AADL Lite Requirements and early definition

Initial statement

- **AADL has evolved into a large language**
  - » Prototypes, arrays, renames, extension/refinement capabilities
  - » Large property set with additional semantics
    - E.g. matching rules, connection patterns, Implemented_as
  - » Several property sets, but usage depend on the analysis
    - Scheduling? Code generation? Model checking? Safety? Security?
    - Must clarify usage of each, part of AADL3.0 effort
  - » E.g. AADLv2.2 has this, is it redundant with ARINC653 annex?

```
Time_Slot: list of aadlinteger
    applies to (thread, thread group, process, virtual processor, system);
Slot_Time: Time
    applies to (processor, virtual processor);
Frame_Period: Time applies to (processor, virtual processor);
```

- Non AADL experts are puzzled, to say the least

General roadmap

> **Diagnostic: AADL has evolved into a large language**

> **Regular concerns about**

» Coverage of the language when writing AADL analysis

» Fuzziness of the language in some aspects (e.g. arrays, modes)

» Patterns for using a given analysis (see discussion on subsets)

> **General objective**

» Leverage AADL Constraints Language to see how one can define subsets in an efficient way

» One candidate: AADL-Lite

- Later work for other existing subsets (ARINC653, Ravenscar, synchronous, etc…)

Rationale for AADL-Lite

- **Reduce complexity of the AADL language**
  - To make it easier to write tools that can be **qualified**
  - Backwards compatible with AADL, by Appendix G
- **Already in place in G. Lasnier PhD thesis, and RAMSES**
  - Models as a result of AADL to AADL normalization, simplification
- **Key idea: leverage AADL core, BA and Resolute**
  - Core to provide basic constructs, property set
  - BA to specify semantics coming from properties
  - Resolute to mimic properties that are actually contracts on model
- **Rule#1: clarify the usage of every construct and property**
  - If you cannot give one use case, forbid it

Roadmap / List of actions

1. **Define set of restrictions on AADL language grammar**
   - » Option#1: textual definition + Resolute checks
   - » Option#2: suppress corresponding definition from BNF for tools

2. **Review property sets**
   - » Embed documentation in .aadl files
   - » Review usage of each property w.r.t. other annexes
   - » Review usage of each property w.r.t. toolsets

3. **Stay compatible with AADL as objective, but also transition:**
   - » Plug-in to go from AADL to AADL-Lite
     - Notionally equivalent to G. Lasnier thesis, and RAMSES

Option#1: definition + resolute checks

- **Similar to pragma Restrictions from Ada**
  - Configure your modeling environment, e.g. OSATE
  - OSATE + plug-ins checks the model conforms to restrictions
  - The model can then be passed *safely* to other tools
- **Resolute checks available for**
  - All features connected, no prototypes, no arrays, no feature group, no abstract features, no subprogram access on threads (aka rendez-vous), no thread group, no subprogram group
- **Option#2: Limitation on the BNF is not necessary, a tool may implement it the way it wants.**
  - Would be a mess to define anyway
  - Rule#1 is that a regular AADL model processor has checked compliance before

Review of property sets

> **Current limitation: no human readable text with properties**
  - » Matter of formatting, provide an basic level of documentation

> **About properties**
  - » Rule of thumb: “If I don’t know how to use it (from V&V, code gen.), then disallow it”

> **Examples**
  - » *Source_Text -> do we need them ?
  - » Also, some can be replaced with Resolute checks
    - • E.g. Allowed_*_Binding

Full list

> **List of properties not necessary from Ocarina/Code generation perspective**

- » Implemented_As, Prototype_Substitution_Rule, Acceptable_Array_Size,
- » Hardware_Source_Language, Hardware_Description_Source_Text,
- » Finalize_Entrypoint, Finalize_Entrypoint_Call_Sequence, Finalize_Entrypoint_Source_Text, Deactivate_Entrypoint_Source_Text, Deactivate_Entrypoint_Call_Sequence, Deactivate_Entrypoint, Activate_Entrypoint_Source_Text, Initialize_Entrypoint_Source_Text, Recover_Entrypoint_Source_Text, Activate_Entrypoint, Activate_Entrypoint_Call_Sequence, Activate_Entrypoint_Source_Text, Write_Time,Read_Time,Access_Time,
- » Subprogram_Call_Rate, Transmission_Type, Required_Connection,
- » Slot_Time, Execution_Time, Frame_Period, Client_Subprogram_Execution_Time,
- » Synchronized_Component, Subprogram_Call_Type, Runtime_Protection, Deactivation_Policy, Active_Thread_Queue_Handling_Protocol, Active_Thread_Handling_Protocol, Resumption_Policy, Mode_Transition_Response, Time_Slot, Dispatch_Able,
- » Priority_Map, Not_Collocated, Collocated, Allowed_Memory_Binding_Class, Allowed_Processor_Binding_Class, Allowed_Processor_Binding, Allowed_Memory_Binding, Allowed_Connection_Binding_Class, Allowed_Connection_Binding, Allowed_Subprogram_Call, Allowed_Subprogram_Call_Binding, Provided_Virtual_Bus_Class, Required_Virtual_Bus_Class, Provided_Connection_Quality_Of_Service, Required_Connection_Quality_Of_Service, Allowed_Connection_Type, Allowed_Dispatch_Protocol, Allowed_Period, Allowed_Physical_Access_Class, Allowed_Physical_Access, Thread_Limit