

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.

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

DM18-0077

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

- Name path (dot) for property sets
- Nested property sets (nesting hierarchy according to name path)
- Do we need a separate property set or allow property definitions in packages? Ok.
- Alias for properties:
 - We have relabeled source code size to code size
 - FASTAR defined properties independently and then wanted to align

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

Property Association

Property association syntax

- Harmonization of syntax with use in expressions (BA, Constraint)
- #Period => 20 ms;
 - Allows for elimination of **properties** section label
- Process1.thread2#Deadline => 10 ms;
 - Replaces **applies to** declaration
 - Deadline => 10 ms **applies to** Process1.thread2;

Applies to allows a list of targets. How to do that in new notation? Pattern, list of items before #

Property Association in Annexes

Syntax in context of an annex

- $\wedge \text{Process1.thread2@Failstop\#Occurrence} \Rightarrow 2.3\text{e-}5;$
 - \wedge escape to core model as context
 - $@$ enter same annex type as original
 - $@(\text{BA})$ enter specified annex

Syntax in context of EMV2 annex

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

Property Values

Scoped default value

- Inherit of property value from enclosing component
 - All properties potentially inherit
- Interaction with configuration
 - Pattern notation for assignment

Value in terms of another property: Needed?

- Use example: Deadline => Period;

Final property value

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

Need for classifier or model element references?

Property Applicability

Specification of which properties apply to a component

- “Stereo” type, “property set”
- Stereo type identifies a set of property definitions
 - May or may not include a (default) property value
 - Gets associated with component classifier
- Component can have multiple associated stereo types
 - Property definition reference in multiple stereo types is acceptable (without conflicting values)

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

```
device GPS  
  with GPSProperties, Periodic;  
End GPS;
```

```
device GPS  
  with Periodic, GPSPropertyset::Sensitivity,  
  GPSPropertyset::Hardening;  
End GPS;
```

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

Use in specification of what properties must have values for an analysis.

Stereo : type specific values: e.g., estimated, measured exec time