

AADL V3 Property Language

Peter Feiler

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2019 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM19-0625



Property Definitions

Define in packages

Utilize unified type system

- No more **aadlinteger**, ...
- Record, list, set, map
- Union of types:
- Integration of proposed Units system (ISO, SysML)

Identify assignment targets (V2 **applies to**)

- No need to list enclosing categories for **inherit**
- Component categories
- Specific classifiers
- Other model elements
- Use type system to express model element types, classifiers
- **property** mine : **int** for (feature);

Property Association

- Property reference always with #

```
process interface LocatorProcess
is
#Period => 20;
end;
```

- Properties on classifier elements
 - Directly attached
 - Via model element reference (aka contained property association)

```
interface subsub
is
  p1 : in port signal ;
  p2 : in port date { #Size => 3; };

  p1#Size => 3;
end;
```



Property Values

Property value can be overridden many times in V2

- As part of definition
- Inherited from enclosing component
- Inherited from interface (ancestor)
- Inherited from implementation (ancestor)
- Inherited from subcomponent definition
- Multiple layers of contained property associations



Property Values in V3

Assignment in interface or implementation

- modifiable assignment ==>
- final assignment =

Value determination potential options

- Property on component
 - Interface and interface extension (#Size ==> 1 on classifier)
 - Implementation, implementation extension (#Size ==> 1 on classifier)
 - Configuration, configuration extension (#Size ==> 1 on classifier)
 - Local subcomponent({ #Size ==> 2; } and sub1#Size ==> 2;)
 - Configuration assignment (#Size ==> 1 on classifier)
 - Configuration assignment nested {#Size ==> 2; }
 - Contained property association outer overrides inner (reach down)
- Values on model elements (features, connections, etc)
 - Local ({ #Size ==> 2; } and feat1#Size ==> 2;)
 - Interface and interface extension
 - Implementation and implementation extension
 - Configuration and configuration extension
 - Configuration assignment (feat1#Size ==> 1 in classifier)
 - Contained property association outer overrides inner (reach down)



Property Values in V3

Simple approach

- Component and element properties specified as part of “spec”
 - Defined through classifiers during design
 - #P for instances of classifier
 - {} or identifier#P for all directly contained model elements except subcomponent.
 - Override rules according to extends hierarchy of interface, implementation
- Configurations finalize a design
 - Final property value assignment in configuration specification
 - Reach down property associations in implementations
 - Containment with first element a subcomponent
- Single final assignment vs. multiple same value ok?



Property Values in V3

Conflicting assignment

- Composition of interfaces
 - #P assigned values: only one or must be the same value
- Multiple assignment through reach down & multiple configurations
 - Assignment is final: only one or must be the same value



Scoped Default Property Values in V3

V3: Scoped value assignment

- #Period \Rightarrow 20ms;
- Scope of configuration, implementation, or interface with assignment
- Used if no value assigned explicitly for contained model element
- Replaces **inherit** in V2



Property Association in Annexes

Syntax in context of an annex

- FailStop#Ocurence => 2.3e-4;
- ^Process[1].thread2@Failstop#Occurrence => 2.3e-5;
 - ^ escape to core model as context
 - @ enter same annex type as original
 - @(BA) enter specified annex: if we have annex specific properties in the annex rather than core we may not need this
 - [x] array index

Mode specific property value assignment #8

- Currently: => 2.3e-5 in **modes (m1)**, 2.4e-4 in modes (m2);
- => { m1 => 2.3 , m2 => 2.4 };
- Event#Occurrence.m1 =>
- See also error type specific property value and binding specific value
 - Use map type: mode, error type, binding target as key
 - Syntax for identifying map key in path (.)



Property Set

Definition of property set

- List of property references
- Property set can be listed as element of a set
- Same property reference can be in multiple sets

```
Periodic : properties {  
  Dispatch_Protocol => constant Periodic,  
  Period, Deadline, Execution_time  
};
```

```
GPSPProperties : properties {  
  Period, GPSPPropertyset::Sensitivity,  
  GPSPPropertyset::Hardening  
};
```

Usage: Analysis specific property set

- Must be present for analysis
- Analysis supporting multiple fidelities
 - Minimum, maximum set
 - Precondition vs. validation