

Bindings, Resources & Layers

Peter Feiler
July 2016



Bindings between System Hierarchies

AADL supports a (primary) containment hierarchy

Semantic connections represent flow between and within subtrees

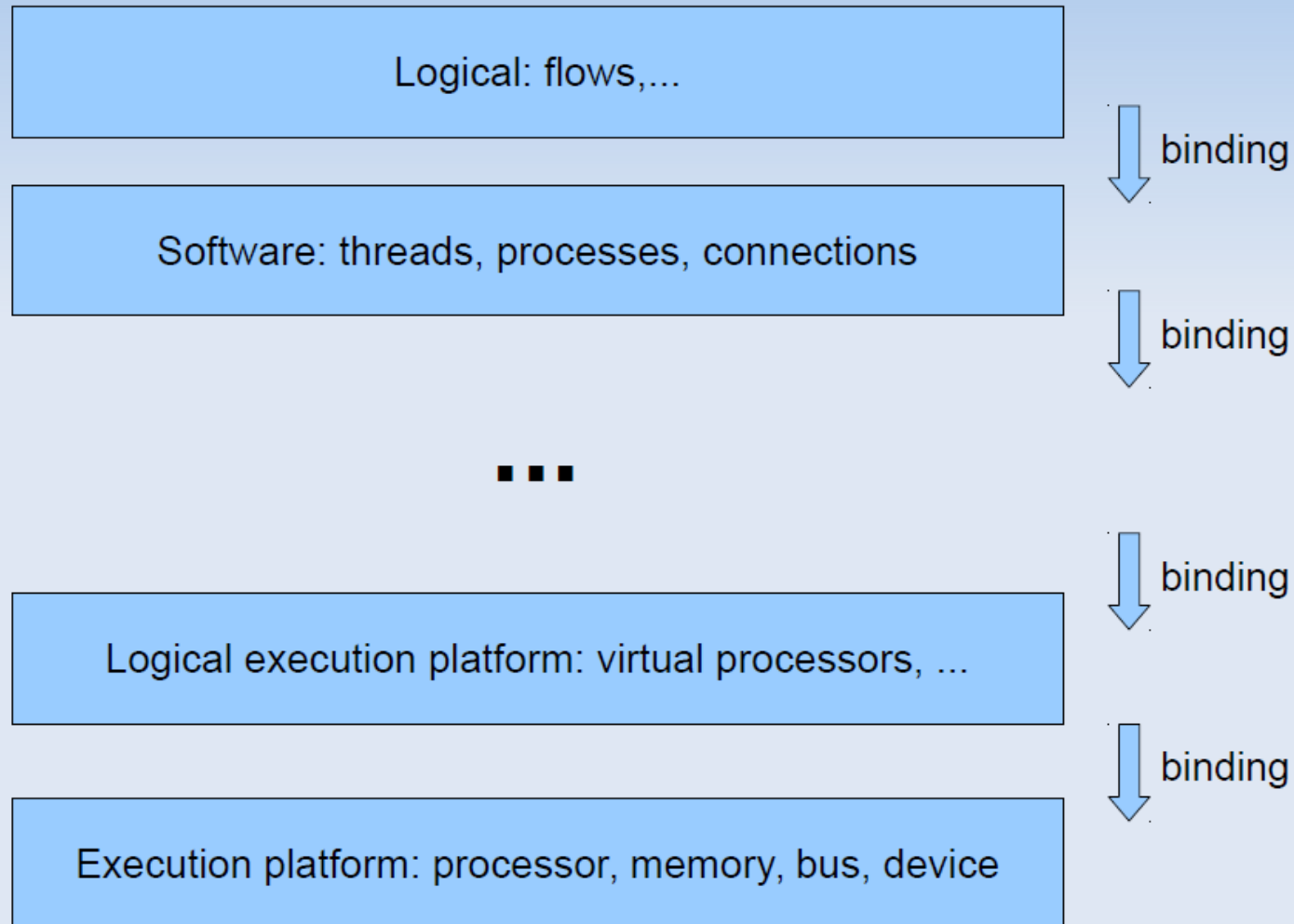
- Managed interaction complexity by requiring connections up and down the hierarchy to restrict arbitrary connectivity
- Note: for subprogram calls we offer both a connection and a mapping specification

Deployment bindings are a mapping from elements of one subtree to elements of another subtree

- The subtrees represent different virtual machine layers with the lower layer typically representing resources to the higher layer
- Bindings represent resource allocation



Multi Level Architecture



Issues in Current Binding Approach

Bindings are currently expressed by properties

Binding related properties are not distinguishable from others

- Properties that express bindings
- Properties that relate to bindings

EMV2 propagation paths are derived from bindings

- binding points currently are identified by special keyword

Binding properties reach down the instance containment hierarchy

- A primary driver for introducing contained property associations



Resource Flow and Resource Allocation

Resource flow within an architecture: follows interface rules

- Electricity, fluid flow, ...
- Directional flow with continuous characteristics: producer → consumer
- Can be modeled with abstract feature and connections
- Resource type represented as type (or abstract component type)
 - Annotations for discrete or continuous behavior
- Fan out/in of flow “volume”
 - multiple features & multiple connections from one feature

Resource allocation/binding: Across different architecture layers

- Resource usage that needs to be allocated/scheduled
- SW to computer platform
- Logical to physical
- Resource capacity and demand as provides and requires features
 - Feature acts as binding point: resource type as classifier
 - Multiple



Binding Specification Proposal

Binding points

- Properties, constraints

Binding instances

- Source and target configurations
- Deployment bindings



Binding Point Specification

Explicit in features section:

- Named features to be used as source or target of binding
- Identification of type of binding (type of resource involved in binding)
- optional classifier(s) to restrict the type of target

```
thread task1
```

```
features
```

```
RequiredCycles: Requires ProcessingCycles;
```

```
Processor IntelX86
```

```
features
```

```
ProvidedService: Provides ProcessingCycles;
```

```
Virtual bus myprotocol
```

```
features
```

```
RequiredService: Requires ProtocolX;
```

```
ProvidedService: Provides MyProtocol ; -- self or other protocols
```



Binding Point Specification

One component can have multiple binding points

- Binding points of same type
 - Provider: subsets of total resource capacity
- Binding points of different types
 - Need/provision of different resources, e.g., at system level

Binding related properties

- Number of acceptable bindings
 - Provider: multiple binding points and one per binding point
 - Requestor: multiplicity of resource providers
- Resource related
 - Provider: capacity per binding point & provider component
 - Requestor: demand(budget)



Binding Constraints

AADL V2 had constraint on classifier and instances

- (“Allowed_xx” property)

Constraint notation or special syntax?

- Classifier of provides and requires must match
- Constraint on category or classifier of target component
 - E.g., Processor type
 - as part of **requires**
- Constraint on source of binding
 - As part of **provides**
- Use of constraint language

```
thread task1
```

```
features
```

```
RequiredCycles: Requires X86.ProcessingCycles; ProcessingCycles of ;
```



Binding Instances

- Binding of unchangeable source and target hierarchies (Configurations)
- Declared as a configuration section
- Associated with system configuration that contains both source and target subtrees

```
System bindings AS.deploymentconfig
For applayer: myapp:ASApp.config -> platformlbayer:
myplatform:Asplatform.config
(
  Appsys.sub.proc.thread1.RequiredCycles -> node.cpu1.ProvidedService,
  Appsys.sub.proc.thread2.bp1 -> node.cpu2.bp2 );
System implementation AS.deploymentconfig
Subcomponents
Platformlayer: system myplatform:Asplatform.config
Applayer: system myapp:ASApp.config;
Vplatform: system myvplatform;
connections
  b1: binding Appsys.sub.proc.thread1.RequiredCycles ->
node.cpu1.ProvidedService,
  c1: port Appsys.sub.proc.thread2.bp1 -> node.cpu2.bp2 b;
```



Binding Instances

- Binding of unchangeable source and target hierarchies (Configurations)
- Declared as a configuration section
- Associated with system configuration that contains both source and target subtrees

```
System bindings AS.deploymentconfig
For applayer: myapp:ASApp.config -> platformlbayer:
myplatform:Asplatform.config
(
  Appsys.sub.proc.thread1.RequiredCycles -> node.cpu1.ProvidedService,
  Appsys.sub.proc.thread2.bp1 -> node.cpu2.bp2 );
```

```
System implementation AS.deploymentconfig
```

```
Subcomponents
```

```
Applayer: myapp:ASApp.config;
```

```
Vplatform: myvplatform;
```

```
Platformlayer: myplatform:Asplatform.config
```

```
bindings
```

```
Appsys.sub.proc.thread1.RequiredCycles -> node.cpu1.ProvidedService,
Appsys.sub.proc.thread2.bp1 -> node.cpu2.bp2 );
```



Partial and Nested Bindings

Partial binding configurations

- Partially configured source and target system
 - Only for those elements that have been configured
- Subset of elements are bound
 - Bindings cannot be overridden

Configurations with binding points

- System may make part of its resources externally available, e.g., camera provides some of its processing capacity for a user plugin
- System may have some driver software that needs to run on an external resource



Connection Bindings

Currently: sequence of target elements

Connection acts as binding point

- Propagation identifies connection by name

connections

```
Conn1: port sub1.p1 -> sub2.p1 Requires XferBandwidth;
```

```
Conn2: abstract sub1.fe1 -> sub2.fe1 Requires WattsPerHour;
```

Binding target options:

- end-to-end flows in next/platform level
- sequence as binding target



Flows and Connection Bindings

End-to-end flow across virtual and hardware platform elements

- Expressed by end to end flow declaration
- Source and destination of ETE flow must match binding target of connection source and destination
- Each element of the flow has binding point of matching type



Binding Multiplicity

Types of multiplicities

- Alternatives: one of x
 - Statically chosen: binding point specifies choices: multiple classifiers
- Scheduling over multiple
 - On one or on one at a time is a scheduling issue
 - Virtual processor (scheduler) responsible for scheduling multiple resources
- Static allocation
 - Partial allocation: multiple bindings each with percentage
 - Replicated allocation: multiple binding targets
- Sequence: ordering, same demand on all elements of sequence
 - If needed we would introduce a property on the binding to indicate the type



Binding of Ports Across Layers

Processors provide ports and subprogram access

Processor features

Portx: **provides** in data port DT;

Applications declare processor port proxies in the processor features section of an implementation.

Move to features section of type

Portx: **requires** in data port DT;

Actual binding

- Once a binding of the application to the processor is specified a “connection” between the application level and the platform level is inferred by name matching of port
- Do we need to separately define the binding of the two or keep inferring?

