AADL for Cyber-Physical Systems: Semantics and beyond, validate what's next

Jérôme Hugues, ISAE/DMIA
jerome.hugues@isae.fr
Outline

• Introduction
• Quick overview of AADL
• AADL for CPS projects
• Conclusion
Some remarks

- Rigorous method used for safety-critical applications
  Code review, testing, model checking ..
- Increasing complexity will make “bugs” happen more often
  Information scattered in design, implementation, testing plan
  Update on changes of one parameter hard to trace in all dimensions
  Make re-validation costly
- “Use engineers’ time wisely: to design spacecraft mission software and integrate it to on-board computer, not to code”, E. Conquet, ESA
  Similar concerns raised by French CEA, Thales, Astrium, Airbus, leading to big investment on MBSE, using SysML, MARTE or AADL
Multiple impact of a single design choice

- Increased confidentiality requirement
  - change of encryption policy

- Key exchange frequency changes
  - Message size increases
    - increases bandwidth utilization
    - increases power consumption

- Increased computational complexity
  - increases WCET
  - increases CPU utilization
  - increases power consumption
  - may increase latency

- RESOURCE CONSUMPTION
  - Bandwidth
  - CPU Time
  - Power Consumption

- SECURITY
  - Intrusion
  - Integrity
  - Confidentiality

- REAL-TIME PERFORMANCE
  - Deadlock/Starvation
  - Latency
  - Execution Time/Deadline
  - Confidence
AADL: Consistent Architecture & Analysis Concepts

AADL Offers!
• Domain concepts with strong semantics!
• Extensible domain model!

Safety! Analysis!
Reliability! Analysis!
Performance! Analysis!
Resource! Analysis!
Data Quality! Analysis!

Architecture Meta Model!
Error Occurrence & Propagation Behavior!
Error Model Annex!

AADL Semantic Model!
Meta model & semantic spec!
Static SW Architecture!
Packages, data, subprograms, abstract components!
Runtime Architecture!
Processes, threads, connections!
Modal runtime configurations!
Computer System & Platform!
Processor, memory, bus, device! system components!

Component & Interaction Behavior Behavior Annex

Textual AADL!
Graphical AADL!
UML Profile via MARTE!
Database Schema & Form-based Frontend!
Import via XML/XMI interchange format!
## Outline

<table>
<thead>
<tr>
<th>• Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Quick overview of AADL</td>
</tr>
<tr>
<td>• AADL for CPS projects</td>
</tr>
<tr>
<td>• Conclusion</td>
</tr>
</tbody>
</table>
• AADL design rationale is to keep the engineer’s vocabulary
  SAE (ARD 5296): help validating and generating complex systems
• Software components:

  Thread ~ data ~ subprogram ~ Threadgroup ~ process

• Hardware components:

  Device ~ Memory ~ bus ~ Processor

• Each component has its own set of legality rules
  Containment of other subcomponents, dedicated properties, etc
  Match typical engineering process from the industry
AADL provides both textual and graphical

Depend on the usage scenario
Properties usually not shows graphically

```
<category> foo!
features
  -- list of features
  -- interface
properties
  -- list of properties
  -- e.g. priority
end foo; !

<category> foo.i [extends <bar>]!
subcomponents
  -- …
calls
  -- subprogram subcomponents
connections
properties
  -- list of properties, e.g. priority
flows
end foo.i; !
```
Some Standard Properties

Dispatch_Protocol => Periodic;
Period => 100 ms;
Compute_Deadline => value (Period);
Compute_Execution_Time => 10 ms .. 20 ms;
Compute_Entrypoint => "speed_control";
Source_Text => "cruise.adb";
Source_Code_Size => 12 KB;
Thread_Swap_Execution_Time => 5 us.. 10 us;
Clock_Jitter => 5 ps;
Allowed_Message_Size => 1 KB;
Propagation_Delay => 1ps .. 2ps;
Bus_Properties::Protocols => CSMA;
Architecture Execution Semantics Defined

- Components “schedule” its subcomponents
- Nominal & recovery
- Fault handling
- Resource locking
- Mode switching
- Initialization
- Finalization

Thread Example Diagram
Controlling dispatch of events

- Timing of events, data governs the stability of the system
  - Multiple policies exist to control arrival of data:
    - Immediate, delayed, sampling
  - Typical policies from control theory, high-integrity systems
- Synchronous, asynchronous exchange of events
  - Queue size, overfull policy, urgency are defined
  - Linked to thread’s dispatch protocol
AADL & semantics

- AADL is a modeling language and set of validity rules
- AADL semantics has been defined from existing one
  To support industrial needs for validating their systems
- AADL has been demonstrated to support
  Mono-processor Synchronous semantics
  Mono-processor Ravenscar system
  FIFO within priorities, one suspension point per cycle, periodic or sporadic only behaviors, static sched. Parameters, communication through shared objects, Priority Ceiling Protocol.
  IMA-like systems -> part of AS5506/2 annexes, using v2 constructs
  MILS systems -> restrictions on flows
- One semantic /= UML profile, but set of restrictions à-la Ada
• AADL is an interesting framework to model and validate complex systems: clear syntax, semantics, low overhead
  “only” 300 pages for the core document
  Increasing number of supporting tools for validation
  MARTE standard to provide guidelines to model AADL patterns
• Scheduling analysis, resource dimensioning, behavior analysis, mapping for formal methods, fault analysis,
  Discussed in another seminar tomorrow
• Today’s focus: AADL at work for validating CPS projects
Outline

• Introduction
• Quick overview of AADL
• AADL for CPS projects
• Conclusion
AADL @ Work

• AADL: validation and generation of complex systems
  A rather vague agenda
  How complex is complex? What to validate? Generate?
• Various projects have been defined to test this claim
  Power consumption of UAV platform
  Heterogeneous modeling of space systems using TASTE
  Virtual integration (part of SAVI)
• A subset of many projects around AADL
  See https://wiki.sei.cmu.edu/aadl/index.php/Main_Page
UAV/UGV platforms at ISAE

- Variations around captors/actuators, field buses and CPUs
- Under-documented, hard to track variants and evaluate architecture trade-offs
- Test bench for AADLv2

Consistency
Power consumption
Analysis of the EMAXX2 variant

- Use of MS Vision for high-level architectural modeling
- Connection to datasheet
- Easy visualization
Modeling EMAXX

- Modular modeling
  Private/pubic, package
  “plain old software engineering”
  Property set for power consumption

- Software/Hardware view
  Reverse engineering existing
  Model has all information for electrical compatibility
  current drain
  Max/avg power consumption
Average Power Consumption

• AADL has all required information
  Software activity (period, WCET,..)
  Devices, processors charact. (peak, $I_{\text{run}}$, $I_{\text{stdby}}$, $V$)
  Power bus (additional converters)

• Use of REAL DSL
  Allow to compute on AADL models
  Implemented functions to compute average power

• Results
  Measured: **230mW**
  Model #1: **245mW** (no peak current)
  Model #2: **248mW** (with peak current)
AADL and ASN.1 are combined to provide a precise, and complete description of the system architecture and data.

**AADL** and **ASN.1** are combined to provide a precise, and complete description of the system architecture and data.

FDIR-command ::= ENUMERATED {
  safe-mode,
  switch-to-redundant,
  ...
}

AOCS-tm ::= SEQUENCE {
  attitude Attitude-ty,
  orbit Orbit-ty,
  ...
From TASTE DSL to code

Generate “application skeletons” in Simulink, SDL, C, and Ada

Generate a software real-time architecture (in AADL)

Generate glue code to put everything together on a real-time operating system
Code generation, Ocarina

**ISAE**

- AADL defines the full architecture of a system
  
  Can use it to generate tricky part
  
  Threads, buffers, driver management, handle portability, etc

  The architecture is **static**
  
  Allows for many optimizations, no need for a framework (like CORBA)
  
  Code penalty in the range of 5%

- Ocarina is a code generator from AADL to C and Ada
  
  C/RT-POSIX, C/RTEMS, C/VxWorks, C/Xenomai, Ada/Ravenscar

- Link with WCET tool: close the loop with scheduling

- TASTE demonstrates rapid system prototyping using AADL
  
  Early validation on meaningful platform
  
  Early verification using all AADL tools (model checking, resource, )
SAVI Proof Of Concept Demo

Incremental Multi-Fidelity Multi-dimensional Multi-Layered Architecture Modeling & Analysis

- Aircraft system: (Tier 1)
  - Engine, Landing Gear, Cockpit, ...
  - Weight, Electrical, Fuel, Hydraulics,
- IMA System: (Tier 2)
  - Hardware platform, software partitions
  - Power, MIPS, RAM capacity & budgets
  - End-to-end flow latency
- Subcontracted software subsystem: (Tier 3)
  - Tasks, periods, execution time
  - Software allocation, schedulability
  - Generated executables
- OEM & Subcontractor:
  - Subsystem proposal validation
  - Functional integration consistency
  - ARINC 429 protocol mappings
- Additional Opportunities:
  - Safety & security analysis
  - Fault modeling & impact analysis
  - What-if trade studies

- System & software system
- Integrator & subcontractor virtual integration

• Based on OSATEv1, large model represented
  Check videos on AADL wiki
Outline

• Introduction
• Quick overview of AADL
  AADL for CPS projects
• Conclusion
Conclusion

• Just a quick overview of AADLv2 capabilities
  Scheduling analysis, resource dimensioning, behavior analysis, mapping for formal methods, fault analysis,
  To be discussed in another seminar tomorrow

• Other projects focus on
  Virtual integration at system-level for avionic system: SAVI
  Integration of SysML/AADL: Rockwell Collins, ISAE
  Incremental modeling, reference architecture: ESA, ISAE
  Virtual upgrade V&V, modernization of aircrafts: DoD, SEI
  Academic work on full formal semantics: IRIT, INRIA, U. Illinois

• AADLv2 has enough expressive power for modeling complex systems. MDE tools can exploit models for V&V at various levels, usually limited by 3rd party tools