

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.

DM19-0166

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

Property Profile

Constraints between properties in profile
Literal value specific sub-profile

Definition of property profile

- List of property references that are part of a profile
- Other profiles can be listed in a profile
- Same property reference can be in multiple profiles

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

```
GPSProperties : properties {  
  Period, GPSPropertyset::Sensitivity,  
  GPSPropertyset::Hardening  
};
```

Usage

- Classifier specific property profile
- Profile assignment to classifier
 - Multiple configuration assignments
 - Unnamed profile
- Analysis specific property profile

```
device GPS  
  use properties GPSProperties;  
End GPS;
```

```
MyPackage::GPS => properties  
  #SecurityLevel, #SafetyLevel;
```

Property Profiles for Model Elements

- Identification of model element “type”
 - By key word
 - By Meta model element name
 - By enumeration type for core and each annex
 - Union of enumeration subtypes
- Granularity of model elements
 - Component categories
 - Feature categories
 - Association categories
 - Flow specifications
- Usage
 - Property definition
 - Profile assignment

```
property Period : applies to Thread;
```

```
Thread => properties #Period, #Deadline;
```

Property Association

- Property reference always with #

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

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

```
process interface subsub |
features
  p1 : port date ;
  p2 : port date { #Size => 3; };
properties
  p1#Size => 3;
end ;
```

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 (.)
 - One value multiple modes?

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

Property value assignment in design space

- Assignment in interface or implementation
- Value override
 - in interface extension
 - Implementation, implementation extension

Property value assignment in configuration

- Assign only if not previously assigned
- At most once via configuration

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