An Approach for Execution of MARTE-based Application Models

Workshop on Distributed Object Computing for Real-time and Embedded Systems

Washington, DC, USA
July 16th, 2008

C. Mraidha, A. Cuccuru and S. Gérard

MDE for real-time applications

A modeling language
- Capturing real-time specificities in models
  - UML
  - Specialization for real-time domain: MARTE

A modeling methodology
- Modeling rules
- Well formed models

A tool support
- Modeling real-time applications
  - Modeling language editors
  - Methodology rules tooling
- Executing real-time models
  - Execution framework
Agenda

- UML profile for MARTE
  - A Real-time MoCC

- Accord approach
  - A modeling methodology
  - An execution framework for DRES

- MARTE models execution
  - Mapping of MARTE models to an execution platform

- Conclusions
UML profile for MARTE Overview

Foundations for RT/E systems modeling and analysis:
- CoreElements
- NFPs
- Time
- Generic resource modeling
- Generic component modeling
- Allocation

Specialization of MARTE foundations for modeling purpose (specification, design, …):
- RTE model of computation and communication
- Software resource modeling
- Hardware resource modeling

Specialization of foundations for annotating model for analysis purpose:
- Generic quantitative analysis
- Schedulability analysis
- Performance analysis
MARTE model of computation

- High-level modeling concepts for RT/E design
  - Qualitative aspects
    - E.g. concurrency and behavior
  - Quantitative aspects as real-time feature
    - E.g. deadline or period

- Allows expressing real-time constraints on component interfaces and connectors
  - Applicable whether component are active or passive

- For active components, introduces specific models of computation
  - Currently, active objects (e.g. Rhapsody, Rose RT, Accord)
  - Alternative MoCC can be defined using the MARTE foundations
MARTE model of computation (cont.)

Provides high-level concepts for modeling qualitative real-time features on classes / structured classes / components

- **Real-Time Unit (RTUnit)**
  - Specialization of the Active Objects of the UML 2
  - Owns at least one schedulable resource
  - Resources are managed either statically (pool) or dynamically
  - May have operational mode description (similar to AADL modes)

- **Protected Passive Unit (PPUnit)**
  - Specialization of the Passive Objects of the UML2
  - Requires schedulable resources to be executed
  - Supports different concurrency policies (e.g. sequential, guarded)
  - Policies are specified either locally or globally
  - Execution is either immediateRemote or deferred
MARTE model of computation (cont.)

- Provides high-level concepts for modeling quantitative real-time features on classes / structured classes / components
  - Real-Time Behavior (RtBehavior)
    • Message Queue size and policy bound to a provided behavior
  - Real-Time Feature (RTF)
    • Extends UML Action, Message, Signal, BehavioralFeature
    • Relative/absolute/bound deadlines, ready time and miss ratio
  - Real-Time Connector (RteConnector)
    • Extends UML Connector
    • Throughput, transmission mode and max blocking/packet Tx time
Agenda

- UML profile for MARTE
  - A Real-time MoCC

- Accord approach
  - A modeling methodology
  - An execution framework for DRES

- MARTE models execution
  - Mapping of MARTE models to an execution platform

- Conclusions
Accord is a platform of MDE for DRES
➤ Offers high level of abstraction approach
➤ Based on UML/MARTE

Powered with Papyrus: http://www.papyrusuml.org
Outlines of the Accord modeling methodology

Specifier
Initial Requirements Document
Specification Modeling Rules

build Specification Model
Prototyping Rules

Prototyper
Specification Model

build Prototype Model

Validator
Prototype Model

build Test Model

Timing Model
analyse schedulability

Test Model

ACCORD / UML

specifier
prototyper
validator
Accord model structure

- Consistency & complementarities
  - Separation of concerns
  - Verify consistency between views

- Modeling system behavior
  - Separation of concerns improves:
    - Readability
    - Maintainability
    - Reusability

⇒ Use MARTE (RTEmoCC) to capture RTE characteristics
Execution platform

- **OS abstraction layer**
  - POSIX compliant operating systems
    - Linux, Solaris, VxWorks…
  - Offers basic mechanisms (not dependent of the OS)
    - Thread, semaphore, mutex…

- **Real-time kernel**
  - Relies on the OS abstraction layer
  - Provides resources for MARTE RtUnits execution
    - Real-Time Objects
  - Provides a deadline controller that verifies RT constraints at runtime
    - Real-Time Features
  - Provides a scheduler
    - Service scheduling
  - Provides intra concurrency management policy of RtUnits
    - n reader / 1 writer protocol

→ Transformation: from MARTE-based models to execution platform specific models
Agenda

- UML profile for MARTE
  - A Real-time MoCC

- Accord approach
  - A modeling methodology
  - An execution framework for DRES

- MARTE models execution
  - Mapping of MARTE models to an execution platform

- Conclusions
Mapping MARTE concepts on the execution platform

- **ProtocolStateMachine**
  - behavior

- **RealTimeObject**
  - currentState : State
  - itsRTO
  - globalCtrl

- **GlobalController**
  - 0..1 globalCtrl

- **RealTimeFeature**
  - tRef : TimeVal
  - deadline : TimeVal
  - readyTime : TimeVal
  - period : TimeVal
  - perNb : integer

- **ConcurrencyFeature**
  - concurrency : ConcurrencyMode

- **Service**
  - itsService
  - running

- **MailBox**
  - itsMB

- **« MARTE RtService »**

- **« MARTE RtBehavior »**

- **« MARTE RTF »**
From MARTE-based model to RT executable code

Automatic transformations (Patterns applications)

C++ code generation

C++ code using Accord kernel facilities
Current limitations and next steps

Current limitations
- Partial support of MARTE MoCC family
- C++ as the action language
- Component-based models are not executable

Next steps and ongoing work
- Larger support of MARTE MoCC by Accord Kernel (parameterization)
- C++ code generator for an action language
- Provide transformations to the execution platform for component-based models
Agenda

- UML profile for MARTE
  - A Real-time MoCC

- Accord approach
  - A modeling methodology
  - An execution framework for DRES

- MARTE models execution
  - Mapping of MARTE models to an execution platform

- Conclusions
Conclusions

- **MARTE-based modeling methodology**
  - Provides executable MARTE-based models

- **Execution platform**
  - Provides execution resources for MARTE concepts

- **Tool support**
  - Modeling
    - Papyrus UML ([http://www.papyrus-uml.org](http://www.papyrus-uml.org))
      - Advanced modeling for UML / MARTE / SysML
  - Automated transformations
    - MARTE-based models to execution platform mapping (patterns)
    - Automatic C++ code generation

- **Ongoing work for a larger support of MARTE MoCC and modeling processes**
Thank you for your attention!