AADL: Bindings and Resources

Peter Feiler

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

DM18-0073

Bindings between System Hierarchies

AADL supports a (primary) containment hierarchy

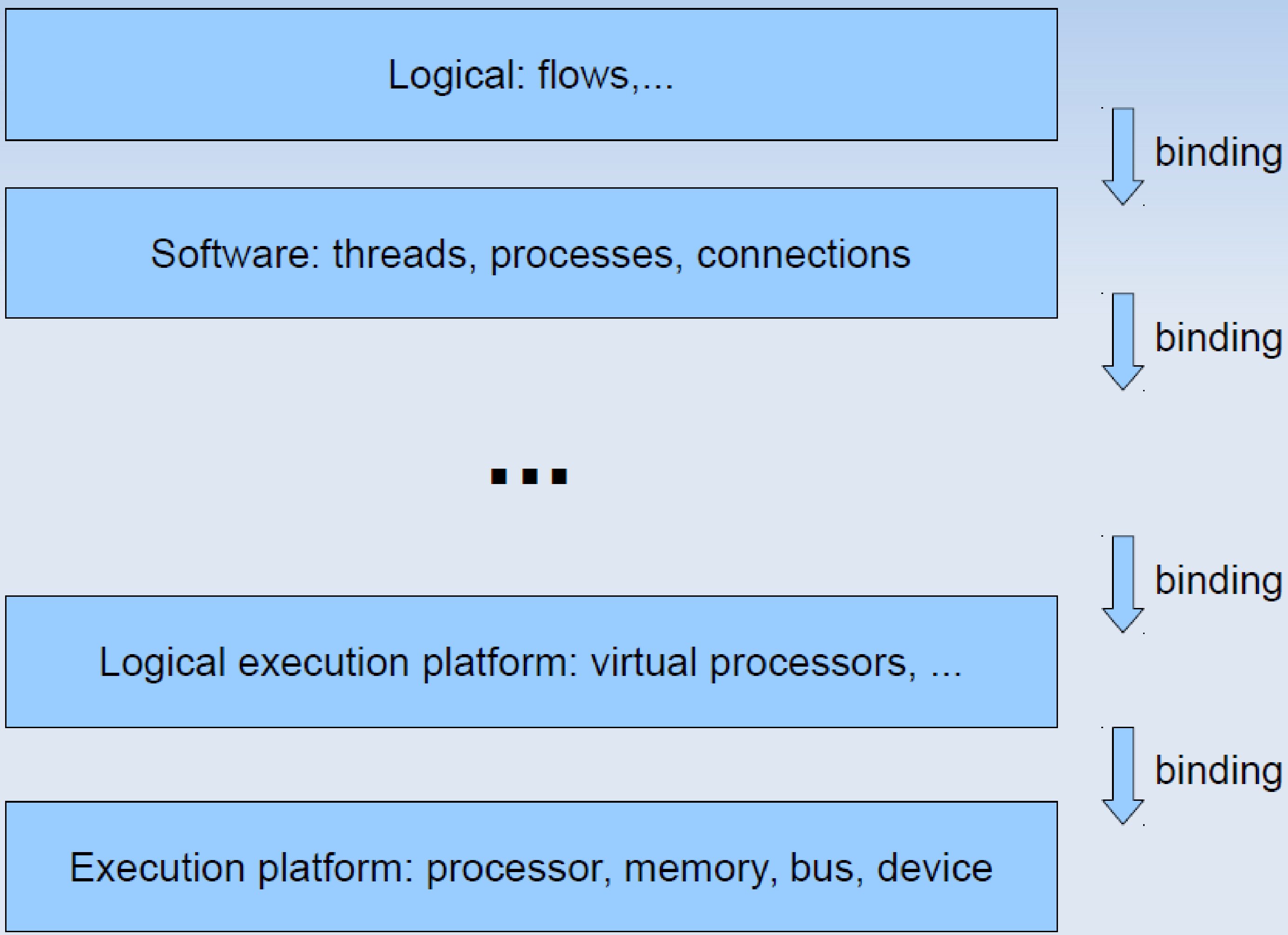
Semantic connections represent flow between and within subtrees

- 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 (aka. allocations) 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 Layer 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

- Continuous: Electricity, fluid flow, … Discrete: data samples, messages
- Directional flow with continuous characteristics: producer –> consumer
- Resource type represented as type (or abstract component type)
  - Annotations for discrete or continuous flow
- 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 aka allocations

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
RequiredCycles: Requires Resources::ProcessingCycles;

Processor IntelX86
features
ProvidedService: Provides Resources::ProcessingCycles;

Virtual bus myprotocol
features
RequiredService: Requires Resources::Bandwidth;
ProvidedService: Provides Resources::Bandwidth ;
```

Do threads, processors etc. have predeclared binding points?

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

Resource Types

Resource type

- May be separate from type of binding target
- Predefined set of resource types
- Use type system to represent resource types

```
ProcessorCycles: type real units CyclesPerSec;
MemorySpace: type int units Size_Units;
```

- Memory speed vs. Memory type as target type constraint

```
MemoryResource: record [
        Size: MemorySpace;
        AccessSpeed: Bandwidth;
];
```

- Memory allocation position

```
AllocatedMemorySpace: record [
        Location: MemoryAddress;
        Size: MemorySpace;
];
MemoryAddress: type int;
```

Generic Binding Types

Binding type

- Generic type as binding type

```
FunctionalBinding: type;
```

- Usage

```
Abstract WBSFunction
features
PhysicalComponent: Requires FunctionalBinding;
```

Target Type & Binding Constraint

Target Type of Binding:

- Target classifier as binding type

```
Virtual bus myprotocol
features
RequiredService: Requires ProtocolY;
End myprotocol ;
Virtual bus ProtocolY
End ProtocolY;
```

ProtocolY provides functionality but not resource

Binding Constraint:

- Optional classifier(s) to restrict the type of target

```
Virtual bus myprotocol
features
RequiredService: Requires Resources::Bandwidth of ProtocolX;
ProvidedService: Provides Resources::Bandwidth ;
End myprotocol;
```

Classifier as separate constraint specification. Classifier must provide specified resource type.

Quantified Resource Binding Specification

Quantity specification

- Leverage directionality of binding point feature
- Separate property

```
thread task1
features
RequiredCycles: Requires Resources::ProcessingCycles => 200 MIPS;

Processor IntelX86
features
ProvidedService: Provides Resources::ProcessingCycles => 1200 MIPS;
```

Relationship of binding point (value) to execution time?

Binding Point Specification

One component can have multiple binding points

- Binding points of different types
  - Need/provision of different resources, e.g., at system level
- Binding points of same type
  - Provider: subsets of total resource capacity
  - Other characteristics: address range for memory, encryption

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)
- Other characteristics

Binding Instances

- System implementation contains subtrees to be mapped to each other
  - Elements of one subtree to be bound to element of another

```
System implementation AS.impl
Subcomponents
  Platform: system myplatform::Asplatform;
  Appsys: system myapp::ASApp;
End AS.impl;
configuration AS.deploymentconfig extends AS.impl {
Platform => myplatform::Asplatform.config,
Appsys => myapp::ASApp.config
};
```

Configurations of platform and Appsys

Binding Instances

- Binding of from source to target hierarchies (Configurations)
- Multiple bindings for same configuration

```
configuration AS.boundconfig1 extends AS.deploymentconfig
{
  Appsys.sub.proc.thread1.RequiredCycles -> Platform.cpu1.ProvidedService,
  Appsys.sub.proc.thread2.cache -> Platform.cpu2.cache
};
configuration AS.boundconfig2 extends AS.deploymentconfig
{
  Appsys.sub.proc.thread1.RequiredCycles -> Platform.cpu2.ProvidedService,
  Appsys.sub.proc.thread2.cache -> Platform.cpu2.cache
};
```

Do bindings apply to subcomponents?

- Binding of process implies binding of all threads

Match pattern notation from configuration:

```
Appsys.sub.proc.*.RequiredCycles ->
Platform.cpu2.ProvidedService
```

Visibility of Binding Points

How far down can the allocation declaration reach

- Processor, memory, bus as boundary within design space
- Parameterized configuration as boundary for external use

Map binding point at configuration interface to component(s) in implementation that manage or represent resource

```
System ASplatform
features
  ComputeCycles: provides Resources::ProcessingCycles;
  Storage: provides Resources::cache;
End ASPlatform;
ASplatform.boundconfig configures ASPlatform.impl {
Cpu => MyHW::X86.i7,
Storage => MyHW::FasstMem.L1,
ComputeCycles -> cpu,
Storage -> Cachememory
};
```

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;
```

Platform End-to-end flow as binding target

- 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

Virtual bus as binding target

- Virtual bus itself needs to be bound to a sequence of items => ETE flow

End-to-end flow as closed platform configuration binding point

- How to expose platform internal ETEF as external binding point? => access to virtual bus that is bound to ETE flow

Binding of Features to Platform

Processors provide ports and subprogram access

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

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

Move to features section of type

Portx: **requires** **in 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?

Resource Scheduling & Binding Multiplicity

Scheduling over multiple resources

- Virtual processor (scheduler) responsible for scheduling multiple resources
  - VP binding to processors represents the set to be scheduled
- Scheduling protocol reflect in virtual processor type and Scheduling_Protocol property on VP/Processor

Memory allocation

- Starting location & size
- Relation to virtual memory?

Allocation across multiple targets

- Replicated allocation: multiple binding targets
- Partial allocation
  - multiple bindings each with percentage
  - Segmentation of data component – handling via virtual memory?