MODEL-BASED DEVELOPMENT FOR RTSJ PLATFORMS

Java Technologies for Real-time and Embedded Systems
24-26 October 2012
Copenhagen, Denmark

MIGUEL ÁNGEL DE MIGUEL
mmiguel@dit.upm.es

EMILIO SALAZAR
esalazar@dit.upm.es
Topics

- RTSJ Java code generation process
- RTSJ code generation approaches
  - RTSJ
  - MARTE
- Integration MARTE-MAST-RTSJ
- RTSJ generation code
  - Handled RTSJ Classes
  - NOT handled RTSJ Classes
- Transformation Patterns
- Application example
- Tools availability
RTSJ Java code generation process

1. MARTE UML Model
   - MARTE to MAST transformation
     - MAST Model
   - MARTE to Java AST transformation
     - Java AST (javaxmi)
2. MAST Model
   - MAST to Java AST transformation
     - Java AST (javaxmi)
3. Java AST to Text transformation
   - RTSJ Java code
RTSJ code generation approaches

• Models that reuse the RTSJ UML model library
  » Load of RTSJ in UML models
  » Reuse of RTSJ Types and Methods
  » Software models and Scheduling Analysis models are independent

• Models annotated with MARTE (GRM, SwConcurrency, GQAM and SAM)
  » Application of profiles
  » Stereotype applications
  » Scheduling analysis models and Java RTSJ code consistent
RTSJ Models and RTSJ Generation

- Import of RTSJ Model library
- Reuse, in user model, of RTSJ classes and methods (*RealtimeThread*, *AsyncEvent*, ...)
- Generate RTSJ Java code
  - Generate Java AST (this the L0 of Java applications)
  - Generate Java code
Scheduling Analysis Annotations MARTE-MAST

- MARTE-Scheduling analysis and software model are supported in the same UML model.
  
  » MARTE-Scheduling analysis introduces information to make scheduling analysis

- Both can contain inconsistencies
UML+MARTE to Java RTSJ and MAST

- Model with MARTE: timing information is represented with MARTE and not with UML-RTSJ libraries
- Integrated generation of MAST and RTSJ Java code
  » Generate Java AST and MAST analysis model
  » Generate Java code (the same for any javaxmi)
Integration of MARTE-MAST-RTSJ (1)

- UML2RTSJ supports the mapping from UML 2.2 L0 to \textit{javaxaxmi}
- MARTE\_SA\_Gen extends this for the integration of \textit{MARTE Scheduling analysis} into \textit{javaxaxmi} models
- Java AST models based on RTSJ (\textit{javaxaxmi}) are consistent with MAST models generated
Integration of MARTE-MAST-RTSJ (2)

- UML+RTSJ Model Library and UML+MARTE differences:
  - UML+RTSJ includes explicit references to RTSJ library (types, method calls, generalizations, ...)
  - UML+MARTE does not include any reference to RTSJ. Any reference to RTSJ in Java code is generated based on MARTE
  - UML+RTSJ are based on Java primitive types (e.g. long, void, ...)
  - UML+MARTE do not use Java types.
  - The consistency of MAST model and RTSJ behaviour depends on modeller
  - Generator guarantees the consistency of MAST and RTSJ behaviour
RTSJ Classes handled in mapping

• Threads
  » `RealtimeThread`

• Scheduling
  » `SchedulingParameters`, `PriorityParameters`, `ReleaseParameters`, `PeriodicParameters`, `AperiodicParameters`, `SporadicParameters`, `ProcessingGroupParameters`

• Synchronization
  » `MonitorControl`, `PriorityCeilingEmulation`, `PriorityInheritance`

• Asynchrony
  » `AsyncEvent`, `AsyncEventHandler`, `BoundAsyncEventHandler`
RTSJ Classes NOT handled in mapping

- MARTE handles memory as resource, but we cannot make classifications as: *Immortal, Heap, Scoped*
  » Fundamental in RTSJ execution time predictability
- Threads
  » *NoHeapRealtimeThread*
- Memory Management
  » *Scoped Memory, Immortal Memory, Physical Memory Manager, Raw Memory Access*
- System and Options
  » *POSIXSignalHandler, RealtimeSecurity*
Transformation Patterns (1)

• MARTE2RTSJ generates Java anonymous classes for handling `ScheduledResource` and handlers of `GaWorkloadEvent`. These classes
  » Template for each RTSJ thread pattern
    - Periodic, Sporadic, Aperiodic
  » Handling of exceptions
    - Deadlines, Execution times

• MARTE2RTSJ configures object monitor based on `SaSharedResource`

• MARTE2RTSJ supports combination of `AsyncEvent` and `AsyncEventHandler` based on `InterruptResource/SchedulableResource`, and `GaSteps` sequences
Transformation Patterns (2)

- MARTE2RTSJ `AperiodicThread` template for handling Bursty and Deferred MARTE aperiodic
  » `release` method for notification of aperiodic event arrival
  » `run` method: thread cycle
  » array of timers for handling aperiodic events deadlines

- `AsyncEvent -> BundedAsyncEventHandler` sequence are represented with

```
GaWorkloadEvent → GaStep → GaStep
     ↓                  ↓
  InterruptResource, SchedulableResource   SchedulableResource
```
Application Example: tele-presence car

- Tele-drive for a tele-presence car. Main components:
  - Steering wheel and throttle pedal handlers
  - Cruise control system: handlers of brake, throttle, lever, speed and engine, and control
  - Combined obstacles detector: indoor localization sensor, webcam images filter, IR detector
Application Example: tele-presence car

- Transformations and generators provide:
  - MAST scheduling analysis for schedulable resources
  - RTSJ Java code -> RTSJ code generated + logical code in Opaque Behaviors
http://www.erca-ma-assets.org

- Eclipse Indigo with ERMA 64K and 32K
- Eclipse Indigo 64K with ERMA and Papyrus
- ERMA plug-ins site
- ERMA plug-ins for RSA 8.0.3
- ERMA Safety-Analysis Assets
  » S&D Profiles and Model Libraries
  » FTA & FMECA Eclipse Languages
  » UML to FTA & FMECA transformations
  » Item Toolkit Bridges
- ERMA Real-Time Assets
  » MARTE 1.1 Conform Standard and Stereotypes Editor
  » RTSJ and Ada-Ravenscar Modelling tools
  » MAST Eclipse Languages
  » MARTE to MAST and RTSJ
Thank you for your attention!

MIGUEL ÁNGEL DE MIGUEL
mmiguel@dit.upm.es

EMILIO SALAZAR
esalazar@dit.upm.es