Introduction to AADL[1]

M. Filali
joint work with Bernard Berthomieu, Jean-Paul Bodeveix,
Christelle Chaudet, Silvano Dal Zilio, François Vernadat

IRIT-CNRS ; University of Toulouse, France
LAAS-CNRS ; University of Toulouse, France

Monday April 12th 2010
KTH
CYBER WEEK 2010
Panel

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

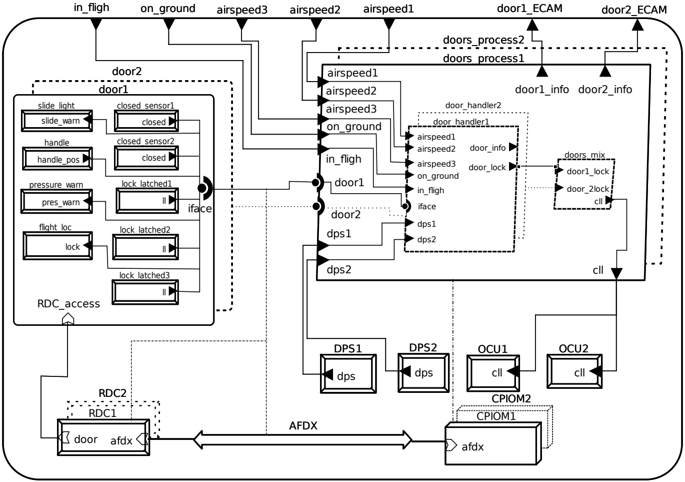
Introduction

AADL: Architecture Analysis & Design Language
Bruce Lewis Peter Feiler (main investigators).
main features:

- architecture description language
- quantitative analysis: timing, sizing, performance, energy
- reliability
- extensibility
- standard of the SAE

AADL description language

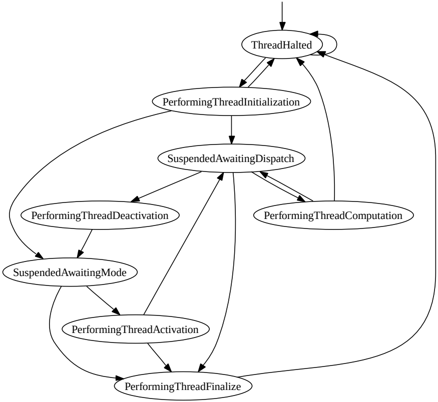
- Software components
  - process
  - thread
  - data
- Hardware components
  - processor
  - device
  - memory
  - bus
- Composition
  - system
- Interaction
  - shared memory
  - port and port groups
    - data port
    - event port
    - event data port
- Connexions
- Modes



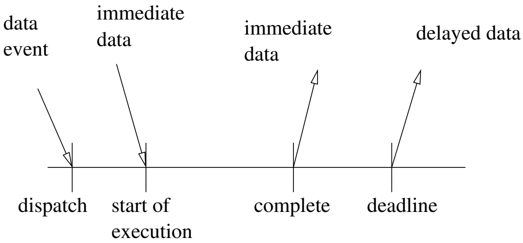
(J.-F. Rolland (IRIT))

AADL execution model

- Precise dispatch & communication protocols semantics.


ThreadHalted
PerformingThreadInitialization
SuspendedAwaitingDispatch
PerformingThreadDeactivation
PerformingThreadComputation
SuspendedAwaitingMode
PerformingThreadActivation
PerformingThreadFinalize

The AADL data port protocol



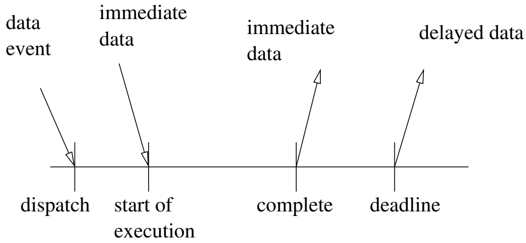
The behavioural annex

(Joint work of Jean-Paul Bodeveix, Pierre Dissaux, Peter Feiler, Mamoun Filali.)
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 behaviour annex and execution model



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
- explicits 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 have been introduced through properties and annexes.

```
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) };
  **};
```

Behavioural annex 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

- AADL results from a long experience in real-time.
- AADL is supported by the OSATE-TOPCASED project.
- The behavioural annex is now part of the AADL standard.
- Verification support through the FIACRE language : pivot verification language of the TOPCASED project.
- Used for case studies in industrial research projects.