

Cotre Annex HRT-HOOD embedding

FéRIA

17th October 2005

Outline

- 1 Basic Principles
- 2 Data and types and subprograms
 - Subprogram call
- 3 Message exchanges
- 4 HOOD objects
 - Periodic Threads
- 5 Cotre property set
- 6 Specification of system properties (preliminary study)

Outline

- 1 Basic Principles
- 2 Data and types and subprograms
 - Subprogram call
- 3 Message exchanges
- 4 HOOD objects
 - Periodic Threads
- 5 Cotre property set
- 6 Specification of system properties (preliminary study)

Outline

- 1 Basic Principles
- 2 Data and types and subprograms
 - Subprogram call
- 3 Message exchanges
- 4 HOOD objects
 - Periodic Threads
- 5 Cotre property set
- 6 Specification of system properties (preliminary study)

Outline

- 1 Basic Principles
- 2 Data and types and subprograms
 - Subprogram call
- 3 Message exchanges
- 4 HOOD objects
 - Periodic Threads
- 5 Cotre property set
- 6 Specification of system properties (preliminary study)

Outline

- 1 Basic Principles
- 2 Data and types and subprograms
 - Subprogram call
- 3 Message exchanges
- 4 HOOD objects
 - Periodic Threads
- 5 Cotre property set
- 6 Specification of system properties (preliminary study)

Outline

- 1 Basic Principles
- 2 Data and types and subprograms
 - Subprogram call
- 3 Message exchanges
- 4 HOOD objects
 - Periodic Threads
- 5 Cotre property set
- 6 Specification of system properties (preliminary study)

Outline

- 1 Basic Principles
- 2 Data and types and subprograms
 - Subprogram call
- 3 Message exchanges
- 4 HOOD objects
 - Periodic Threads
- 5 Cotre property set
- 6 Specification of system properties (preliminary study)

- Embedding of HRT-HOOD concepts in AADL
 - Property sets
 - Behavioral annex (non deterministic specifications over AADL data)
- Expression of system properties

Outline

- 1 Basic Principles
- 2 Data and types and subprograms**
 - Subprogram call
- 3 Message exchanges
- 4 HOOD objects
 - Periodic Threads
- 5 Cotre property set
- 6 Specification of system properties (preliminary study)

Simple and compound types

- Simple types : package with data declarations for integers, reals, ...
- Compound types (records) : user defined hierarchies of data

```

package Cotre
  data integer
    properties Source_Data_Size => 32 bits;
  end integer;

  data float    end float;

  data boolean end boolean;
end Cotre;

```

Arrays

- No array type in AADL
 - Reuse of UML multiplicity
- ↪ new AADL property

```
tab: data Cotre::integer {CotreProperties::Multiplicity=
```

Multiplicity attribute ↪

- Implicit declaration of access subprograms
- Usual array notation defined in Cotre behavioral annex
- Equivalent to new `data` declaration

Subprograms

- Behavior attached to subprogram implementations
- Access to subprogram parameters
- Access to visible data declared in AADL

Annex for subprogram behavior

- Specified by an automaton
- Reuse of mode automata syntax
- Action part associated to a transition
- Guard added to event
- Final state declaration: when reached,
 - output parameters are transmitted to caller,
 - control returns to caller.

Example (implementation)

```
subprogram implementation addition.default
annex cotre_behavior {**
states
  s0 : initial state;
  s1 : final state;
transitions
  s0 -[ ]-> s1 { r := x + y ; ovf := false; }
  s0 -[ ]-> s1 { r:= 0; ovf := true; }
**};
end start_read;
```

Subprogram call

- AADL control flow: specification of unconditional call sequences
- proposed annex:
 - data dependant control flows
 - subprogram calls
 - raise of events

Raise of an event

```
subprogram addition
```

```
features
```

```
  x: in parameter std::integer;   y: in parameter std::i
```

```
  r: out parameter std::integer;  ovf: out event;
```

```
end addition;
```

```
subprogram implementation addition.default
```

```
annex cotre_behavior {**
```

```
states
```

```
  s0 : initial state;  s1 : final state;
```

```
transitions
```

```
  s0 -[ ]-> s1 { r := x + y ; ovf := false; }
```

```
  s0 -[ ovf! ]-> s1 { }
```

```
**};
```

```
end start read;
```

Outline

- 1 Basic Principles
- 2 Data and types and subprograms
 - Subprogram call
- 3 Message exchanges**
- 4 HOOD objects
 - Periodic Threads
- 5 Cotre property set
- 6 Specification of system properties (preliminary study)

Message exchanges

- use of ports and port groups of AADL
- syntax similar to that of subprogram calls
 - $p!(t_1, \dots, t_n)$ sends t_i to port group p
 - $p?(x_1, \dots, x_n)$ receives x_i from port group p
- use of () to delimit port groups

Example of message transmission

```
thread implementation test.default
subcomponents
  x: data Cotre::integer;
annex cotre_behavior {**
  states
    s0: initial state;
    s1: state;
  transitions
    s0 -[p_in?x]-> s1 {}
    s1 -[p_out!x+1]-> s0 {}
**};
end test.default;
```

Outline

- 1 Basic Principles
- 2 Data and types and subprograms
 - Subprogram call
- 3 Message exchanges
- 4 HOOD objects**
 - Periodic Threads**
- 5 Cotre property set
- 6 Specification of system properties (preliminary study)

HOOD passive objects

Direct translation to AADL data declarations

```
subprogram put
  features v: in parameter Cotre::integer;
end put;
subprogram get
  features v: out parameter Cotre::integer;
end get;
subprogram empty
  features v: out parameter Cotre::boolean;
end empty;
subprogram full
  features v: out parameter Cotre::boolean;
end full;
```

data specification

data stack

features

```
put: subprogram put;
```

```
get: subprogram get;
```

```
empty: subprogram empty;
```

```
full: subprogram full;
```

```
end stack;
```

HOOD protected objects

- supported by AADL
- ↪ data with property `Concurrency_Control_Protocol`
- Access control left unspecified by AADL
- ↪ property
`Supported_Concurrency_Control_Protocols` to be defined

HOOD active objects

- separation of processing and synchronization
- subprogram behavior sequential
- object behavior: synchronization part
- AADL mechanisms weaker than HRT-HOOD:
 - asynchronous communications through ports,
 - highly synchronous communications through client/server subprograms,
 - no synchronization conditions

Asynchronous mode of HRT-HOOD

- Entry points are AADL ports
- Asynchronous message sending
- Thread associated to the server
- Bounded message queue (AADL attribute `Queue_Size`)
- Acceptance conditions specified by the server thread

Highly synchronous mode of HRT-HOOD

- The called service can return a result to the client.
- Partially implemented by AADL client/server subprogram
- Size of the server queue bounded (`Queue_Size` attribute)
- Activation conditions not specified in AADL
- ~> Use of the server thread to express conditions
- ~> the server thread waits for allowed input events
- ~> the client waits for a transition to a `final` state

Semi-synchronous mode of HRT-HOOD

- The client wakes up when message is taken into account
- No value is returned
- The server thread calls the subprogram associated to the entry point
- ↪ implementation of client wake up: transition to a *return* state
- ↪ specification of semi-synchronous mode: new property attached to entry point

Server_Call_Protocol:

```
type enumeration (LSER,HSER) → HSER
applies to (server subprogram);
```

Elapse of time

defined as new actions

- `Computation(min, max)`: non deterministic CPU usage
- `Delay(min, max)`: non deterministic wait

Periodic threads

Reuse of AADL properties attached to threads:

- `Dispatch_Protocol=>Periodic`
- `Period=>...`

The behavior defined by the behavioral annex starts from an initial state and must reach all final states before `Compute_Deadline`.

Bounded synchronization time: example

```
s0 -[ g & p! ]-> s1 { }
s0 -[ g timeout T ]-> s2 { }
```

- if synchronization on p occurs less the T t.u. after g becomes true, send message and go to s_1
- else wait for T t.u. and go to s_2

Outline

- 1 Basic Principles
- 2 Data and types and subprograms
 - Subprogram call
- 3 Message exchanges
- 4 HOOD objects
 - Periodic Threads
- 5 Cotre property set**
- 6 Specification of system properties (preliminary study)

Cotre property set

```
property set CotreProperties is  
  Server_Call_Protocol:type enumeration (ASER,LSER,HSER)  
    → HSER applies to (server subprogram);  
  Multiplicity : aadlinteger applies to (data);  
  Multiplicities : list of aadlinteger applies to (data)  
end CotreProperties;
```

Outline

- 1 Basic Principles
- 2 Data and types and subprograms
 - Subprogram call
- 3 Message exchanges
- 4 HOOD objects
 - Periodic Threads
- 5 Cotre property set
- 6 Specification of system properties (preliminary study)

Environment of a component

- Verification needs a closed system
- Compositional verification needs environment hypothesis
- ~> attach an environment component to each component
- ~> one to one correspondance between declared features
- Environment can be specified hierarchically
- Environment has a behavior
- Verification of the product (closed system)

Specification of system properties

- CTL-based domain specific properties
-