

AADL V3 Property Language

Peter Feiler

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2018 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.

DM18-0659

Property Language

(Property) types (unified type system)

- Distinguish between types to be used in model specification (BA, constraint, property values) and types used as application data types
- No more **aadlinteger**, ...
- Record: map can represent record
 - Require all fields?
 - Require naming of fields in assignment or assume ordering?
- Lists (sequence), sets, bag (multiset), map for properties
 - Explicit types: on value assignment same syntax for lists/sets of values
 - Map use case: key based value
- Union of types:
 - for application types
 - For properties?:
 - e.g., compute entry point
 - Value: invalid or actual value
- Intersection of types?
- Integration of proposed Units system (ISO, SysML)

Property Language

Property set

- Place property definitions in packages
- Name path (dot) for property sets
- Alias for properties: consider
 - We have relabeled source code size to code size
 - FASTAR defined properties independently and then wanted to align
 - Subset definition: with and without different name

Property Definition

Identify scope of application (**applies to**)

- No need to list enclosing categories for **inherit**
 - Known from standard
 - Need for inherit (use pattern notation?) see other slide
- Component categories, etc
 - Meta model elements
- No user defined classifiers
 - This creates dependency of property types on user defined packages
 - handled via stereotype
- No default value as part of definition
 - Scoped defaults via inherited property values
 - All properties can be inherited
 - Leave it in place: but no reference to another property, but allow reference to constant
 - Default value for record fields: assign whole record
- Lists, sets: append

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)
 - P1 => { #prop1 => 2; #prop2 => 3;}

```
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

- \wedge Process1.thread2@Failstop#Occurrence \Rightarrow 2.3e-5;
 - \wedge 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

Syntax in context of EMV2 annex

- \wedge Process1.thread2@Failure{Overheated}#Occurrence \Rightarrow 2.3e-5;
 - {} syntax for types in EMV2

Mode specific property value assignment #8

- Currently with **in modes** specification
 - Mode may be for an enclosing component
 - Include mode identifier as part of path (m1, m2)
 - Similar to array element as part of path [idx1]

Property Values

Scoped default value

- Inherit of property value from enclosing component
 - All properties potentially inherit
 - Inherit vs. wildcards in property associations #11
 - Proc.*#Period => 20 ms; vs. #Period on proc inherited by contained threads
 - Pattern notation for assignment (see configuration)

Value in terms of another property: Needed? no

- Use example: Deadline => Period;

Final property value

- Explicit: **constant** tagging in assignment
- Implicit: via configuration

Need for classifier or model element references?

Compute values

Property Applicability

Specification of which properties apply to a component

- “Stereo” type, “property set”
- Stereo type identifies a set of property definitions

Do we want to use the keyword **stereotype**?

- May or may not include a (default) property value
 - Gets associated with component classifier
- Use configuration mechanism to attach threadprops stereotype to all thread classifiers.?best way?

```
ThreadProperties : properties {  
    Dispatch_Protocol, Period, Deadline, Execution_time  
};
```

```
Dispatch_Protocol: union (Periodic, Aperiodic, );  
Periodic : record {  
    Period, Deadline, Execution_time  
};
```

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

Records will lead us to specify values multiple times. The stereo allows property in several stereo but as a single property.

```
device GPS  
use properties Periodic;  
End GPS;
```

A set of properties with values:
property value configuration

Property Applicability

Specification of which properties apply to a component

- Component can have multiple associated stereo types
 - Property definition reference in multiple stereo types is acceptable (without conflicting values)

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

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

Do we need collections of collections?

```
device GPS  
    use properties GPSPROPERTYSET::Sensitivity,  
    Periodic;  
End GPS;
```

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

Property Applicability

Use in precondition specification for analyses

- Component categories/classifiers that are expected to have a value for a given set of properties