AADL Application modeling with MARTE
Madeleine Faugère, Timothée Bourdeau – THALES Research and Technology
Robert de Simone – INRIA
Sébastien Gérard – CEA List
AADL in a nutshell

- Synchronous data flow applications modeling
- Architecture Description
  - Specific layer in the development life cycle
- Implicit execution platform model
  - AADL execution platform model based on specific thread execution automata
  - Applications and platform execution semantics have to be in line
- Non-functional properties come as a model decoration
  - A better model integration would improve end-users application modeling understanding
- Existing open source and commercial tools
  - TopCased, Cheddar, aDes, OSATE, Ocarina, STOOD....
MARTE in a nutshell

- A Unified Modeling Language for RTE Systems
  - Too much specific Real-Time and Embedded approaches, RTE languages, different tools

- Relationships with generic OMG standards
  - Profile for UML 2
    - All UML 2 can load the profile and can design application with MARTE
  - The UML profile for Systems Engineering: SysML
    - Design & Analyse RTE systems
  - Replace UML profile for SPT
    - Integrate issues coming from the SPT
  - Use OCL2 and MOF2 QVT where needed
    - Align with the MDA approach (Model transformations … )

- Align with almost RT/E specific OMG standards
  - The UML profile for Modeling QoS and FT Characteristics and Mechanisms
  - The UML profile for SoC
  - The Real-Time CORBA profile & UML profile for CCM/LwCCM
MARTE overview

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

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

Specialization of foundations for modeling purpose (specification, design...):
- RTE model of calculus & computation
- Software resource modelling
- Hardware resource modelling
- Generic for Real time Embedded System applications modeling and analysis
- Architecture Description
  - Address early design stages
- Complementary and consistent view make the model more understandable
  - time properties, performance, scheduling features,…
  - platform execution model can be explicitly modelized
- Full integration of non-functional properties in the model
- Young standard
  - Tool chain in development
AADL application modeling with MARTE

- **AADL – MARTE Bridge benefits**
  - MARTE will provide AADL users with
    - early design capabilities
    - enhanced modeling capabilities through consistent views
    - rich modeling information
  - AADL will provide integrated validation tool suites for synchronous data flows based applications design with MARTE

Specific data flow synchronous domain

Generic Real Time

Embedded System Domain
Most AADL concepts can be represented using UML and MARTE Core concepts

- HRM for Hardware components → HRM packages
- Software components → SRM packages
- Bindings → Alloc packages (issued from SysML)
- Flow ports and associated features → GCM packages (issued from SysML)
- Aadl properties → NFP and Marte libraries

- Subcomponents → UML Parts
- Port Connections → UML delegation connectors between ports
- Parameter connections → Object Flows on activity diagrams
- Flow specification → UML Object flows between UML Object Pins
- Modes → UML state machines and collaboration diagram
## AADL Software components in MARTE

<table>
<thead>
<tr>
<th>AADL</th>
<th>MARTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Type</td>
<td>UML::DataType</td>
</tr>
<tr>
<td>Thread</td>
<td>MARTE::Sw_SchedulableResource</td>
</tr>
<tr>
<td>Thread Group</td>
<td>MARTE::Sw_SchedulableResource_group</td>
</tr>
<tr>
<td>Process</td>
<td>MARTE::MemoryPartition</td>
</tr>
</tbody>
</table>
## AADL Execution Platform Components in MARTE

<table>
<thead>
<tr>
<th>AADL</th>
<th>MARTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor (Abstraction of HW + associated SW)</td>
<td>MARTE::Hw_Processor (ASICs, GPPs, DSPs, FPGAs)</td>
</tr>
<tr>
<td>Memory (code + data)</td>
<td>MARTE::Hw_Memory (RAM, ROM, Drive...)</td>
</tr>
<tr>
<td>Bus (HW channel + communication protocols)</td>
<td>MARTE::Hw_Bus (CAN, I2C, SPI, USB, PCI...)</td>
</tr>
<tr>
<td>Device (Input/Output)</td>
<td>MARTE::Hw_Device (I/O, Power supply, Cooling...)</td>
</tr>
<tr>
<td>System (composite HW/SW component)</td>
<td>UML4SysML::Block</td>
</tr>
</tbody>
</table>
Use of UML Extension and Generalization mechanism
system implementation a_client.impl
subcomponents
the_processor : processor a_client_processor;
the_process : process a_client_process;
properties
Actual_Processor_Binding => reference the_processor
applies to the_process;
end a_client.impl;

⇒ Marte allocation between software and hardware components
Subcomponents are modelized as UML Parts
processor powerPC
Features
MemBus : requires bus access Memory_Bus;
Dev_Bus : requires bus access Device_Bus
End PowerPC;

bus Memory.Bus
end Memory.Bus;

bus and data components may provide internal data/service access via
an UML Interface
Ports and Port connections

thread read_data
features
  in_data : in data port data1;
  out_data : out data port data1;
end read_data;

thread basic_control
features
  in_data : in data port data2;
  out_data : out data port data2;
end basic_control;

process implementation control_speed.impl
subcomponents
  read_data : thread read_data;
  control : thread basic_control;
connections
  delayed_C1 : data port read_data.out_data --> control.in_data;
properties
  Period => 50ms;
end control_speed.impl

AADL

⇒ Ports are represented by SysML flow Ports
⇒ Port connections are represented by delegation connectors
Subprogram are represented as Operation, and calls through UML Messages on sequence diagrams.
Flight simulator: a dynamic re-configuration thought mode switching
Modes in UML

Mode transition modelized by an UML state machine

Different mode configuration through Collaboration diagrams
Length_Unit : type units ( mm, cm => mm * 10, m => cm * 100, km => m * 1000 );

OnOff : type aadlboolean;

Speed_Range : type range of aadlreal 0 .. 250 units ( kph );

mass_t : type aadlreal units mass_u;

mass_u : type units (g, kg => g*1000, t => kg*1000);

Wheel_speed : aadlintege 0 rpm .. 5000 rpm units ( rpm applies to (system):

allowed_mass: mass_range_t applies to memory, processor, bus, device, system;

actual mass: mass_t applies to (memory, processor, bus, device, system);
First AADL – MARTE alignment based on AADL constructs and features and MARTE artifacts

A MARTE2AADL code generator has been developed (Thales RT)

Future work

AADL Profile will be extended by explicit AADL properties modelization