Schedulability Analysis with MARTE and MAST

Julio Medina
Universidad de Cantabria

Topics

- A flash of schedulability analysis.
- RT Modeling constructs in MAST.
- Mapping.
- Structural and behavioral modeling elements in MARTE.

Options for managing time

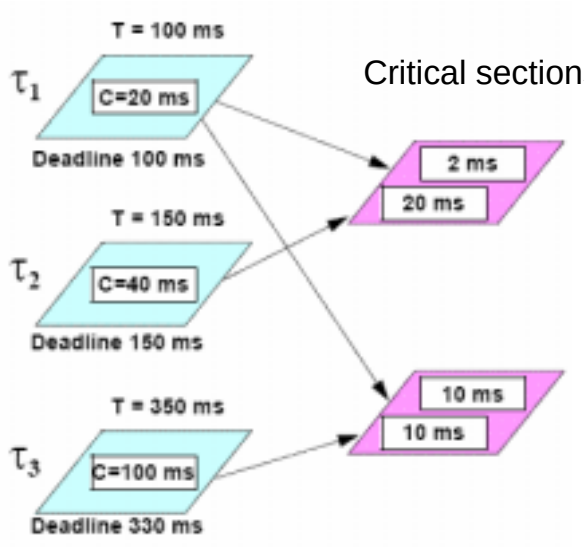
- Compile-time schedules:
  - Known as cyclic executives
  - Predictability through static schedule
  - Logical integrity often compromised by timing structure
  - Difficult to maintain, even with model based strategies
- Run-time schedules:
  - Priority-based schedulers
  - Preemptive or non preemptive
  - Fixed priority or dynamic priority (deadline based i.e.)
  - Analytical methods needed for predictability
  - Separates logical structure from timing

→ RMA

- Server-based frameworks:
  - Combine static or dynamic schedules with budgets and periods enforced at run-time

A simple example

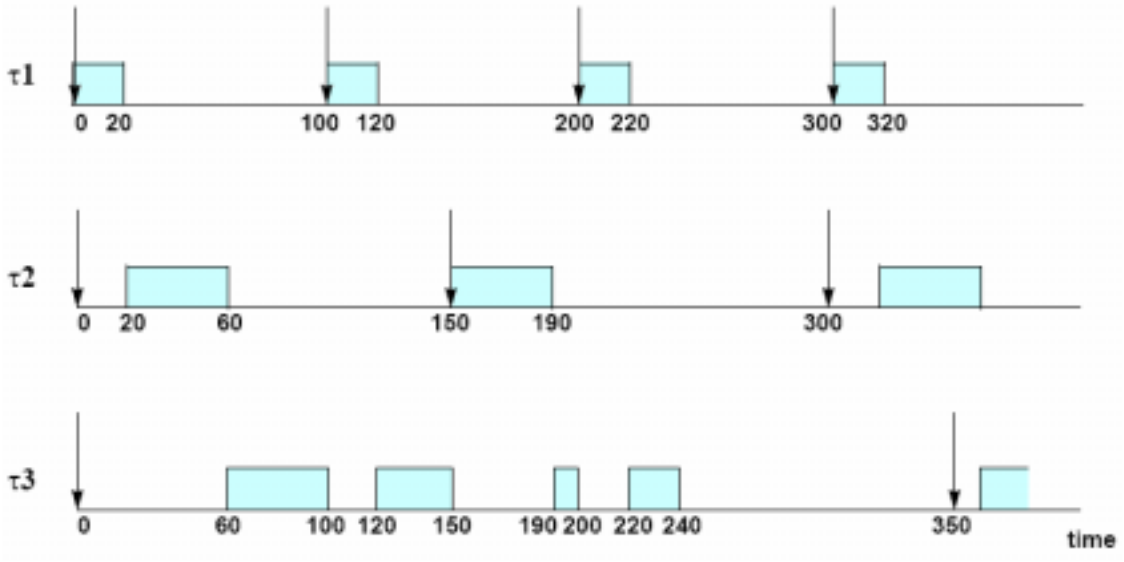
Tasks



Basic Principles of RMA

- 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
- Based on these concepts, several results arise:
  - Optimality of rate monotonic priorities
  - Utilization bound test
  - Exact test

Critical instant



Utilization and Response Time tests

- Utilization Bound(UB) Test: A set of n independent periodic tasks, with deadlines at the end of the periods, scheduled by the rate monotonic algorithm will always meet its deadlines, for all task phasings, if: U <= U(n) = $n(2^{1/n} - 1)$
- Completion time test:
  - For a number the tasks according to priority (highest priority=t_1, lowest priority=t_n)
  - Under a critical instant condition, the amount of work Wi(t) of tasks at priority Pi or higher started before t is

$$W_i(t) = \left\lceil \frac{t}{T_1} \right\rceil C_1 + \ldots + \left\lceil \frac{t}{T_{i-1}} \right\rceil C_{i-1} + C_i$$

Response time evaluation

- *R*i may be computed by the following iterative formula:

$$a_0 = C_1 + C_2 + \ldots + C_i$$

$$a_{k+1} = W_i(a_k) = \left\lceil \frac{a_k}{T_1} \right\rceil C_1 + \ldots + \left\lceil \frac{a_k}{T_{i-1}} \right\rceil C_{i-1} + C_i$$

- The iteration ends when:

$$a_{k+1} = a_k = R_i$$

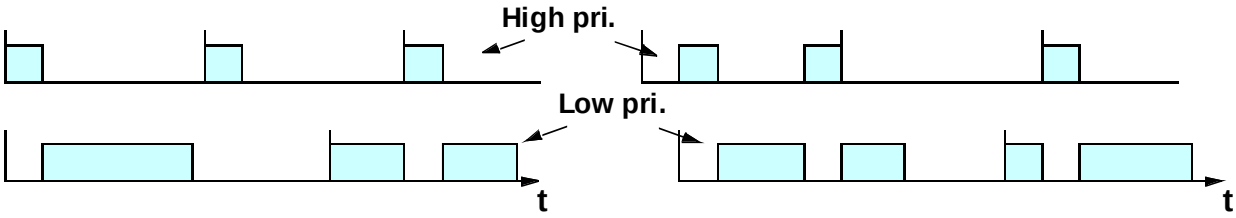
- Task t_i is schedulable if:

$$R_i \leq D_i$$

Effects of jitter

- Periodic events with jitter have an arrival time which may be early or late, within a bounded interval:
  - events arrive at $t_o + nT +/- J$

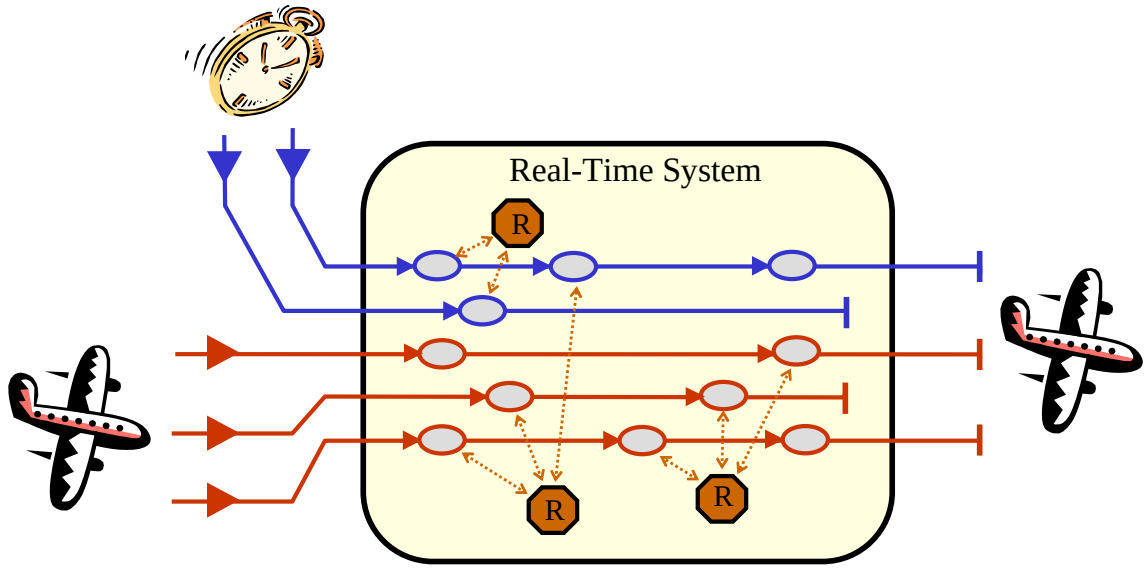
- Jitter may have a delay effect on lower priority tasks



Execution sequence for two periodic tasks. Worst case is one preemption

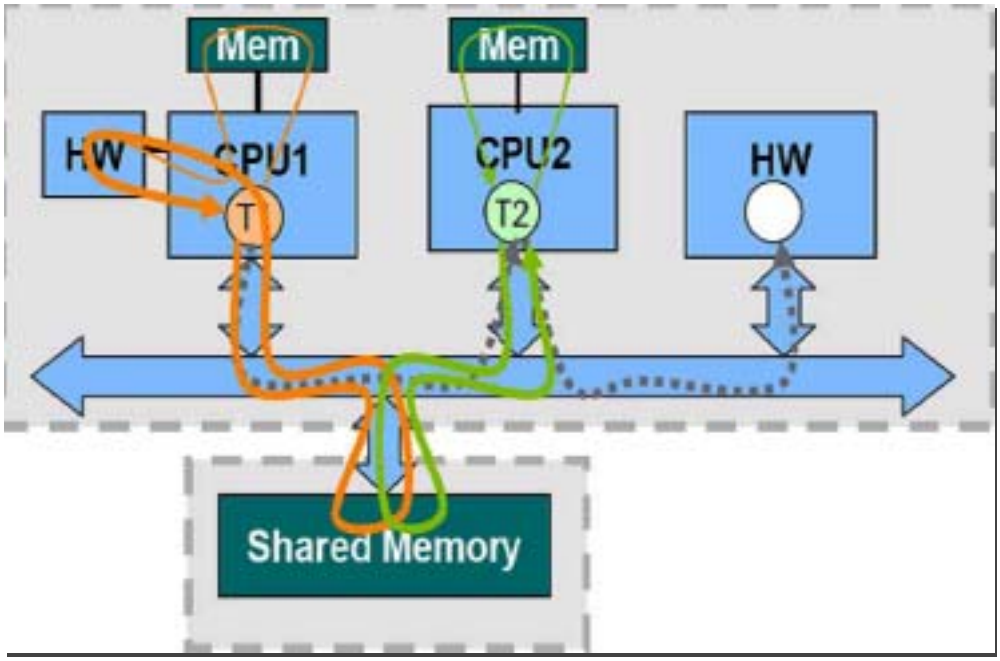
Execution sequence with jitter. Worst case is two preemptions

More general approach



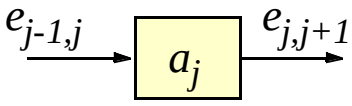
Scenario(instance) based, distributed, control-flow dependencies

Distributed System



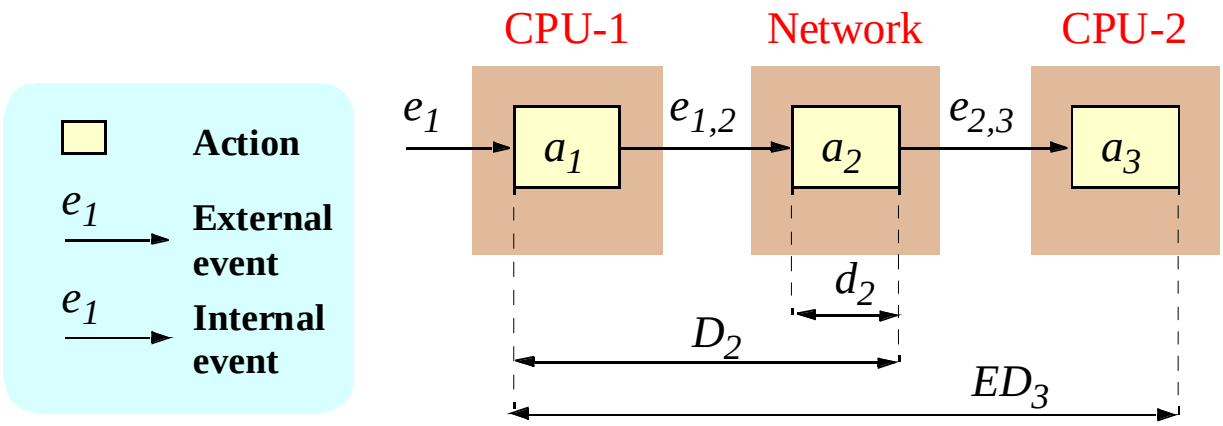
Distributed system model

Linear Action:

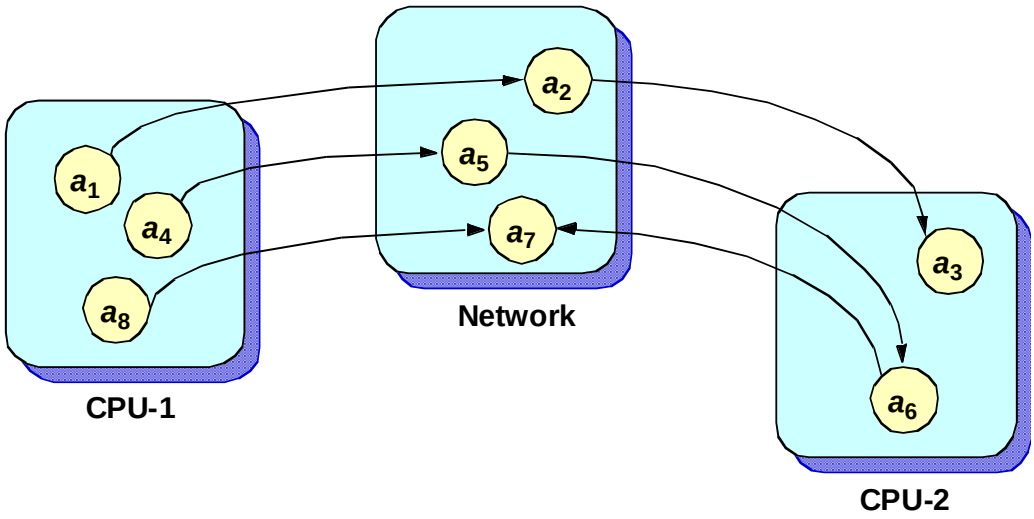


$$T_{j-1,j} = T_j = T_{j,j+1}$$

Linear Response to an Event:

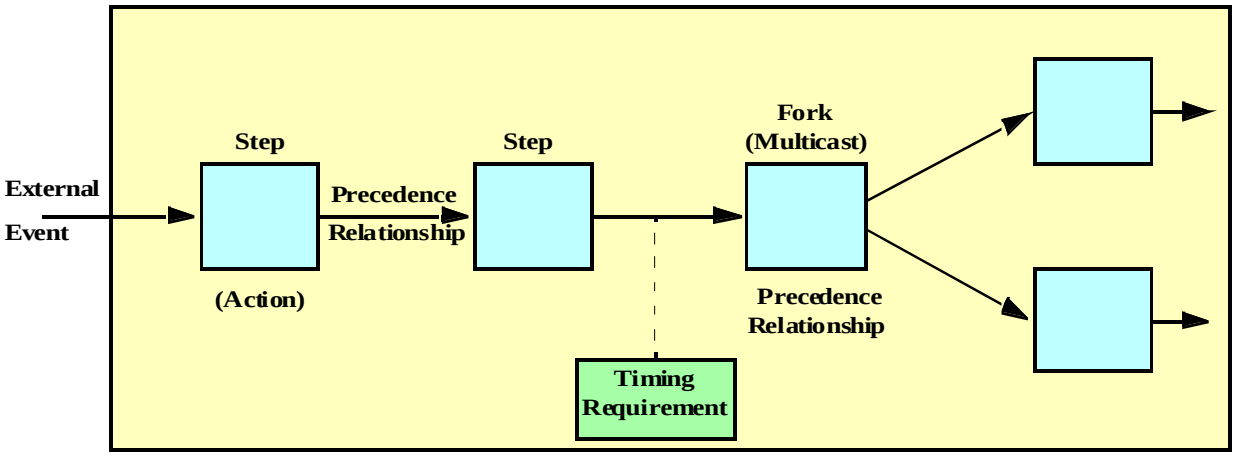


Jitter in distributed systems

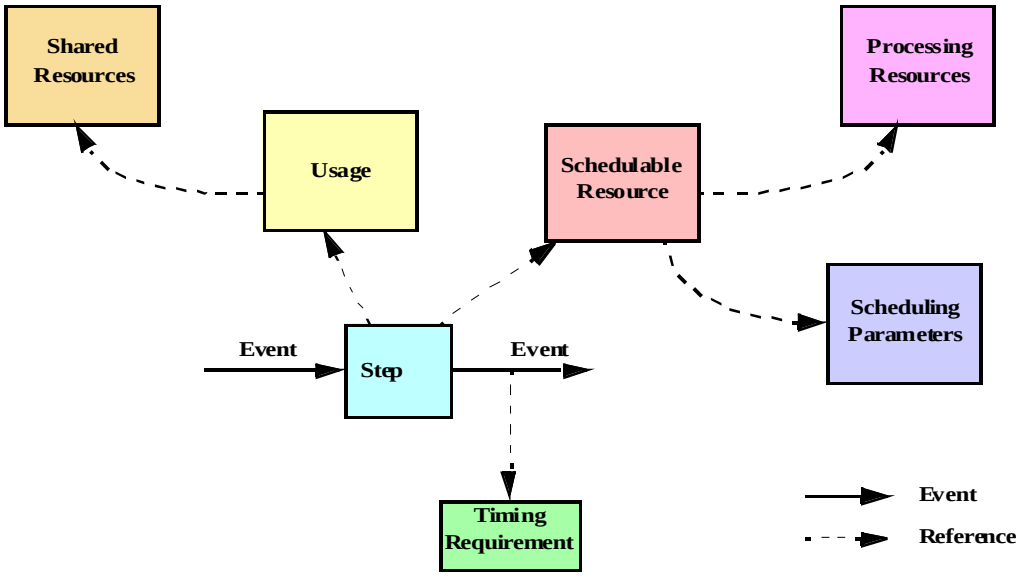


- Jitter in one processing resource depends on the response times in other processing resources
- Response times depend on jitters

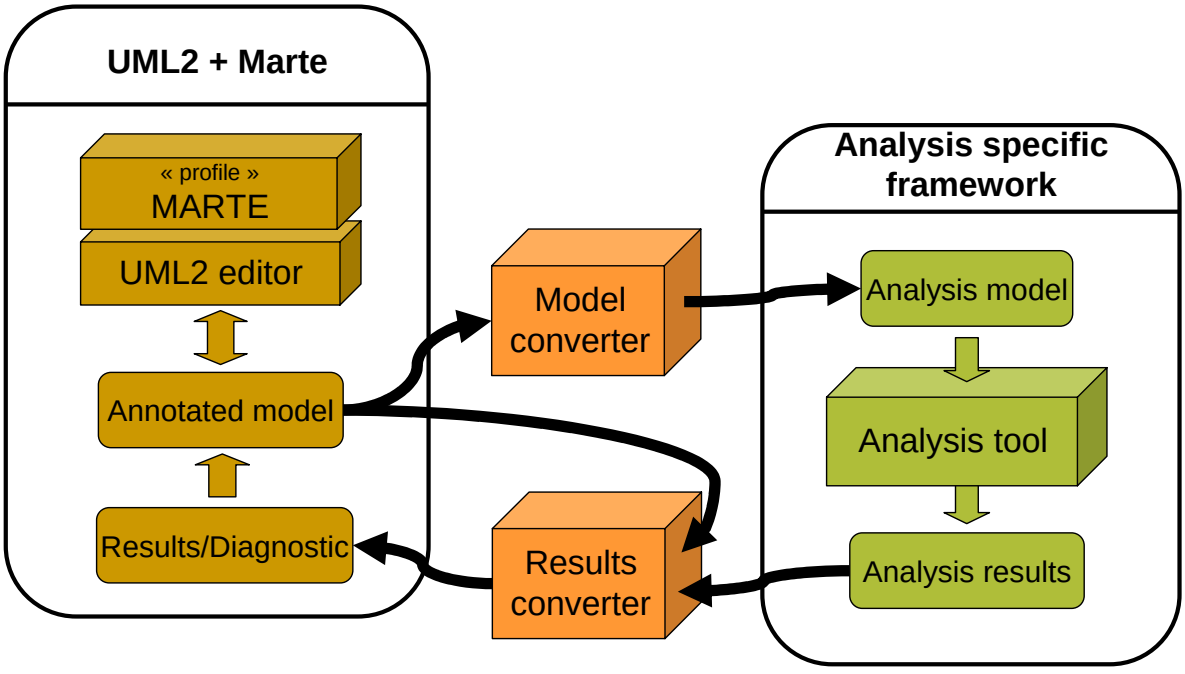
Transactions



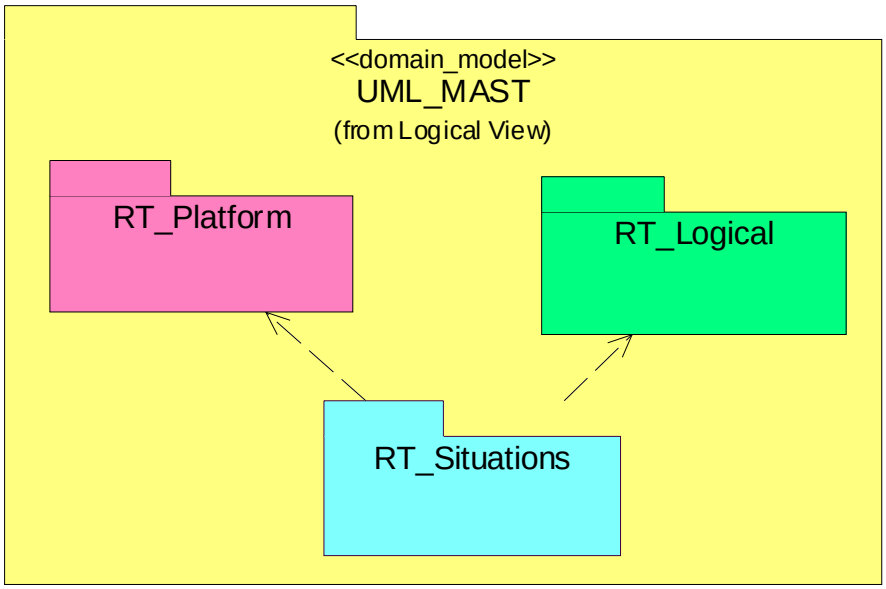
Transactions



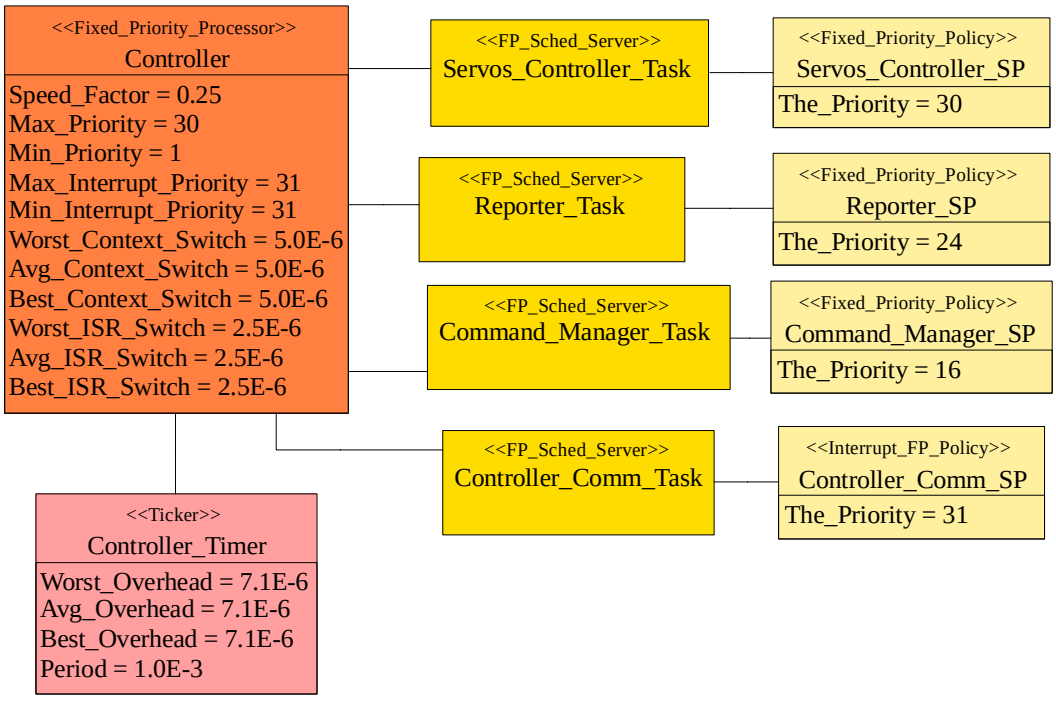
Processing schema for model-based analysis



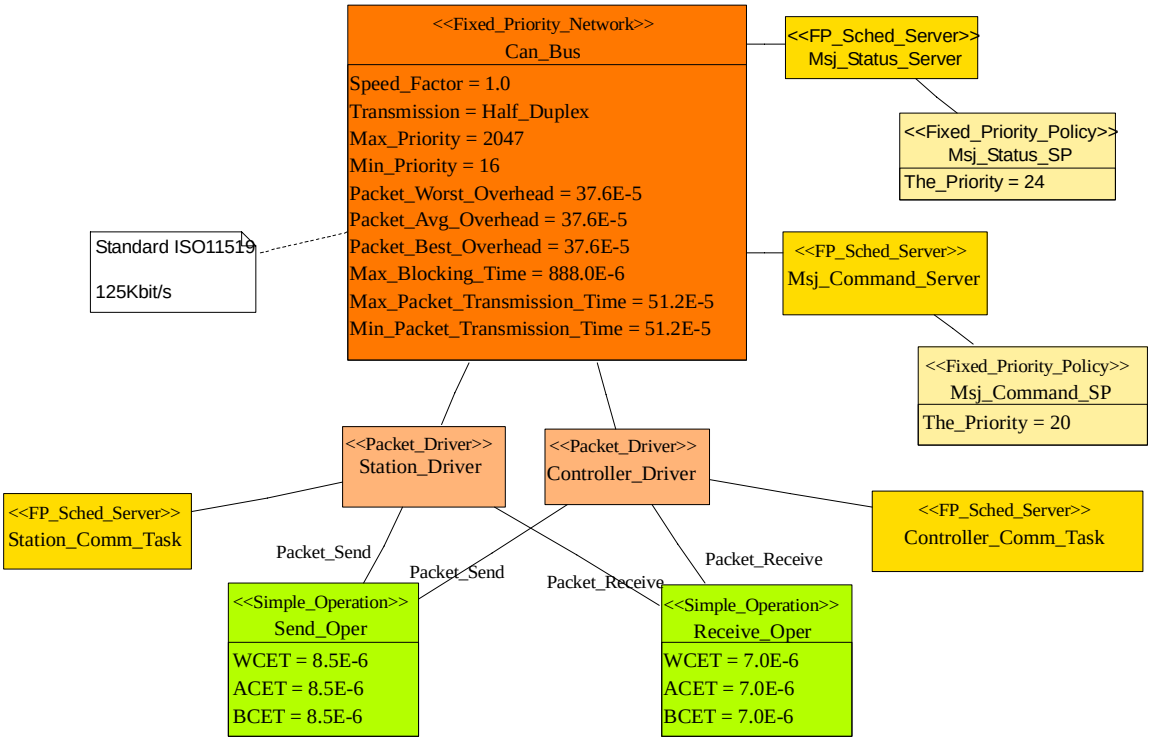
Concepts in MAST



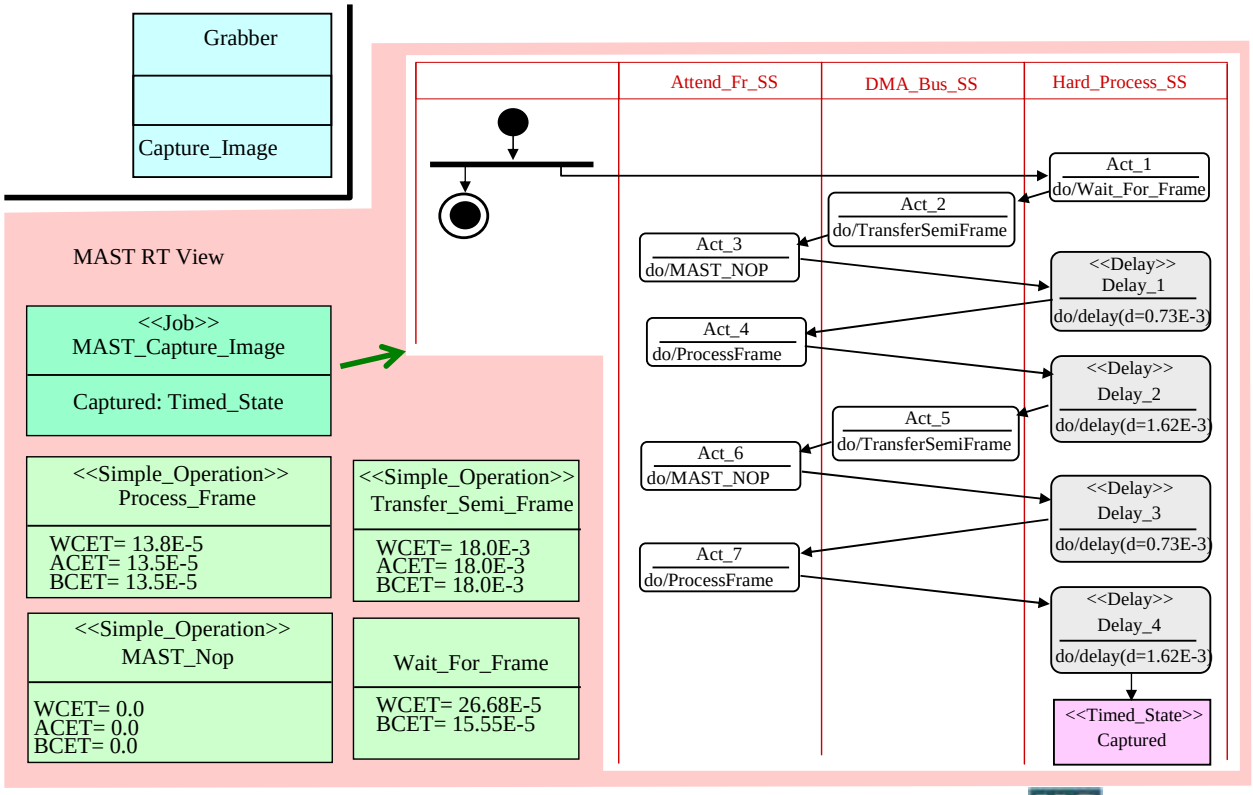
Processors and threads



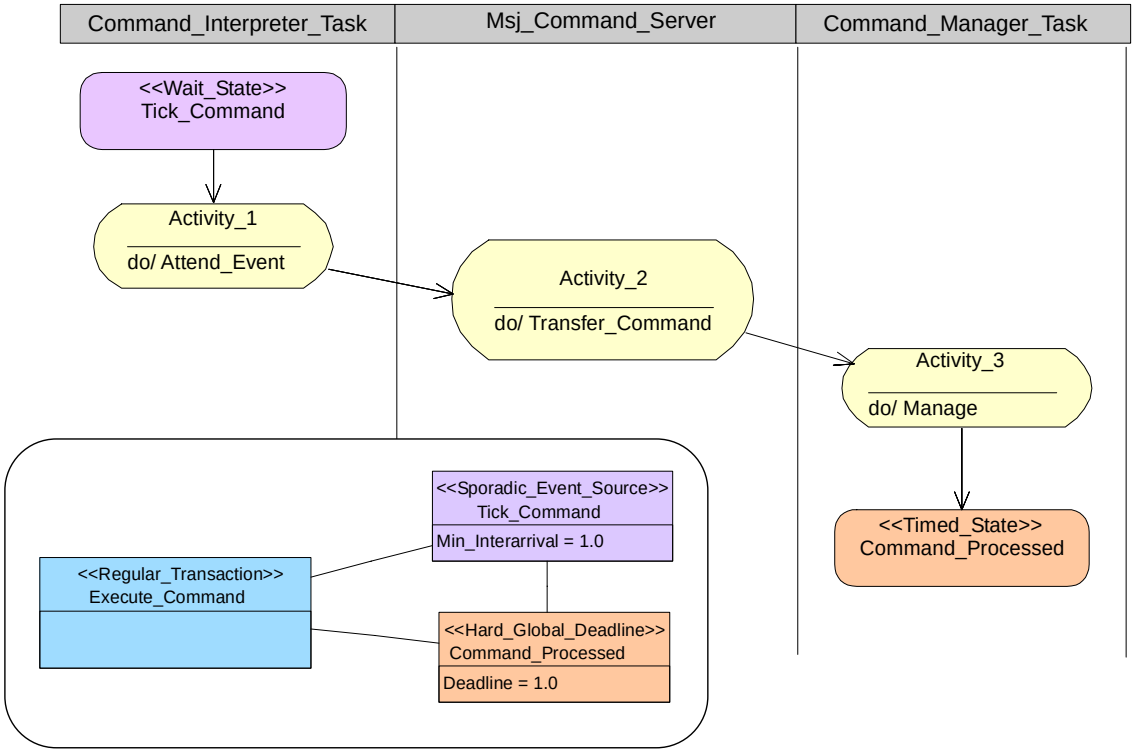
Networks



Operations



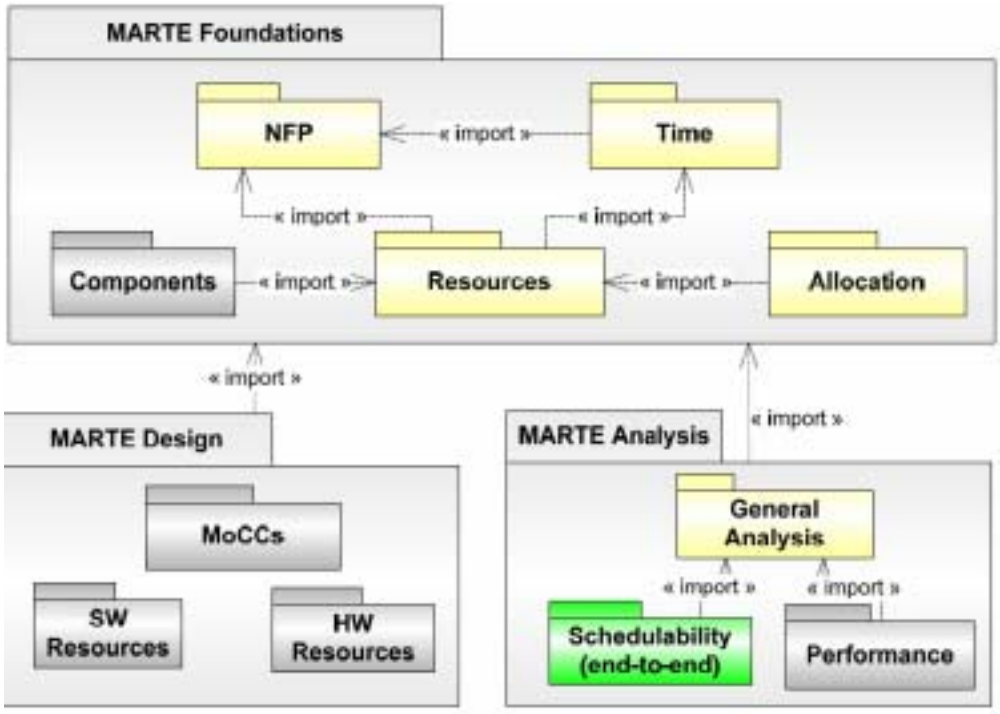
Transactions



Mapping to MAST

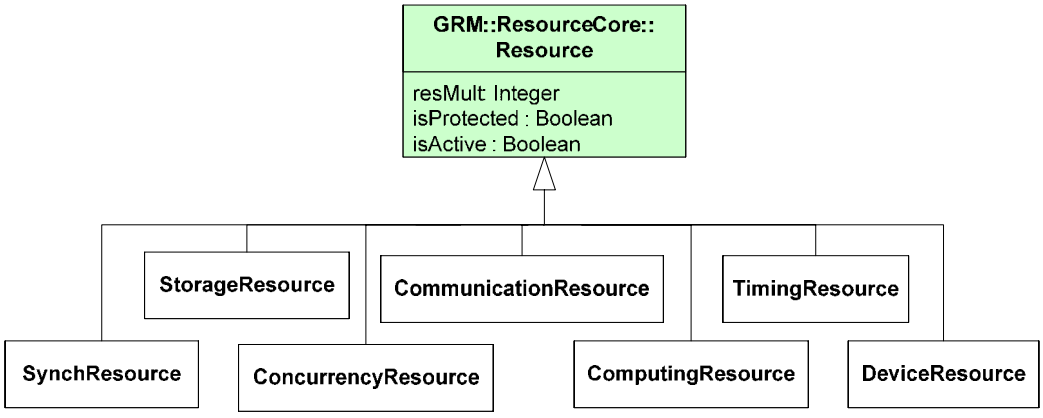
- Platform:
  - Processor → SaExecHost
  - Network → SaCommHost
  - Drivers → AnalisisContext w/EndtoEndFlows
  - Shared_Resources → SaSharedResource
- Logical components:
  - Operation → SaStep (GaStep, ResourceUsage)
  - Job → GaScenario
- RT Situations:
  - Transactions → SaEndToEndFlow
  - TimingRequirement → SaSchedObs, GaLatencyObs
  - ExternalEvent → GaWorkloadEvent

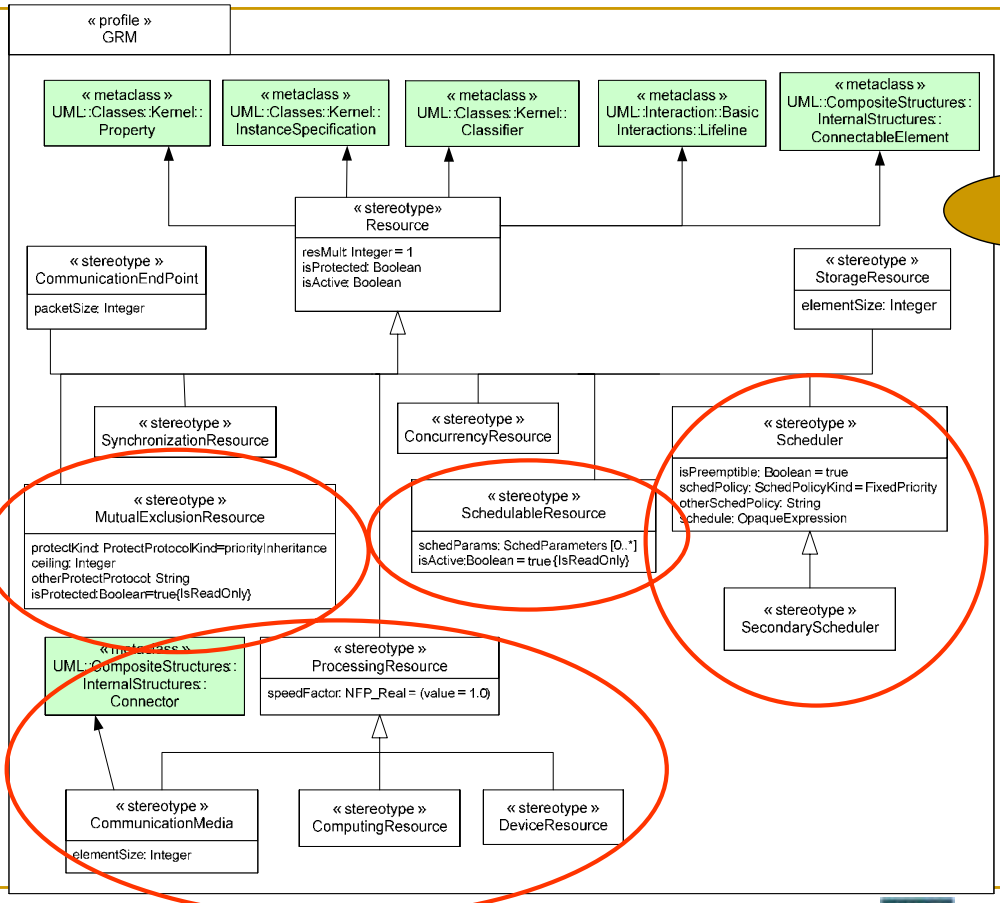
SAM in MARTE



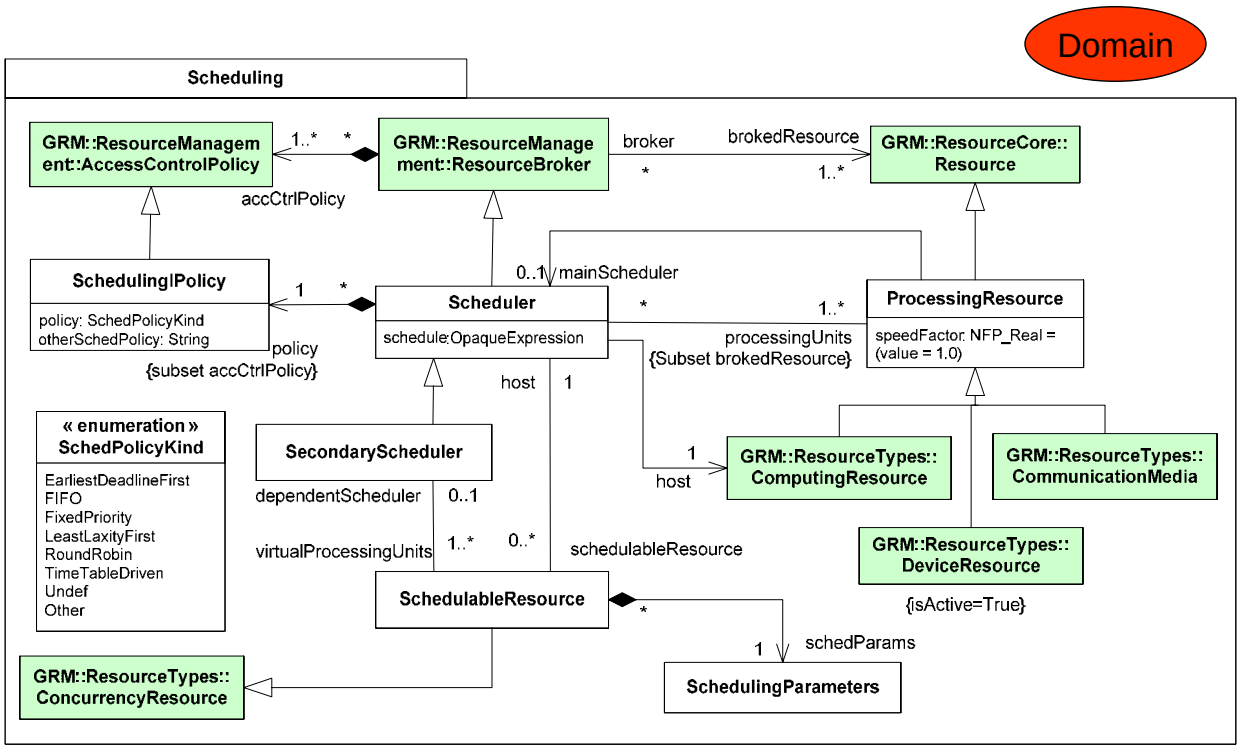
Resources

Domain



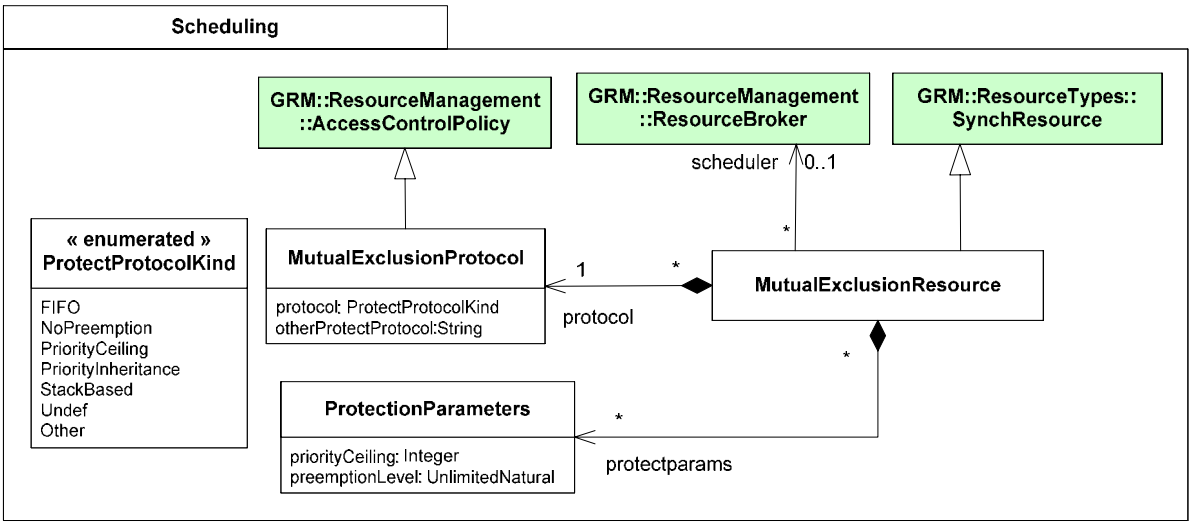


Domain model for Schedulable resources



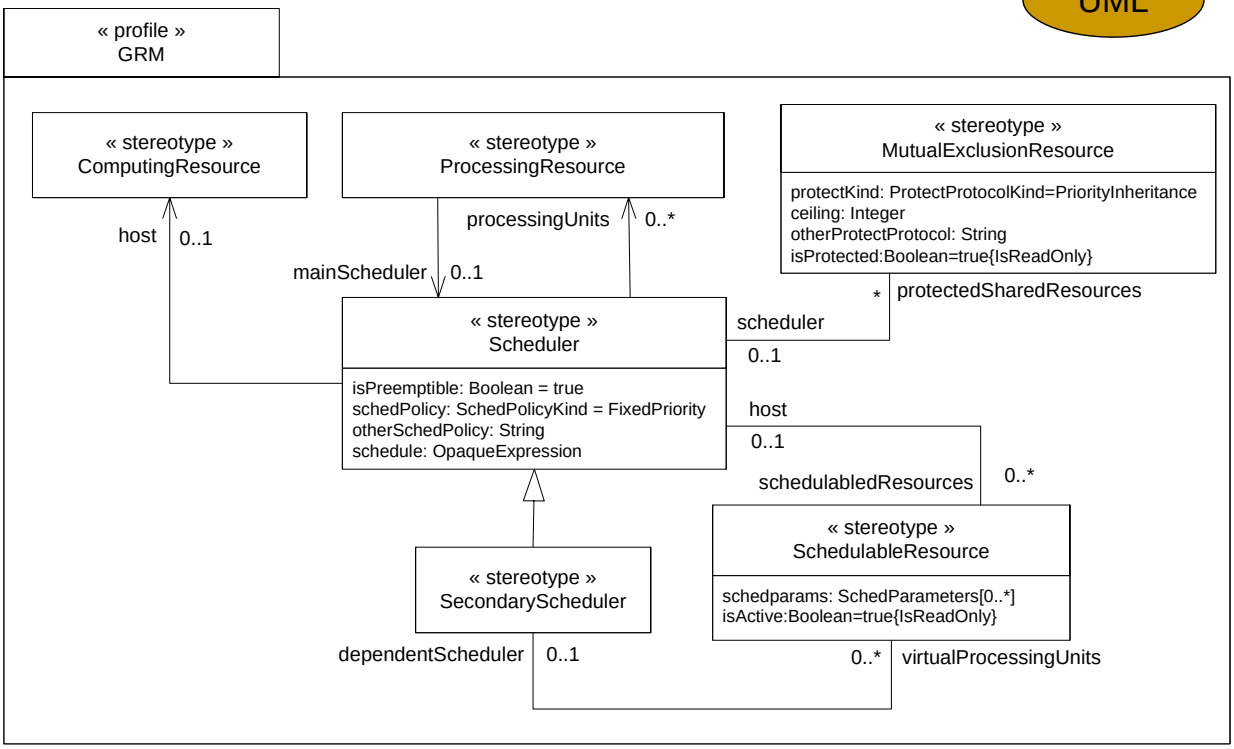
Shared resources

Domain

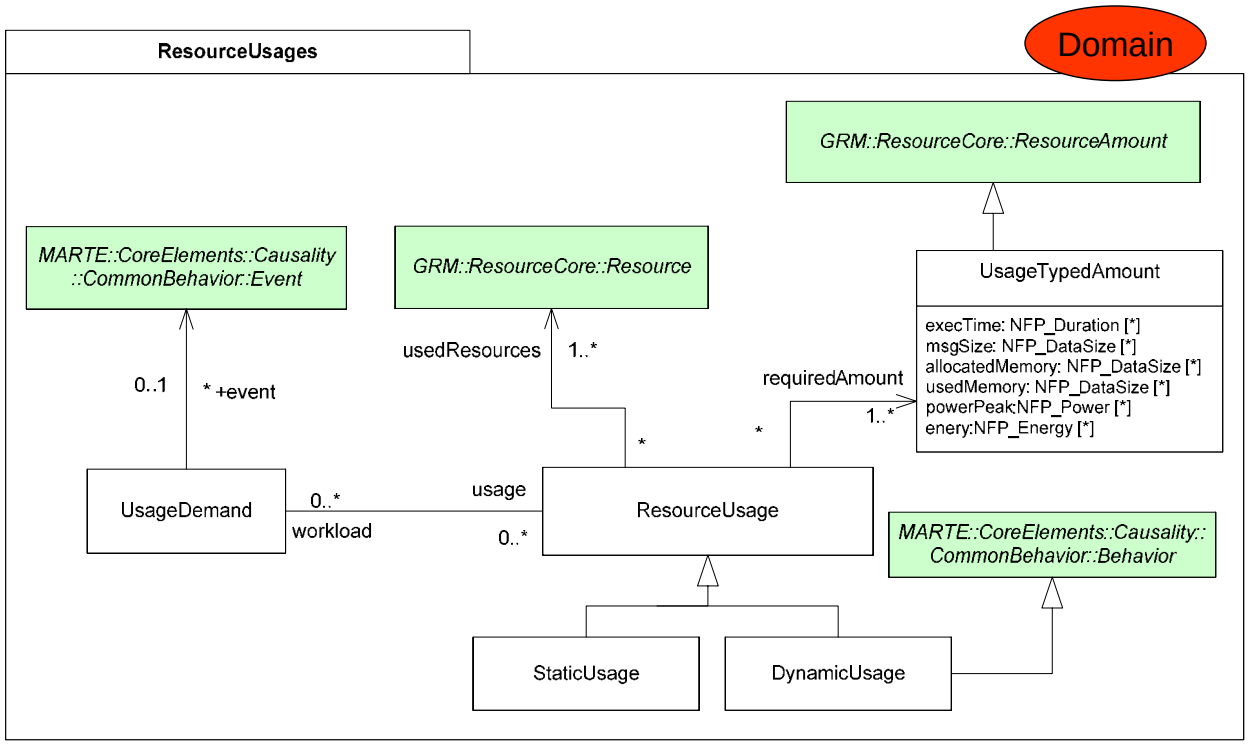


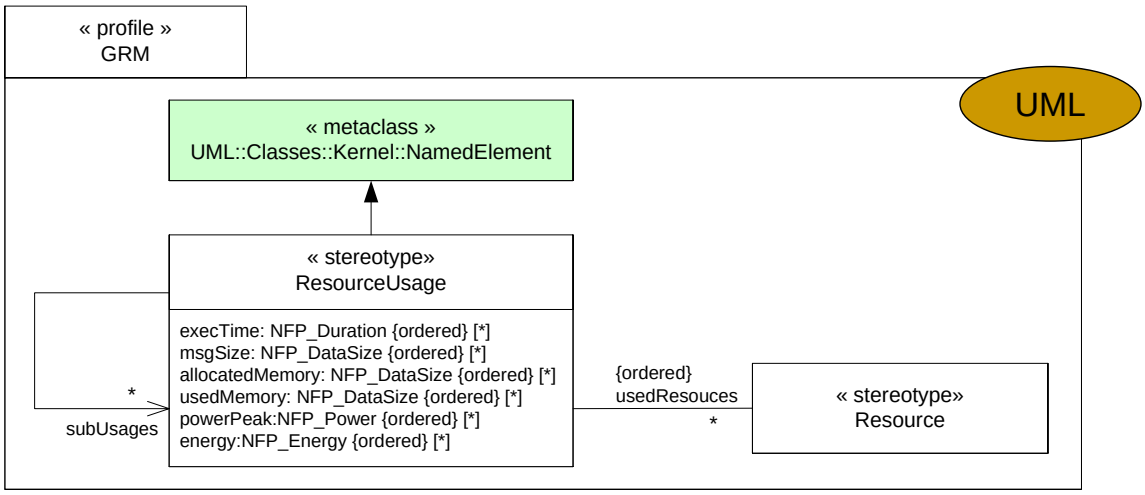
Extensions for scheduling

UML



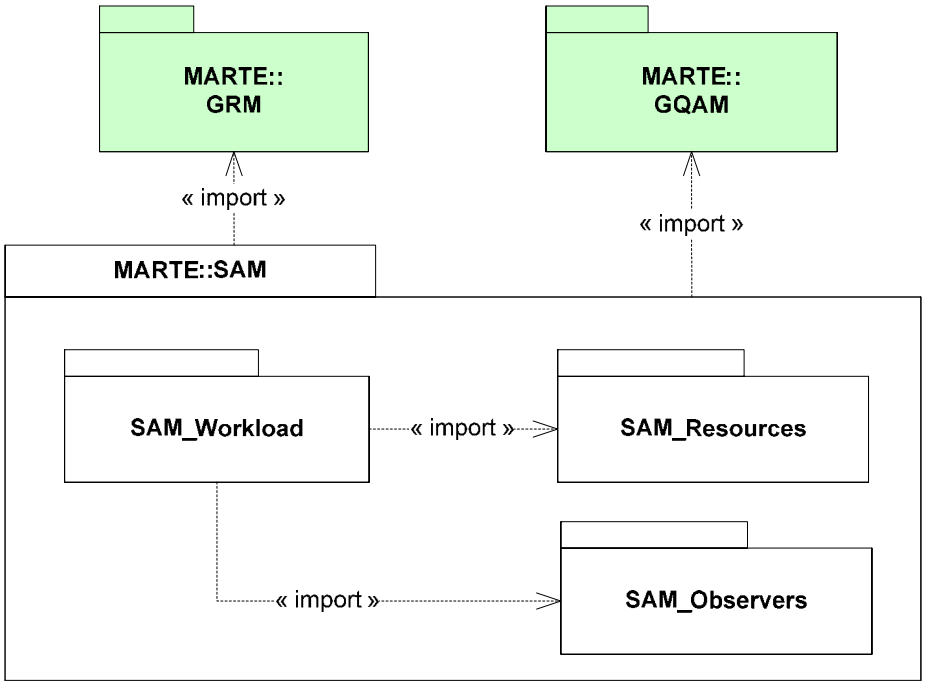
Resource Usage



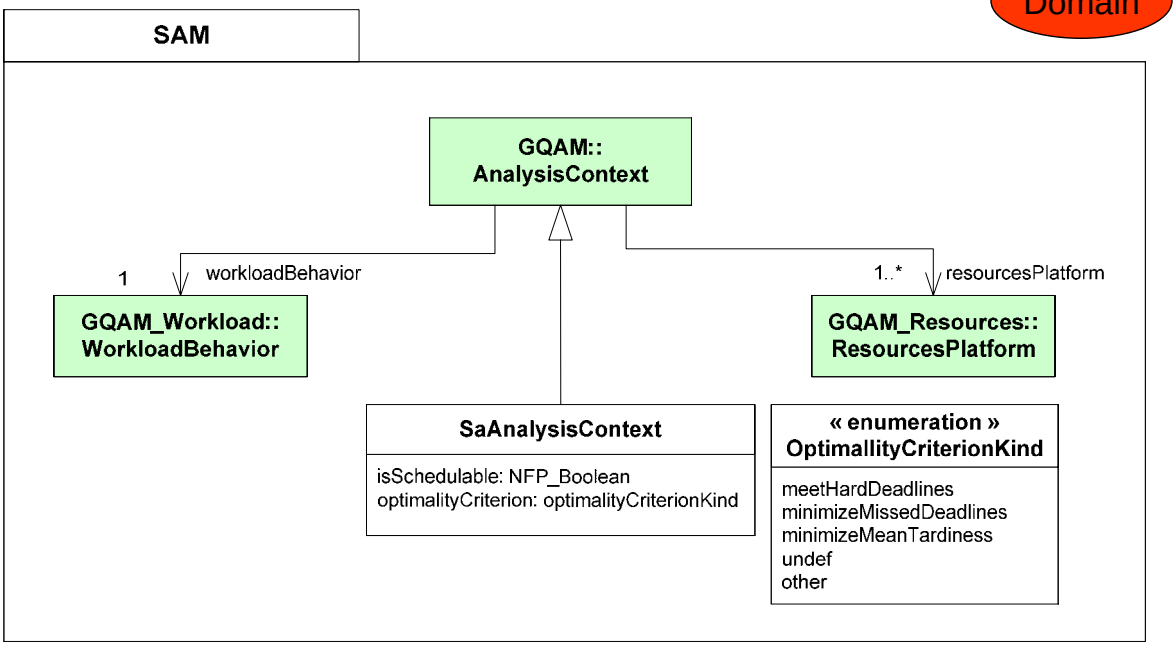


1. If the list usedResources is empty the list subUsages should not be empty and viceversa.
2. If the list usedResources has only one element, all the optional lists of attributes refer to this unique Resource and at least one of them must be present.
3. If the list usedResources has more than one element, 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.

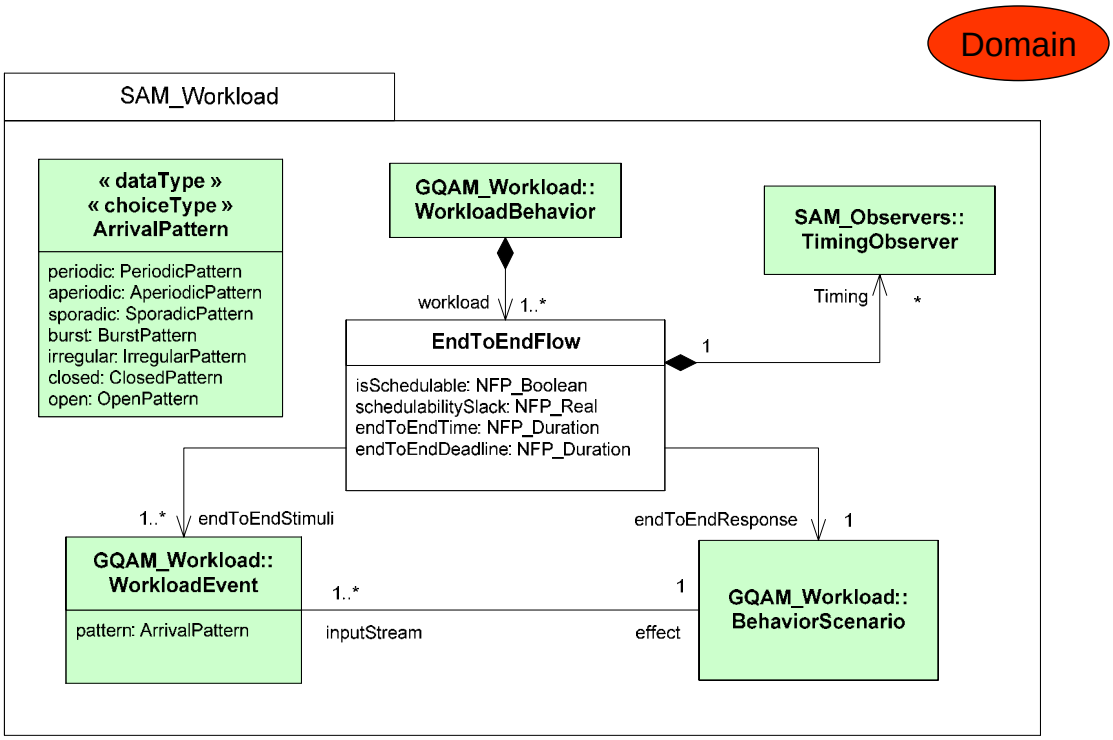
Structure of SAM



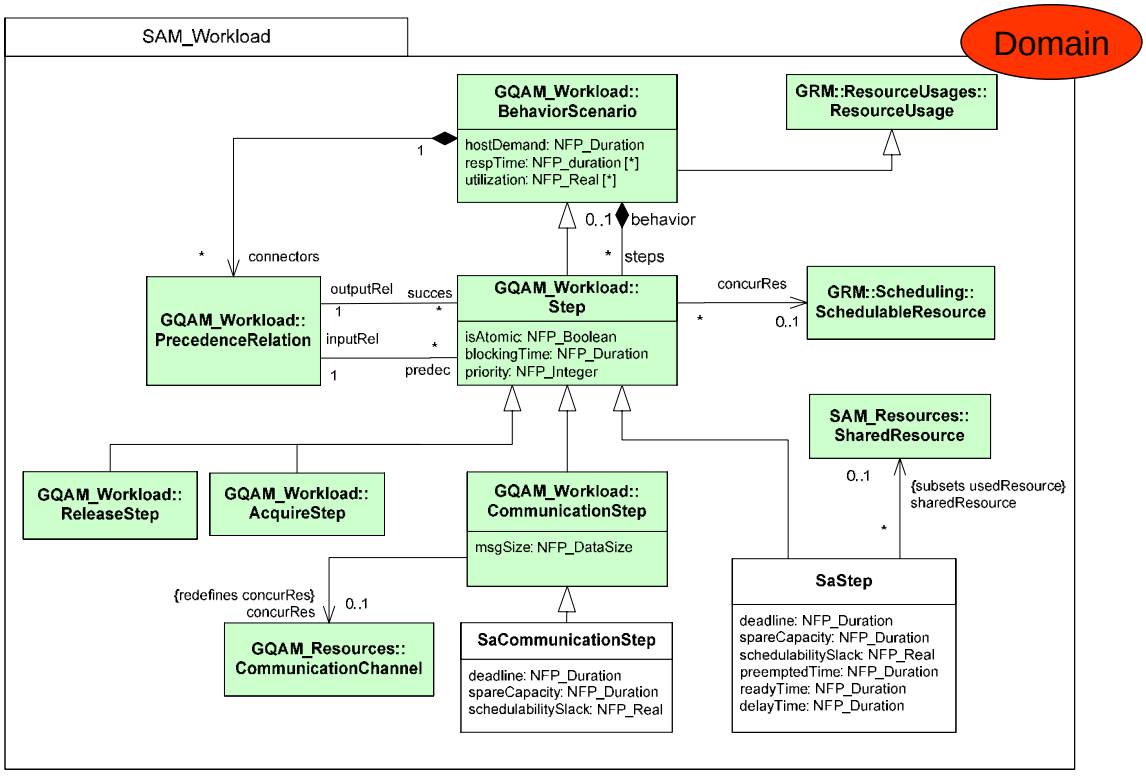
Domain



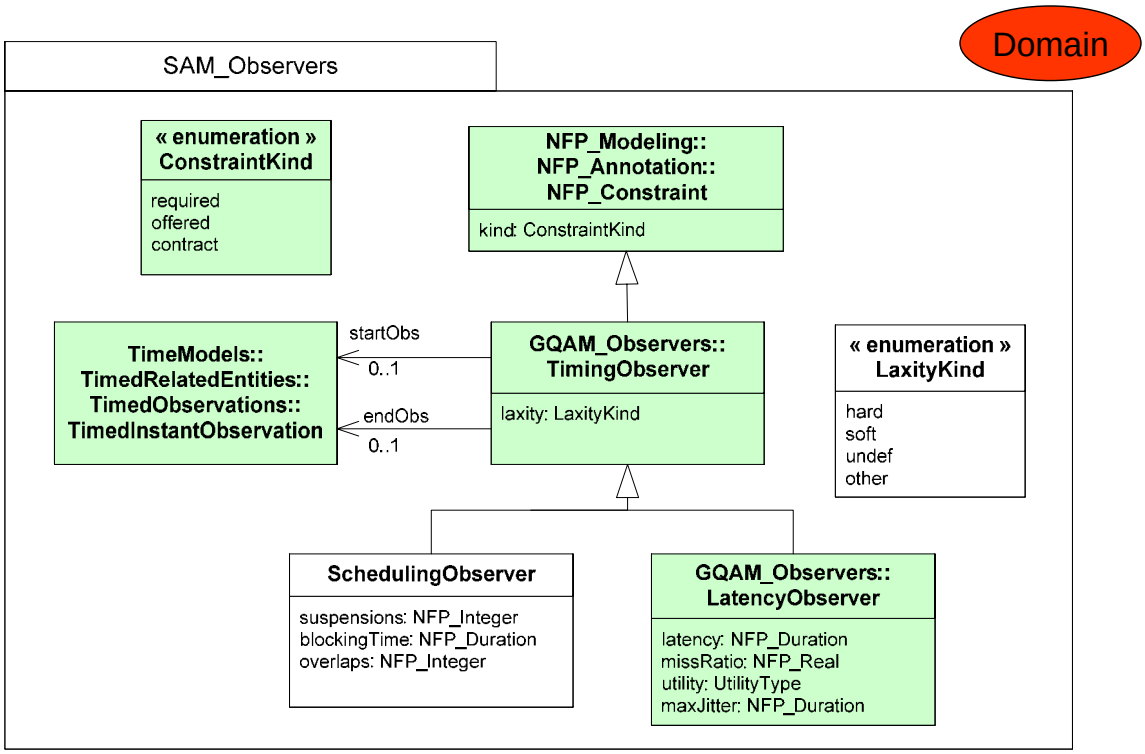
SAM_Workload: End-to-endFlow



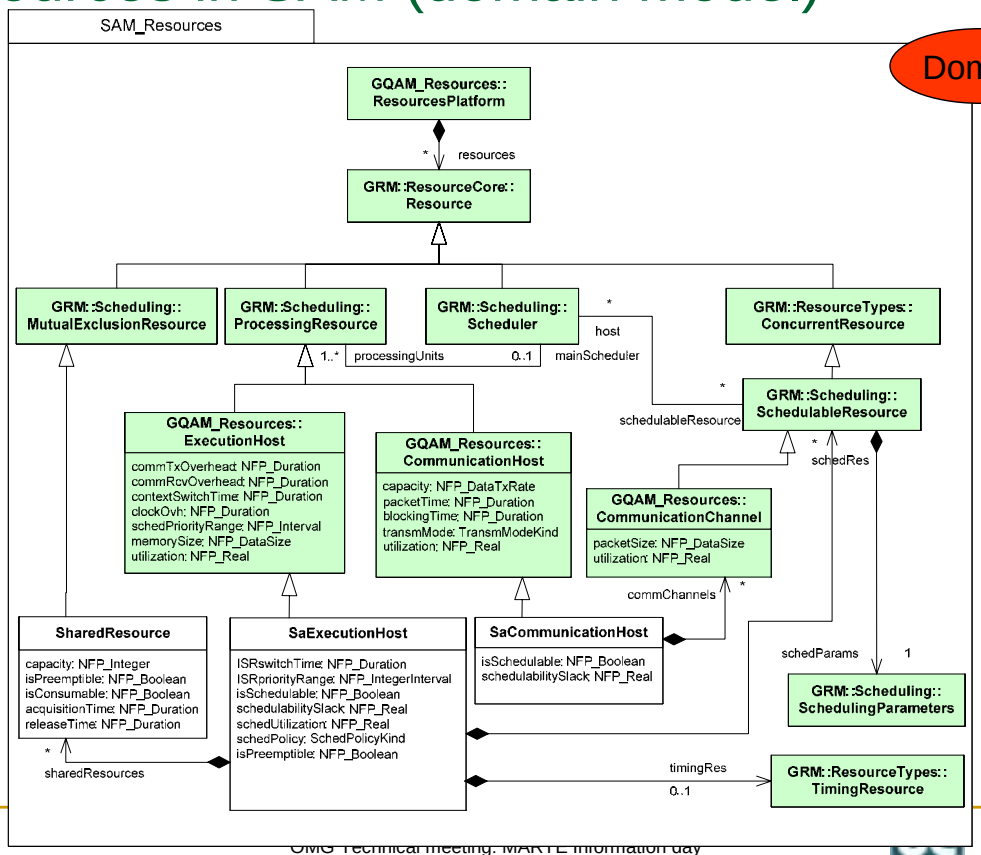
SAM_Workload: BehaviorScenario



SAM_Observers

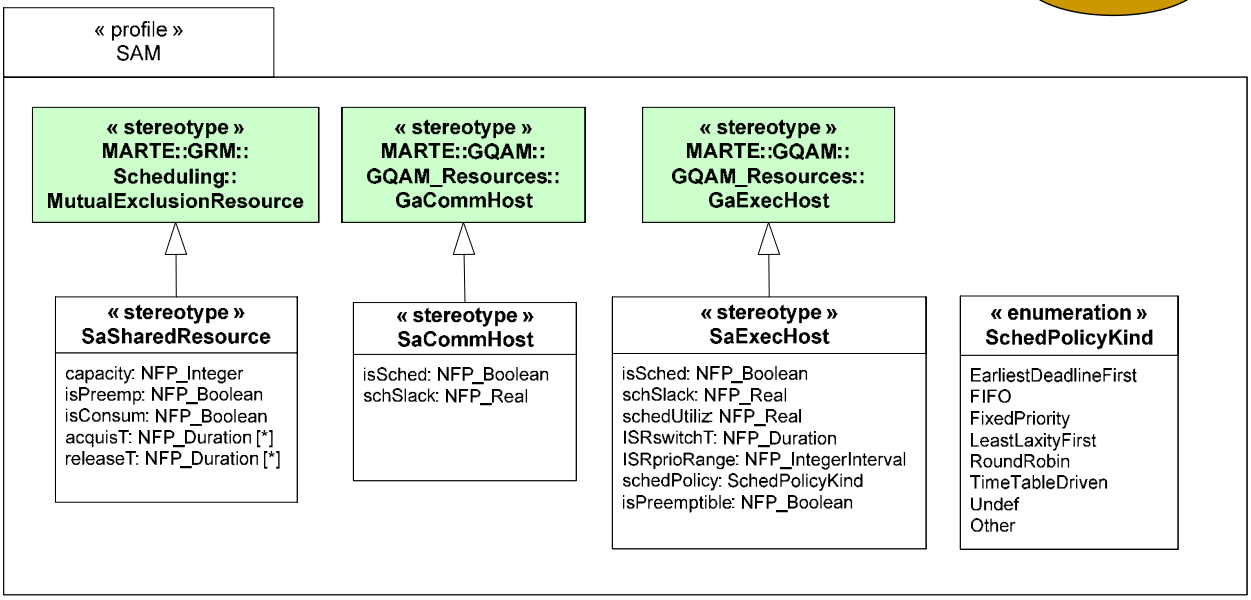


Resources in SAM (domain model)



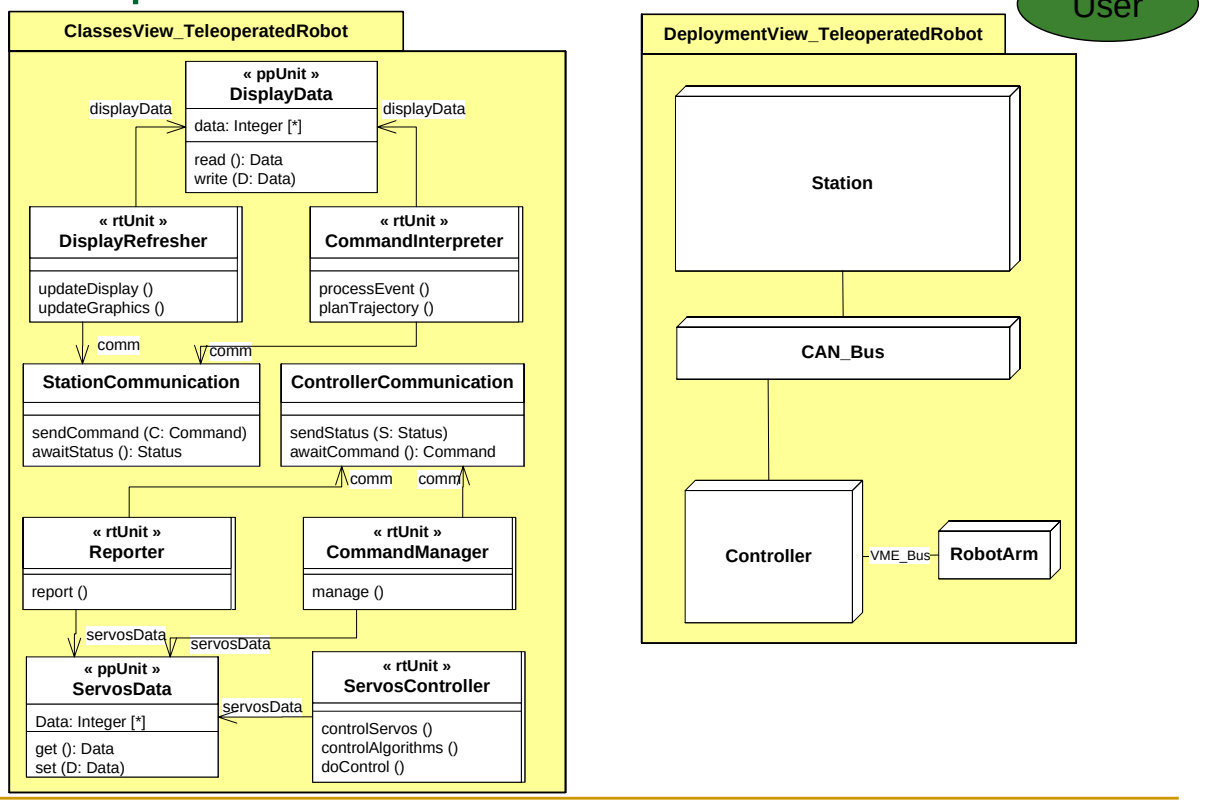
Resources in SAM (profile)

UML



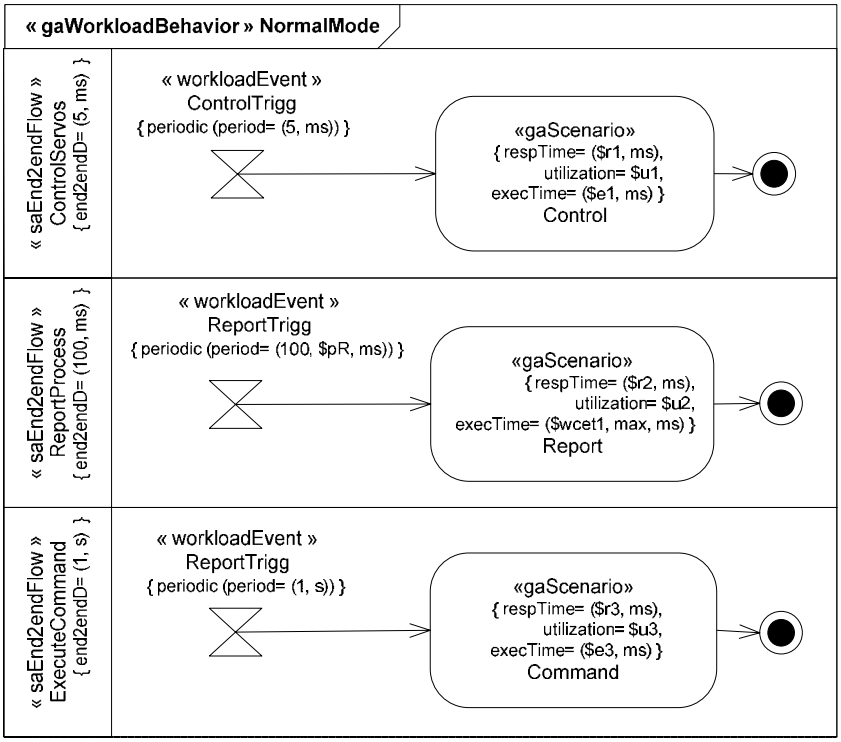
Example

User



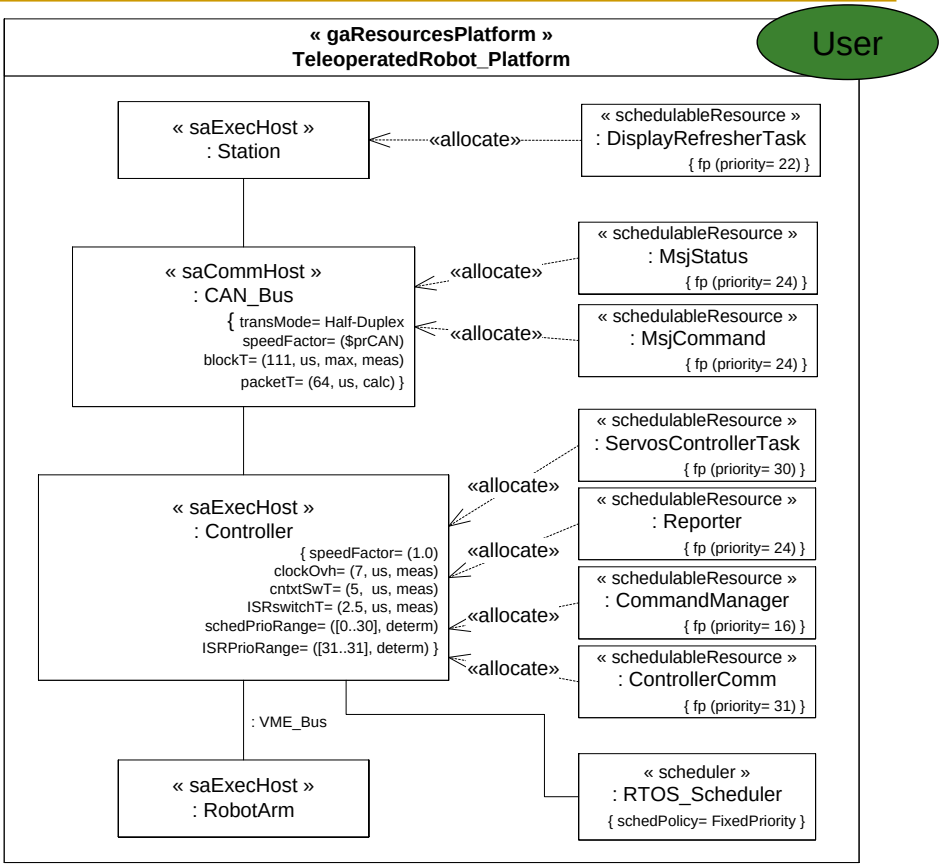
Real-time situation

User



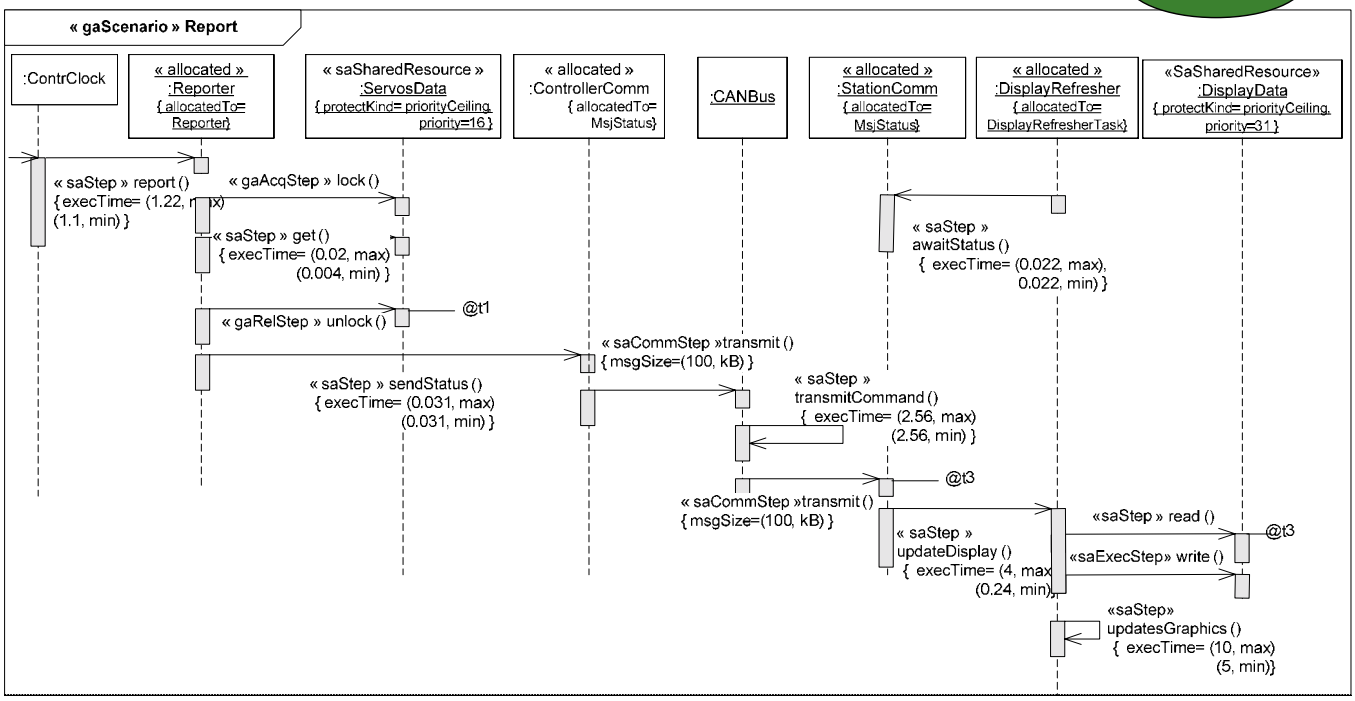
Platform

User



A behavior scenario

User



Parametric analysis contexts

User