Schedulability Analysis with the UML Profile for Modeling and Analysis of Real-Time and Embedded Systems

Julio Medina
Universidad de Cantabria

Topics

- A flash of schedulability analysis:
  - RMA: foundations
  - Offset-based techniques: high level models.
- Structural and behavioral modeling elements in MARTE.
- Examples of the notation used.

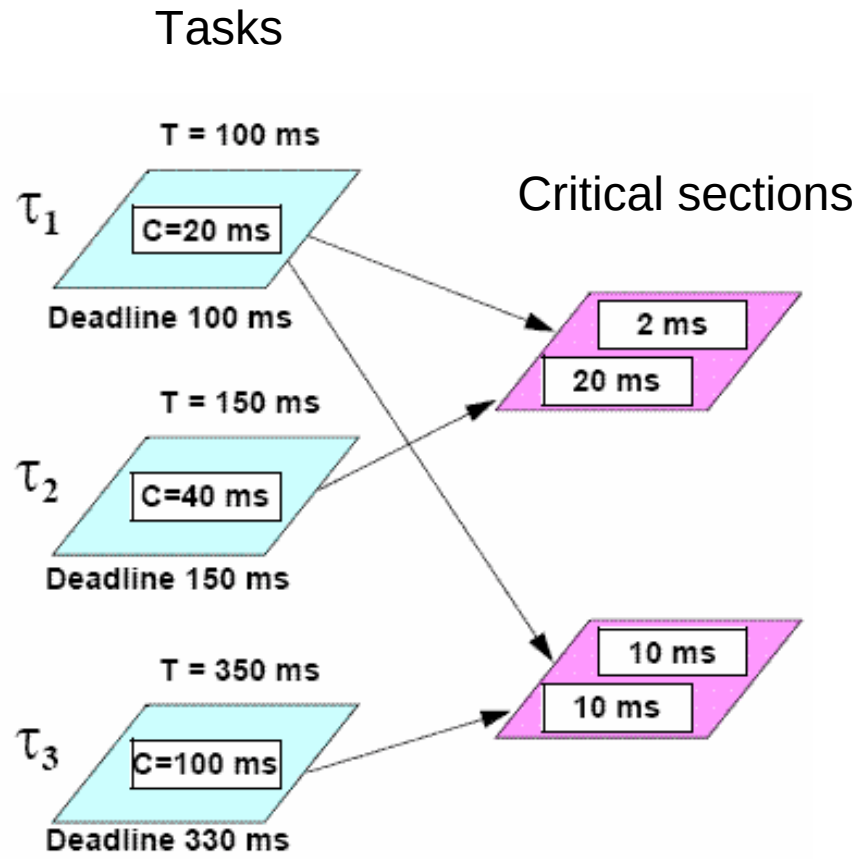
Options for managing time

- 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



Why Are Deadlines Missed?

- For a task to meet its deadline, it must accommodate
  - preemption: time waiting for higher-priority tasks
  - execution: time to do its own work
  - blocking: time delayed by lower-priority tasks
- The task is schedulable if the sum of its preemption, execution, and blocking is less than its deadline for the worst possible case:
  - Execution is unavoidable (unless requirements change).
  - Preemption can be minimized by choosing an optimum priority assignment
  - Main focus: identify and reduce blocking

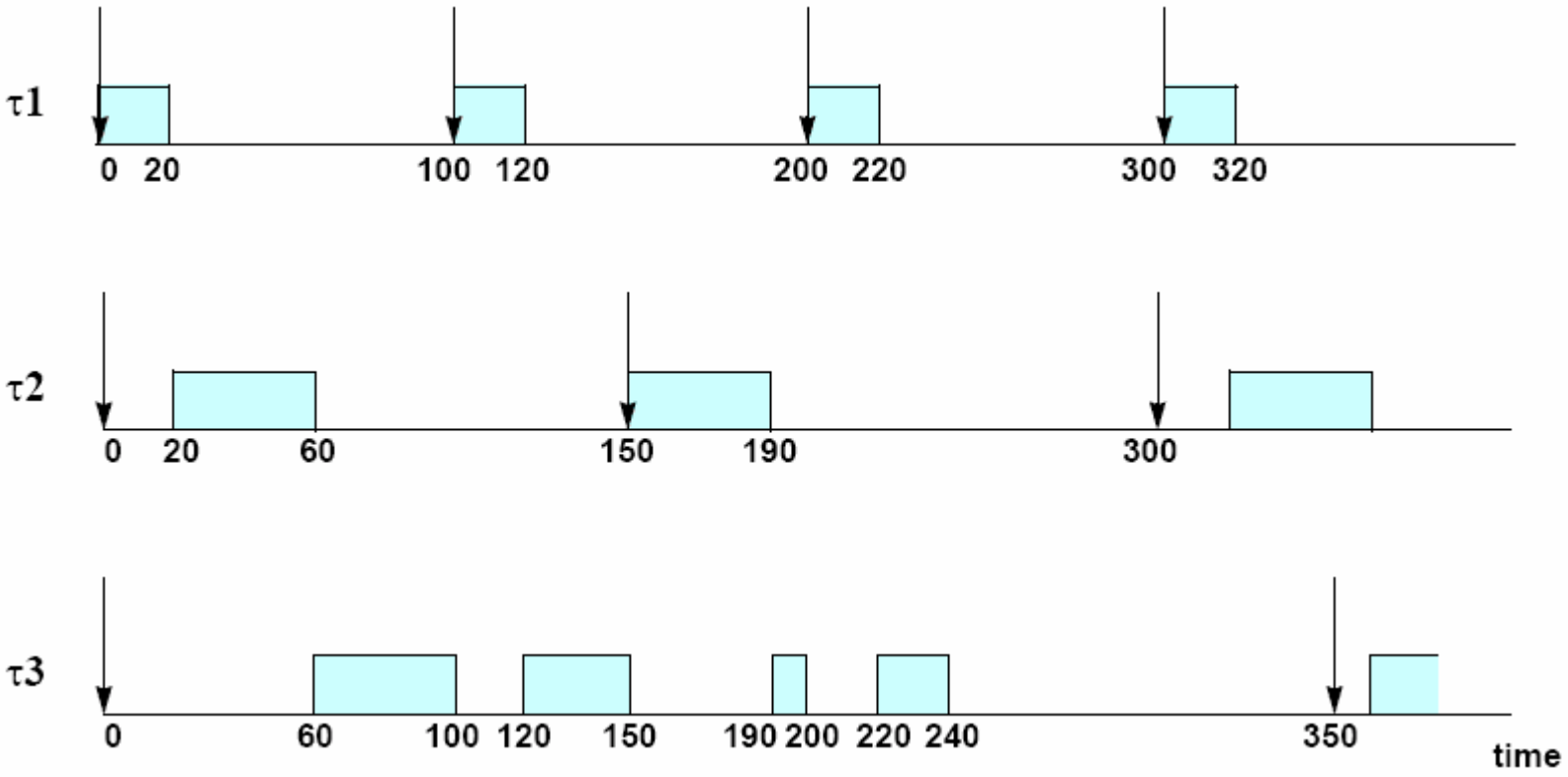
Concepts and Definitions (periodic tasks)

- Periodic task
  - initiated at fixed intervals
  - must finish before start of next cycle
- Task's CPU utilization: $U_i = C_i / T_i$
  - C_i = compute time (execution time) for task ti
  - T_i = period of task t_i
  - P_i = priority of task t_i
  - D_i = deadline of task t_i
  - f_i = phase of task t_i
  - R_i = response time of task t_i
- CPU utilization for a set of tasks: $U = U_1 + U_2 + \ldots + U_n$

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:

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

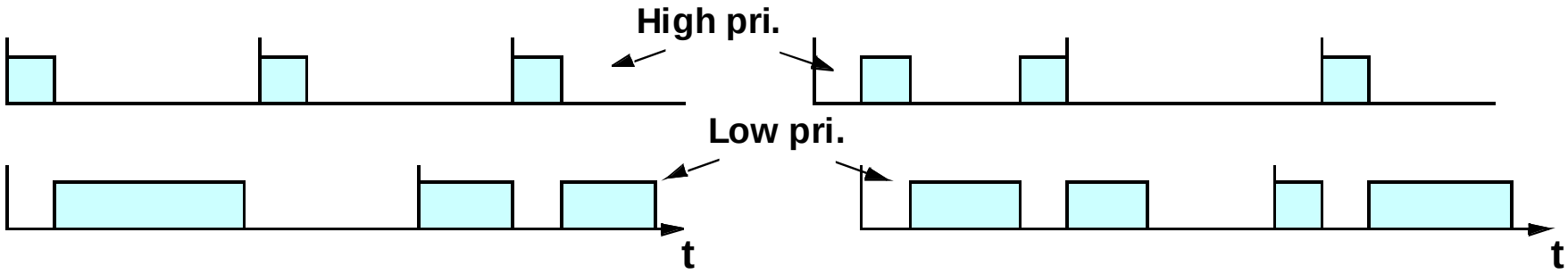
- Task t_i is schedulable if:

$$R_i \leq D_i$$

Effects of jitter

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

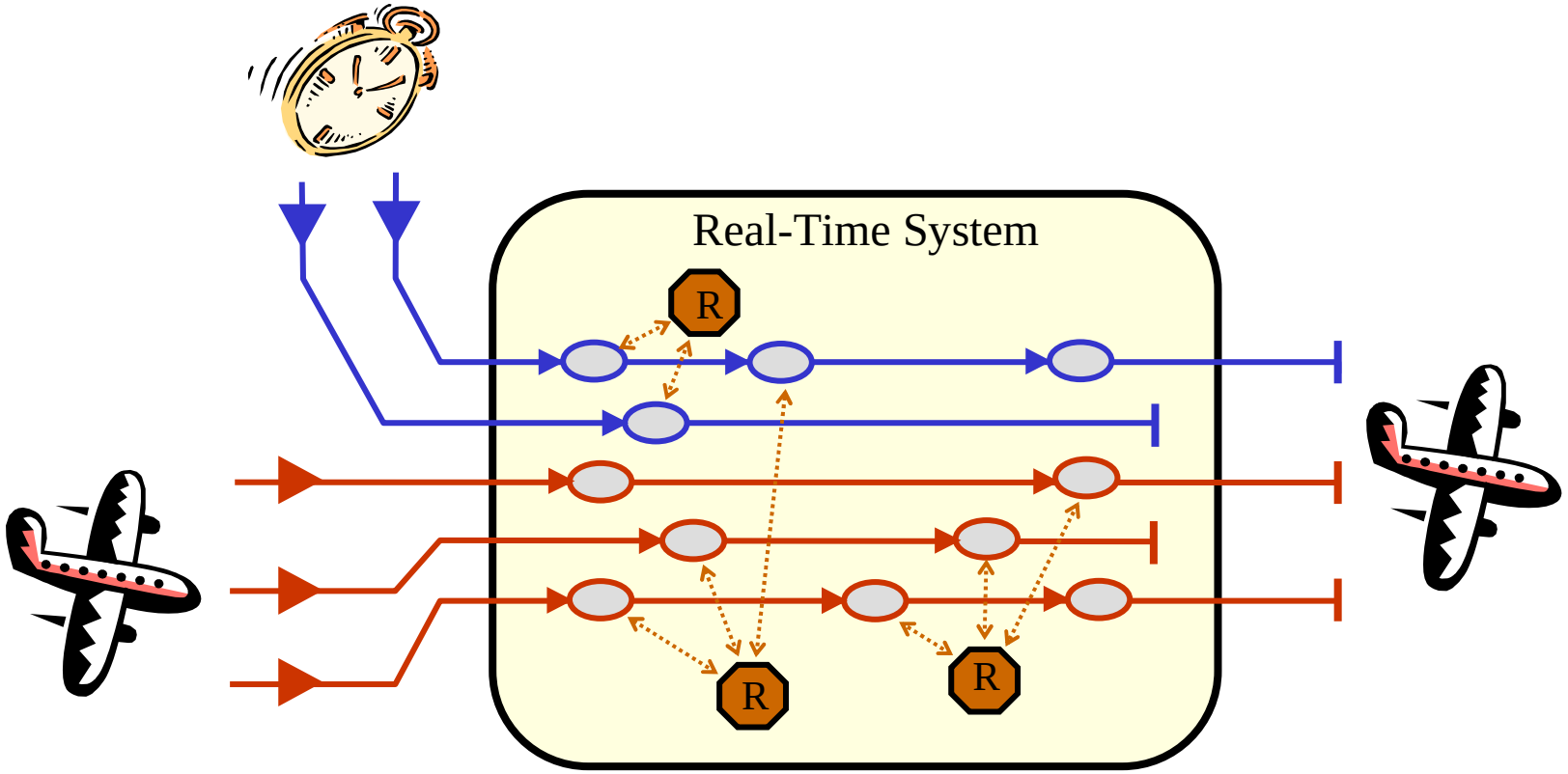
- Jitter may have a delay effect on lower priority tasks



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

Execution sequence with jitter. Worst case is two preemptions

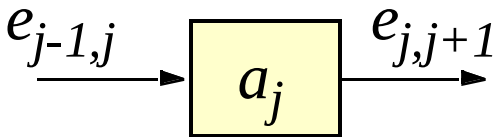
More general approach



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

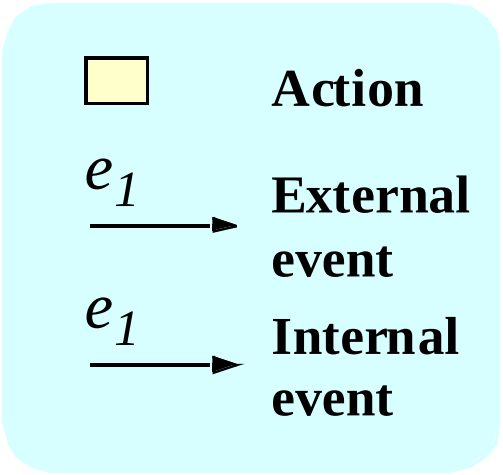
Distributed system model

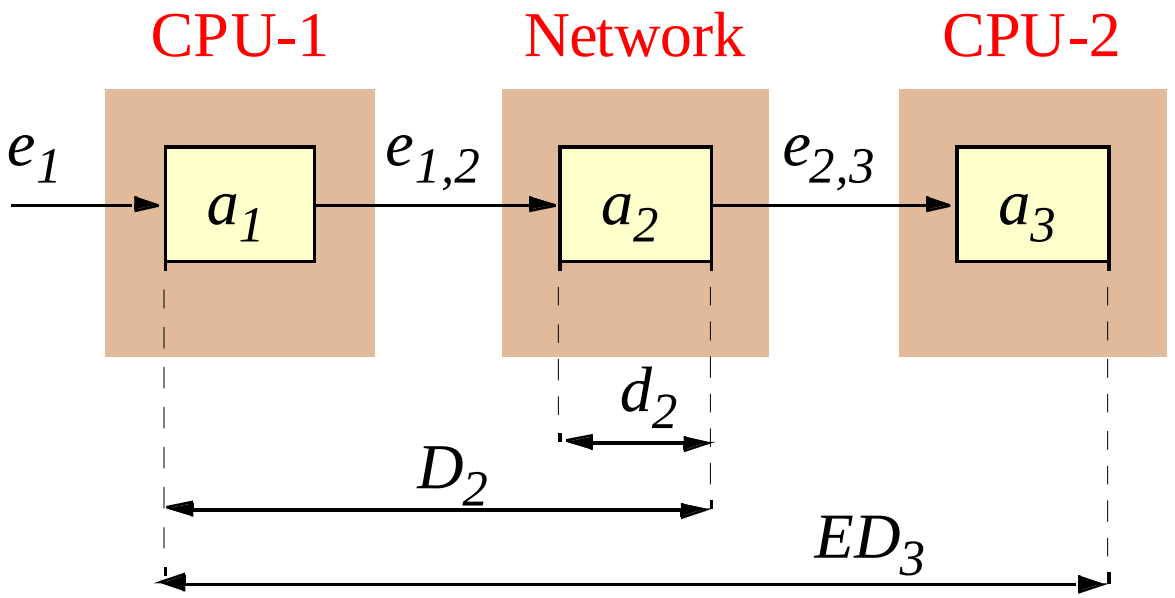
Linear Action:



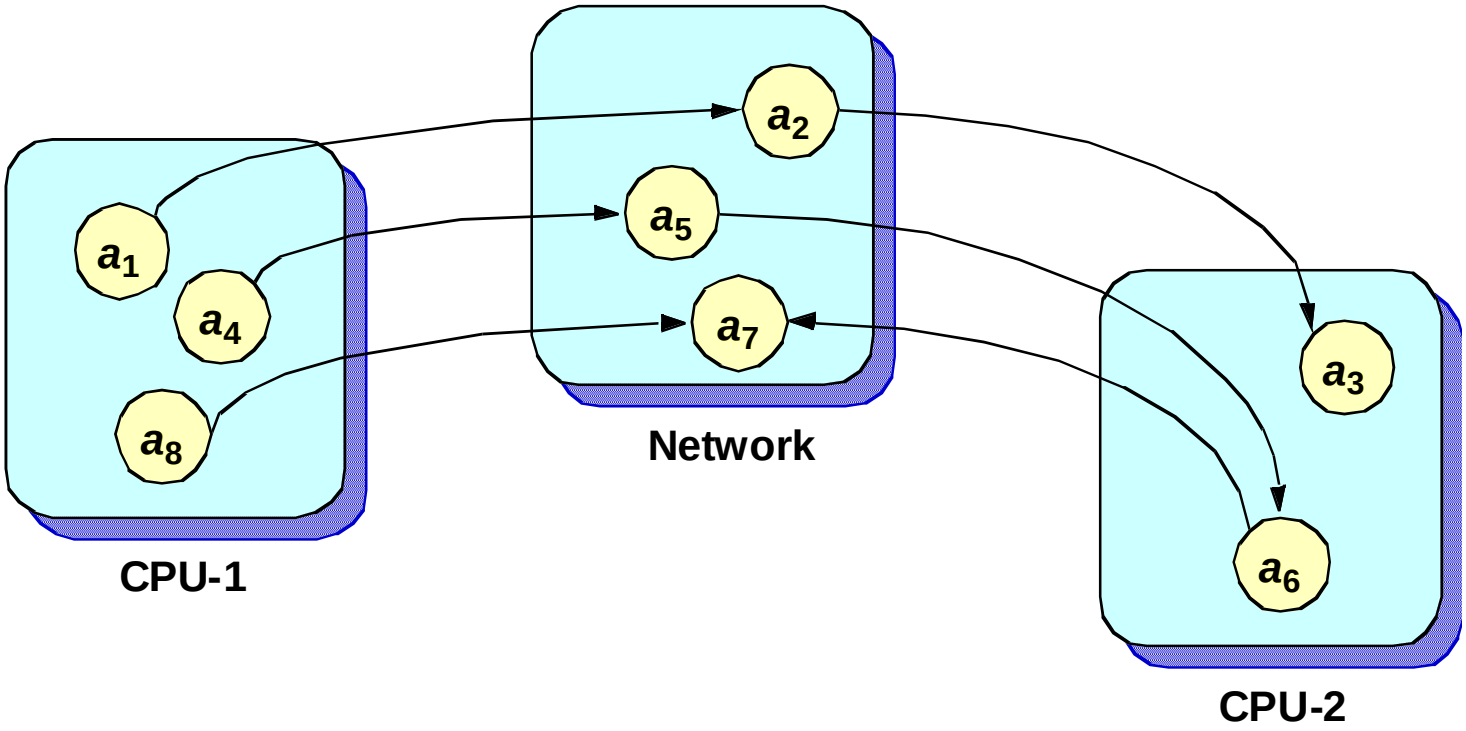
$$T_{j-1,j} = T_j = T_{j,j+1}$$

Linear Response to an Event:



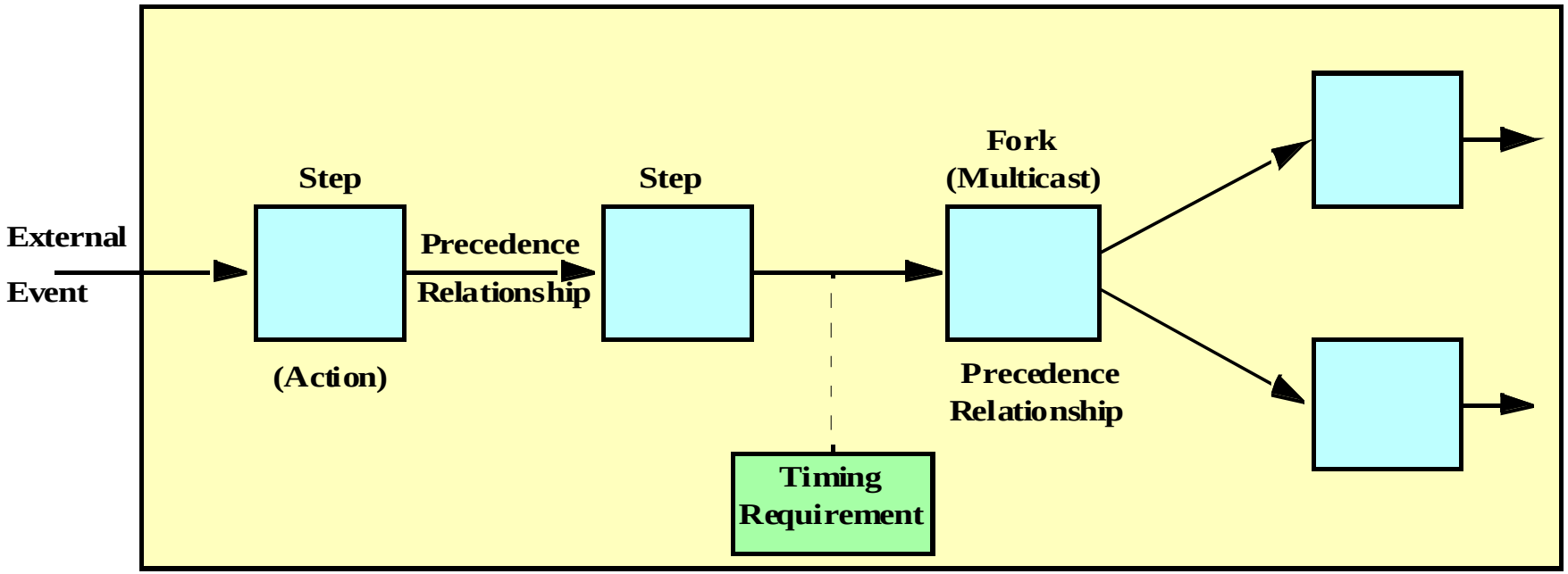


Jitter in distributed systems

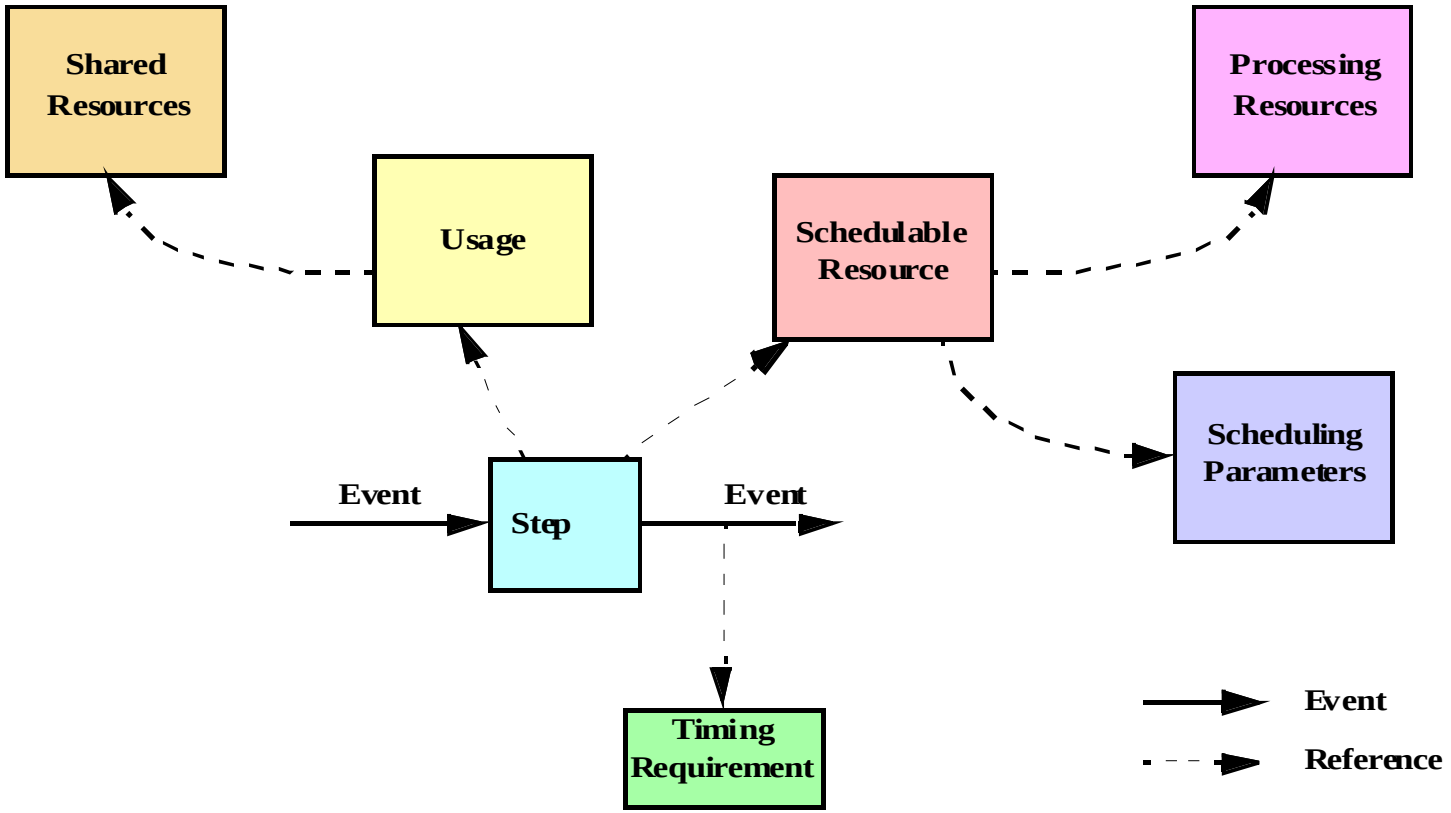


- Jitter in one processing resource depends on the response times in other processing resources
- Response times depend on jitters

Transactions

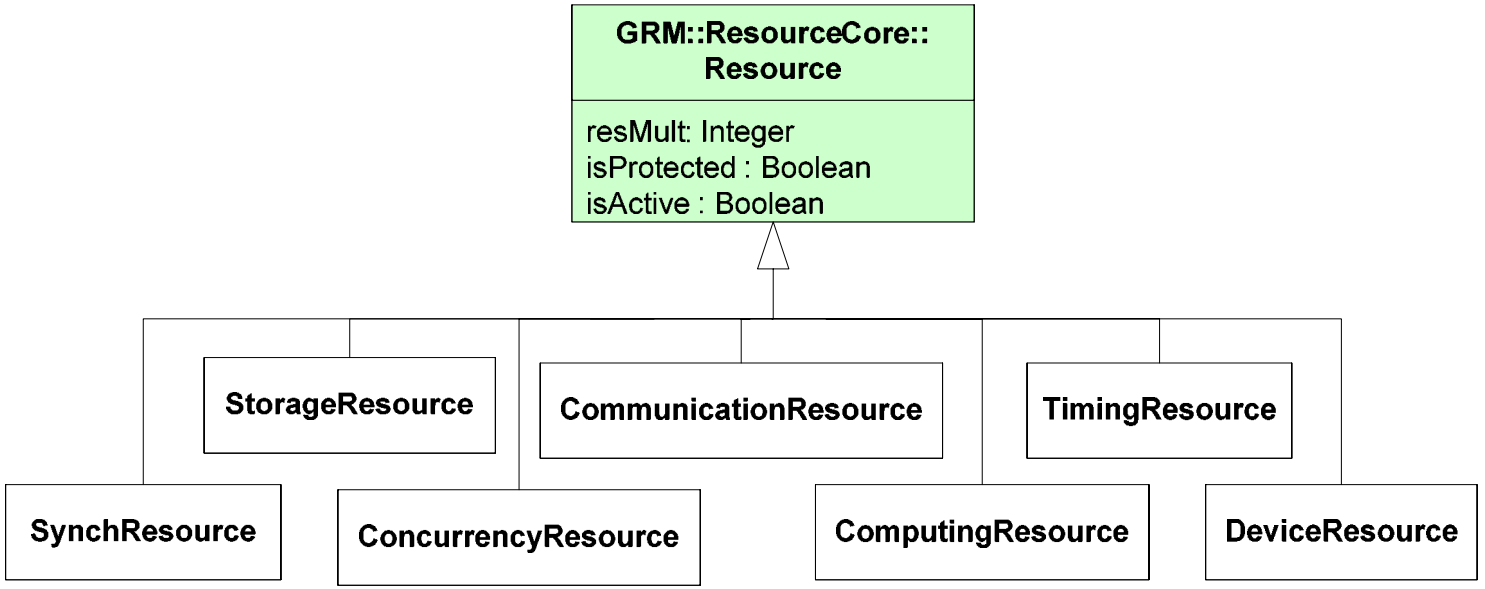


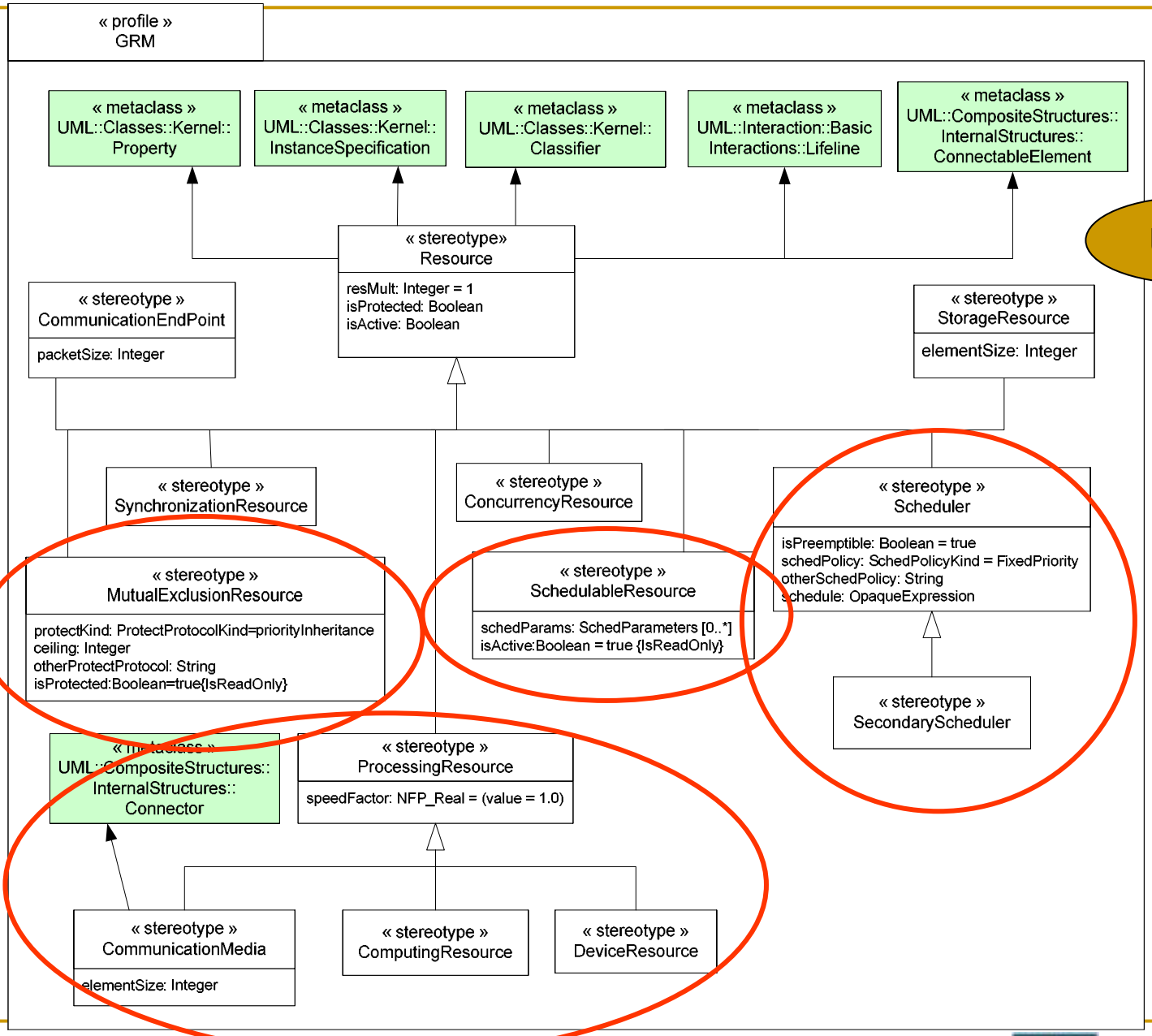
Transactions



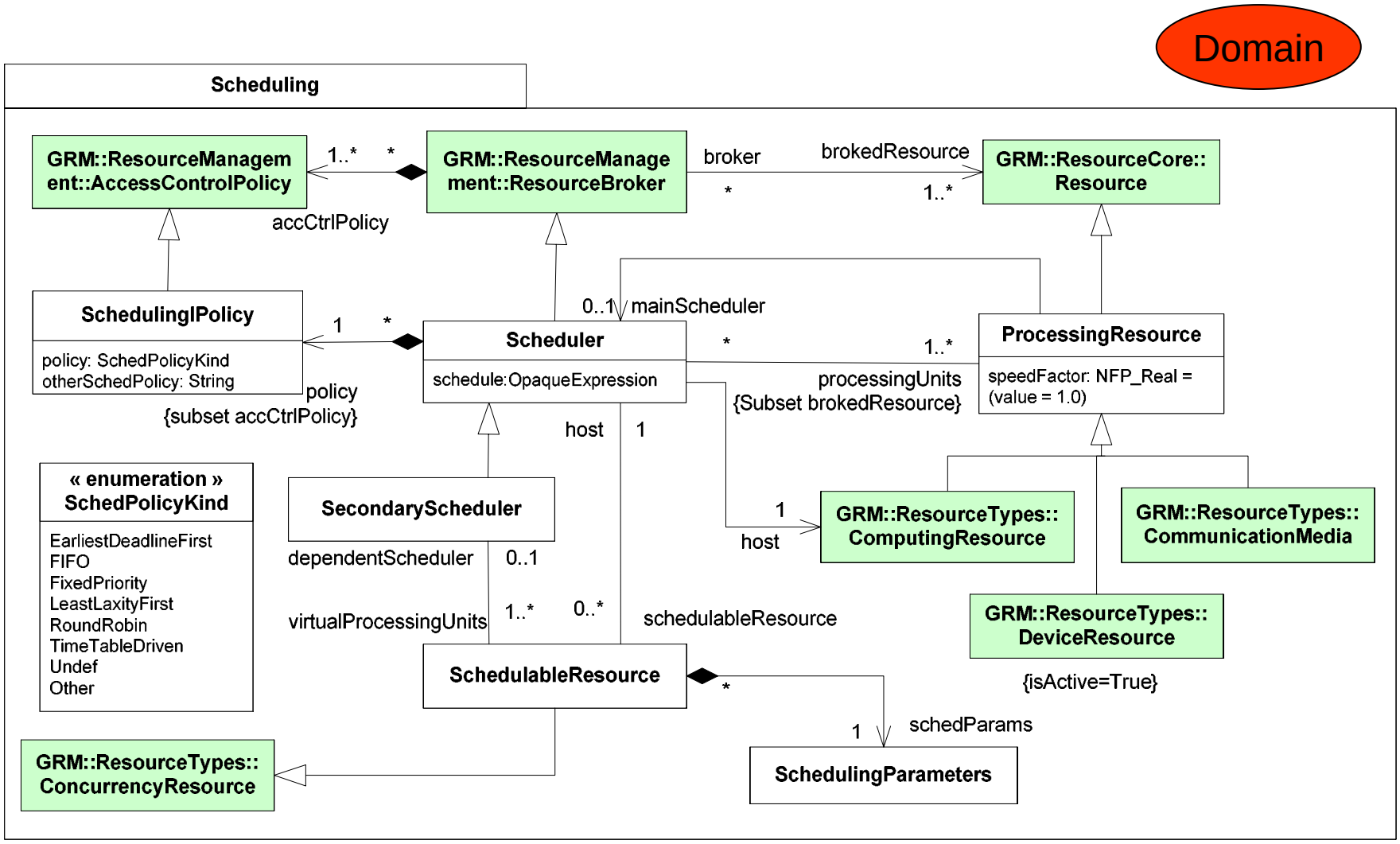
Resources

Domain



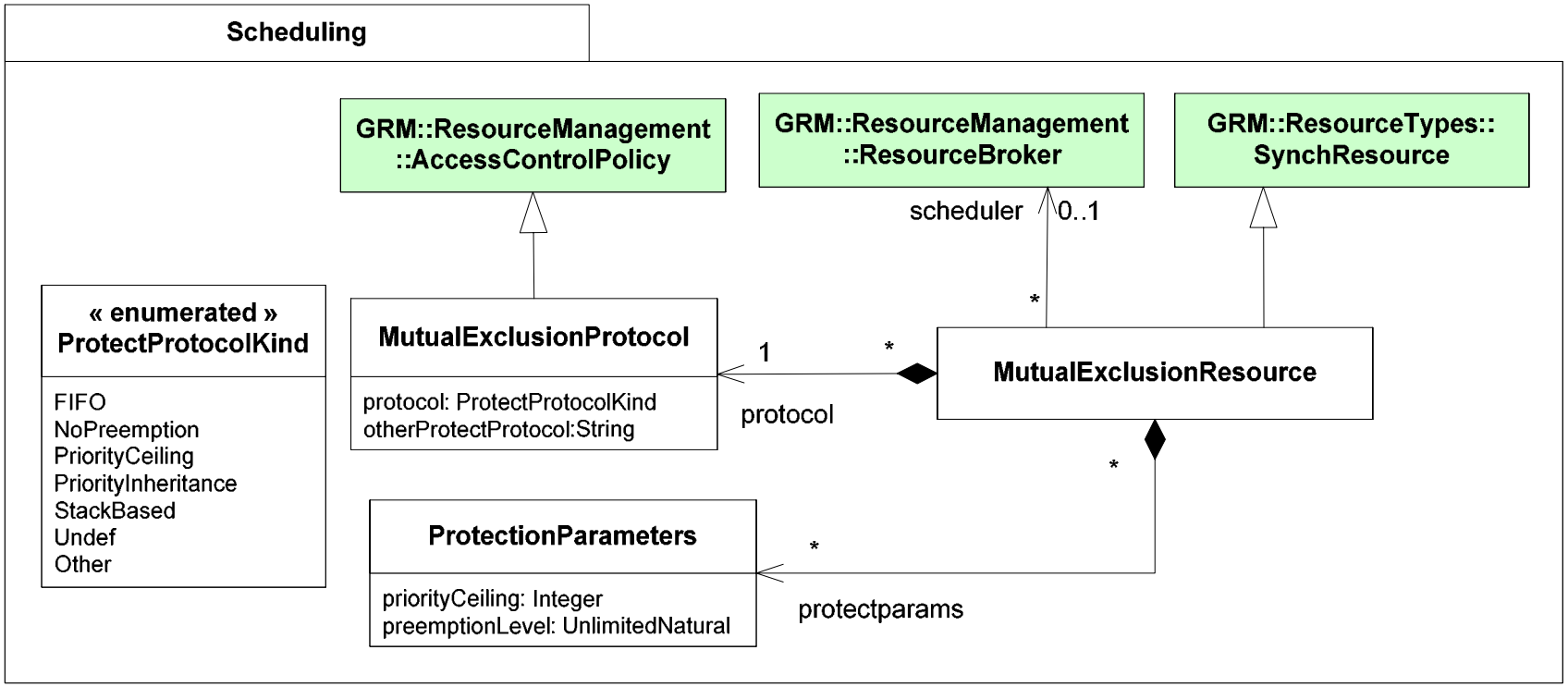

« profile »
GRM
« metaclass »
UML::Classes::Kernel::
Property
« metaclass »
UML::Classes::Kernel::
InstanceSpecification
« metaclass »
UML::Classes::Kernel::
Classifier
« metaclass »
UML::Interaction::Basic
Interactions::Lifeline
« metaclass »
UML::CompositeStructures::
InternalStructures::
ConnectableElement
UML
« stereotype»
Resource
resMult: Integer = 1
isProtected: Boolean
isActive: Boolean
« stereotype »
CommunicationEndPoint
packetSize: Integer
« stereotype »
StorageResource
elementSize: Integer
« stereotype »
SynchronizationResource
« stereotype »
ConcurrencyResource
« stereotype »
Scheduler
isPreemptible: Boolean = true
schedPolicy: SchedPolicyKind = FixedPriority
otherSchedPolicy: String
schedule: OpaqueExpression
« stereotype »
MutualExclusionResource
protectKind: ProtectProtocolKind=priorityInheritance
ceiling: Integer
otherProtectProtocol: String
isProtected:Boolean=true{IsReadOnly}
« stereotype »
SchedulableResource
schedParams: SchedParameters [0..*]
isActive:Boolean = true {IsReadOnly}
« stereotype »
SecondaryScheduler
« metaclass »
UML::CompositeStructures::
InternalStructures::
Connector
« stereotype »
ProcessingResource
speedFactor: NFP_Real = (value = 1.0)
« stereotype »
CommunicationMedia
elementSize: Integer
« stereotype »
ComputingResource
« stereotype »
DeviceResource

Domain model for Schedulable resources

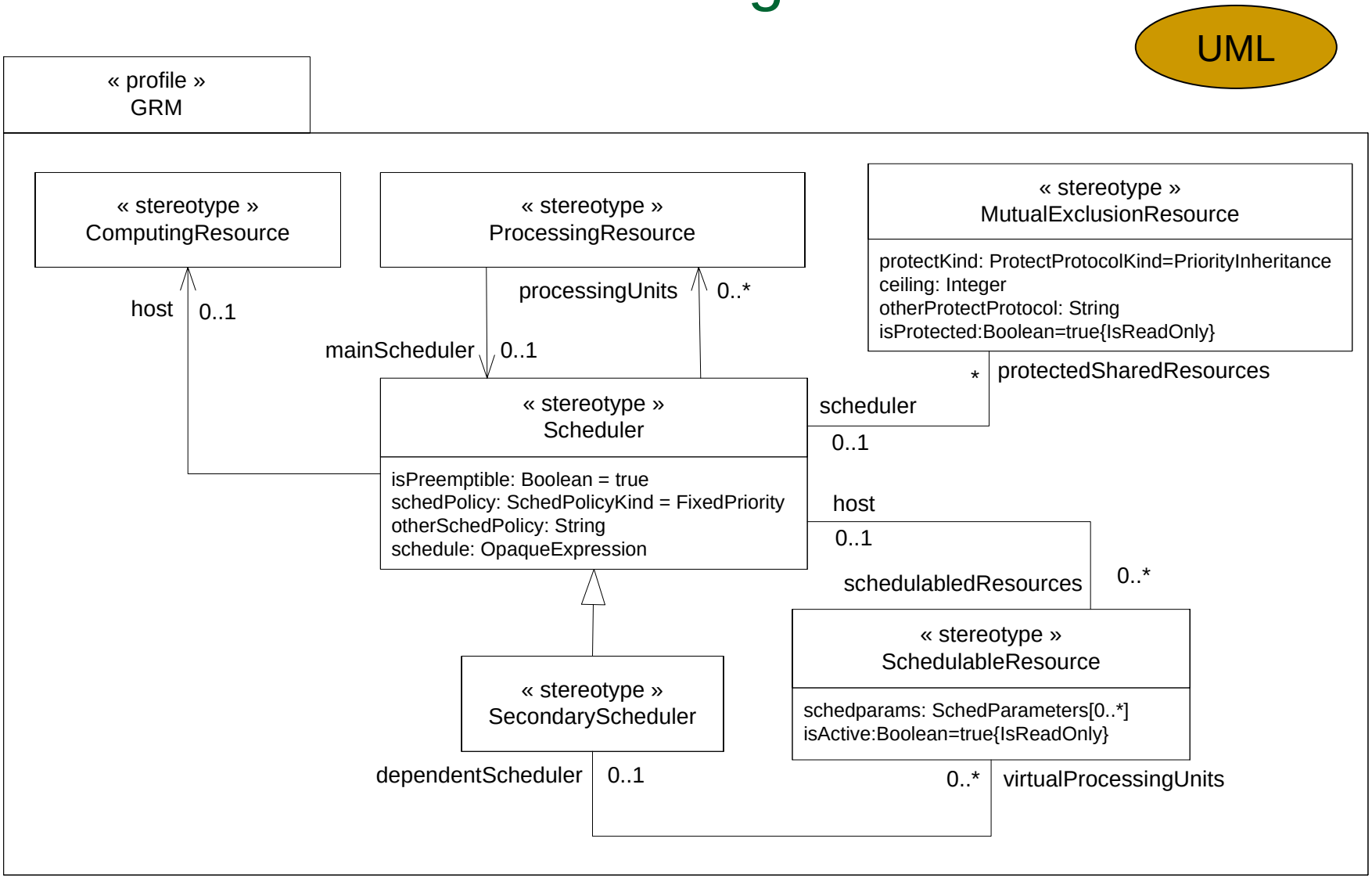


Shared resources

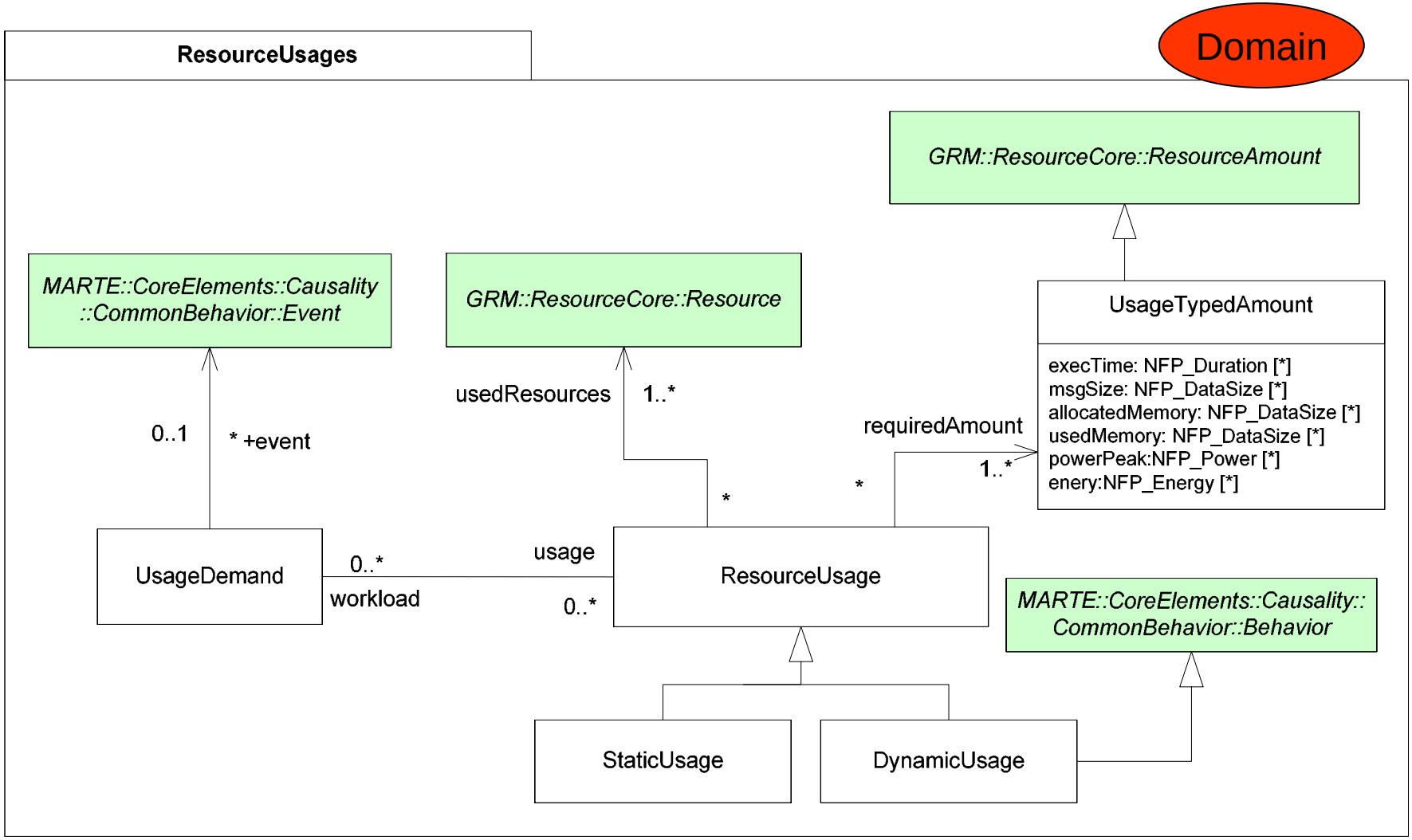
Domain

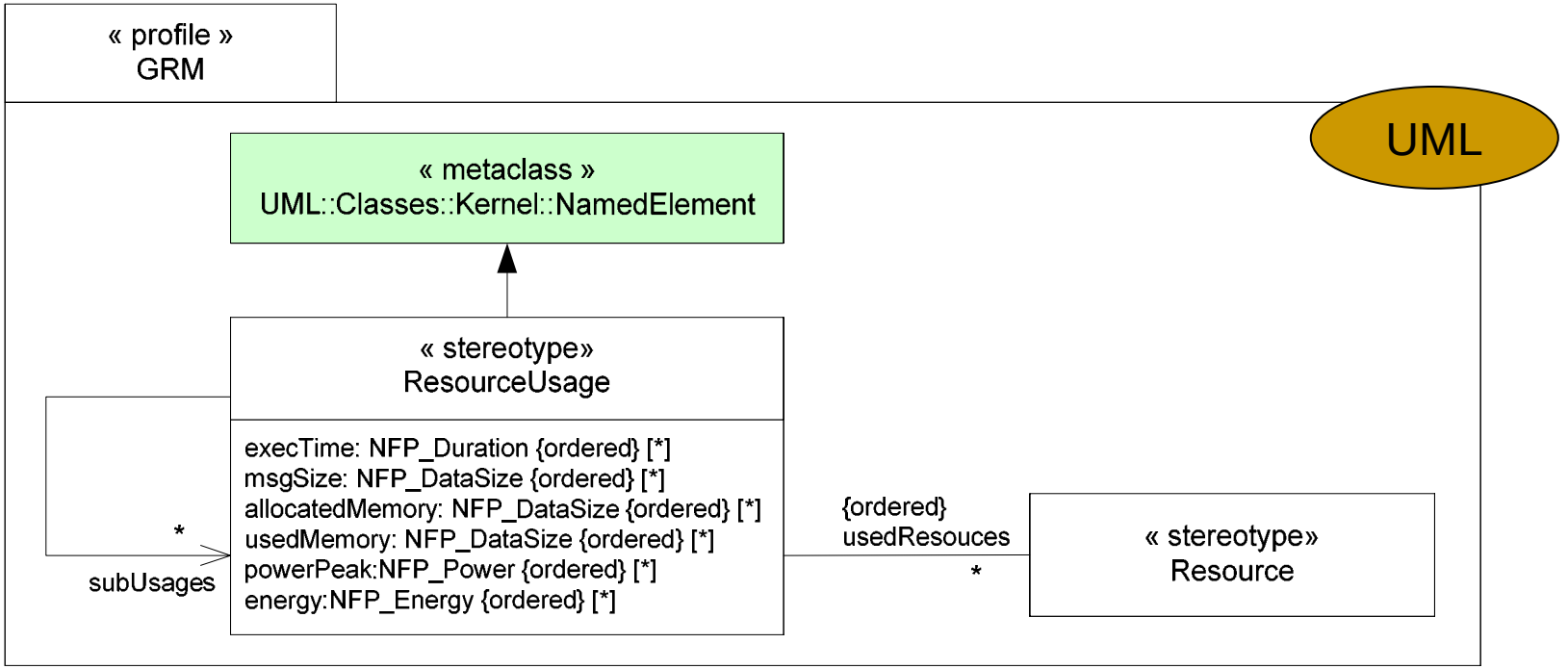


Extensions for scheduling



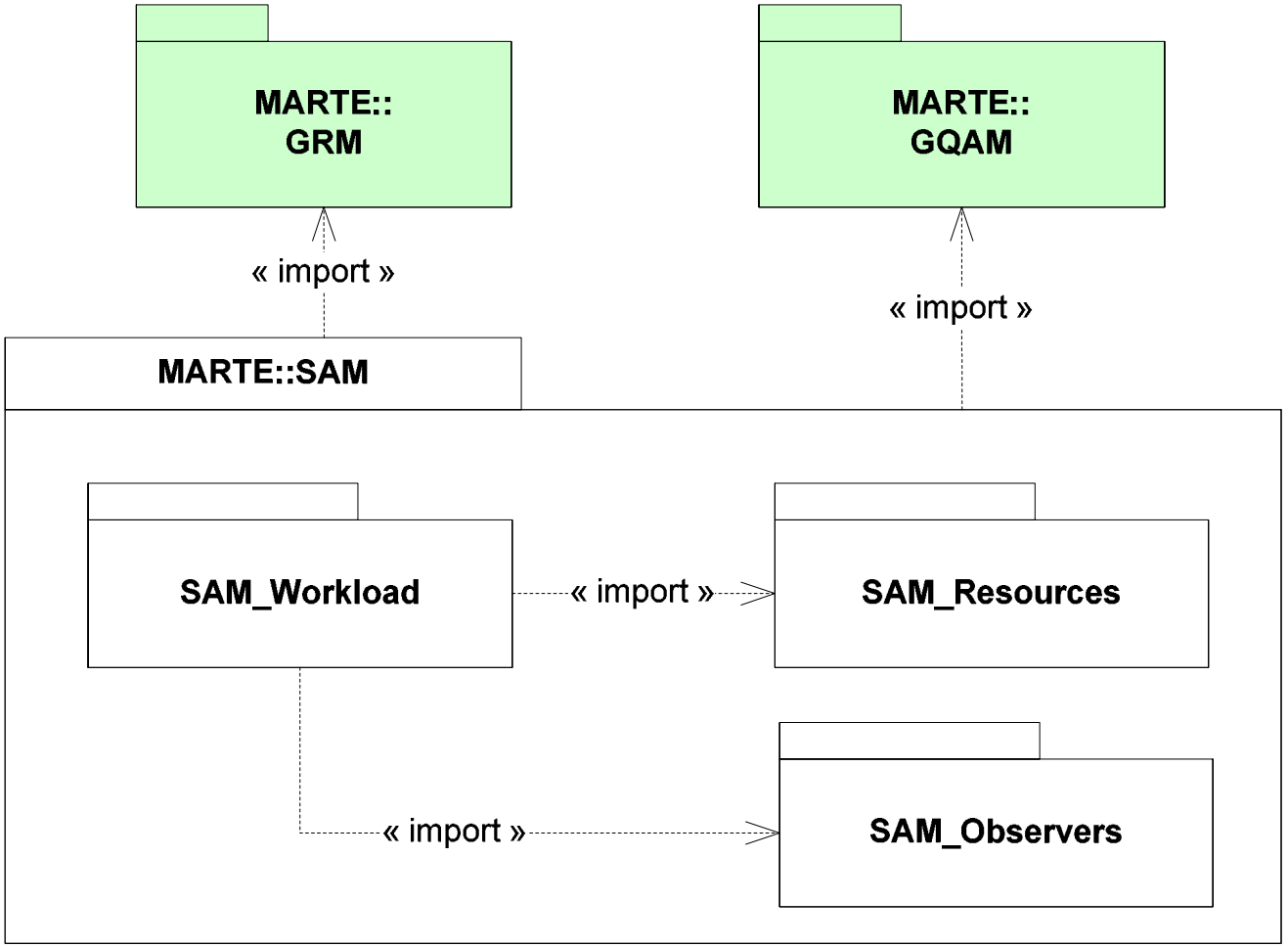
Resource Usage



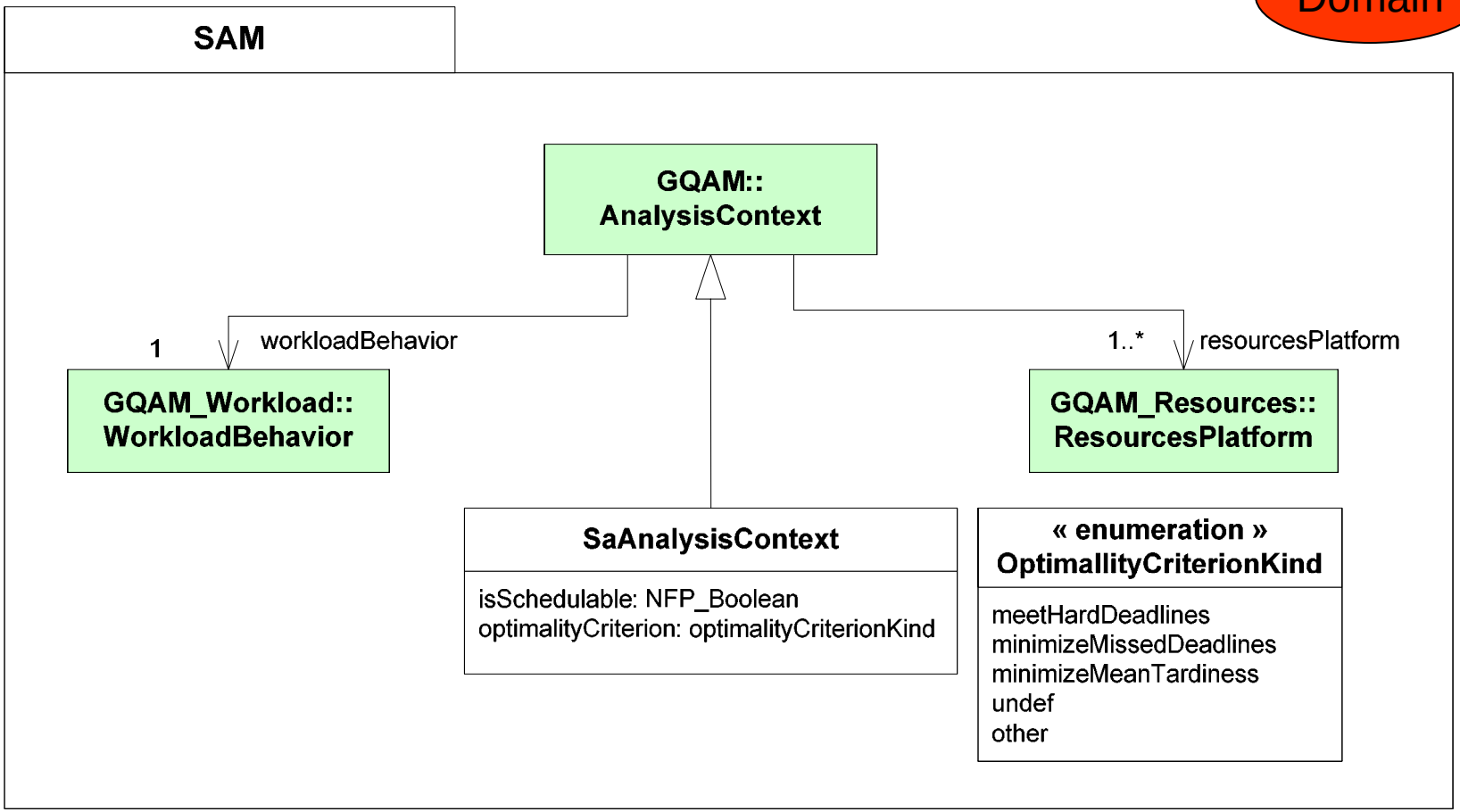


1. If the list usedResources is empty the list subUsages should not be empty and viceversa.
2. If the list usedResources has only one element, all the optional lists of attributes refer to this unique Resource and at least one of them must be present.
3. If the list usedResources has more than one element,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

Structure of SAM

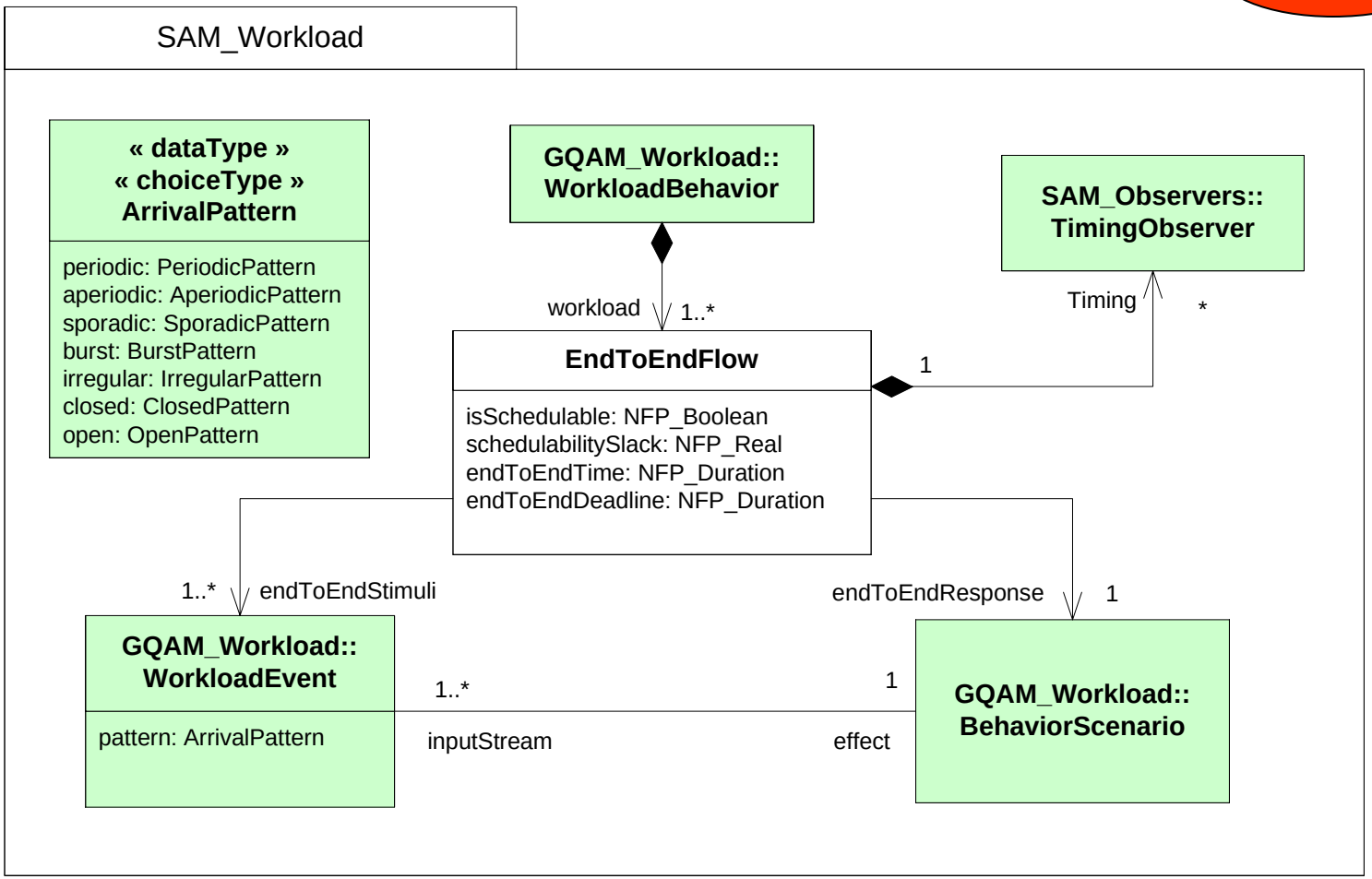


Domain

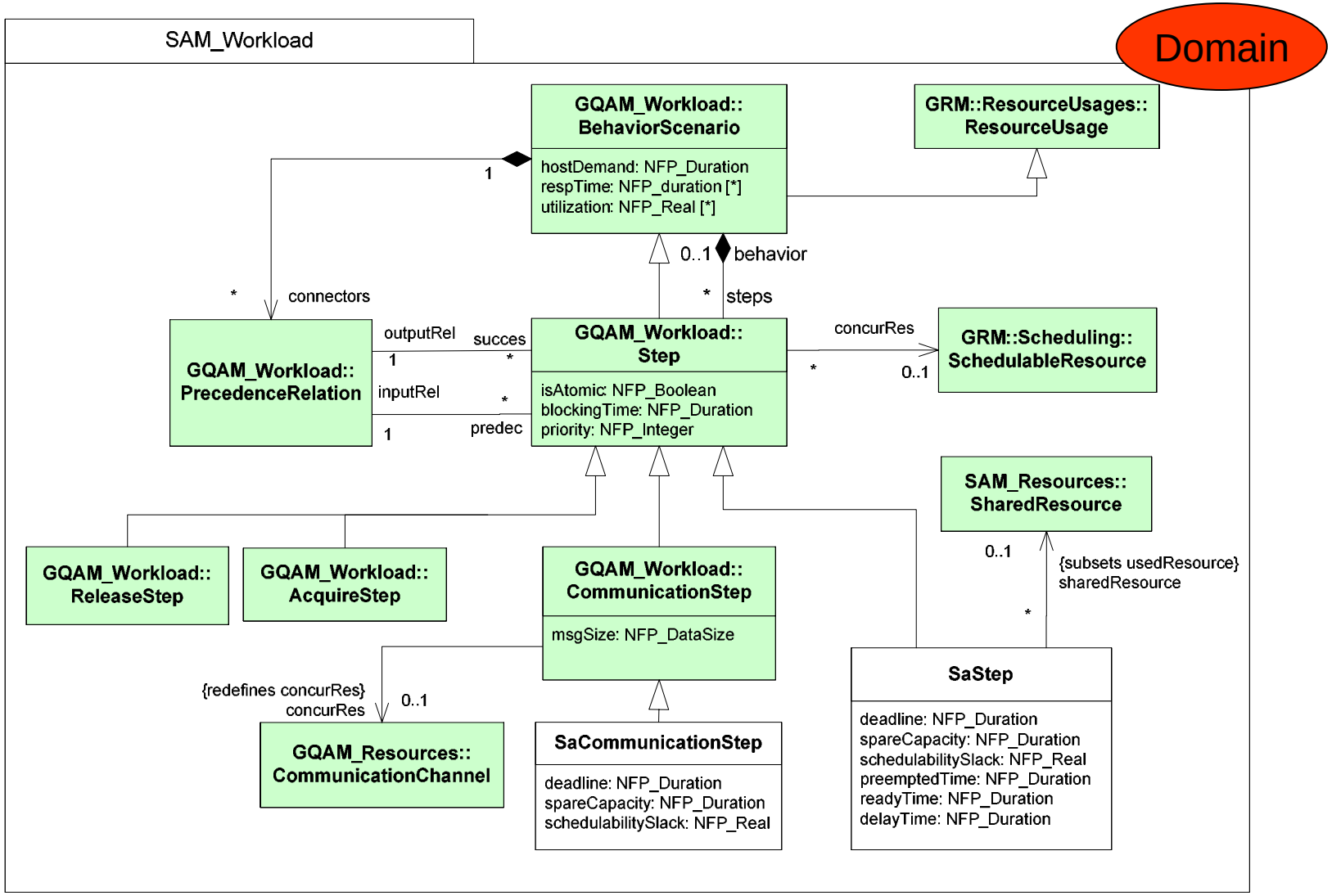

SAM
GQAM::
AnalysisContext
1
workloadBehavior
1..*
resourcesPlatform
GQAM_Workload::
WorkloadBehavior
GQAM_Resources::
ResourcesPlatform
SaAnalysisContext
isSchedulable: NFP_Boolean
optimalityCriterion: optimalityCriterionKind
« enumeration »
OptimallityCriterionKind
meetHardDeadlines
minimizeMissedDeadlines
minimizeMeanTardiness
undef
other

SAM_Workload: End-to-endFlow

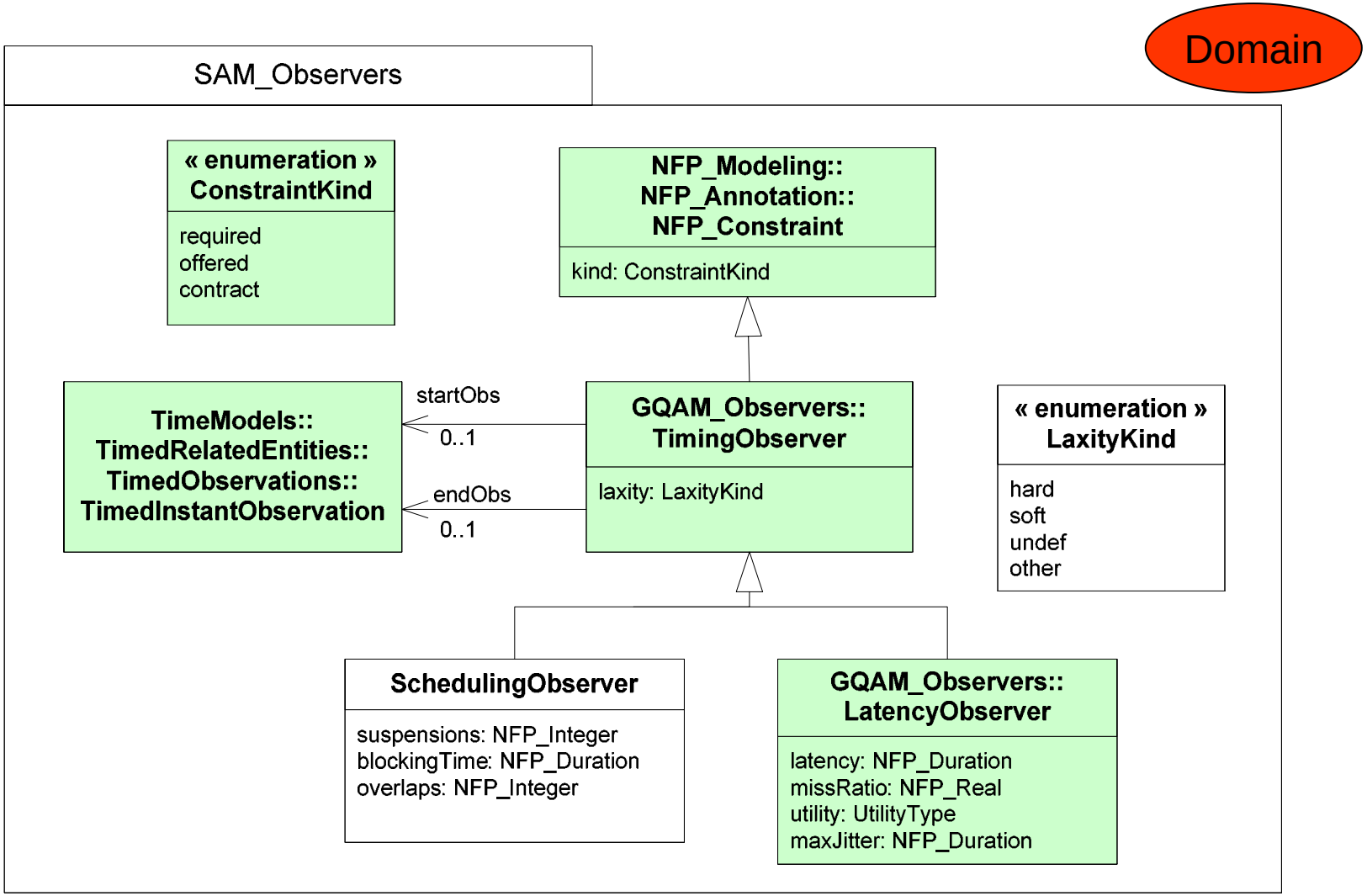
Domain



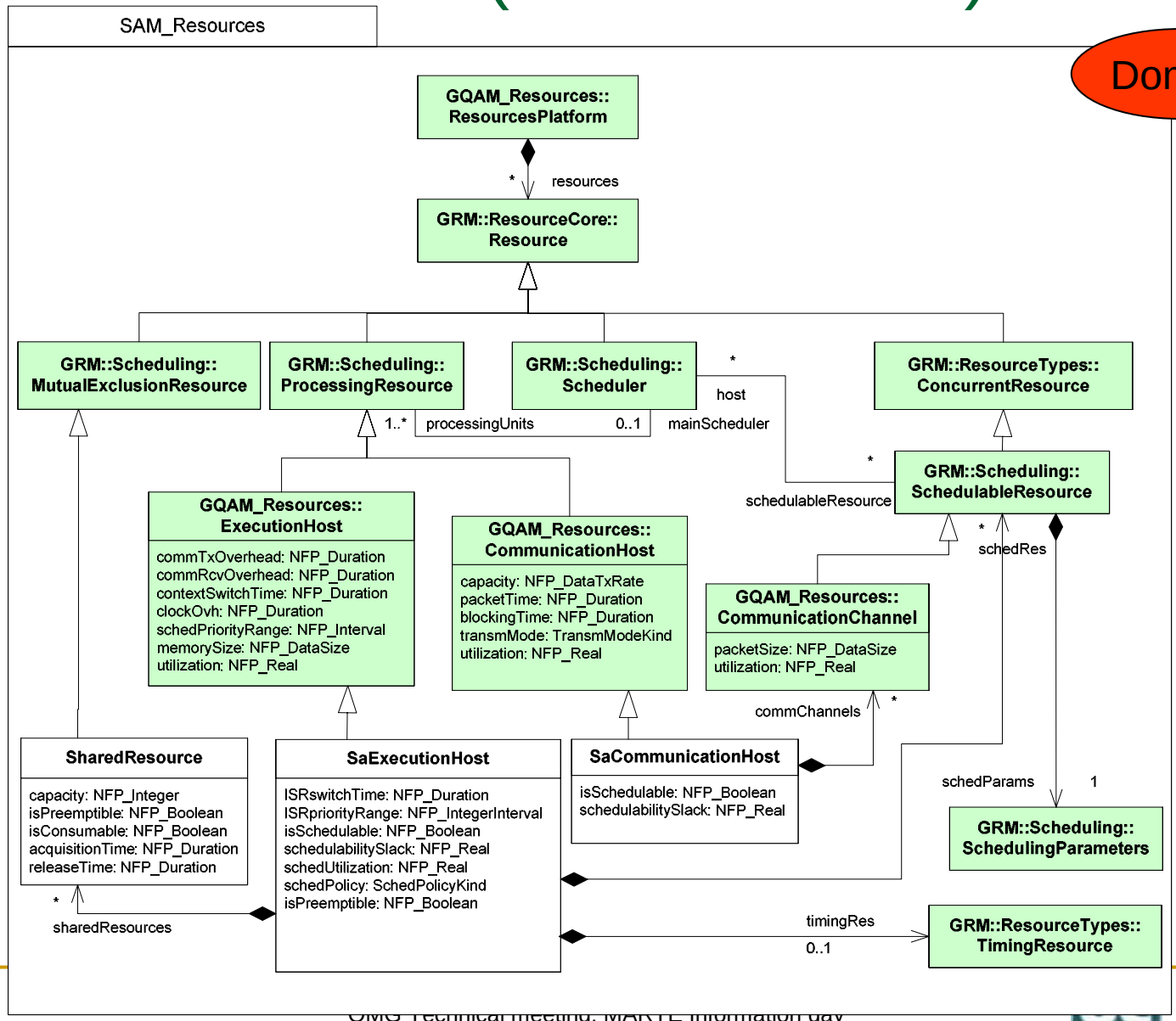
SAM_Workload: BehaviorScenario



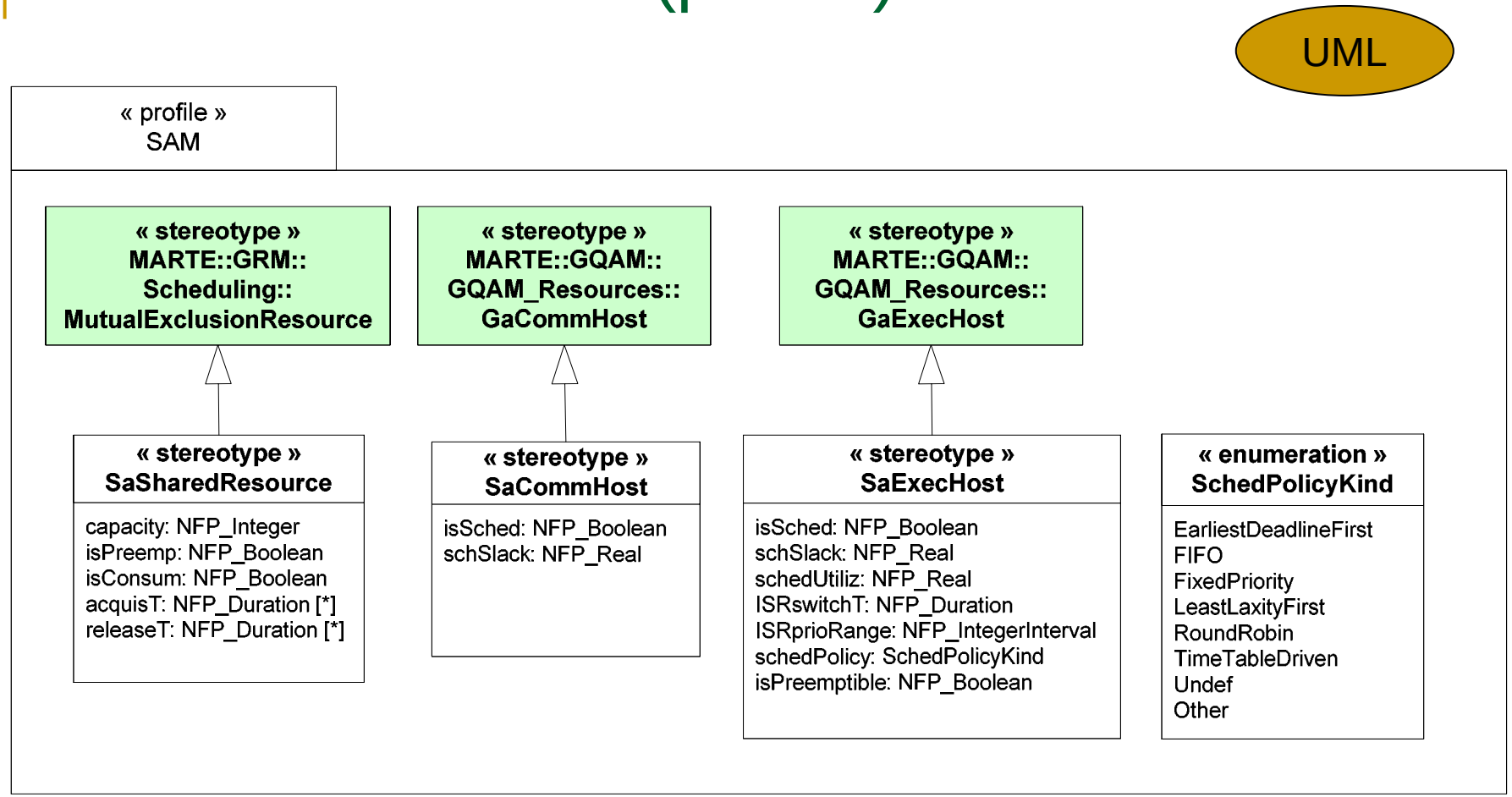
SAM_Observers



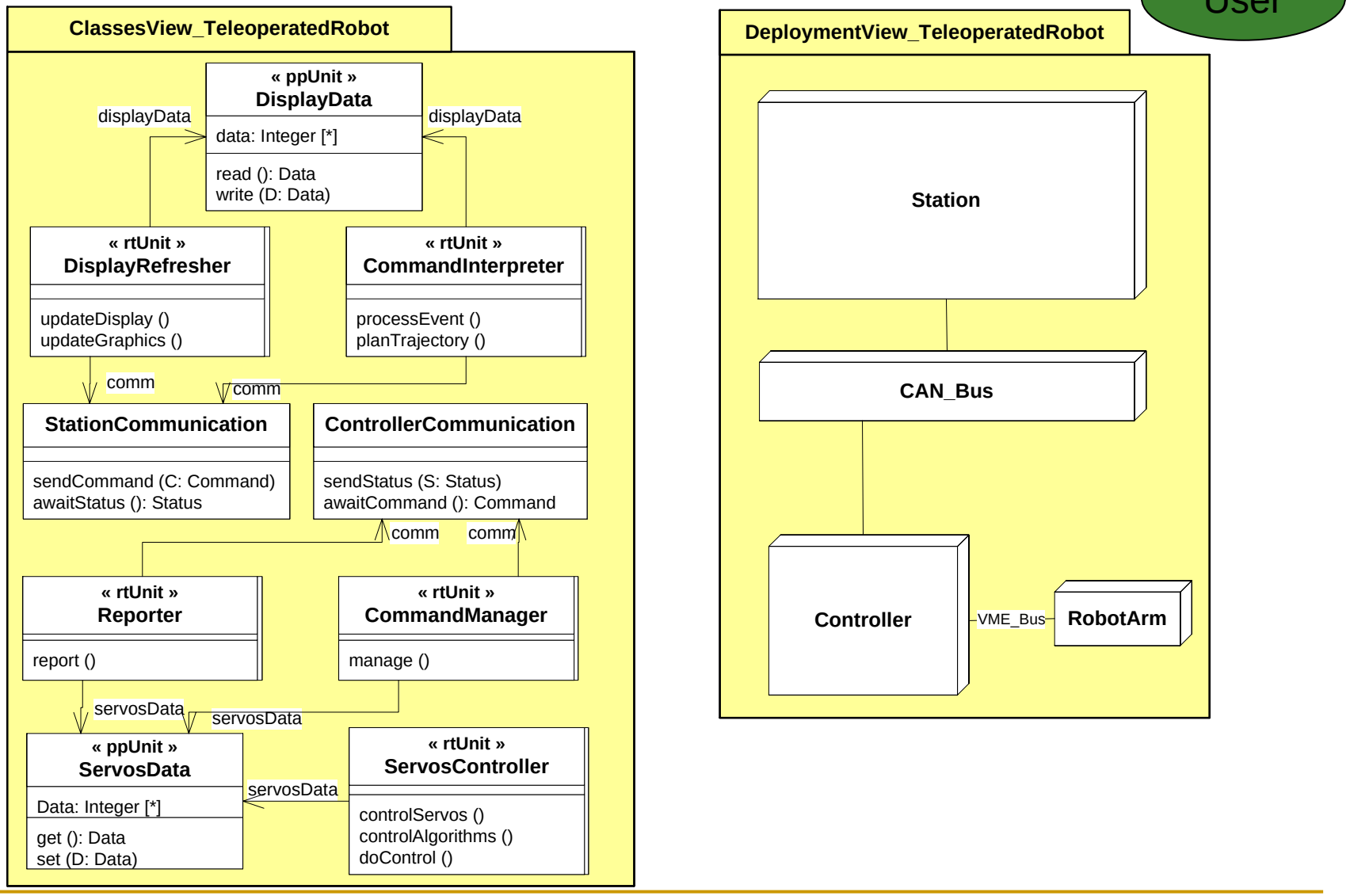
Resources in SAM (domain model)



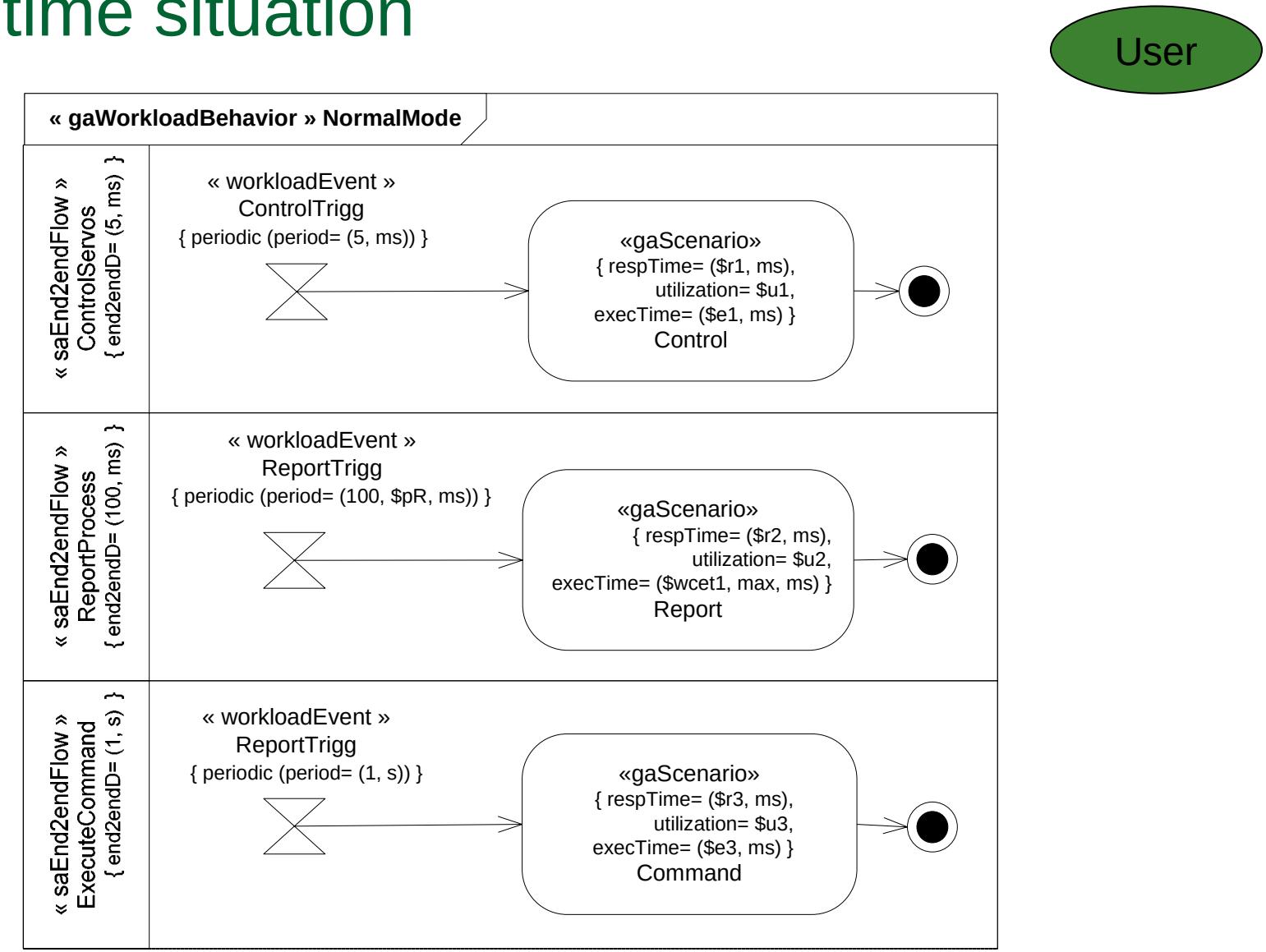
Resources in SAM (profile)



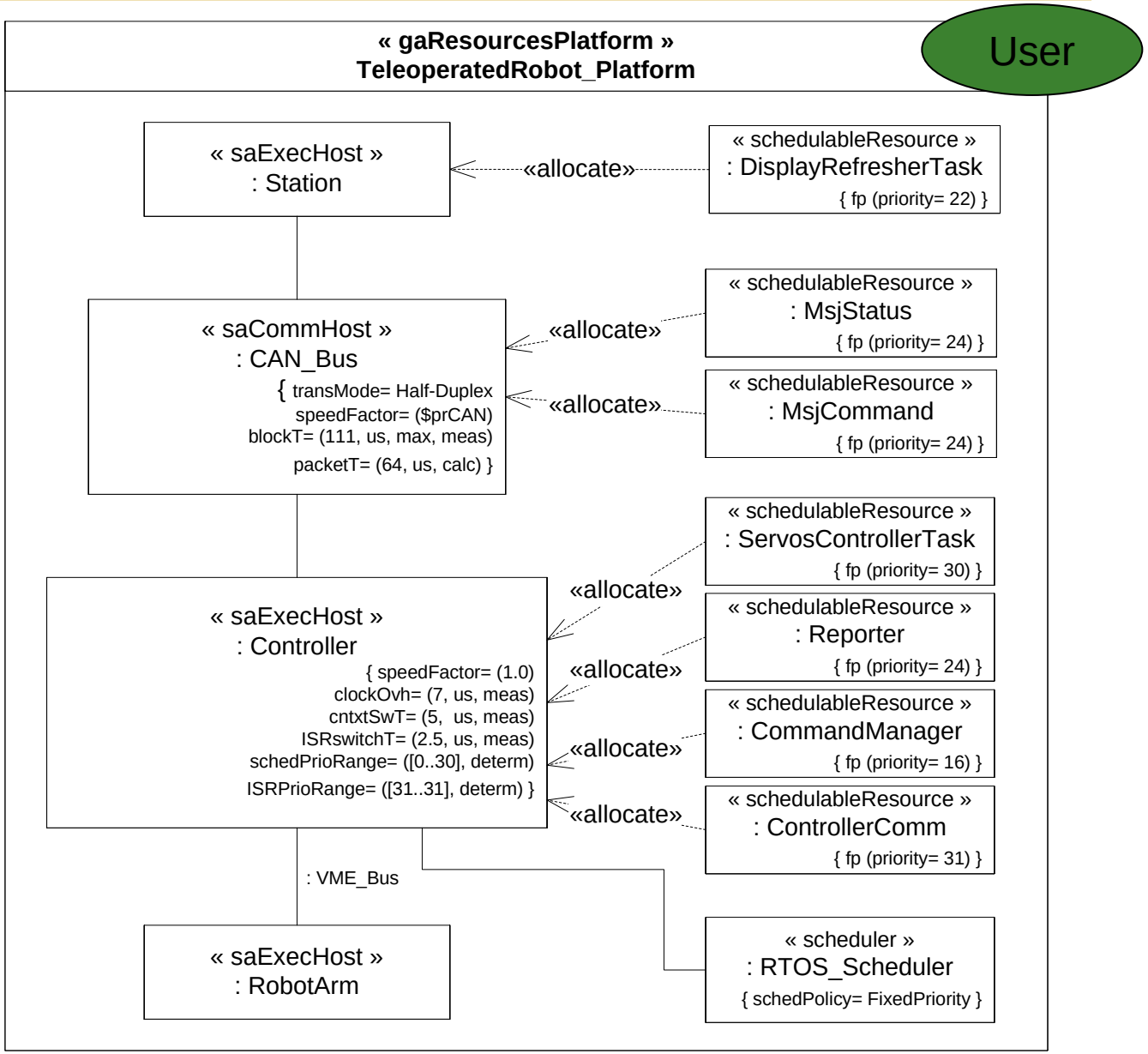
Example


ClassesView_TeleoperatedRobot
« ppUnit »
DisplayData
data: Integer [*]
read (): Data
write (D: Data)
displayData
displayData
« rtUnit »
DisplayRefresher
updateDisplay ()
updateGraphics ()
« rtUnit »
CommandInterpreter
processEvent ()
planTrajectory ()
comm
comm
StationCommunication
sendCommand (C: Command)
awaitStatus (): Status
ControllerCommunication
sendStatus (S: Status)
awaitCommand (): Command
comm
comm
« rtUnit »
Reporter
report ()
« rtUnit »
CommandManager
manage ()
servosData
servosData
« ppUnit »
ServosData
Data: Integer [*]
get (): Data
set (D: Data)
servosData
« rtUnit »
ServosController
controlServos ()
controlAlgorithms ()
doControl ()
User
DeploymentView_TeleoperatedRobot
Station
CAN_Bus
Controller
VME_Bus
RobotArm

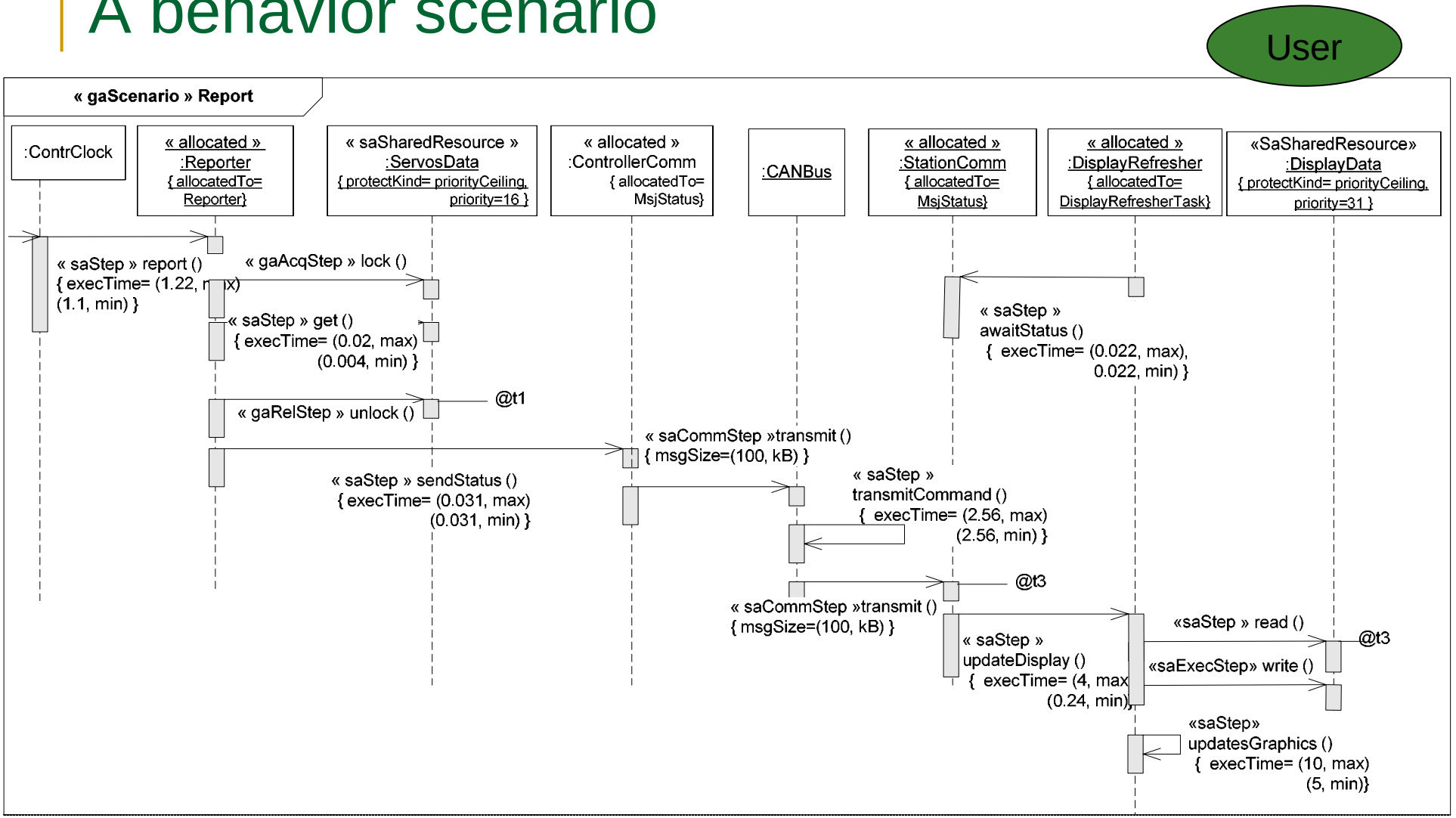
Real-time situation



Platform


« gaResourcesPlatform »
TeleoperatedRobot_Platform
User
« saExecHost »
: Station
«allocate»
« schedulableResource »
: DisplayRefresherTask
{ fp (priority= 22) }
« saCommHost »
: CAN_Bus
{ transMode= Half-Duplex
speedFactor= ($prCAN)
blockT= (111, us, max, meas)
packetT= (64, us, calc) }
«allocate»
« schedulableResource »
: MsjStatus
{ fp (priority= 24) }
«allocate»
« schedulableResource »
: MsjCommand
{ fp (priority= 24) }
« saExecHost »
: Controller
{ speedFactor= (1.0)
clockOvh= (7, us, meas)
cntxtSwT= (5, us, meas)
ISRswitchT= (2.5, us, meas)
schedPrioRange= ([0..30], determ)
ISRPrioRange= ([31..31], determ) }
«allocate»
« schedulableResource »
: ServosControllerTask
{ fp (priority= 30) }
«allocate»
« schedulableResource »
: Reporter
{ fp (priority= 24) }
«allocate»
« schedulableResource »
: CommandManager
{ fp (priority= 16) }
«allocate»
« schedulableResource »
: ControllerComm
{ fp (priority= 31) }
: VME_Bus
« saExecHost »
: RobotArm
« scheduler »
: RTOS_Scheduler
{ schedPolicy= FixedPriority }

A behavior scenario



Parametric analysis contexts