AADL V3 Property Language

Peter Feiler

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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 peoperties 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

- ^Process1.thread2@Failstop#Occurrence => 2.3e-5;
  - ^ escape to core model as context
  - @ enter same annex type as original
  - @(BA) enter specified annex

Syntax in context of EMV2 annex

- ^Process1.thread2@Failure{Overheated}#Occurrence => 2.3e-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
};
```

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

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

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

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

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