuli information**

**End-to-end response and deadline times**
SAM: Workload Domain Metamodel (detailed behav.)

- **Processing unit (execution or communication)**
- **Execution units accessing shared resources**
- **Concurrency resources (e.g., threads, channels)**

**SAM_Workload**

- **GQAM_Workload::BehaviorScenario**
  - hostDemand: NFP_Duration
  - respTime: NFP_duration[*]
  - utilization: NFP_Real[*]

- **GQAM_Workload::Step**
  - isAtomic: NFP_Boolean
  - blockingTime: NFP_Duration
  - priority: NFP_Integer

- **GQAM_Workload::PrecedenceRelation**
  - outputRel
  - succes
  - inputRel
  - predec

- **GQAM_Workload::ReleaseStep**

- **GQAM_Workload::AcquireStep**

- **GQAM_Workload::CommunicationStep**
  - msgSize: NFP_DataSize

- **GQAM_Resources::CommunicationChannel**

- **GRM::ResourceUsages::ResourceUsage**

- **GRM::Scheduling::SchedulableResource**

- **SAM_Resources::SharedResource**

- **SaStep**
  - deadline: NFP_Duration
  - spareCapacity: NFP_Duration
  - schedulabilitySlack: NFP_Real
  - preemptedTime: NFP_Duration
  - readyTime: NFP_Duration
  - delayTime: NFP_Duration
### SAM: Example of Stereotype Extensions Usage

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

---

Copyright THALES – Copyright CEA – Copyright INRIA – Copyright UC
SAM: Examples of Behavior Annotations

async. message transmission

syncr. execution message

lock and unlock of a shared resources

behavior scenario
SAM: Example of Precedence Relations Annotation

**Sequential:**

```
{execT=d(a1)} → a1 → a2
```

**Fork:**

```
{execT=d(a2)} → a1
```

**Join:**

```
{execT=d(a3)} → a1
```

```
{execT=d(a4)} → a4
```

```
{execT=d(a2)} → a2
```

```
{execT=d(a3)} → a3
```

```
{execT=d(a4)} → a4
```

**User**
SAM: Example of Workload Annotations

end-to-end flow

triggering events (async. messages)

use of an interaction

concurrent fragments
SAM: Resources Concepts

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

Processing resources (execution and communication)

Scheduler

Shared Resources

Schedulable resources (e.g., threads, channels,...)

- $\epsilon_i$: execution host
- $\kappa_j$: communication host
- $\xi$: shared resource
- SCH: scheduler
- $\tau_m$: schedulable resource

Domain

Extract of the MARTE Tutorial – www.omgmar.te.org in WATERS 2010

Copyright THALES – Copyright CEA – Copyright INRIA – Copyright UC
# SAM: Examples of the Stereotypes Usage

<table>
<thead>
<tr>
<th>SAM Domain Model</th>
<th>SAM Stereotype</th>
<th>UML Metaclasses</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourcesPlatform</td>
<td>GaResourcesPlatform</td>
<td>UML::StructuredClasses::StructuredClass</td>
<td>Main container of resources</td>
</tr>
<tr>
<td>SaExecutionHost</td>
<td>SaExecHost</td>
<td>UML::StructuredClasses::Property</td>
<td>Parts of the resources platform</td>
</tr>
<tr>
<td>SaCommunicationHost</td>
<td>SaCommHost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRM::Scheduler</td>
<td>Scheduler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRM::SchedulableResource</td>
<td>SchedulableRes</td>
<td>UML::StructuredClasses::Property</td>
<td>Parts of processing resources</td>
</tr>
<tr>
<td>SaCommChannel</td>
<td>SaCommChannel</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SAM: Example of Resources Stereotype Usage

- User
- « gaResourcesPlatform » RP1
- « saCommHost » commH1
  - « saCommChannel » ch1 : CH
  - « saCommChannel » ch2 : CH
- « saExecHost » execH1
  - « schedulableRes » schR1 : SCHR
  - « schedulableRes » schR2 : SCHR
- « scheduler » sch1 : SCH

Concurrency resources as nested parts
Scheduler as UML part

Resources platform under analysis
Processing Resources as UML parts

Extract of the MARTE Tutorial – www.omgmar.te.org in WATERS 2010

Copyright THALES – Copyright CEA – Copyright INRIA – Copyright UC
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

GQAM::AnalysisContext

GQAM_Workload::WorkloadBehavior

GQAM_Resources::ResourcesPlatform

SaAnalysisContext

Global analysis annotations

isSchedulable: NFP_Boolean
optimalityCriterion: optimalityCriterionKind

« enumeration » OptimallityCriterionKind

meetHardDeadlines
minimizeMissedDeadlines
minimizeMeanTardiness
undefined

other
SAM: Example of Analysis Context Stereotype Applic.

Interaction representing an analysis context

Allocation to Schedulable resources (link to platform Resources)
Example of Global Development Process

- **REQUIREMENTS MODEL**
  - Use cases
  - Scenarios
  - QoS

- **LOGICAL MODEL**
  - Interactions
  - Behaviour
  - SAM

- **PROTOTYPE MODEL**
  - Components
  - Activities
  - HRM&SRM

- **PLATFORM MODEL**
  - Components
  - Hardware
  - Software

- **SCHED. & PERF. MODEL**
  - Workload
  - Allocation
  - Behaviour
  - WCET Calculation

Copyright THALES – Copyright CEA – Copyright INRIA – Copyright UC
General Procedure to Use the SAM Profile

1. Design Models
   - Annotated Behavior Models
   - Annotate Resources Models

2. Workload Behavior Models (PIM)

3. Resources Platform Models (PDM)

4. Specify Parameterized Analysis Context Model

5. Analysis File

6. Non-functional values for specific analysis contexts

Analysis Tools

Determine Desired NFPs of Interest (given and predicted parameters)
Example: A Teleoperated Robot
Example of Annotated Scenario with SAM
Example of Annotated Resources Model with SAM

```
« gaResourcesPlatform »
TeleoperatedRobot_Platform

« saExecHost »
: Station

<table>
<thead>
<tr>
<th>« schedulableRes »</th>
</tr>
</thead>
<tbody>
<tr>
<td>: DisplayRefresherTask</td>
</tr>
<tr>
<td>{ fp (priority= 22) }</td>
</tr>
</tbody>
</table>

« saCommHost »
: CAN_Bus

<table>
<thead>
<tr>
<th>« schedulableRes »</th>
</tr>
</thead>
<tbody>
<tr>
<td>: MsjStatus</td>
</tr>
<tr>
<td>{ fp (priority= 24) }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>« schedulableRes »</th>
</tr>
</thead>
<tbody>
<tr>
<td>: MsjCommand</td>
</tr>
<tr>
<td>{ fp (priority= 24) }</td>
</tr>
</tbody>
</table>

« saExecHost »
: RobotArm

<table>
<thead>
<tr>
<th>« schedulableRes »</th>
</tr>
</thead>
<tbody>
<tr>
<td>: CommandManager</td>
</tr>
<tr>
<td>{ fp (priority= 16) }</td>
</tr>
</tbody>
</table>

« saExecHost »
: Controller

<table>
<thead>
<tr>
<th>« schedulableRes »</th>
</tr>
</thead>
<tbody>
<tr>
<td>: ServosControllerTask</td>
</tr>
<tr>
<td>{ fp (priority= 30) }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>« schedulableRes »</th>
</tr>
</thead>
<tbody>
<tr>
<td>: Reporter</td>
</tr>
<tr>
<td>{ fp (priority= 24) }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>« schedulableRes »</th>
</tr>
</thead>
<tbody>
<tr>
<td>: MsjCommand</td>
</tr>
<tr>
<td>{ fp (priority= 24) }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>« scheduler »</th>
</tr>
</thead>
<tbody>
<tr>
<td>: RTOS_Scheduler</td>
</tr>
<tr>
<td>{ schedPolicy= FixedPriority }</td>
</tr>
</tbody>
</table>

Threads owned by the processing resource
```
Example of Analysis Context Model

End To End Flows (end2end deadlines and predicted times)

Workload Events (arrival patterns)

Workload Behavior

Scenario (response times, hosts utilization…)

Extract of the MARTE Tutorial – www.omg.org/marte

www.omg.org/marte

Copyright THALES – Copyright CEA – Copyright INRIA – Copyright UC
Example of Parametric Analysis Context

Context under Analysis

TeleoperatedRobotSAM

{ isSched= ($isSchSys) }

Instance of a WorkloadBehavior model

« saAnalysisContext »

Schedulability: TeleoperatedRobotSAM

isSched_System= (true, $v0, calc)
wcte_Report= (5, ms, determ)
procRate_CAN= (1, determ)
period_Report= (30, ms, determ)

Instance of a WorkloadBehavior model

« gaWorkloadBehavior »

: NormalMode

« GaResourcesPlatform »

: TeleoperatedRobot_Platform

Context-specific variables

Sensitivity Analysis context

Simple Schedulability Analysis context

« saAnalysisContext »

SensitivityAnalysis: TeleoperatedRobotSAM

isSched_System= (true, req)
wcte_Report= (50, $v1, ms, max, calc)
procRate_CAN= (0.2, $v2, min, calc)
period_Report= (10, $v3, ms, min, calc)
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](http://www.papyrusuml.org)
IBM RSA:  [www.omgmarje.org](http://www.omgmarje.org)
Conclusions on MARTE’s Analysis

- **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
Elements for discussion

- Techniques comparison
- Models
  - Semantics
  - Syntax
  - Automation
- Tools
- Are all these domain specific?
  Or they are simply different flavors of complexity management