The AADL Behavioural annex[1]

J.-P. Bodeveix, P. Dissaud, P. Feiler, M. Filali

IRIT-CNRS ; Université de Toulouse, France
Ellidis Software France-UK
SEI CMU USA

Wednesday March 24th 2010
OXFORD
UML-AADL 2010
Panel

[1]This work was partly supported by the French AESE project Topcased and by the region Midi-Pyrénées.

Introduction

Architecture languages $\rightsquigarrow$ verification of dynamic aspects:

- composition, temporal properties,
- quantitative analysis: sizing, performance, energy,
- reliability.

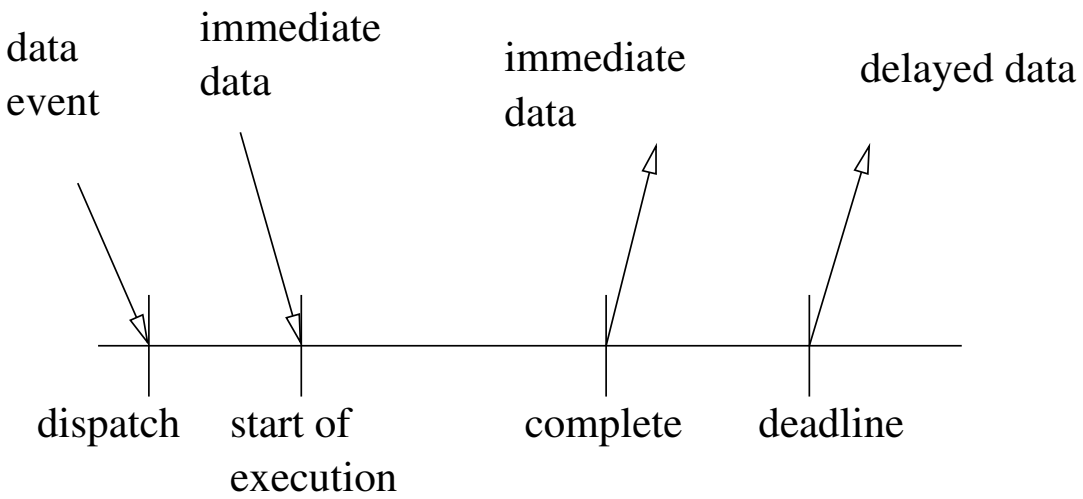
The behavioural annex

Why a behavioural annex for AADL?

- AADL relies on the analysis of source text.
- AADL calls and flows do not depend on data.
- AADL behaviour is basically described by:
  - the AADL execution model
  - quantitative aspects like wcet.

The behavioural annex allows a refinement of these aspects.

The AADL data port protocol



The AADL behavioural annex:

- describes how the dispatch is triggered.
- gives access to the received and sent data
- describes what happens when control is gained
- says when control is relinquished

The behavioural annex

- Describe the internal behavior of component implementations as a state transition system with guards and actions.
- Extend the default run-time execution semantics that is specified by the core of the standard, such as thread dispatch protocols.
- Provide behavioral refinement for mode transitions.
- Introduce subprogram calls synchronization protocols.

These extensions are introduced through properties and annexes.

```
behavior_annex ::=
  [ variables  behavior_variable + ]
  [ states  behavior_state + ]
  [ transitions  behavior_transition + ]

behavior_state_kind ::=
  [ initial ][ complete ][ final ]

execution_behavior_transition ::=
  [ behavior_transition_label : ]
  source_state_identifier  , source_state_identifier *
  -[ [ behavior_condition ] ]->
  destination_state_identifier [  behavior_actions  ] ;
```

```
thread merger
  features
    p1 : in event data port Basic_types::integer;
    p2 : in event data port Basic_types::integer;
    m : out event data port Basic_types::integer;
end merger;
```

```
thread implementation merger.twopersistentstates
  annex behavior_specification {**
  variables
    x1 : data Basic_types::integer;
    x2 : data Basic_types::integer;
  states
    s0 : initial complete state;
    comp : state;
    next1, next2 : complete state;
  transitions
    s0 -[ on dispatch p1 ]-> next2 { x1 := p1 };
    s0 -[ on dispatch p2 ]-> next1 { x2 := p2 };
    next1 -[ on dispatch p1 ]-> comp { x1 := p1 };
    next2 -[ on dispatch p2 ]-> comp { x2 := p2 };
    comp -[ x1 < x2 ]-> next1 { m!(x1) };
    comp -[ x2 <= x1 ]-> next2 { m!(x2) };
  **};
```

Synchronization protocols

- subprogram calls in AADL is synchronous.
- alternate synchronization protocols (HRT-HOOD):

```
Server_Call_Protocol: type enumeration (ASER,HSER,LSER)
    applies to (provides subprogram access);
```

  - ASER : the caller is never blocked.
  - LSER : the caller waits for the acceptation of the request.
  - HSER : the caller waits for the completion of the request and gets results if any.

Conclusion

- The behavioural annex is now part of the AADL standard.
- Supported by the OSATE-TOPCASED project.
- Verification support through the FIACRE language : pivot verification language of the TOPCASED project.
- Used for case studies in industrial research projects.