The Real-Time UML Standard

Ben Watson
Tri-Pacific Software Inc.
The Real-Time A&D Group in OMG

- An OMG working group
  - mission: to investigate and issue requests (RFPs) for standard ways and means to apply UML to real-time problems
- Three principal areas of investigation:
  - Time-related modeling issues
  - General quality of service/fault tolerance modeling issues
  - Architectural modeling issues
- Status:
  - first RFP issued (April 1999)
  - second RFP being drafted
Overview

- *The Real-Time UML Profile*
  - *Background*
  - Modeling Resources and Quality of Service
  - Modeling Time
  - Modeling Concurrency
  - The Schedulability Analysis Sub-Profile
  - The Performance Analysis Sub-Profile
  - The Real-Time CORBA Schedulability Analysis Sub-Profile
  - Real-Time CORBA UML Model
  - Model Processing Cycle

- Summary
The Real-Time UML RFP

- “UML Profile for Schedulability, Performance and Time”
  - First in a series of real-time specific RFPs
  - RFP issued in March 1999
  - Initial proposal submitted in August 2000 (ad/2000-08-04)
  - Standard adopted in September 2001

- Standard methods for UML modeling of:
  - Physical time
  - Timing specifications
  - Timing services and mechanisms
  - Modeling resources (logical and physical)
  - Concurrency and scheduling
  - Software and hardware infrastructure and their mapping
  - ..including specific notations for the above where necessary
Response to the RFP

- Just one submission throughout
- Consortium team:
  - ARTiSAN (UML tool vendor)
  - I-Logix (UML tool vendor)
  - Rational (UML tool vendor)
  - Telelogic (UML tool vendor)
  - TimeSys (RT tool and technology vendor)
  - Tri-Pacific Software (RT tool vendor)
- In consultation with many of the top real-time system experts (tool builders, analysis technique experts, academics)
RT Profile: Guiding Principles

- Ability to specify quantitative information directly in UML models
  - key to quantitative analysis and predictive modeling
- Flexibility:
  - users can model their RT systems using modeling approaches and styles of their own choosing
  - open to existing and new analysis techniques
- Facilitate the use of analysis methods
  - eliminate the need for a deep understanding of analysis methods
  - as much as possible, automate the generation of analysis models and the analysis process itself
Quantitative Methods for RT Systems

Once we have included QoS information in our models, we can use *quantitative methods* to:

- predict system characteristics (detect problems early)
- analyze existing system
- synthesize elements of the model

Methods considered for the profile:

- **Schedulability analysis**
  
  *will the system meet all of its deadlines?*

- **Performance analysis** based on queueing theory
  
  *what kind of response will the system have under load?*
Desired Development Model

- Seamless integration of technologies and tools based on standards for real-time modeling
Structure: Normative and Non-Normative Parts

General Resource Model
- Client
- Resource
- etc.

Specialized Analysis Domain Model
- Analysis
- Client
- Analysis Resource
- etc.

What (non-normative)

How (normative)

RTResourceModeling

ADprofile

Class

Object

ADresource

etc.

etc.
UML Real-Time Profile Structure (Normative Part)
Overview

- The Real-Time UML Profile
  - Background
  - *Modeling Resources and Quality of Service*
  - Modeling Time
  - Modeling Concurrency
  - The Schedulability Analysis Sub-Profile
  - The Performance Analysis Sub-Profile
  - The Real-Time CORBA Schedulability Analysis Sub-Profile
  - Real-Time CORBA UML Model
  - Model Processing Cycle

- Summary
Quality of Service Concepts

- An abstract, technology-independent representation of the engineering model can be specified using the general concept of Quality of Service (QoS):

  a specification (usually quantitative) of how a particular service is (to be) performed

  - e.g. throughput, capacity, response time

- The specification of a model element can include:
  - **offered QoS**: the QoS that it provides to its clients
  - **required QoS**: the QoS it requires from other components to support its QoS obligations
Resources and Quality of Service

- **Resource**: an element whose service capacity is limited, directly or indirectly, by the finite capacities of the underlying physical computing environment.

- The services of a resource are characterized by one or more *quality of service (QoS)* attributes:
  - capacity, reliability, availability, response time, etc.

![Diagram](image_url)

\[
\{\text{RequiredQoS} \leq \text{OfferedQoS}\}
\]
Simple Example

- Concurrent tasks accessing a monitor with known response time characteristics
The General Resource Model
Instances Vs. Specifications

- Practically all analysis methods are concerned with instance-based models
  - E.g., class models are too abstract to be analyzable
  - E.g., two executions of the same action (specification) may have very different execution times

- However, it is often useful to associate QoS characteristics with classes
  - Used to define default values that may be overridden for specific instances

- Need to apply a stereotype to both spec elements and instance elements
  - E.g., Class and Object
Basic Resource Usage Model

EventOccurrence
(from CausalityModel)

UsageDemand

ResourceUsage

AnalysisContext

ResourceInstance
(from CoreResourceModel)

StaticUsage

DynamicUsage

ResourceServiceInstance
(from CoreResourceModel)

+workload

+usedServices

+usedResources

0..1

1..n

0..n

0..n

1..n

1..n

1..n

1..n

1..n
Basic Causality Loop

- Used in modeling dynamic scenarios
Logical (Conceptual) Viewpoint

- A technology-independent view of the software
  - a “virtual” mechanism realized by a computer
Engineering (Realization) Viewpoint

- The realization of a specific set of logical components using facilities of the run-time environment
Viewpoints and Mappings

**Logical Viewpoint**

- INSTRUCTOR STATION
- AIRFRAME
- ATMOSPHERE MODEL
- GROUND MODEL
- ENGINES
- CONTROL SURFACES
- PILOT CONTROLS

**Engineering Viewpoint**

- Processor
  - OS process
  - TCP/IP socket
  - Ethernet LAN
    - TCP/IP socket
  - OS process
  - OS process

**Realization mappings**
Realization Mappings

- A correspondence between elements of two distinct models (logical and engineering)
- Semantics: the logical elements are *implemented* by the corresponding engineering model elements
  - logical elements can be viewed as “residing” on the corresponding engineering elements

![Diagram showing realization mappings between models](image-url)
Forms of Realization
Realization Relationships

- For sophisticated multi-layer deployment modeling
## Defined Stereotypes

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies To</th>
<th>Tags</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>«GRMacquire»</td>
<td>Stimulus, Message, Action, Operation, Reception…</td>
<td>GRMblocking [0..1]</td>
<td>An operation or invocation that acquires access rights to an exclusive resource</td>
</tr>
<tr>
<td>«GRMcode»</td>
<td>Abstraction</td>
<td>GRMmapping [0..1]</td>
<td>Relates a logical element model to the component that contains its code</td>
</tr>
<tr>
<td>(subclass of «GRMrealize»)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>«GRMdeploy»</td>
<td>Abstraction</td>
<td>GRMmapping [0..1]</td>
<td>Identifies where logical model elements are deployed</td>
</tr>
<tr>
<td>(subclass of «GRMrealize»)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>«GRMrealize»</td>
<td>Abstraction</td>
<td>GRMmapping [0..1]</td>
<td>A realization mapping</td>
</tr>
<tr>
<td>«GRMrelease»</td>
<td>Stimulus, Message, Action, Operation, Reception…</td>
<td>GRMexclServ [0..1]</td>
<td>An operation or invocation that releases an exclusive resource acquired previously</td>
</tr>
<tr>
<td>«GRMrequires»</td>
<td>Abstraction</td>
<td>GRMmapping [0..1]</td>
<td>A specification of a required environment for one or more logical model elements</td>
</tr>
<tr>
<td>(subclass of «GRMrealize»)</td>
<td>(subclass of «GRMrealize»)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview

- The Real-Time UML Profile
  - Background
  - Modeling Resources and Quality of Service
  - *Modeling Time*
  - Modeling Concurrency
  - The Schedulability Analysis Sub-Profile
  - The Performance Analysis Sub-Profile
  - The Real-Time CORBA Schedulability Analysis Sub-Profile
  - Real-Time CORBA UML Model
  - Model Processing Cycle

- Summary
General Time Model

![Diagram of General Time Model]

- TimeModel
- TimedEvents
- TimingMechanisms
- TimingServices
Physical and Measured Time

Physical Time

Time Interval

Time Value

Clock (from TimingMechanisms)

Physical Instant

Duration

{ordered} +start 1 1 +end

+start 1 1 +end

+measurement

kind : {discrete, dense}

+measurement

0..n

0..n

0..n

0..n

1 1 +end

1 +start

+referenceClock

1

0..n
Timing Mechanisms Model

ResourceInstance (from CoreResourceModel)

TimingMechanism
- stability
- drift
- skew
- set(time : TimeValue)
- get() : TimeValue
- reset()
- start()
- pause()

TimeInterval (from TimeModel)
- +resolution
- +accuracy
- +offset
- +referenceClock

Clock

Timer
- isPeriodic : Boolean

TimeValue (from TimeModel)
- +currentValue
- +maximalValue

TimedEvent (from TimedEvents)
- +timestamp
- +origin

ClockInterrupt (from TimedEvents)

Timeout (from TimedEvents)

generatedTimeouts 0..n
+generatedInterrupts 0..n
+generatedTimeouts 0..n
+generatedInterrupts 0..n

Tri-Pacific Software, Inc. © 2002
# Example Timing Stereotype

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Base Class</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>«RTaction»</td>
<td>Action</td>
<td>RTstart</td>
</tr>
<tr>
<td></td>
<td>ActionExecution</td>
<td>RTEnd</td>
</tr>
<tr>
<td></td>
<td>Message</td>
<td>RTduration</td>
</tr>
<tr>
<td></td>
<td>Stimulus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Method</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ActionSequence</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ActionState</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SubactivityState</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transition</td>
<td></td>
</tr>
<tr>
<td></td>
<td>State</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Tag Type</th>
<th>Multiplicity</th>
<th>Domain Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTstart</td>
<td>RTtimeValue</td>
<td>[0..1]</td>
<td>TimedAction::start</td>
</tr>
<tr>
<td>RTEnd</td>
<td>RTtimeValue</td>
<td>[0..1]</td>
<td>TimedAction::end</td>
</tr>
<tr>
<td>RTduration</td>
<td>RTtimeValue</td>
<td>[0..1]</td>
<td>TimedAction::duration</td>
</tr>
</tbody>
</table>
Specifying Time Values

- Time values can be represented by a special stereotype of Value («RTtimeValue») in different formats; e.g.
  - 12:04 (time of day)
  - 5.3, ‘ms’ (time interval)
  - 2000/10/27 (date)
  - Wed (day of week)
  - $param, ‘ms’ (parameterized value)
  - ‘poisson’, 5.4, ‘sec’ (time value with a Poisson distribution)
  - ‘histogram’ 0, 0.3 1, 0.4 2, 0.3, 3, ‘ms’
Specifying Arrival Patterns

- Method for specifying standard arrival pattern values
  - Bounded: ‘bounded’, <min-interval>, <max-interval>
  - Bursty: ‘bursty’, <burst-interval> <max.no.events>
  - Irregular: ‘irregular’, <interarrival-time>, [<interarrival-time>]*
  - Periodic: ‘periodic’, <period> [, <max-deviation>] 
  - Unbounded: ‘unbounded’, <probability-distribution>

- Probability distributions supported:
  - Bernoulli, Binomial, Exponential, Gamma, Geometric, Histogram, Normal, Poisson, Uniform
Overview

- The Real-Time UML Profile
  - Background
  - Modeling Resources and Quality of Service
  - Modeling Time
  - *Modeling Concurrency*
  - The Schedulability Analysis Sub-Profile
  - The Performance Analysis Sub-Profile
  - The Real-Time CORBA Schedulability Analysis Sub-Profile
  - Real-Time CORBA UML Model
  - Model Processing Cycle

- Summary
General Concurrency Modeling

Diagram showing relationships between various components:

- ResourceServiceInstance
- DeferredService
- ImmediateService
- threading
- Scenario
- SynchronousInvoke
- AsynchronousInvoke
- ActiveResource
- ConcurrentUnit
- StimuliQueue
- ProtectedResource
- Stimulus
- MessageAction
- ActionExecution
- StimulusGeneration
- ImmediateService
- +method
- +step
- +cause
- +effect
- isAtomic
- Boolean
- ResourceServiceInstance
- CoreResourceModel
- ResourceTypes
- DynamicUsageModel
- CausalityModel
## Defined Stereotypes

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies To</th>
<th>Tags</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>«CRAction»</td>
<td>Action, ActionExecution, Stimulus, Action, Message, Method…</td>
<td>CRAAtomic [0..1]</td>
<td>An action execution</td>
</tr>
<tr>
<td>«CRAsynch»</td>
<td>Action, ActionExecution</td>
<td></td>
<td>An asynchronous invocation</td>
</tr>
<tr>
<td>«CRC Concurrent»</td>
<td>Node, Component, Artifact, Class, Instance</td>
<td>CRMMain [0..1]</td>
<td>A concurrent unit concept</td>
</tr>
<tr>
<td>«CRContains»</td>
<td>Usage</td>
<td></td>
<td>A generalized usage dependency</td>
</tr>
<tr>
<td>«CRDeferred»</td>
<td>Operation, Reception, Message, Stimulus</td>
<td></td>
<td>A deferred receive</td>
</tr>
<tr>
<td>«CRImmediate»</td>
<td>Operation, Reception, Message, Stimulus</td>
<td>{remote, local} [0..1]</td>
<td>An instance of an immediate service</td>
</tr>
<tr>
<td>«CRmsgQ»</td>
<td>Instance, Object, Class, ClassifierRole</td>
<td></td>
<td>A stimuli queue</td>
</tr>
<tr>
<td>«CRSynch»</td>
<td>Action, ActionExecution</td>
<td></td>
<td>A synchronous invoke</td>
</tr>
</tbody>
</table>
Overview

- The Real-Time UML Profile
  - Background
  - Modeling Resources and Quality of Service
  - Modeling Time
  - Modeling Concurrency
  - *The Schedulability Analysis Sub-Profile*
  - The Performance Analysis Sub-Profile
  - The Real-Time CORBA Schedulability Analysis Sub-Profile
  - Real-Time CORBA UML Model
  - Model Processing Cycle

- Summary
Schedulability Analysis Sub-Profile
Schedulability Example: Annotation

```plaintext
Sensors : SensorInterface

TelemetryGatherer : DataGatherer

TelemetryDisplayer : DataDisplayer

TelemetryProcessor : DataProcessor

SensorData : RawDataStorage

Display : DisplayInterface

TGClock : Clock
```

### Sensors
- **SensorInterface**

### TelemetryGatherer
- **DataGatherer**

### TelemetryDisplayer
- **DataDisplayer**

### TelemetryProcessor
- **DataProcessor**

### SensorData
- **RawDataStorage**

### Display
- **DisplayInterface**

### TGClock
- **Clock**

### SAAction
- **SAPriority=2, SWorstCase=(93,'ms'), RTduration=(33.5,'ms')**
- **SAAction**
  - **SAPriority=3, SWorstCase=(177,'ms'), RTduration=(46.5,'ms')**

### SAResource
- **SACapacity=1, SAAccessControl=PriorityInheritance**

### SATrigger
- **SAScheduled=$R1, RTat=('periodic',100,'ms')**
- **SAResponse**
  - **SAAbsDeadline=(100,'ms')**

### SAAction
- **RTstart=(10,'ms'), RTend=(31.5,'ms')**
- **RTstart=(3,'ms'), RTend=(5,'ms')**
- **RTstart=(16.5,'ms'), RTend=(33.5,'ms')**
- **RTstart=(12.5,'ms'), RTend=(16.5,'ms')**

### SAResponse
- **SAPriority=2, SWorstCase=(93,'ms'), RTduration=(33.5,'ms')**
- **SAPriority=1, SWorstCase=(50.5,'ms'), RTduration=(12.5,'ms')**

### SATrigger
- **SAScheduled=$R2, RTat=('periodic',60,'ms')**
- **SAScheduled=$R3, RTat=('periodic',200,'ms')**

### SAResponse
- **SAPriority=2, SWorstCase=(93,'ms'), RTduration=(33.5,'ms')**
- **SAPriority=3, SWorstCase=(177,'ms'), RTduration=(46.5,'ms')**

### SAAction
- **RTstart=(16.5,'ms'), RTend=(33.5,'ms')**
- **RTstart=(10,'ms'), RTend=(31.5,'ms')**
- **RTstart=(3,'ms'), RTend=(5,'ms')**
- **RTstart=(12.5,'ms'), RTend=(16.5,'ms')**

### SAResponse
- **SAPriority=1, SWorstCase=(50.5,'ms'), RTduration=(12.5,'ms')**
- **SAPriority=1, SWorstCase=(50.5,'ms'), RTduration=(12.5,'ms')**
- **SAPriority=2, SWorstCase=(93,'ms'), RTduration=(33.5,'ms')**
- **SAPriority=3, SWorstCase=(177,'ms'), RTduration=(46.5,'ms')**

---

Tri-Pacific Software, Inc. © 2002
Schedulability Example: Deployment

```
«SASchedulable»
TelemetryDisplayer : DataDisplayer

«SASchedulable»
TelemetryGatherer : DataGatherer

«SASchedulable»
TelemetryProcessor : DataProcessor

«GRMdeploys»

«SAEngine»
{SARate=1, SASchedulingPolicy=FixedPriority}
: Ix86Processor

«SAOwns»

«SAResource»
SensorData : RawDataStorage
```
<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies To</th>
<th>Tags</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>«SAAction» (subclass of «RTAction» and «CRAction»)</td>
<td>Action, ActionExecution, Stimulus, Action, Message, Method…</td>
<td>SAPriority [0..1] SAAActualPty [0..1] SABlocking [0..1] SARready [0..1] SADelay [0..1] SARelRelease [0..1] SAPreempted [0..1] SAWorstCase [0..1] SALaxity [0..1] SAPriority [0..1] SAAbsDeadline [0..1] SARelDeadline [0..1] SAusedResource [0..1] SAhost [0..1]</td>
<td>An action</td>
</tr>
<tr>
<td>«SAEngine»</td>
<td>Node, Instance, Object, Classifier, ClassifierRole</td>
<td>SASchedulingPolicy [0..1] SAAccessPolicy [0..1] SARate [0..1] SAContextSwitch [0..1] SAPriorityRange [0..1] SAPreemptible [0..1] SAUtilization [0..1] SASchedulable [0..1] Saresources [0..1]</td>
<td>An execution engine</td>
</tr>
</tbody>
</table>
# Defined Stereotypes (2 of 3)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies To</th>
<th>Tags</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>«SAOwns» (subclass of «GRMrealize»)</td>
<td>Abstraction</td>
<td></td>
<td>Identifies ownership of resources</td>
</tr>
<tr>
<td>«SAPrecedes»</td>
<td>Usage</td>
<td></td>
<td>A precedence relationship between actions and triggers</td>
</tr>
</tbody>
</table>
| «SAResource»                        | Classifier, ClassifierRole, Instance, Object, Node | SAAccessControl [0..1]  
SAConsumeable [0..1]  
SACapacity [0..1]  
SAAcquisition [0..1]  
SADeacquisition [0..1]  
SAPtyCeiling [0..1]  
SAPreemptible [0..1] | A resource of some kind                    |
| «SAResponse»                        | Action, ActionExecution, Stimulus, Action, Message, Method… | SAUtilization [0..1]  
SASpare [0..1]  
SALack [0..1]  
SAOverlaps [0..1] | A response to a stimulus or action         |
| «SASchedulable» (subclass of «SAAction») | Classifier, ClassifierRole, Instance, Object, Node |                                               | A schedulable resource                                       |
## Defined Stereotypes (3 of 3)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies To</th>
<th>Tags</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>«SAScheduler»</td>
<td>Classifier, ClassifierRole, Instance, Object</td>
<td>SASchedulingPolicy [0..1]</td>
<td>A scheduler</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAExecutionEngine [0..1]</td>
<td></td>
</tr>
<tr>
<td>«SAPrecedes»</td>
<td>Usage</td>
<td></td>
<td>A precedence relationship between actions and triggers</td>
</tr>
<tr>
<td>«SASituation»</td>
<td>Collaboration, CollaborationInstance, ActivityGraph</td>
<td></td>
<td>A schedulability analysis context</td>
</tr>
<tr>
<td>«SATrigger»</td>
<td>Message, Stimulus</td>
<td>SAScheduled [0..1]</td>
<td>A trigger</td>
</tr>
<tr>
<td>(subclass of</td>
<td></td>
<td>SASAprecedents [0..1]</td>
<td></td>
</tr>
<tr>
<td>«SAAction»)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>«SAusedHost»</td>
<td>Usage</td>
<td></td>
<td>Identifies schedulable resources used for execution of actions</td>
</tr>
<tr>
<td>«SAUses»</td>
<td>Usage</td>
<td></td>
<td>Identifies sharable resources</td>
</tr>
</tbody>
</table>
Policies Supported

- **Scheduling Policies:**
  - Rate Monotonic, Deadline Monotonic, HKL, Fixed Priority, Minimum Laxity First, Maximize Accrued Utility, Minimum Slack Time
  - …may be extended in the future

- **Access Control Policies:**
  - FIFO, Priority Inheritance, No Preemption, Highest Lockers, Priority Ceiling
  - …may be extended in the future
Overview

- The Real-Time UML Profile
  - Background
  - Modeling Resources and Quality of Service
  - Modeling Time
  - Modeling Concurrency
  - The Schedulability Analysis Sub-Profile
  - The Performance Analysis Sub-Profile
  - The Real-Time CORBA Schedulability Analysis Sub-Profile
  - Real-Time CORBA UML Model
  - Model Processing Cycle

- Summary
Performance Analysis Concepts
### Defined Stereotypes (1 of 2)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies To</th>
<th>Tags</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>«PAclosedLoad»</td>
<td>Action, ActionExecution, Stimulus, Action, Message, Method…</td>
<td>PArespTime [0..*] PApriority [0..1] PApopulation [0..1] PAextDelay [0..1]</td>
<td>A closed workload</td>
</tr>
<tr>
<td>«PAcontext»</td>
<td>Collaboration, CollaborationInstanceSet, ActivityGraph</td>
<td>PAutilization [0..*] PAschdPolicy [0..1] PARTate [0..1] PActxtSwT [0..1] PAprioRange [0..1] PApreemptible [0..1] PAthroughput [0..1]</td>
<td>A performance analysis context</td>
</tr>
<tr>
<td>«PAhost»</td>
<td>Classifier, Node, ClassifierRole, Instance, Partition</td>
<td>PArespTime [0..*] PApriority [0..1] PApopulation [0..1] PAextDelay [0..1]</td>
<td>A deferred receive</td>
</tr>
<tr>
<td>«PAopenLoad»</td>
<td>Action, ActionExecution, Stimulus, Action, Message, Method…</td>
<td>PArespTime [0..*] PApriority [0..1] PAoccurrence [0..1]</td>
<td>An open workload</td>
</tr>
</tbody>
</table>
## Defined Stereotypes (1 of 2)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies To</th>
<th>Tags</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>«PAresource»</td>
<td>Classifier, Node, ClassifierRole, Instance, Partition</td>
<td>PAutilization [0..*] PAschdPolicy [0..1] PAcapacity [0..1] PAmaxTime [0..1] PArespTime [0..1] PAwaitTime [0..1] PAthroughput [0..1]</td>
<td>A passive resource</td>
</tr>
<tr>
<td>«PStep»</td>
<td>Message, ActionState, Stimulus, SubactivityState</td>
<td>PADemand [0..1] PArespTime [0..1] PAprob [0..1] PArept [0..1] PAdelay [0..1] PAextOp [0..1] PAinterval [0..1]</td>
<td>A step in a scenario</td>
</tr>
</tbody>
</table>
Specifying Performance Values

- A complex structured string with the following format
  - \(<\text{kind-of-value}> , <\text{modifier}> , <\text{time-value}>\)
- Where:
  - \(<\text{kind-of-value}> ::= \text{‘req’} \mid \text{‘assm’} \mid \text{‘pred’} \mid \text{‘msr’}\)
  - Required, assumed, predicated, measured
  - \(<\text{modifier}> ::= \text{‘mean’} \mid \text{‘sigma’} \mid \text{‘kth-mom’} , <\text{Integer}> \mid \text{‘max’} \mid \text{‘percentile’} <\text{Real}> \mid \text{‘dist’}\)
  - E.g.:
    - \{\text{PAdemand = (‘msr’, ‘mean’, (20, ‘ms’))}\}
Overview

- The Real-Time UML Profile
  - Background
  - Modeling Resources and Quality of Service
  - Modeling Time
  - Modeling Concurrency
  - The Schedulability Analysis Sub-Profile
  - The Performance Analysis Sub-Profile
  - *The Real-Time CORBA Schedulability Analysis Sub-Profile*
  - Real-Time CORBA UML Model
  - Model Processing Cycle

- Summary
Real-Time CORBA: Schedulability Sub-Profile
Example: RT CORBA

- `RTclock`: Clock
- `RSAclient`: RSAclient timeout=(105,'ms'), RSAclPrio=12 TelemetryGatherer : DataGatherer
- `RSAserver`: SACapacity=4
- `RSAserver`: SACapacity=10
- `RSAorb`: SASchedulingPolicy='RateMonotonic'
- `SAEngine`: SASchedulingPolicy='RateMonotonic', SARate=1
  - `P1`: I586
- `SAEngine`: SASchedulingPolicy='RateMonotonic', SARate=0.5
  - `P2`: Ix86
Example: RT CORBA Usage Scenario

```
«RTclock»
TGClock

«RSAclient»
TelemetryGatherer
: DataGatherer

«RSAserver»
Sensor1
: SensorInterface

«RSAserver»
Sensor2
: SensorInterface

«RTevent» {RTat=('periodic', 100, 'ms')}
tick()

«SAAction»
{SAWorstCase=(15,'ms'),
RTduration=(10,'ms')}
readSensor()

«SAAction»
{SAWorstCase=(30,'ms'),
RTduration=(20,'ms')}
readSensor()

«SAAction»
{SAWorstCase=(25,'ms'),
RTduration=(5,'ms')}
compare()

«SAAbsDeadline»
(100,'ms')

«SAResponse»
```

```
«SASituation»
```
## Defined Stereotypes

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies To</th>
<th>Tags</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>«RSAclient» (subclass of «SASchedulable»)</td>
<td>Classifier, ClassifierRole, Instance, Object, Node</td>
<td>RSAtimeout [0..1] RSAclPrio [0..1] RSAprivate [0..1] RSAhost [0..1]</td>
<td>An RT CORBA client</td>
</tr>
<tr>
<td>«RSAconnection» (subclass of «SASchedulable» and «SAResource»)</td>
<td>Classifier, ClassifierRole, Instance, Object, Node</td>
<td>SAAccessControl [0..1] RSAshared [0..1] RSAhiPrio [0..1] RSAloPrio [0..1] RSAserver [0..1]</td>
<td>An RT CORBA connection</td>
</tr>
<tr>
<td>«RSAmutex» (subclass of «SAResource»)</td>
<td>Classifier, ClassifierRole, Instance, Object, Node</td>
<td>SAAccessControl [0..1] RSAhost [0..1]</td>
<td>An RT CORBA mutex</td>
</tr>
<tr>
<td>«RSAorb» (subclass of «SAResource»)</td>
<td>Classifier, ClassifierRole, Instance, Object, Node</td>
<td>SAschedulingPolicy [0..1]</td>
<td>An RT CORBA ORB</td>
</tr>
<tr>
<td>«RSAserver» (subclass of «SAResource»)</td>
<td>Classifier, ClassifierRole, Instance, Object, Node</td>
<td>RSAsrvPrio [0..1] SACapacity [0..1]</td>
<td>An RT CORBA server</td>
</tr>
</tbody>
</table>
Overview

- The Real-Time UML Profile
  - Background
  - Modeling Resources and Quality of Service
  - Modeling Time
  - Modeling Concurrency
  - The Schedulability Analysis Sub-Profile
  - The Performance Analysis Sub-Profile
  - The Real-Time CORBA Schedulability Analysis Sub-Profile
  - Real-Time CORBA UML Model
  - *Model Processing Cycle*

- Summary
Model Processing

Model Configurer
- Configuration Data Set
  - Parametrized UML Model (XMI)

Model Convertor
- Configured UML Model (XMI)

Model Analyzer
- Domain Model
  - Processing Control Specification

Results Convertor
- Results UML Model (XMI)

Results

Model Processor

Model Editor

The Tag Value Language

- Tagged value format:
  \{<\text{tag-name}> = <\text{tag-value}>\}

- Used to specify complex (structured) tagged values

- Based on a small proper subset of the freeware Perl language
  - Includes: variables, numbers, booleans, strings, lists, expressions (including conditionals), operators, and functions

- Suitable for:
  - expressing complex dependencies between values
  - writing processing scripts
Summary (1 of 2)

- Defines a set of extensions for directly expressing real-time domain concepts and their key quantitative characteristics in UML:
  - resources
  - concurrency mechanisms
  - time and timing mechanisms

- Models constructed using the real-time UML standard can be exchanged with specialized analysis tools
  - automated model validation
  - eliminates need for expensive and rare expertise
  - predictive engineering models
Summary (2 of 2)

- UML provides a common and standardized underpinning that supports all the components of our solution
  - for object-oriented modeling
  - for predictive QoS modeling (via the real-time profile)
  - for design analysis and synthesis (tool interchange)
  - for architectural definition
  - for implementation (through full automatic code generation)

- Furthermore, as a standard, it enables model interchange between specialized tools and is a basis for significant automation of the RT software development process