Non-Functional Analysis
for UML Models

Model Processing for Analysis
Ben Watson
Tri-Pacific Software, Inc.
watson@tripac.com
The (So Called) Real-Time UML Profile

- Officially, it is *The UML Profile for Schedulability, Performance and Time*
- The profile was adopted at the September OMG meeting in Toronto
- The profile addresses the time related non-functional characteristics of a UML model:
  - Models for time, resources, concurrency
  - Sub profiles (and models) for schedulability and performance
  - Software and hardware infrastructure and their mapping
  - Specific notations for the above where necessary
    - Stereotypes
    - Tagged values
Desired Development Model

- Seamless integration of technologies and tools based on standards for real-time modeling

Model Editing Tool
- UML model
- 3.1
- 4
- 5
- quantitative annotations

Model Analysis Tool
- Automated model conversion
- Inverse model conversion

UML model
Instance- vs Class-Based Models

- Practically all analysis methods are concerned with instance-based models
- 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
Example: Collaboration
Example: Annotated Sequence

```
<<SAAction>>
SAAbsDeadline=(1000, 'ms')
RTduration=(335, 'ms')
SAPriority=32
<<Local::SAAction>>
RTduration=(20, 'ms')
<<SATrigger>>
SAOccurrencePattern=(1000, 'ms')

<<SAAction>>
RTduration=(150, 'ms')

1.2: c2 : createItem
1.2.1:
<<Local::SAAction>>
RTduration=(165, 'ms')
```
What We Needed to Build

- A schedulability analysis model processor
- Two issues to address
  - The program architecture
  - Extracting a timing model from the UML model
- Starting point for the model processor was RapidRMA, our Rate Monotonic Analysis (RMA) tool
- Our goal: To make the integration with multiple UML tools as seamless as possible
  - Make it unobtrusive (look like the host application)
  - Provide complete RMA tool capability
  - Do it interactively
Program Architecture

UML System

Adaptor

UMLRMA

RMA Analysis Engine

RapidRMA
Schedulability Analysis Sub-Profile

- SAction
  - Priority
  - Worst-case Completion Time
  - Delay Time
  - Preempted Time
  - Ready Time
  - Release Time
  - Blocking Time
  - Laxity
  - Absolute Deadline
  - Relative Deadline
  - isAtomic

- Response
  - Utilization
  - Spare Capacity
  - Overlaps
  - Slack Time

- Trigger
  - isSchedulable

- SResource
  - Capacity
  - Acquisition Time
  - Deacquisition Time
  - isConsumable
  - Priority Ceiling
  - isPreemptible

- RealTimeSituation

- ExecutionEngine
  - Priority Range
  - Processing Rate
  - Context Switch Time
  - Utilization
  - isPreemptible
  - isSchedulable

- SchedulingPolicy

- SchedulingJob
### Defined Stereotypes (1 of 3)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies To</th>
<th>Tags</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>«SAAAction» (subclass of «RTaction» and «CRAction»)</td>
<td>Action, ActionExecution, Stimulus, Action, Message, Method…</td>
<td>SAPriority [0..1] SAAActualPty [0..1] SABlocking [0..1] SAReady [0..1] SADelay [0..1] SARelease [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>
<tr>
<td>«SAResource»</td>
<td>Classifier, ClassifierRole, Instance, Object, Node</td>
<td>SAAccessControl [0..1] SAConsumable [0..1] SACapacity [0..1] SAAcquisition [0..1] SADeacquisition [0..1] SAPtyCeiling [0..1] SAPreemptible [0..1]</td>
<td>A resource of some kind</td>
</tr>
<tr>
<td>«SAResponse»</td>
<td>Action, ActionExecution, Stimulus, Action, Message, Method…</td>
<td>SAUtilization [0..1] SASpare [0..1] SASlack [0..1] SAOverlaps [0..1]</td>
<td>A response to a stimulus or action</td>
</tr>
<tr>
<td>«SASchedulable» (subclass of «SAAction»)</td>
<td>Classifier, ClassifierRole, Instance, Object, Node</td>
<td></td>
<td>A schedulable resource</td>
</tr>
</tbody>
</table>
## 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] SAExecutionEngine [0..1]</td>
<td>A scheduler</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] SASAprecedents [0..1]</td>
<td>A trigger</td>
</tr>
<tr>
<td>(subclass of «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>
Minimum Annotations for Schedulability

- External signals and time triggered internal signals
  - Occurrence pattern
  - Deadline
- Actions that process the signals
  - Execution time
  - Action sequence
    - Precedence
    - Synchronous / asynchronous
- Deployment
  - Processor
  - Device
  - Instance
Minimum Annotations

[Diagram of a UML sequence diagram with annotations and a tree structure diagram showing elements such as CPU1, sensor, disk, and display with specific attributes like ElementName, ElementType, and others, including SAAccessPolicy, SASchedulingPolicy, SPriority, and SARate.]
Classifiers and Instances

- All schedulability analysis is instance-based
- Annotations on a classifier are permitted
  - Default value for the entire class
  - An annotation on an instance overrides the classifier annotation
- Weak support for instances in UML tools
  - No method to correlate instances on different sequence diagrams
  - Adopt the convention that identical instance names refer to the same instance
- It is important to know when actions belong to the same instance of an object due to run-to-completion semantics
Rules to Extract Timing Model

- The sequence diagrams determine the timing model
- Locate all external signals
  - Incoming from the environment
- Determine arrival pattern and deadline from "<<SATrigger>>" and "<<SAAction>>" stereotypes
- Determine the action that is the response to the trigger event
  - Single action
  - Action sequence (precedence)
    - "<<SAAction>>" and "<<local::SAAction>>"
  - Action sequence inherits the trigger occurrence pattern
  - End-to-end deadline
- Determine tasks and resources
  - Synchronous vs asynchronous messages
Timing Model

tasks

Resources

sensorData

sensors

telemetryGatherer
Another Example
Timing Model

Tasks

/o3:O3::a3,1 → /o5:O5::a3,3

/o3:O3::a3,2
Results Example
Conclusion

- We have implemented a model processor for the RT UML profile
  - Conforms to the standard
  - Meets our “seamless” goals
- Future work
  - Implement the entire standard
    - Layered models
    - Parameterized tagged values
  - Extensions to the standard
    - Stochastic analysis
    - Scripting interface
Questions?