Component Interaction and AADL modeling

Oleg Sokolsky
Real-Time System Group
University of Pennsylvania

SAE AADL Working Group Meeting
July 10-13, 2006
Outline

• **Motivation**
  - Enhance AADL models with information on protocol-based component interaction
• Simple case study
  - Based on virtual buses
• Suggestions
  - Interaction annex
• Summary
AADL: connections and flows

• Flows: abstraction of communication

• Associate QoS properties with flows
  - End-to-end delays, transfer rates, etc.
AADL: connections and flows

- Implementations map flows to connections

- Architectural checks enable validation of implementations
Motivation: flows vs. interactions

• Flows are very suitable for stream processing
  - Sense, transform/compute, actuate
• Flows are restrictive for reactive components
  - independent
  - unidirectional
• Component interactions in a reactive system
  - bi- and multi-directional
  - follow a protocol
Related ports and connections

- We need to be able to specify that
  - A group of ports are semantically related
  - Connections on these ports are established together and provide a certain protocol
- Port groups serve this purpose to some extent
Interacting flows

• We need to be able to specify that
  - A group of flows establishes a protocol
  - Collectively provide certain QoS
Refinement of interactions

• We need to be able to refine multi-party interactions into patterns of coordinated connections and flows
Outline

- **Motivation**
  - Enhance AADL models with information on protocol-based component interaction
- **Simple case study**
  - Based on virtual buses
- **Suggestions**
  - Interaction annex
- **Summary**
Case study

- Consider an event data flow passing through a bus
  - Standard AADL
system Passthrough
  features
    p_out: out event data port;
    p_in: in event data port;
  flows
    pflow: flow path p_in -> p_out;
end Passthrough;

system PassthroughBus extends Passthrough
  features
    b_acc: requires bus access Itype;
end PassthroughBus;

bus Ibus
end Ibus;

system implementation Passthrough.Interaction
  subcomponents
    sender: system PassthroughBus;
    receiver: system PassthroughBus;
    bcon: bus Ibus;
  connections
    dflow: event data port sender.p_out -> receiver.p_in
      {Allowed_Connection_Binding => reference bcon;};
    s_bus: bus access bcon -> sender.b_acc;
    r_bus: bus access bcon -> receiver.b_acc;
end Passthrough.Interaction;
One level of refinement

- Communication follows a simple protocol

```plaintext
port group SenderRole
  data: out event data port;
  ack: in event port;
end SenderRole;

port group ReceiverRole
  data: in event data port;
  ack: out event port;
end ReceiverRole;
```
bus IbusProto extends Ibus
features
  role1: port group SenderRole
  role2: port group ReceiverRole
end IbusProto;

system PassthroughSender extends PassthroughBus
features
  sendRole: port group inverse of SenderRole;
end PassthroughSender;

system PassthroughReceiver extends PassthroughBus
features
  recvRole: port group inverse of ReceiverRole;
end PassthroughReceiver;

system implementation Passthrough.IProto
  extends Passthrough.Interaction
subcomponents
  sender: refined to PassthroughSender;
  receiver: refined to PassthroughReceiver;
  bcon: refined to IbusProto;
connections
  s_role: port group sender.sendRole -> bcon.role1;
  r_role: port group bcon.role2 -> receiver.recvRole;
end Passthrough.IProto;
What is missing?

- Relationship between ports in a role port group
  - Interaction along the port group connection
- Interaction between roles
- Mapping of the abstract connection
Outline

• Motivation
  - Enhance AADL models with information on protocol-based component interaction
• Simple case study
  - Based on virtual buses
• Suggestions
  - Interaction annex
• Summary
Suggestions I

• **Mapping of ports can be done similar to the binding of connections**

```plaintext
dflow: event data port sender.p_out -> receiver.p_in
    { Allowed_Connection_Binding =>
        reference bcon;
    Allowed_Source_Mapping =>
        reference sender.sendRole.data;
    Allowed_Dest_Mapping =>
        reference receiver.recvRole.data;
    }
```

• **Alternatively, mapping may be a property of each port**
Suggestions II

• Interaction annex
  - Interaction diagrams (message sequence charts)
  - Use 1: Associated with port groups for protocol roles
    • interactions between two sides of the connection
    • ports in the group serve as interaction steps
  - Use 2: A feature in a bus type
    • possibly other components
Interactions in protocol roles

**port group** SenderRole

`data: out event data port;`
`ack: in event port;`

```latex
annex interaction {**
    other -[data]--> self;
    other <-[ack]--> self;
**}
```

end SenderRole;

**port group** ReceiverRole

`data: in event data port;`
`ack: out event port;`

```latex
annex interaction {**
    self -[data]--> other;
    self <-[ack]--> other;
**}
```

end ReceiverRole;
Interaction between roles

bus IbusProto extends Ibus
features
  role1: port group SenderRole
  role2: port group ReceiverRole
annex interaction {**
  role1 ->[data] self;
  self ->[data] role2;
  self <-[ack] role2;
  role1 <-[ack] self;
**}
end IbusProto;
Do we need “self” role?

- Separate distinct interaction steps
- Sequencing information may be lost

```plaintext
bus IbusLog extends Ibus
  features
  role1: port group SenderRole
  role2: port group ReceiverRole
  roleL: port group LoggerRole
  annex interaction {**
    role1 -[data]-> self;
    self   -[log]-> roleL;
    self   <-[done]-> roleL;
    self   -[data]-> role2;
    self   <-[ack]-> role2;
    role1 <-[ack]-> self;
  **}
end IbusLog;
```
Interactions vs. behavior

- Specification of interactions is complementary to specification of component behaviors
- Implementation of an interaction is done in terms of state machines
- Component behavior can be used to check that the component will comply with the protocol
Summary

• Interactions between components in the system can and should be modeled on the architectural level
  - Interactions are part of the component type
• Interactions are typically specified by message sequence charts
  - Interaction annex specifies syntax
• Interactions modeling is built on top of virtual buses extension