Bindings & Resources

Peter Feiler
May 2016

Bindings between System Hierarchies

AADL supports a (primary) containment hierarchy

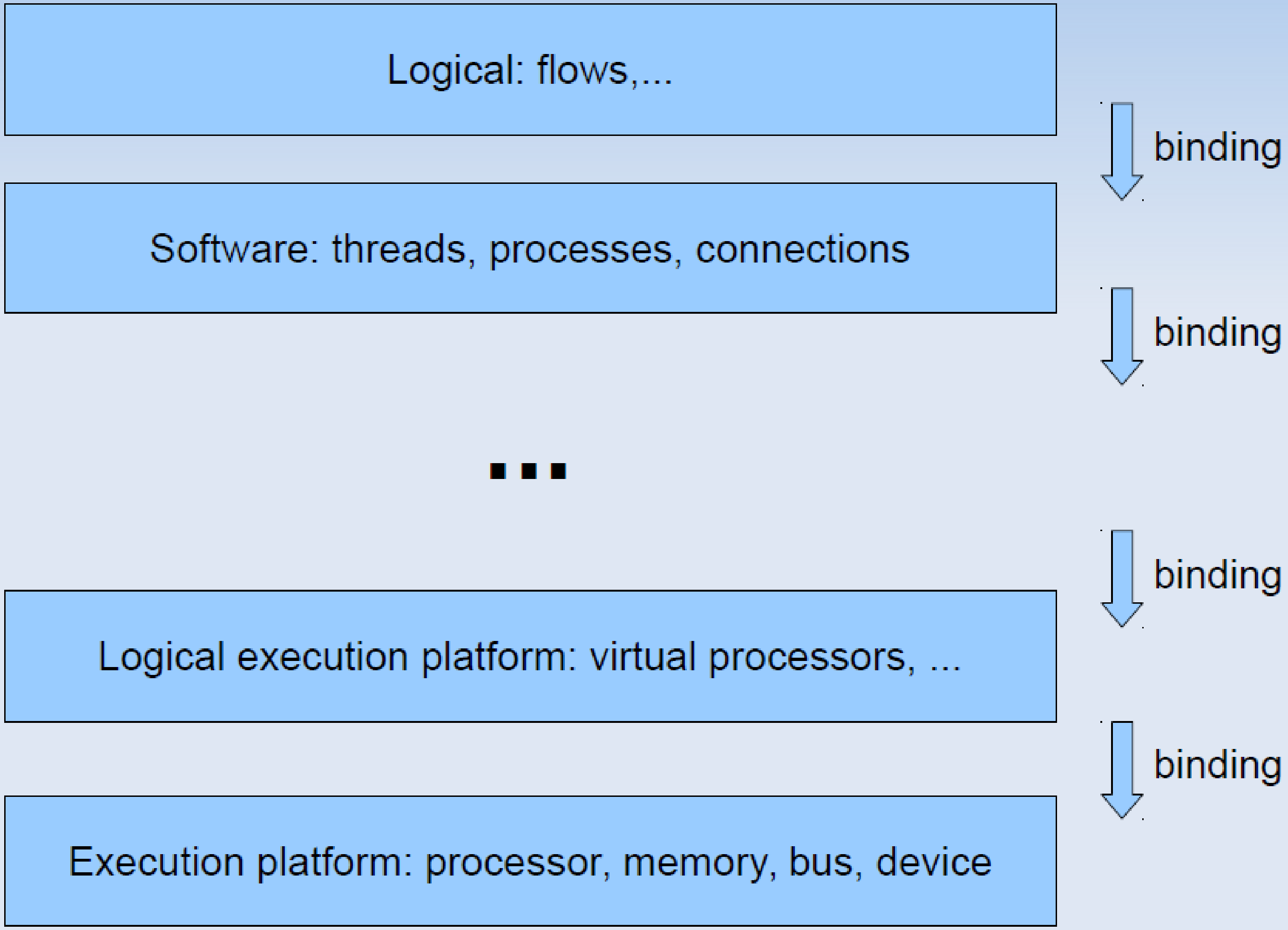
Semantic connections represent flow between and within subtrees

- 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


Logical: flows,...
binding
Software: threads, processes, connections
binding
...
binding
Logical execution platform: virtual processors, ...
binding
Execution platform: processor, memory, bus, device

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

f

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 reflected as 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 architectures (Levels)

- 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 bindingss

User-definable binding types

- Introduce name and source/target model elements

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;
RequiredCycles: Requires X86ProcessingCycles;

Processor IntelX86
features
ProvidedService: Provides X86ProcessingCycles;

Virtual bus protocol
features
RequiredService: Requires ProtocolX;
ProvidedService: Provides ProtocolX ;
```

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)

Bindings and EMV2

In EMV2 binding related propagation points are identified by

- Special keywords (AADL V2)
  - For sources: processor, memory, connection, binding,
  - For targets: bindings (all bindings with component as target)
- the binding point (feature) name (AADL V3)
  - In/out indicates direction of propagation: could be from source to target or vice versa

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;
RequiredCycles: Requires ProcessingCycles of X86;
```

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 configuration AS.deploymentconfig
binds ASApp.config to Asplatform.config
bindings
-- single binding target
T1ProcessorBinding : Appsys.sub.proc.thread1.bp1 -> platform.node.cpu1.bp2;
T2ProcessorBinding : Appsys.sub.proc.thread2.bp1 -> platform.node.cpu2.bp2;
-- multiple targets
Conn1Protocols : Appsys.sub.proc.conn1 -> <op>(platform.proto1,
platform.proto2);
```

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

Nested binding configurations

- Subsystem may be internally fully bound
- Configuration that includes binding has been configured in

Binding points of closed system architecture

- External resource demand and capacity as part of interface specification
- Accessible system internal resources mapped to system binding points

Connection Bindings

Currently: sequence of target elements

Binding point on connection:

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

Options:

- Binding to end-to-end flows in next/platform level
- Support sequence multiplicity

Flows and Connection Bindings

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

- Expressed by end to end flow declaration
- Assumes access connections between virtual buses and virtual processors
- Assumes flow declarations across interconnected virtual components
- Binding of connections to these flows
- Binding of virtual platform elements to flows at next level

Binding points of targets in end-to-end flow

- Each element of the flow has binding point of matching type
- If more than one of same type – then name matching

Binding Multiplicity

Multiplicity of targets

- Inherent in binding type
- Specific to binding instances

Types of multiplicities

- Alternatives: one of x
  - Statically or dynamically chosen: single/all whole request to single target
- Distributed across: part of every
  - Distributed In time: single request scheduled across multiple targets
  - Distributed in demand: across multiple memories, parallel processing
- Replication: replicated demand one per target
- Sequence: ordering, same demand on all elements of sequence

Binding Type

Use of abstract classifier

- Represents binding/resource type
- Same as for resource flow classifiers

Explicit binding type with source/target constraints

Needed?

- User defined name, sets of source and target categories

```
Binding ProcessorBinding : { thread, thread_group, process} -> {
processor, virtual processor, system }
```

1-to-n binding: Multiplicity kind specified with binding instance.

Binding of Ports

We currently have proxies for processor ports that are referenced in connection declarations. The processor port is declared in a processor features section (currently in the implementation).

Processor features

Portx: **in data port** DT;

connections

Conn1: port sub1.p1 -> Portx;

Once a binding of the application to the processor is specified a connection between the application level and the platform level is inferred.

Do we need to separately define the binding of the proxy port to the actual processor port in the instance?