Formal Verification of AADL Specifications in the Topcased Environment[1]

B. Berthomieu[1] J.-P. Bodeveix[2,3] C. Chaudet [2,3]
S. Dal Zilio[1] M. Filali[2] F. Vernadat[1,3]

[2]IRIT-CNRS [1]LAAS-CNRS
[3]Université de Toulouse

Brest - Ada Europe
June 8-12 2009

[1]supported by the French AESE project Topcased, EU SPICES project and by region Midi-Pyrénées.

Outline

1. Verification in TOPCASED
2. The AADL and FIACRE languages
   - AADL
   - Fiacre
3. The AADL to Fiacre transformation
   - The Kermeta model transformation language
   - Transformation
4. Verification: from Fiacre to Tina
5. Conclusions

Outline

1. Verification in TOPCASED
2. The AADL and FIACRE languages
   - AADL
   - Fiacre
3. The AADL to Fiacre transformation
   - The Kermeta model transformation language
   - Transformation
4. Verification: from Fiacre to Tina
5. Conclusions

Outline

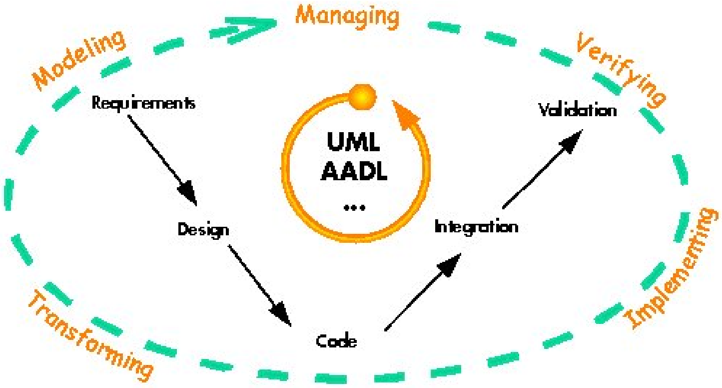
Outline

Outline

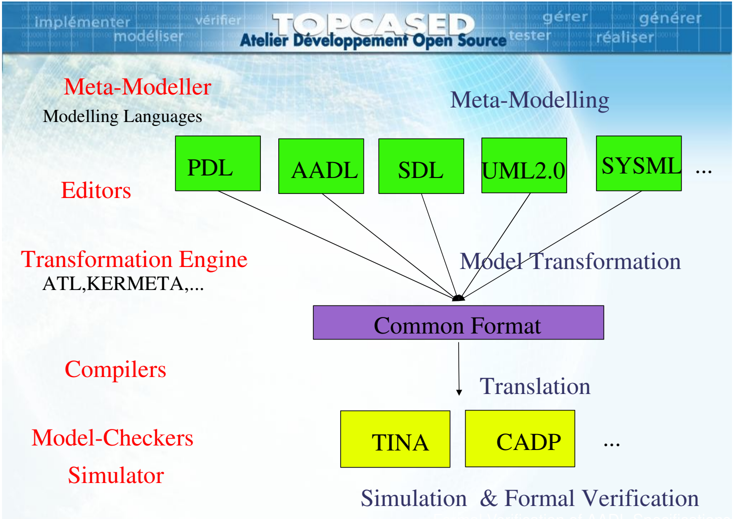
Plan

1. Verification in TOPCASED
2. The AADL and FIACRE languages
   - AADL
   - Fiacre
3. The AADL to Fiacre transformation
   - The Kermeta model transformation language
   - Transformation
4. Verification: from Fiacre to Tina
5. Conclusions

http://www.topcased.org



Verification in TOPCASED



Plan

1. Verification in TOPCASED
2. The AADL and FIACRE languages
   - AADL
   - Fiacre
3. The AADL to Fiacre transformation
   - The Kermeta model transformation language
   - Transformation
4. Verification: from Fiacre to Tina
5. Conclusions

Outline

The AADL language (I)

- The **package** defines component structure.
- Components are classified into categories:
  - software: **data, subprogram, thread, thread group, process**;
  - execution: **memory, processor, bus, device**;
  - composite: **system**.

AADL execution model

AADL has a precise execution semantics.

Basic aspects:

- Threads:
  - Execution.
  - Communication.
  - Behavior
- Modes.

AADL example: a token ring system (I)

```
system root
end root;

system implementation root.i
subcomponents
  p: process network.i;
end root.i;
```

AADL example: a token ring system (II)

```
process network
end network;

process implementation network.i
subcomponents
  s:  thread Start.i;
  n0: thread Node.i;
  n1: thread Node.i;
  n2: thread Node.i;
connections
  event port s.succ → n0.prev;
  event port n0.succ → n1.prev;
  event port n1.succ → n2.prev;
  event port n2.succ → n3.prev;
end network.i;
```

```
thread Node
features
  prev: in event port {OverFlow_Handling_Protocol ⇒ Error; };
  succ: out event port;
properties
  Dispatch_Protocol ⇒ Sporadic; Period ⇒ 10ms;
end Node;

thread implementation Node.i
annex behavior_specification {**
states
  idle: initial complete state;
  wait: complete state;
  cs: state; -- critical section
transitions
  idle -[prev?]→ idle { computation(1ms); succ!; };
  idle -[prev?]→ wait { computation(1ms); succ!; };
  wait -[prev?]→ cs;
  cs -[]→ idle { computation(5ms, 10ms); succ!; };
**};
end Node.i;
```

```
thread Start
features
  succ: out event port;
properties
  Dispatch_Protocol ⇒ Background;
end Start;

thread implementation Start.i
annex behavior_specification {**
states
  s0: initial state;
  s1: complete state;
transitions
  s0 -[]→ s1 { succ!; };
**};
end Start.i;
```

Outline

The Fiacre language
Joint work with the INRIA team VASY

- Factorize transformation efforts between high level languages and model checking tools
- Powerful enough to represent high level languages features (synchronization mechanisms, shared variables, data structures)
- Avoid tool dependant features (timers)
- Potentially efficient model checking (coarse grain transitions)

Main features of Fiacre

Processes: sequential behaviors

- control states
- local variables
- transitions
  - synchronous communications (at most one per execution path)
  - access to shared variables (if no communication)
  - non deterministic choice
  - sequence, conditionals, loops

Components: parallel composition

- synchronous or interleaving composition
- declaration of shared variables
- declaration of ports with timing constraints

```
port p:T in [a,b]
```

Composition: the token ring

```
process Node[prev : none, succ : none, start : in none] is
  states idle, wait, cs, st_1
  from idle select start ; to st_1 [] prev ; to st_1 end
  from st_1 succ ; select to idle [] to wait end
  from wait prev ; to cs
  from cs   succ ; to idle

process Start[start0 : none, start1 : none, start2 : none] is
  states s0, s1
  from s0 select start0 [] start1 [] start2 end ; to s1

component root is
  port s0 : none, s1 : none, s2 : none,
       p0 : none, p1 : none, p2 : none,
  par * in
     Start[s0,s1,s2]
  || Node[p0, p1, s0] || Node[p1, p2, s1] || Node[p2, p0, s2]
end

root
```

Plan

Outline

The Kermeta model transformation language

Main features

- Object oriented, multiple inheritance, genericity
- Easy specification/import/export of EMF models
- OCL-like iterators thanks to functionnal programming features
- Aspect oriented programming

Outline

From AADL to Fiacre

Generic transformations

- Communication network
- Thread behaviors

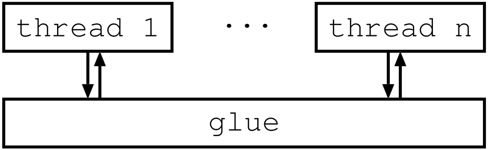
Transformations depending on the execution model
$\rightsquigarrow$ **Definition of subsets of AADL**

- Untimed asynchronous
- Untimed synchronous
- Timed asynchronous without preemption
- Timed asynchronous with preemption (needs Tina semi-decision algorithm)
- Use of modes, remote procedure calls, shared variables, pooling,...

Transformation of an AADL architecture

Principle

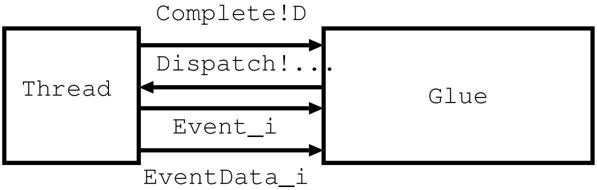
- Introduction of a *Glue* process
- AADL threads communicate through the Glue
- The Glue manages buffering and mode-dependent communication networks



Interaction with the glue

Principle

- At dispatch time, get all input data
- At complete time, send contents of output data ports
- Immediatly send events and valued events



Interaction with the glue

Process interface in Fiacre

```
process thread[
  dispatch: in Tdp#bool# -- data ports
    union
      ev1 -- OneItem event ports
    | evn of nat -- AllItems event ports
    | ed1 of Tedp -- OneItem event data ports
    | edn of queue of N Tedp -- AllItems event data
    end
  oevp: out none,
  oedp: out Tedp,
  complete: out Tdp] is ...
```

Glue internals

Principle

- declare variables for threads input ports
- declare states for AADL operational modes
- manage incoming messages
  as specified by AADL connections for current mode
- manage incoming mode change requests

Plan

1. Verification in TOPCASED
2. The AADL and FIACRE languages
   - AADL
   - Fiacre
3. The AADL to Fiacre transformation
   - The Kermeta model transformation language
   - Transformation
4. Verification: from Fiacre to Tina
5. Conclusions

Verification in Fiacre

- Properties expressed in SE-LTL (state-event LTL)
  - safety: $\Box(cs\_Node_1 + cs\_Node_2 + cs\_Node_3 \leq 1)$;
  - liveness: $\Box(wait\_Node_1 \Rightarrow \Diamond cs\_Node_1)$;
  - realtime properties expressed through observers
- Timed transitions taken into account by Tina
  - Expression of periodic or sporadic behaviors
  - Expression of timeouts, delays or cpu consumptions

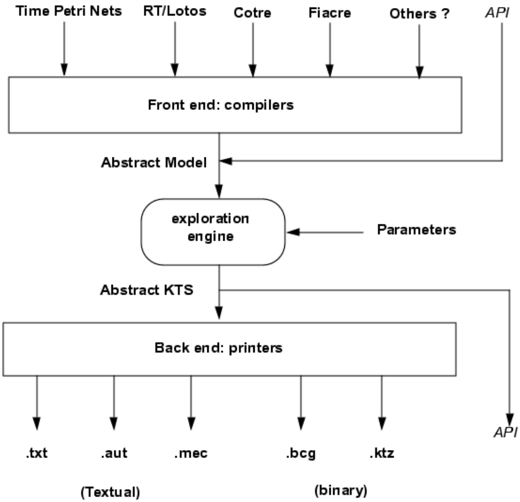
  $\rightsquigarrow$ Verification of schedulability
- Translation to Tina or CADP

The Tina verification environment

Edition and analysis of

- Petri Nets
- Time Petri Nets (with time intervals associated to transitions)
- Extensions
  - handling of data (timed transition systems)
  - priorities
  - preemption

TINA – exploration engine



Plan

Conclusions

- Verification in the TOPCASED environment (open source).
- Front end tool development AADL→ Fiacre
- Back end tools development Fiacre → Tina (also CADP)

⇝ A prototype of the tool is available.

Next steps

- Simplification of the expression of logical properties.
- Improving error reporting.
- Improving and assessing the verification process.
  - higher level Fiacre
  - Formalization and Mechanization of the semantics