Executing AADL models with UML/MARTE

Aoste Team-Project

F. Mallet, C. André, J. DeAntoni
Real-time embedded systems are often made out of communicating components, with varying assumptions on timely aspects of computation and communication progress, according to varying assumptions on implementation platforms.
Motivation

- **Synchronous** vs. **Asynchronous** communities
  - **Time-triggered / Cycle-accurate / Discrete Time**: enhanced predictability and higher dependability, close to system-level and hardware design, or RT simulation.
  - **Event-based / Discrete Event**: more flexible to configuration changes, close to software design

- **Computations** mix periodic and aperiodic (often sporadic) behaviors

- **Communications** can be considered as:
  - (handshake or queued) **message-passing** (should match event-based)
  - shared memory write/read **data sampling** (should match time-triggered)

Complex modeling demands. How well founded?
Objectives

- **Marte** has means to define time models
  - aims to be general-purpose (inside Real-Time Embedded field)
  - MoCCs to be built from time model constructors (by experts, *not* end-users)

- **AADL** offers a broad range of computation and communication styles related to time aspects
  - Specific syntax
  - Good example to illustrate the approach

So,…

- … we attempt here to provide a formal description of **AADL** time aspects using **Marte**
  - Make UML models executable with AADL execution semantics
AADL

• **Architecture** **Analysis & Design Language**
  
  – Avionics/Automotive standard from SAE (Society of Automotive Engineers).

  – Aims at Design and Analysis (scheduling/schedulability/performance/…) more than code production

MARTE

• **Modeling and Analysis of Real-Time and Embedded systems**

  – UML2 Profile standard from OMG (Object Management Group)

  – Aims at Design and Analysis (scheduling/schedulability/performance/…) more than code production
AADL  MARTE

Application modeling

• Thread, Thread Group, Process, Subprogram

• Usual UML2 state and activity diagrams + structured classes

Execution Platform modeling

• Bus, Device, Processor, Memory

• General Resource Modeling, Hw/Sw resource (components)

Allocation modeling

• Binding (should respect constraints on timing aspects)

• Allocation (should respect constraints on timing aspects)
we shall consider varying assumptions on the application pipeline communication and computation timing nature
AADL example(s)

- pipeline above should be event-based *(right!)*
- pipeline below should be time-triggered data-sampled *(really?)*
MARTE representation of this example (pattern)

- OMG UML2 Profile for **M**odeling and **A**nalysis of **R**eal-**T**ime and **E**mbedded systems
  - OMG Adopted Specification (ptc/07-08-04) => FTF
MARTE-rizing it (1)

- **Application part** *(our focus)*
  - UML Activity Diagrams

- Different queueing policies *(event, event-data, data)* ?

With queues *(event-data)*

Without queues *(data)*
MARTE-rizing it (2)

- **Software Execution Platform** (OS, middleware, …)
  - Composite Structure Diagrams

```
 Software Execution Platform

 t1 : AADLTask  t2 : AADLTask  t3 : AADLTask

 AADLTask
```

- **« clock »** identifies a model element that must be scheduled
  - Ordered set of instants (discrete or dense)
  - **Instants** are *points in time* at which something happens
  - Clocks are *a priori independent/concurrent*
  - For **logical clocks**, only the ordering is important, the distance between two successive instants is irrelevant

- **« clockType »** denotes the type of compatible clocks
Hardware execution platform
- Composite Structure Diagrams

MARTE-rizing it (3)
Time Specification (1)

- Aperiodic threads only

```plaintext
ds alternatesWith t1;  // asynchronous communications
t1 alternatesWith t2;
t2 alternatesWith t3;
t3 alternatesWith da;
```
Time Specification (2)

- t1 (periodic=50Hz), t2 (aperiodic), t3 (periodic=25Hz)

Clock clk is idealClk discretizedBy 0.01;  //100Hz

t1 isPeriodicOn clk period 2;  // periodicity (logical)
t1 alternatesWith t2;          // asynchronous communication
t3 isPeriodicOn t1 period 2;
t3 alternatesWith da;

coincidence
MARTE Allocations

Application

Software execution platform

Hardware execution platform
Conclusion and future work

- **AADL** provides rich means for synchronous/asynchronous computations & comm.

- Mix of **time-triggered** and **event-based** behaviors not easily understood (ex: Feiler & Hansson, CMU/SEI-2007-TN-010)

- **Marte CCSL** constraints express the time relations
  - explicitly within the model
  - in simple mathematical terms (relations between logical clocks)

- TimeSquare solver relies on a BDD solver

  "Use the BDD to explore the whole state-space (when possible: not always possible due to CCSL expressivity)"
Questions?

- TimeSquare
  - Available at http://www.inria.fr/sophia/aoste/dev/time_square/
  - Update site or standalone
MARTE Clock Relations and Constraints

- Logical clocks: time threads representing control and activation
  - not to be confused with physical devices
  - …but, they generate sequences of ticks corresponding to time events
  - strongly connected to Lee and Sangiovanni-Vincentelli tag systems
  - unrelated clocks form asynchronous systems
  - strongly related clocks may form synchronous multi-clock systems
  - GALS systems modeled in between
  - Scheduling amounts to adding (and solving) more constraints, making the system more synchronized
  - Constraints come up from execution platform or user-defined Real-Time requirements

- The complete set of relations and constraints:
  **MARTE Clock Constraint Specification Language**