Open Errata AADL V2.1 (some to become V3.0 issues)

Feb 2016

See also www.github.com/saeaadl/aadlv2.1/issues

Table of Contents

8.2 Feature Groups and Types

Inverse of in Feature Group Types

Issue: In AADL V2 we added the ability to specify **inverse of** in the feature group declaration. We still kept the inverse of in the feature group type. On one hand the inverse of still allows features to be explicitly listed. On the other hand there are special restriction to reduce the complexity of **inverse of** regarding **extends**. One of those rule is rather restrictive (an extension must include an inverse of if the original does).

Proposed correction: Eliminate **inverse of** on feature group type. **Inverse of** on feature groups gives the modeler enough expressive power.

Status: Originally discussed 4/13 (Accepted for V3). In AADL V3 see interface composition. Logged in Github.

8.4 Subprograms & Groups

Subprogram and Subprogram Group Ports

Section 5.2 Legality rules only allow out event and out event data ports in subprogram types. Section 5.3 Legality rules do not allow any port features in subprogram groups. It is not possible to declare that a subprogram or subprogram group contains a call to an RTOS API to send/receive messages through a port. This also prevents modeling a FACE Software Unit of Portability as an AADL subprogram group, which would be the most intuitive mapping of a UoP to AADL.

Proposed correction: Allow port features to be declared in subprogram and subprogram group types without restriction. This is backwards-compatible. The rules for connections to ports of subprograms and subprogram groups that are not nested within threads require some thought. Push and pull connections to in and out data ports of subprogram components of processes, and subprograms accessed by and called from multiple threads, have meaningful semantics.

Status: Reopen 2/2016. Logged on Github. [Closed 7/14. It seems that the desired effect of in ports on subprograms and subprogram groups should be passing the port itself as a parameter. Calls to send or receive data from ports are expressed by Send_Output and Receive_Input calls. They take as parameter an identification of the port that is implementation specific, e.g., by port name as string.]

Feb 3, 2016: Recommendation for reopening the issue.

In a follow-up discussion with Steve and with Lutz I came to the conclusion that we reopen the issue as an errata.

Here is the reason:
We already support outgoing ports on subprograms. The interpretation is that effectively a subprogram can perform a send on the port.
For incoming ports we would connect the port to a parameter. In this case, the port content has to be frozen and read before the call (parameters are pass by value). By supporting incoming ports as part of the subprogram signature we allow users to specify that a port is being read from within a subprogram after it has been called.

9 Connections

9.2.3 Connection sets for Component Arrays and Feature Arrays

Issue: The connection set declarations are difficult to specify and read.

Proposed correction: Replace the connection set mechanism by indexing array elements in connection declarations.

Status: In discussion for AADL V3. Logged in Github.

9.6 Align Classifier_Substitution_Rule and Classifier_Matching_Rule

Issue: The type compatibility rules between refinement and connection are dissimilar enough to possibly cause some surprises. The following example illustrates this issue. The proposal would be to allow values of Type_Extension and Signature_Match for Classifier_Matching_Rule, at least for certain kinds of connections. (People aren't expected to edit AADL_Project and customize Supported_Classifier_Equivalence_Matches for individual projects, are they?)

```
package ExtendTypeNotImp
public

        bus ParentCable
        properties
                Classifier_Substitution_Rule => Type_Extension;
        end ParentCable;

        bus ChildCable extends ParentCable
        properties
                Classifier_Substitution_Rule => Type_Extension;
        end ChildCable;

        bus implementation ChildCable.Imp
        properties
                Classifier_Substitution_Rule => Type_Extension;
        end ChildCable.Imp;

        system Network
        features
                p0: provides bus access ChildCable.Imp;
        end Network;
        system PFeatures
        features
                p0: requires bus access ParentCable;
        end PFeatures;

        system CFeatures extends PFeatures
        features
                p0: refined to requires bus access ChildCable.Imp;
        end CFeatures;

        system Test
        end Test;

        system implementation Test.Imp
        subcomponents
                net: system Network;
                parComp: system PFeatures;
                childComp: system CFeatures;
        connections
                cc: bus access net.p0 -> childComp.p0;
                -- The following connection has a type mismatch.
                cp: bus access net.p0 -> parComp.p0;
        end Test.Imp;

end ExtendTypeNotImp;
```

Status: Open. For V3. Logged in Github.

Allow connection refinements to refine source or destination

Issue: It is a useful capability to extend an implementation and then refine some of its connections so they route through new subcomponents added in the extension. For example, a connection between components A and B might be refined by re-routing from A to a new Filter subcomponent and then adding a connection from that Filter to B. A more elaborate version of this would be to allow a connection to be refined to a flow as long as the flow source and destination denote the same features as the refined connection. There also appear to be some inconsistencies in the standard. Section 9.1 (N1) and section 9.2 (N1) talk about connection refinement referencing the source or destinations of the refined declaration, but the grammar does not seem to permit any reference to the actual feature-to-feature connection part. A comment in the example of section 9.5 says feature group connection refinements may only add properties and that the actual feature-to-feature connection part does not have to be repeated. The grammar does not permit a repeat. The rules for all connections should be as similar as possible. There seems no reason to permit this for port connections but not others (the grammar appears to forbid it even for port connections, although the textual descriptions could be interpreted to imply that it is permitted for port connections).

```
-- Illustrates several cases of connection refinements that currently
-- generate errors in OSATE2 but seem very useful and in at least one
-- case possibly legal.

-- The one in system implementation Sys.Refine2 might be an OSATE2
-- bug, or it might be that the standard is unclear about what is
-- legal.  (This error message masks subsequent error messages,
-- some commenting-out of previous errors is needed to see subsequent
-- error messages.)

-- The others are reasonably or clearly illegal in the current AADL
-- standard, but they seem like errata (things that are meaningful
-- and desirable in use).

package RefineConnect
public

	system SysPort
		features
			outp: out data port;
			inp: in data port;
	end SysPort;

	feature group FG
	end FG;

	system SysGroup
		features
			outg: feature group FG;
			ing: feature group inverse of FG;
	end SysGroup;

	system Sys
	end Sys;

	system implementation Sys.Parent
	subcomponents
		sysAP: system SysPort;
		sysBP: system SysPort;
		sysAG: system SysGroup;
		sysBG: system SysGroup;
	connections
		c_abp: port sysAP.outp -> sysBP.inp;
		c_abg: feature group sysAG.outg <-> sysBG.ing;
	end Sys.Parent;

	system implementation Sys.Child extends Sys.Parent
	subcomponents
		sysCP: system SysPort;
		sysCG: system SysGroup;
	end Sys.Child;

	system implementation Sys.Refine1 extends Sys.Child
		connections
			-- Accepted by OSATE2
			c_abp: refined to port {Latency => 200ms .. 200ms;};
	end Sys.Refine1;

	system implementation Sys.Refine2 extends Sys.Child
		connections
			-- Standard grammar seems to rule this out,
			-- but the standard text refers to including
			-- source and destination elements.  This may
```

```
                        -- be an inconsistency in the standard or an
                        -- OSATE bug.
                    c_abp: refined to port sysAP.outp <-> sysBP.inp;
        end Sys.Refine2;

-- The above error masks error reports in the following
-- declarations.  To see the additional errors, comment
-- out the preceding declarations that generated errors.

        system implementation Sys.Refine3 extends Sys.Child
                connections
                        -- I couldn't find anything in the standard forbidding
                        -- this, and this form of refinement (to
                        -- change one end of the connection) is highly
                        -- desirable in practice.  If the standard
                        -- forbids this, that should be an errata.
                    c_abp: refined to port sysAP.outp -> sysCP.inp;
        end Sys.Refine3;

        system implementation Sys.Refine4 extends Sys.Child
                connections
                        -- The standard explicitly forbids this for
                        -- feature group connections, but it is
                        -- as desirable and meaningful as the port
                        -- refinement above.  There should also be as
                        -- much symmetry as possible between port connections
                        -- and feature group connections. This should
                        -- be an errata.
                    c_abg: refined to port sysAG.outg <-> sysCG.ing;
        end Sys.Refine4;

end RefineConnect;
```

Status: Open. Logged in Github.

11.1 Property Sets

11.1.1 Property Types: Predeclared Units

Issue: The standards supports the definition of units for numeric values. Users can define conversion factors between units of the same unit type. It would be useful to provide a specification of unit mappings for certain arithmetic operators. For example, 20 m / 2 sec becomes 10 mpersec.

Proposed correction: Someone develop a proposal for an annex to provide such a mapping, i.e., a specification of derived units. We should leverage the International System of Units.

Status: draft document completed by Alexey. Will be incorporated into properties under AADL V3.

11.1.2 Property Definitions

using list and set

Issue: as of today, we can have a list for a property. A list is an ordered set. The idea would have to add the possibility to define a set instead of a list. We would then have a property that can be either a list or a set.

Proposal: introduce the concept of a set with the concept of a list in AADL (i.e. to allow 'set of' anywhere 'list of' is allowed by AADL Grammar).

Status: To be included in AADL v3.

A.1 Deployment Properties

Bindings property

Issue: the binding concept is currently expressed as contained property association.

Proposed change: introduce a separate binding concept.

Status In discussion AADL V3

Priority, Priority_Range, Preemptive_Scheduler

Issue: Messages on buses are also scheduled, not just threads on processors. Some schedulers and schedulability analysis tools allow preemptive and nonpreemptive threads and messages to be mixed on the same processor and bus. That is, preemptability may be specified on a task-by-task basis rather than a resource-by-resource basis. The standard never specifies whether numerically higher Priority values mean lower or higher scheduling priority.

Proposed correction: Make all these inherit properties. Allow Priority to apply to connection and virtual bus. Allow Priority_Range to apply to bus, virtual bus, and system. Allow Preemptive_Scheduler to apply to system, virtual processor, bus, virtual bus, connection, thread, and thread group; and override any value specified for the resource itself (so the resource property specifies the default). Specify the default semantics for Priority values, whether numerically larger means higher or lower scheduling priorities. (The classical literature typically uses the latter, but many tools and RTOSes use the former. As a practical matter I suggest numerically higher means higher priority as a default. An option is to introduce a new property to reverse this default, which might be more concise than filling in all the details of a Priority_Map.)

Status: Open

A.2 Thread Properties

Time_Slot property

Issue: The time slot is a list of integers that designated which time slot is assigned. But the time slots are not described. The semantics is ambiguous.

Proposed correction: Remove it

Status: Open.

Duration property

Issue: Currently periodic threads are assumed to run forever. They may have a limited duration.

Proposed correction: Add Duration property.

Status: Open.

Synchronization Domains and Reference_Time

Issue: Section 5.4.7 Asynchronous Systems talks about a pre-declared device Reference_Time and a pre-declared property Reference_Time that allows users to specify common time bases for sets of platform resources. A search of both the standard and OSATE and an attempt to create a device whose type is Reference_Time indicate these are not actually supported. When they are, care should be taken that the naming capabilities are the same as those of the binding properties and allow a reference to a Reference_Time object to be associated with a component that is arbitrarily deeply nested in the design hierarchy. Also, it may be useful to simply allow any processor or device or system to be named as a time reference, not just devices having a specific predeclared type. For example, it might be convenient to simply declare a Reference_Time property in a system that names that system itself, meaning all sub-components of that system are in the same synchronization domain.

Proposed Correction:

Status: discussed. Logged in Github.

A.7 Modeling Properties

Implemented_As

Issue: semantics of the property is not well defined. We would like to define what this property intends to do

Proposal: either it would drive the instantiation process and requires some additional consistency checks, either it would be a dedicated mechanism. Need to decide if this is a small change or should be address in AADLv2.2. The component having Implemented_As must not have ANY subcomponent. Might be also implemented using a better refinement mechanism.

Potential Legality rules:

- Instantiated component has the type of the referenced component
- Instantiated component contain the features from the initial and referenced components. If both have the same features declaration, the one from the specified component overrides the one from the type.
- Instantiated component contain the properties from both initial and referenced components. If some properties cannot be applies to the destination component type (legality rules), they are discarded.
- If the same property is defined on both components, properties from the destination override the one from the source

Also, this mechanism might be implemented by extending/improving the refinement mechanism.

Status: discussion AADL V3

A.8 Project-specific property set

Issue: tools have different values for the AADL_Project property file

Proposal: define a set of common properties for easing tool interoperability. Need to potentially define a tool-specific mechanism to extend the values.

Status: in V3 roadmap