Agenda

- Part 1
  - Introduction to MDD for RT/E systems & MARTE in a nutshell
- Part 2
  - Non-functional properties modeling
  - Outline of the Value Specification Language (VSL)
- Part 3
  - The timing model
- Part 4
  - A component model for RT/E
- Part 5
  - Platform modeling
- Part 6
  - Repetitive structure modeling
- Part 7
  - Model-based analysis for RT/E
- Part 8
  - MARTE and AADL
- Part 9
  - Conclusions
AADL Overview

AADL Architecture Analysis Description Language

- Architecture Description Language dedicated to RTES
- International standard at SAE (AS5506, 2004)
- Adapted for many critical computer system domain
  - Automotive, space, robotics, industrial control, medical, avionics, …
- Allow specification, analysis and automated integration of real-time performance critical
  - Timing,
  - Safety,
  - Schedulability,
  - Fault tolerant,
  - Security,
  - Distributed computing systems,…
What does AADL look like?

A graphical representation

```
senseconn data port sense.outed -> compute1.ined;
compute12: data port compute1.outed --> compute2.ined;
compute23: data port compute2.outed --> compute3.ined;
actuateconn: data port compute3.outed --> actuate.ined;
business db --> sense.devbus;
business db --> actuate.devbus;
flows
  etelatency: end to end flow sense.flowl --> senseconn --> com-
  pute1.flowl
    --> compute12 --> compute2.flowl --> compute23 --> com-
  pute3.flowl
    --> actuateconn --> actuate.flowl { latency => 153 ms;};
end application.twosamplesteps;
```

A textual representation
Guidelines for modeling AADL application in MARTE

- **MARTE**
  - Generic for Real time Embedded System application modeling and analysis
  - Address early and detailed design stages
  - Complementary and consistent views make the model more understandable
    - Platform execution model can be explicitly modelized
    - Full integration of non-functional properties in the model (Time, performance, scheduling features...)

- **AADL**
  - Specific to synchronous data flow application modeling and analysis
  - Address detailed design stages
  - Based on an implicit execution platform model
    - Specific thread execution automata
    - Applications and platform execution semantics have to be in line
  - Lack of non-functional properties model integration
AADL useful for....

Non functional aspects through properties specification
- Temporal, safety, reliability,....

Architecture and design
- Components and interfaces, connections
- Data and control flows
- Run time architecture

Analysis and verification aspects
- End-to-end latency
- Age of signal data and jitter
- ...

AADL Elements
- Software components
- Hardware component
- Allocation
- Port and connections
- Flows
- Modes
- Properties (extensible language)
### MARTE-AADL concepts mapping

- First AADL – MARTE alignment based on AADL constructs and features and MARTE artifacts.
- Next step: Deeper map AADL component semantics, AADL properties and implicit AADL platform semantics on MARTE concepts (MARTE concepts properties and NFP, VSL language)

<table>
<thead>
<tr>
<th>AADL concepts</th>
<th>MARTE concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software components</td>
<td>memoryPartition, wSchedulerableRessource,..</td>
</tr>
<tr>
<td>Hardware components</td>
<td>hwProcessor, hwMemory,..</td>
</tr>
<tr>
<td>Binding</td>
<td>Allocated</td>
</tr>
<tr>
<td>AADL features</td>
<td>MARTE flowPorts, UML required/provided interfaces…</td>
</tr>
<tr>
<td>Subcomponents</td>
<td>UML Parts</td>
</tr>
<tr>
<td>Port Connections</td>
<td>UML delegation/assembly</td>
</tr>
<tr>
<td>Flow specification</td>
<td>UML object flows</td>
</tr>
<tr>
<td>Modes</td>
<td>UML state machines</td>
</tr>
<tr>
<td>Properties</td>
<td>Not yet documented</td>
</tr>
</tbody>
</table>
### AADL Component Types

- Specifies a functional interface in terms of features (ports, port groups, flow specifications, subcomponent access...), properties
- UML package containing AADL component declaration

### AADL Component Implementations

- Describes the internal structure and behavior of that component in terms of subcomponents, connections and flows across them, and behavioral modes
- UML package containing AADL component implementations
- AADL implementation liked to AADL declaration through UML Realization

### AADL Component Extension

- UML Generalization
AADL System
- Represents a composite software, execution platform or system components
- Represented by a “SysML” block

AADL Subcomponents
- represented by UML parts

```plaintext
system Flight_System
end Flight_System;

system implementation Flight_System.Generic
subcomponents
  S_HCI : system HCI_System.PowerPC_G4;
  S_NAP : system Nav_Autopilot_System.PowerPC_G4;
  LAN : bus LAN_Bus; ....
```
Required data/bus access

- AADL Bus and Data components access
- Provide data/service to other AADL components
- Access required from other AADL components

- Declaration part: Provide/required access modeled in MARTE via provided / required UML Interfaces

- Implementation part:
  - Access design by delegation / assembly connectors
  - Resources may be specialized with real time features or associated to services (synchronization, concurrency access ...)

processor powerPC
Features
  MemBus : requires bus access Memory_Bus;
  Dev_Bus : requires bus access Device_Bus
End PowerPC;

bus Memory_Bus
end Memory_Bus;

Access declaration

Access connections

Generated code
**AADL Ports**

- Flow ports are represented by MARTE Flow Ports

```aadl
system Nav_Autopilot_System
  AP_Toggle : in event port;
  AP_Position_Input : in event data port Nav_Types::Position.GPS;
```

**AADL Port, Bus and Memory Connections**

- Bus/data access data are represented by UML connectors between the port providing the access and the device

```aadl
bus access Dev_Bus -> Engine_RPM_Controller.Connector_Bus;
```

**Flows ports connections are represented by UML connectors (delegation or assembly connectors)**

```aadl
data port P_Nav_Con.Engine_RPM_Output -> Engine_RPM_Controller.Input;
```

**Generated code**
AADL System bindings
- Are represented by MARTE <<allocation>> stereotyped UML Dependency

Execution platform

AADL Bindings

Software Application
Subprogram

- Are represented by Operation

Subprogram calls

- Are represented through UML Messages on sequence diagrams

Generated code:

```plaintext
thread implementation my_thread.impl
calls {
    first_subpgr : subprogram my_subprogram;
    second_subpgr : subprogram my_second_subprogram;
}
end my_thread.impl;
```
Modes with MARTE guidelines

Mode transition modelized by an UML state machine

Different mode configuration through Collaboration diagrams

Reference MARTE Tutorial – November 2007 – Version 1.1

www.omgmar.te.org

Copyright © Thales, CEA and INRIA 2007 All rights reserved, commercial use strictly prohibited.
Properties

- **AADL Properties**
  - Are represented by `<AADL properties>` stereotyped UML comments
  - Will be aligned using NFPs, VSL language and MARTE concepts properties
MARTE to AADL code generator is already available

Bridge between MARTE/AADL and Cheddar (Scheduling analysis) already tested