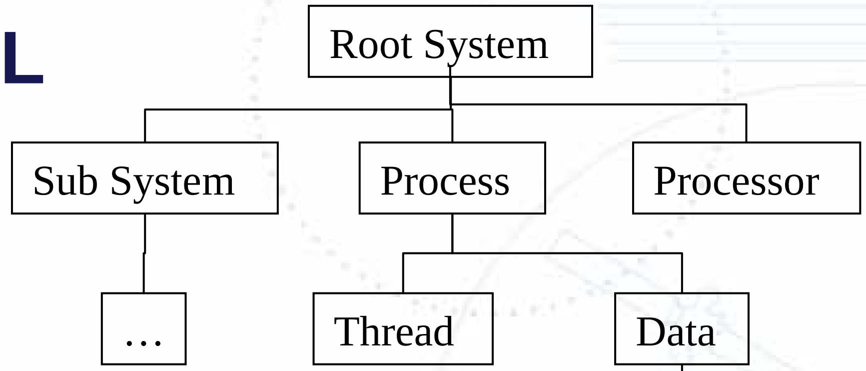
Expressing and enforcing user-defined constraints of AADL models

Olivier Gilles TelecomParisTech, Jérôme Hugues, ISAE/DMIA

About the SAE AADL



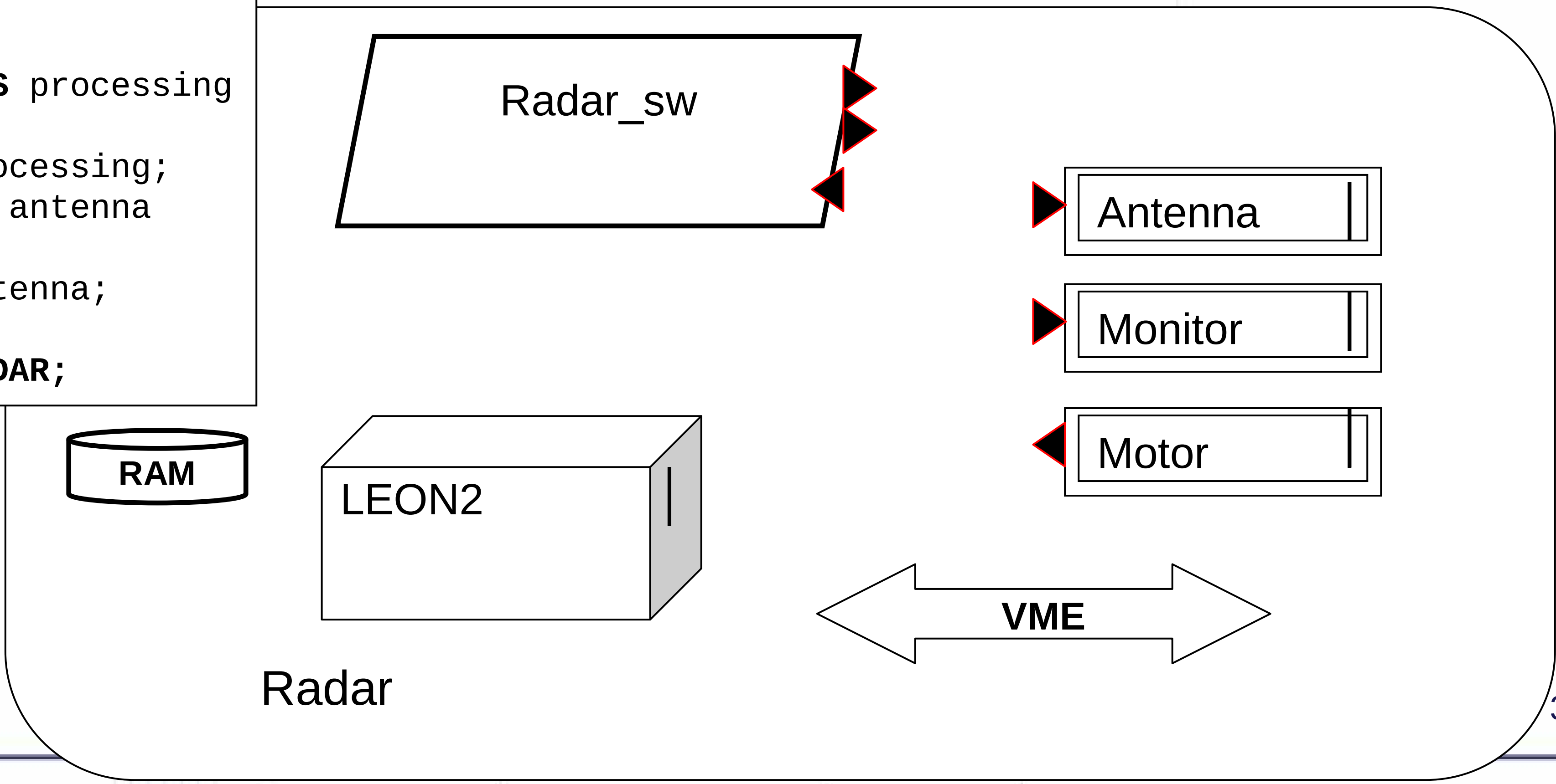
- **AADL model** :
  - A hierarchy from top-most (system) to top-down (subprograms, ..)
- **AADL components:**
  - **Component definition :** model of a software or hardware element, notion of type/interface, one or several implementations organized in package. A component implementation may have subcomponents.
  - **Component interactions :** features (part of the interface) + connections (access to data, to subprograms, ports, …)
  - **Component properties:** valued attributes to model non-functional property (priority, WCET, memory consumption, …)
- AADLv2 defines **both** textual and graphical representations
- UML/MARTE defines guidelines for modeling AADL

AADLv2 Radar example

```
PACKAGE radar
PUBLIC

PROCESS processing
-- …
END processing;
DEVICE antenna
-- …
END antenna;

END RADAR;
```

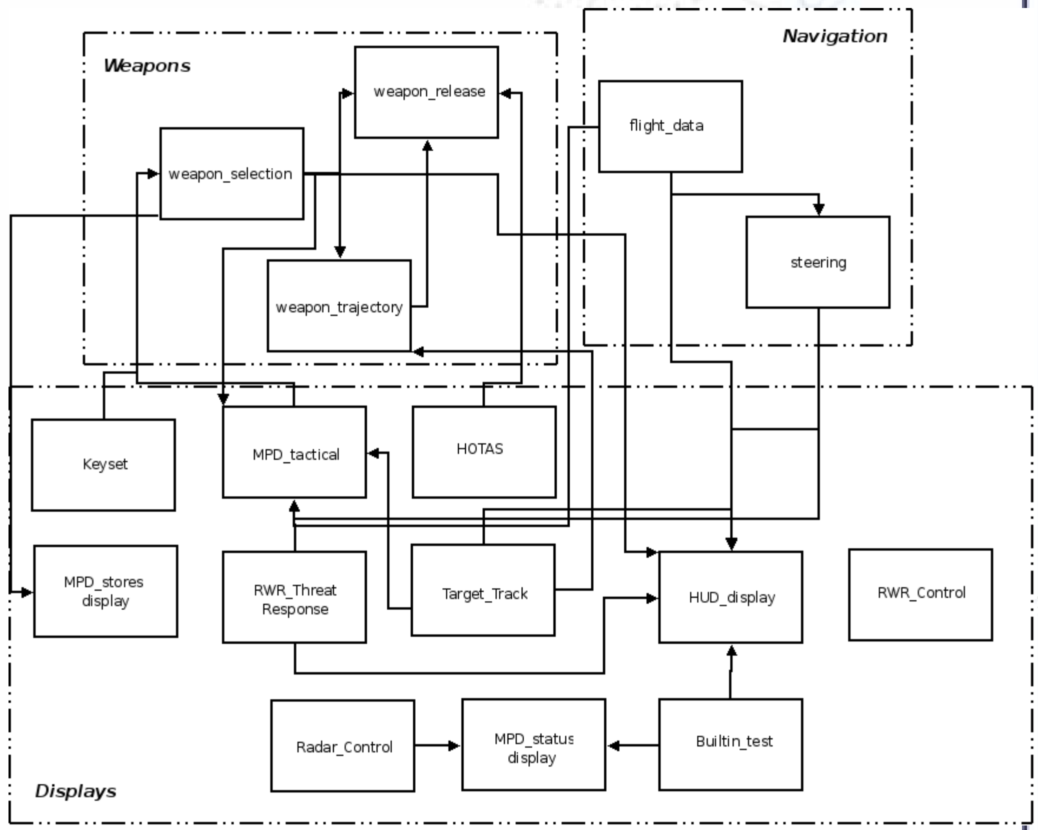


Modeling with AADL, what else ?

- AADL is an interesting framework to model and validate complex systems: clear syntax, semantics, low overhead
- **Increasing** number of supporting tools for validation
  - Scheduling analysis, resource dimensioning, behavior analysis, mapping for formal methods, fault analysis, …
  - More than 14 different projects around all kind of analysis
- But the model needs to be ***"ready"***
  - Rely on carefully chosen modeling patterns
  - Needs to constrain the value of some properties
- And we need to **validate** that the model is ready

Generic Avionics Platform case study

- Real-world AADL model can be large
- GAP models a 1980' avionics platform
- 3 subsystems
  - 14 processes, 30+ types
  - 2000 lines of AADLv2
- How to check the model is amenable to some analysis, verification or code generation ?



AADL annex documents

- Core AADL defines generic guidelines
- These guidelines are completed with domain-specific ones
  - Data modeling annex: modeling user-define types (e.g. records, arrays of records, integers, …) used by software elements
  - Programming language annex: how to bind source code (Ada, C) or other models (e.g. Simulink, SCADE) to AADL models
  - ARINC653 annex: how to express IMA and ARINC653 concepts as AADLv2 model entities
- Each guideline adds some requirements on the model
  - Valid/forbidden combination of properties
  - Validity of some combination of model entities (number of ports, kind of bus to interconnect elements,

REAL, an AADL annex for expressing constraints

- OCL is the OMG mechanism for expressing contraints on models and meta-models following the MDA principles
  - But evolves from mathematical grounds to complex expressions
    - ✓ E.g. MOF!Class.allInstances()->collect(name)
  - Not adapted to AADL, model's lifecycle, tools (except UML profile)
- **Our contribution**: REAL
- A language to express requirements on a model
  - Coupled to AADL model's definition
  - Based on mathematical grounds: set theory, expression
  - Built as a dedicated annex
- Design goal is to have REAL usable for people unaware of AADL meta-model, but knowledgeable of AADL concepts

Request Enforcement & Analysis Language: an example

- An AADLv2 legality rule
  - Each process has at least one thread subcomponent

Request Enforcement & Analysis Language: an example

- An AADLv2 legality rule
  - Each process has at least one thread subcomponent
- Steps
  - **Theorem declaration**

```
theorem no_empty_process



end  no_empty_process;
```

Request Enforcement & Analysis Language: an example

- An AADLv2 legality rule
  - Each process has at least one thread subcomponent
- Steps
  - Theorem declaration
  - **Scope of the theorem (process_set)**

theorem no_empty_process
 foreach p **in** process_set **do**

end no_empty_process;

Request Enforcement & Analysis Language: an example

- An AADLv2 legality rule
  - Each process has at least one thread subcomponent
- Steps
  - Theorem declaration
  - Scope of the theorem (process_set)
  - **Intermediate computation sets (threads_in)**

```
theorem no_empty_process
 foreach p in process_set do
  threads_in := {t in thread_set | is_subcomponent_of (t, p)};

end  no_empty_process;
```

Request Enforcement & Analysis Language: an example

- An AADLv2 legality rule
  - Each process has at least one thread subcomponent
- Steps
  - Theorem declaration
  - Scope of the theorem (process_set)
  - Intermediate computation sets (threads_in)
  - **Verification expression**

```
theorem no_empty_process
 foreach p in process_set do
  threads_in := {t in thread_set | is_subcomponent_of (t, p)};
 check (cardinal (threads_in) >= 1);
end  no_empty_process;
```

REAL by the example

- RMA

```
--  Check weither the threads bound to each processors can be schedduled
--  with RMA (cf. Liu, Layland. "Scheduling Algorithms for Multi-programming
--  in hard-Real-Time Environment", JACM, 01/1973)

theorem RMA

foreach e in Processor_Set do

  Proc_Set(e) := { x in Process_Set | Is_Bound_To (x, e) };

  Threads := { x in Thread_Set | Is_Subcomponent_Of (x, Proc_Set) };

  check (sum (get_property_value (Threads, "RTOS_properties::Utilization")) <=
         (Cardinal (Threads) * (2 ** (1 / Cardinal (Threads))) -1));

end RMA;
```

Chaining theorems

- Some theorems may depend on some others
  - E.g. Ravenscar system are RTA-compliant, PCP-compliant
  - Expressed using “requires” clause
- Theorems depend on some settings
  - E.g. apply only to one entity,
    - ✓ Use AADL annex mechanism to affect one theorem to one AADL component
- Theorems can be expressed to reflect constraints
  - E.g. subprogram “foo” must be used by a thread periodic, of 5Hz
- The objective of REAL is to be used in higher settings, e.g. evaluating model’s performance

Use case #1: data modeling annex

- Defines property sets and guidelines to model data types

```
data Target_Distance
properties
  Data_Model::Data_Representation => integer;
end Target_Distance;
```

- Requires consistent use of properties, e.g.

```
theorem check_data_scale
  foreach d in data_set do

  --  1/ Check that the "Data_Scale" property is applied only to data
  --     type whose representation is fixed
  check ((not property_exists (d, "data_model::data_scale"))
         or (property_exists (d, "data_model::data_representation")
             and get_property_value (d, "data_model::data_representation") =
                       "fixed"));
end check_data_scale;
```

- 15 theorems to check various legality rules

Use case #2: Ravenscar profile

- Set of patterns for deterministic concurrency
  - Initially defined for Ada, expanded to RTSJ, UML and AADL
- Need to constraint architecture to accept only these patterns
  - Periodic/sporadic task, mono processor, use of PCP, …
  - 6 different rules

```
--  All *shared* components use the PCP concurrency control protocol

theorem check_pcp

foreach d in Data_Set do

   accessor_threads := {t in Thread_Set | Is_Accessing_To (t, d)};
   check (Cardinal (accessor_threads) <= 1 or
          (Property_Exists (d, "Concurrency_Control_Protocol") and
             (Get_Property_Value (d, "Concurrency_Control_Protocol") =
              "Priority_Ceiling")));
end check_pcp;
```

Use case #3: ARINC653 annex

- AADLv2 and ARINC653 annex support IMA concepts
  - Notion of partitions, hierarchical scheduler ..
  - Needs to constraint models to respect some invariants
  - 10+ additional legality rules

```
--  Check configuration of partition scheduling
theorem scheduling_major_frame
    foreach cpu in processor_set do
    check ((property_exists(cpu, "POK::Major_Frame") and
            ((float (property (cpu, "POK::Major_Frame")) =
            sum (property (cpu, "POK::Slots")))))
          or
         (float (property (cpu, "ARINC653::Module_Major_Frame")) =
          sum (property (cpu, "ARINC653::Partition_Slots"))));
end scheduling_major_frame;
```

- Integrated to the POK toolchain (see http://pok.gunnm.org)

Conclusion

- REAL has been integrated to Ocarina and POK
  - http://aadl.telecom-paristech.fr
- Ocarina is also integrated to OSATE, STOOD
  - Available to major AADL modeling environments
- Implementation relies on AADL instance model
  - Time and memory effective: most operations have the same complexity as of the theorem to be proved
- Ongoing work
  - Evaluate model's metrics
    - E.g. evaluate performance of models based on computations
    - Weight, power consumption, specific policies ..