

AADL v3 Features

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.

DM18-0075

Features in AADL

Interaction points between components

In V2 (interaction within layer via connection)

- Data, event data, event port
 - No way of specifying just port
- Access: data, bus, subprogram, subprogram group
 - Target can be component itself
 - Direction of connection: target component vs. information flow
- Feature group
- Generic feature
 - Refinable to any of the other feature categories
 - Usable as generic interaction point with type indicating need for matching

In EMV2

- Propagation point (observable feature)

Request for

- Physical feature

Coming in V3

- Binding points (interaction across layers)

Generic Features

Feature

- Directional, non-directional (bi-directional?)

Need for ability to refine feature category?

- Generic feature with bus type can only represent bus access
- Same with subprogram and subprogram group
- Refine to binding point?
- Data interchange:
 - Deferred decision as to whether port based or by shared data?

All features:
Connectivity only feature to feature not feature
to component.

Ports as Features

Ports

- Exchange of discrete logical entities
- Exchanged entity with or without data
- Requires binding to transport mechanism (bus, virtual bus)
- Queuing at receiving end
- Sampled processing
 - Periodic recipient
 - Queue size 1 for single sample
 - Queue size n for down sampling by processing all in queue
- Message driven processing
 - Aperiodic recipient
 - Queue size of n
- Directional
 - Need for bi-directional?

Proposed syntax

- `<id> : (in | out) port <type>? ;`
- Properties for queue size and dequeuing policy as exists

Physical Features

Do we need physical features?

Devices represent the interface to the physical world. Digital sampling of physical world measures is represented by ports. The continuous time physical behavior may be modeled in Modelica, thus, a reference to this model may be sufficient. Alternatively, a property can specify appropriate equations.

Physical features

- Continuous time physical characteristic of a component
 - May be directional (flow) or non-directional (temperature)
- Exchanged with other components
 - Air temperature available as continuous input to a sensor device
 - Flow volume of water, electricity
- Type associated with feature represents physical characteristic of interest

This is similar to resource characteristics on binding points.

Proposal: use generic feature construct

- `<id> : (in | out)? feature <physical type>? ;`
- Type definition includes measure type with unit and optionally dynamic characterization of that measure (diff equation)

Access Features

Access features

- Access to shared resource
- Requires coordination protocol

Can we simplify access features?
Do we need to distinguish between data, bus, subprogram or
is this reflected to the classifier?

Access to data component

- Read/write information flow may be different from direction of user of data (requires) to data component (provides)
- Access means any time during execution a component can w/r multiple times
- Synchronization by participation of data user
- Ability to interconnect ports and data access
 - Agent manages w/r patterns according to port semantics (dispatch/completion)

Access to subprogram

- Remote service that can be used
- Directionality from caller to service (requires to provides)
- Call instance is expressed in call sequence on requires access
- Service itself determines single/multi threaded behavior

Access to subprogram group

- Collection of remote services

Access to buses

- Physical connectivity between hardware components
- May or may not have a directional flow associated
- Is it access to bus or connectivity to bus (generic feature)?

Requires/provides without access
keyword?

In binding features the type indicates the
resource type while in case of access it is
the type of the target component

Binding Point Specification

Explicit in features section:

- Directional features to be used as source or target of binding
- Identify type of binding (type may represent resource or target type)
- Properties related to (resource) type: capacity/budget, other target characteristics

thread task1

features

“of” constraint on container of resource

RequiredCycles: **Requires** Resources::ProcessingCycles **of** IntelX86;

Processor IntelX86

features

ProvidedService: **Provides** Resources::ProcessingCycles;

Virtual bus myprotocol

features

Virtual bus acts a virtual channel resource and requires physical channel resource

RequiredService: **Requires** Resources::Bandwidth ;

ProvidedService: **Provides** Resources::Bandwidth ;

Do we need to distinguish between features that require binding vs. connection?

Access/Usage Features

Provides/requires declaration