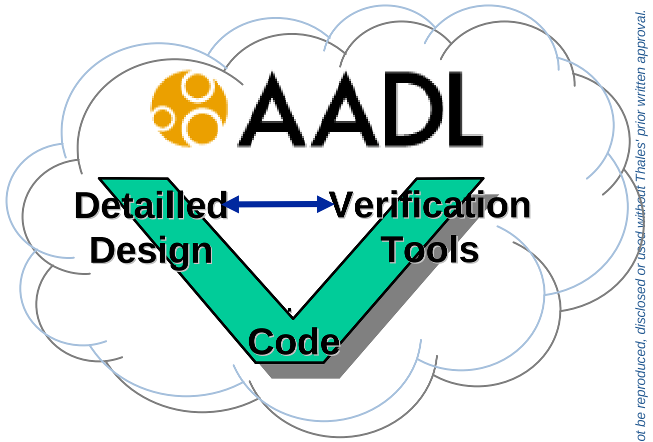
AADL Application modeling with MARTE

Madeleine Faugère, Timothée Bourdeau – THALES Research and Technology
Robert de Simone – INRIA
Sébastien Gérard – CEA List

17/07/2007

AADL in a nutshell

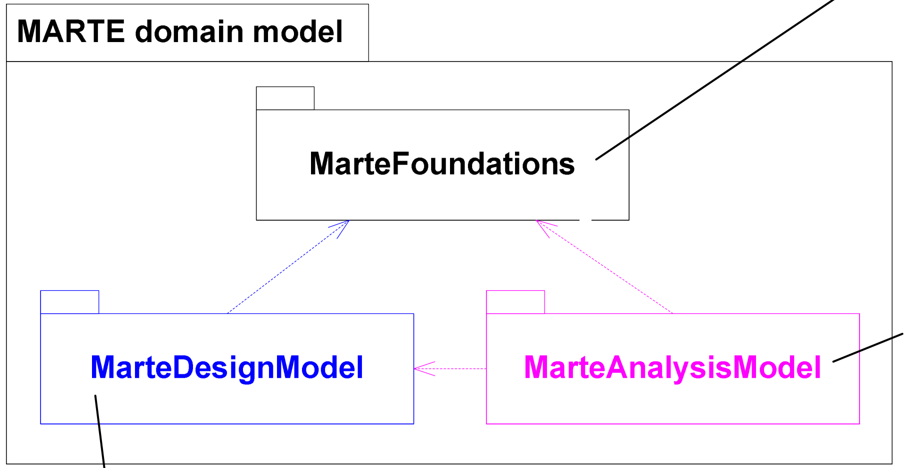
- Architecture Analysis and Design Language (SAE Standard AS-5506 , Nov 2004)
- Synchronous data flow applications modeling
- Architecture Description
  - Specific layer in the developpement life cycle
- Implicit execution platform model
  - AADL execution platform model based on specific thread execution automata
  - Applications and platform execution semantics have to be in line
- Non-functional properties comes as a model decoration
  - A better model integration would improve end-users application modeling understanding
- Existing open source and commercial tools
  - TopCased, Cheddar, aDes, OSATE, Ocarina, STOOD….



MARTE in a nutshell

- A Unified Modeling Language for RTE Systems
  - Too much specific Real-Time and Embedded approaches, RTE languages, diff tools
- Relationships with generic OMG standards
  - Profile for UML 2
    - All UML 2 can load the profile and can design application with MARTE
  - The UML profile for Systems Engineering: SysML
    - Design & Analyse RTE systems
  - Replace UML profile for SPT
    - Integrate issues coming from the SPT
  - Use OCL2 and MOF2 QVT where needed
    - Align with the MDA approach (Model transformations …)
- Align with almost RT/E specific OMG standards
  - The UML profile for Modeling QoS and FT Characteristics and Mechanisms
  - The UML profile for SoC
  - The Real-Time CORBA profile & UML profile for CCM/LwCCM

MARTE overview



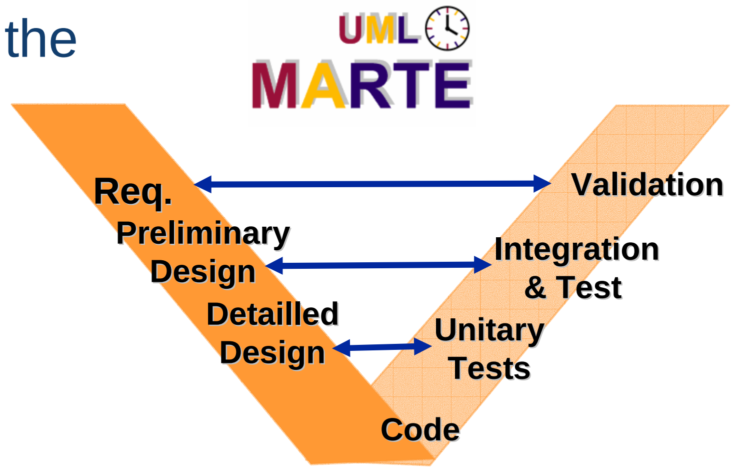
Foundations for RT/E systems modeling and analysis:
- ➔ **CoreElements**
- ➔ **Non Functional Properties**
- ➔ Time
- ➔ **Generic resource modeling**
- ➔ **Generic component modeling**
- ➔ Allocation

Specialization of foundations for annotating model for analysis purpose:
- ➔ **Generic quantitative analysis**
- ➔ **Schedulability analysis**
- ➔ Performance analysis

Specialization of foundations for modeling purpose (specification, design…):
- ➔ **RTE model of calculus & computation**
- ➔ **Software resource modelling**
- ➔ **Hardware resource modelling**

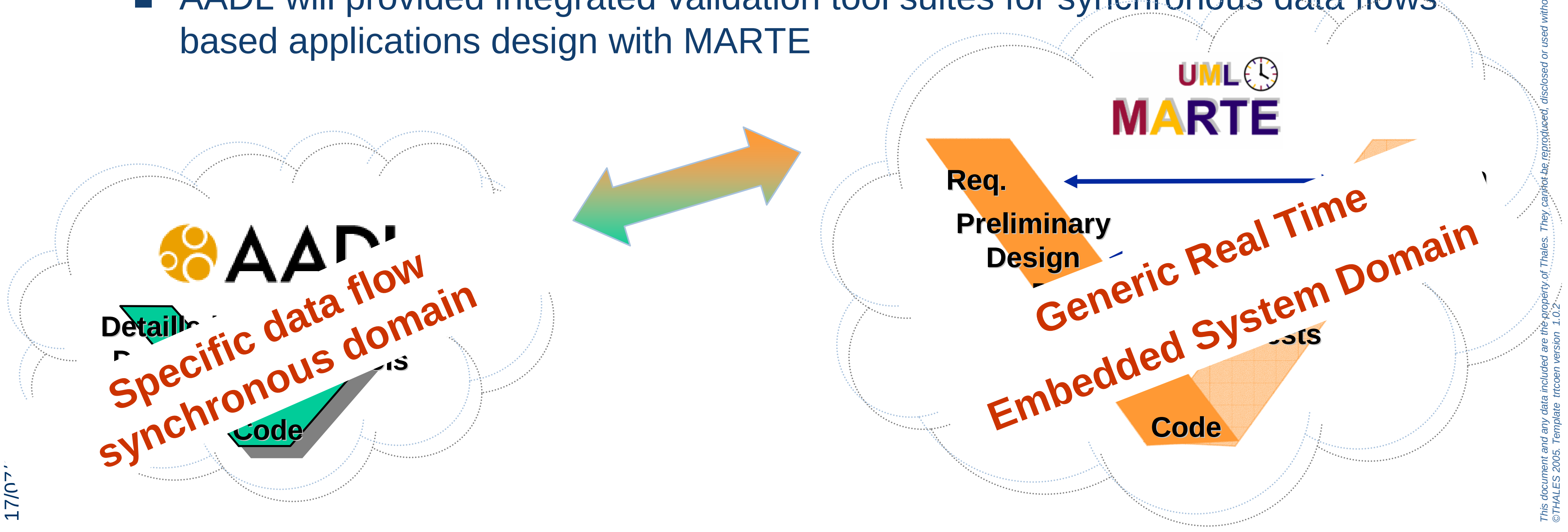
MARTE features

- Generic for Real time Embedded System applications modeling and analysis
- Architecture Description
  - Address early design stages
- Complementary and consistent view make the model more understandable
  - time properties, performance, scheduling features,…
  - platform execution model can be explicitly modelized
- Full integration of non-functional properties in the model
- Young standard
  - Tool chain in developpement



AADL application modeling with MARTE

- AADL – MARTE Bridge benefits
  - MARTE will provide AADL users with
    - early design capabilities
    - enhanced modeling capabilities through consistent views
    - rich modeling information
  - AADL will provided integrated validation tool suites for synchronous data flows based applications design with MARTE



cea list THALES INRIA

AADL Profil presentation and usage

- Most AADL concepts can be represented using UML and MARTE Core concepts
  - HRM for Hardware components ➔ HRM packages
  - Software components ➔ SRM packages
  - Bindings ➔ Alloc packages (issued from SysML)
  - Flow ports and associated features ➔ GCM packages (issued from SysML)
  - Aadl properties ➔ NFP and Marte libraries

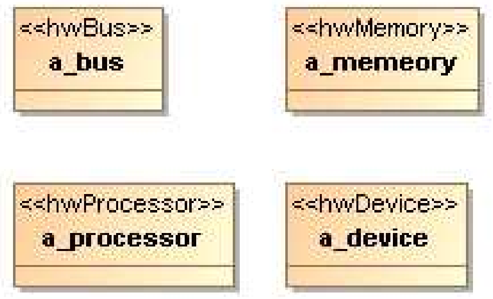
  - Subcomponents ➔ UML Parts
  - Port Connections ➔ UML delegation connectors between ports
  - Parameter connections ➔ Object Flows on activity diagrams
  - Flow specification ➔ UML Object flows between UML Object Pins
  - Modes ➔ UML state machines and collboration diagram

AADL Software components in MARTE

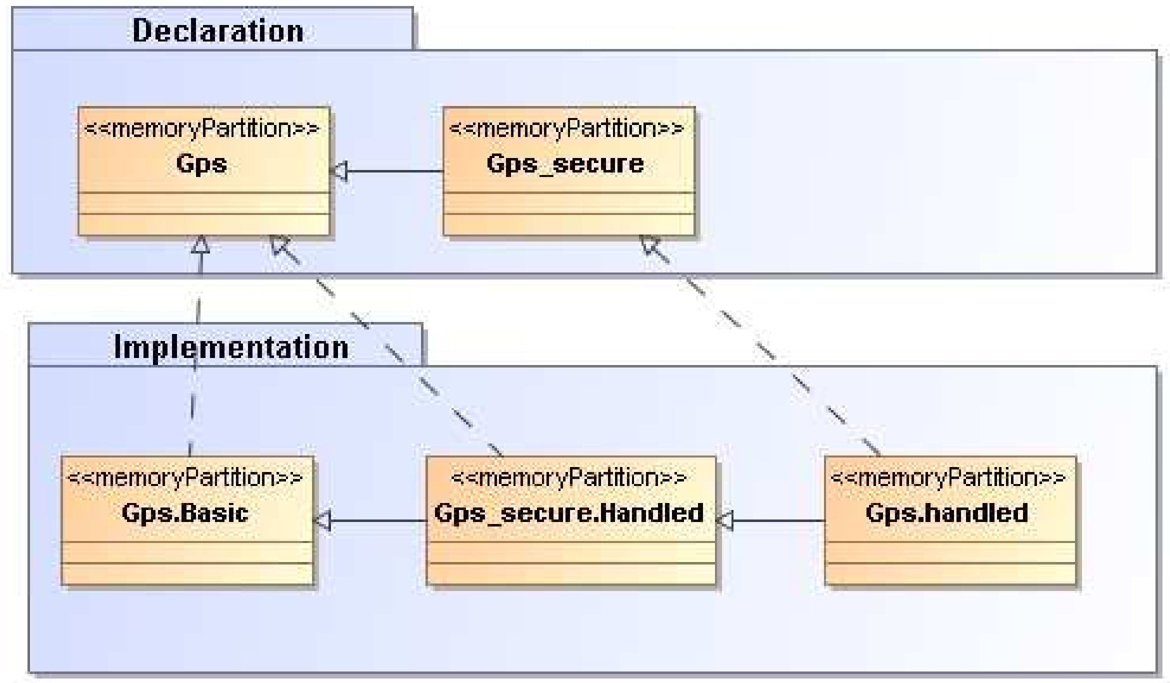
AADL	MARTE
Data Type	***UML::DataType***
Thread	***MARTE::Sw_SchedulableResource***
Thread Group	***MARTE::Sw_SchedulableResource_group***
Process	***MARTE::MemoryPartition***

AADL Execution Platform Components in MARTE

AADL	MARTE
Processor (Abstraction of HW + associated SW)	*MARTE::Hw_Processor* (ASICs,GPPs,DSPs,FPGAs)
Memory (code + data)	*MARTE::Hw_Memory* (RAM,ROM,Drive…)
Bus (HW channel + communication protocols)	*MARTE::Hw_Bus* (CAN, I2C, SPI,USB, PCI…)
Device (Input/Output)	*MARTE::Hw_Device* (I/O, Power supply, Cooling…)
System (composite HW/SW component)	*UML4SysML::Block*



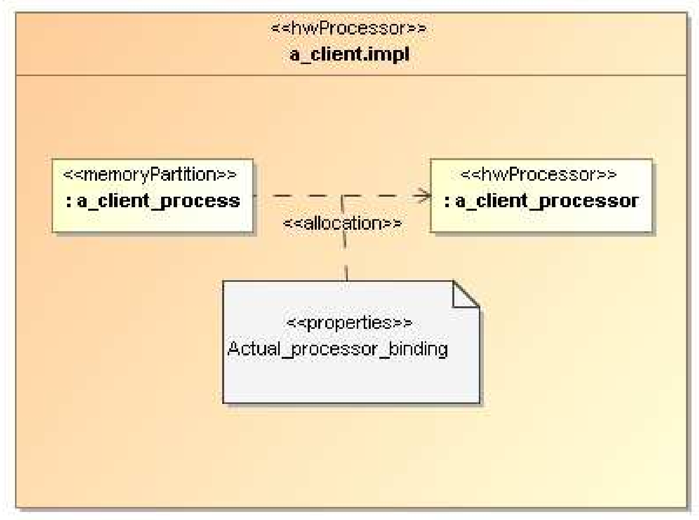
Component Declaration / implementation



➔ Use of UML Extension and Generalization mechanism

System binding

```
system implementation a_client.impl
subcomponents
the_processor : processor a_client_processor;
the_process : process a_client_process;
properties
Actual_Processor_Binding => reference the_processor
applies to the_process;
end a_client.impl;
```



AADL

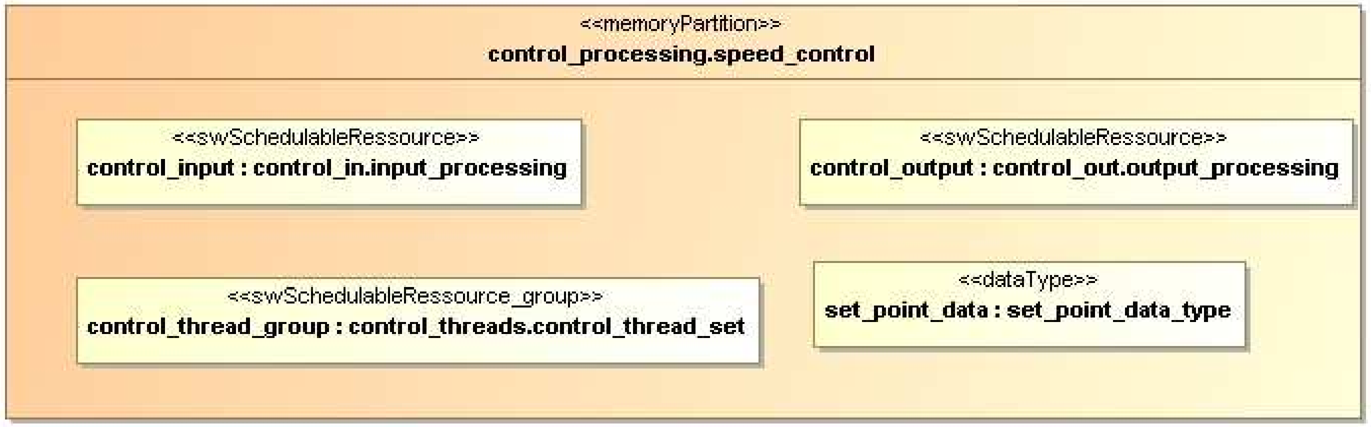
UML MARTE

➔ Marte allocation between software and hardware components

cea list THALES INRIA

Process and contained subcomponents

```
process implementation control_processing.speed_control
subcomponents
        control_input : thread control_in.input_processing;
        control_output : thread control_out.output_processing;
        control_thread_group : thread group control_threads.control_thread_set;
        set_point_data : data set_point_data_type;
end control_processing.speed_control
```



<<memoryPartition>>
control_processing.speed_control
<<swSchedulableRessource>>
control_input : control_in.input_processing
<<swSchedulableRessource>>
control_output : control_out.output_processing
<<swSchedulableRessource_group>>
control_thread_group : control_threads.control_thread_set
<<dataType>>
set_point_data : set_point_data_type

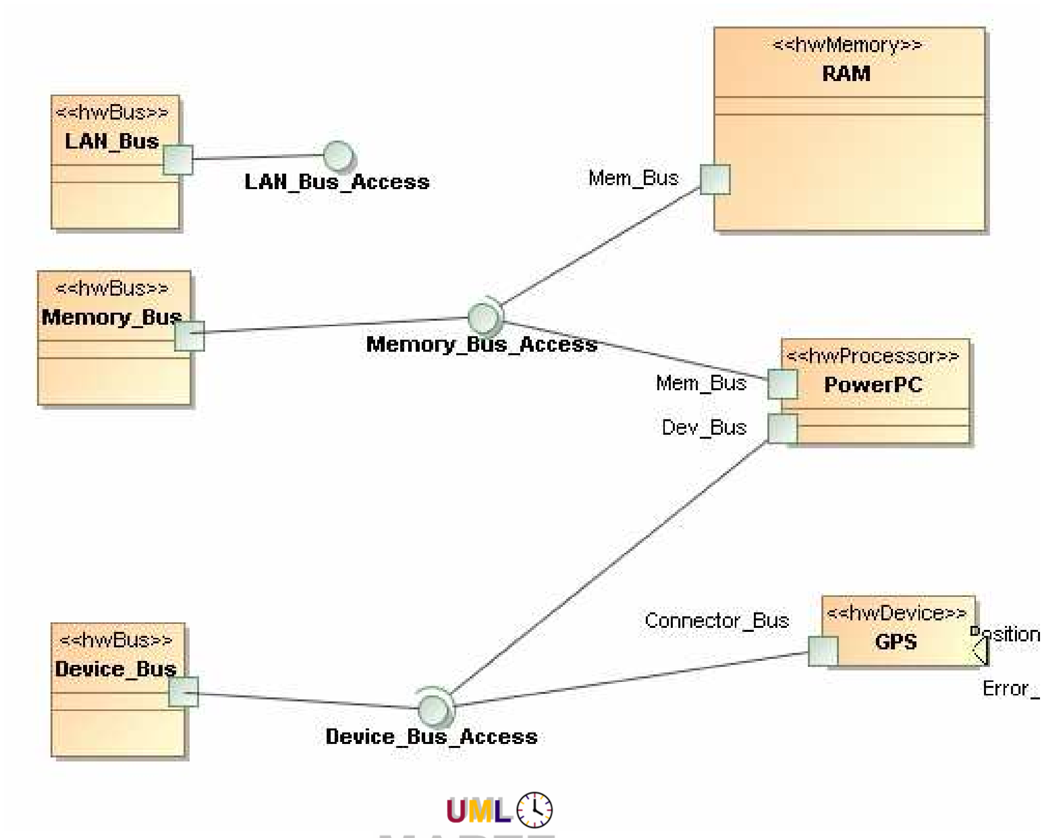
➔ Subcomponents are modelized as UML Parts

Required data/bus acces

```
processor powerPC
Features
                    MemBus : requires bus access Memory_Bus;
                    Dev_Bus : requires bus access Device_Bus
End PowerPC;

bus Memory_Bus
end Memory_Bus;
```

AADL



UML MARTE

➔ bus and data components may provide internal data/service access via an UML Interface

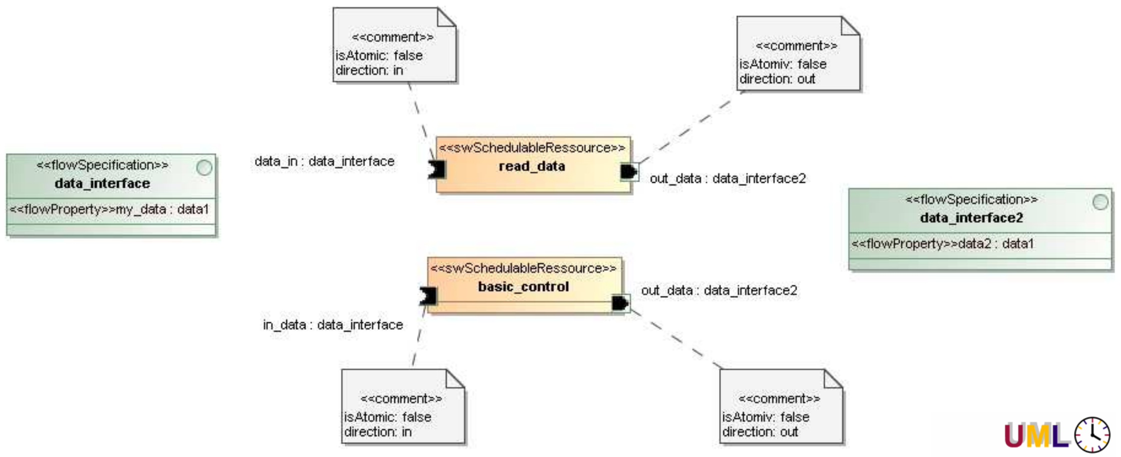
cea list THALES INRIA

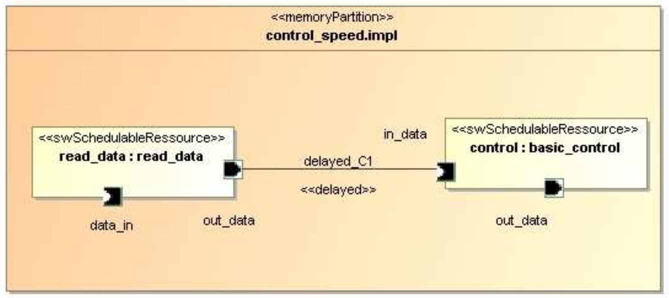
Ports and Port connections

```
thread read_data
features
                in_data : in data port data1;
                out_data : out data port data1;
end read_data;

thread basic_control
features
                in_data : in data port data2;
                out_data : out data port data2;
end basic_control;

process implementation control_speed.impl
subcomponents
                read_data : thread read_data;
                control : thread basic_control;
connections
delayed_C1 : data port read_data.out_data ->> control.in_data;
properties
                Period => 50ms;
end control_speed.impl
```

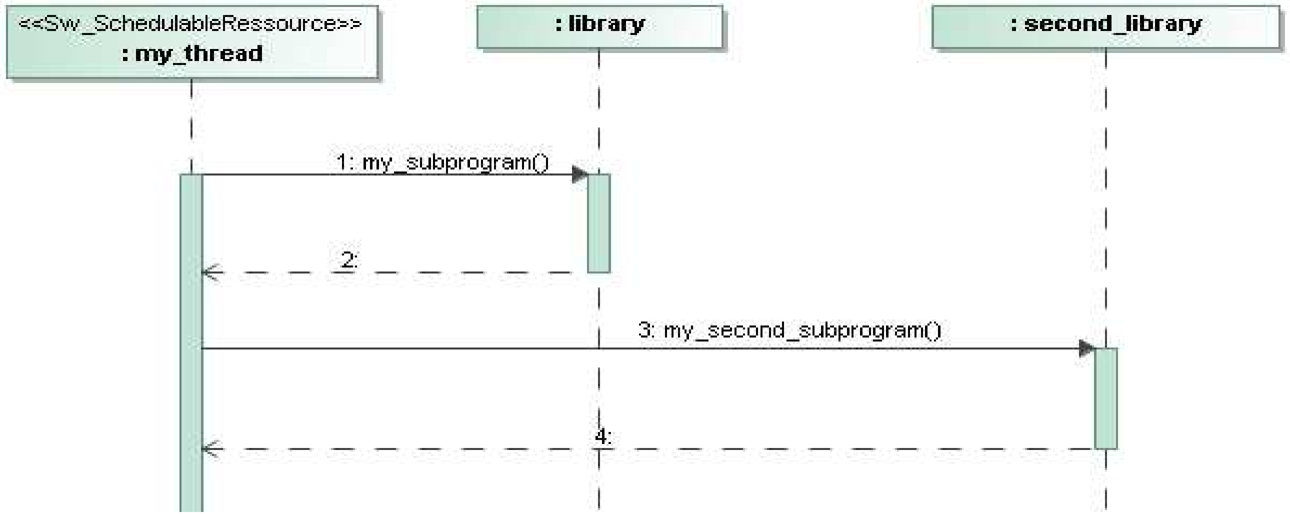




- ➔ Ports are represented by SysML flow Ports
- ➔ Port connections are represented by delegation connectors

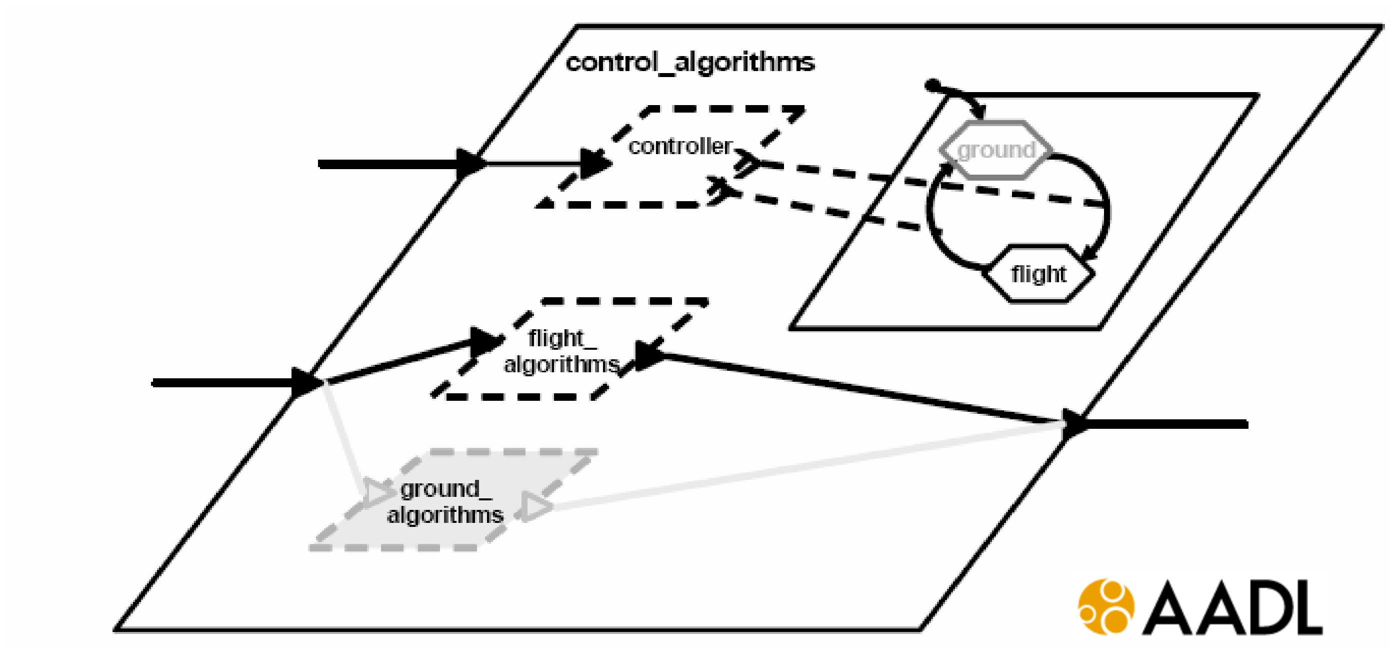
Subprogram Calls

```
thread implementation my_thread.impl
calls {
                first_subpgr : subprogram my_subprogram;
                second_subpgr : subprogram my_second_subprogram;
     }
end my_thread.impl;
```



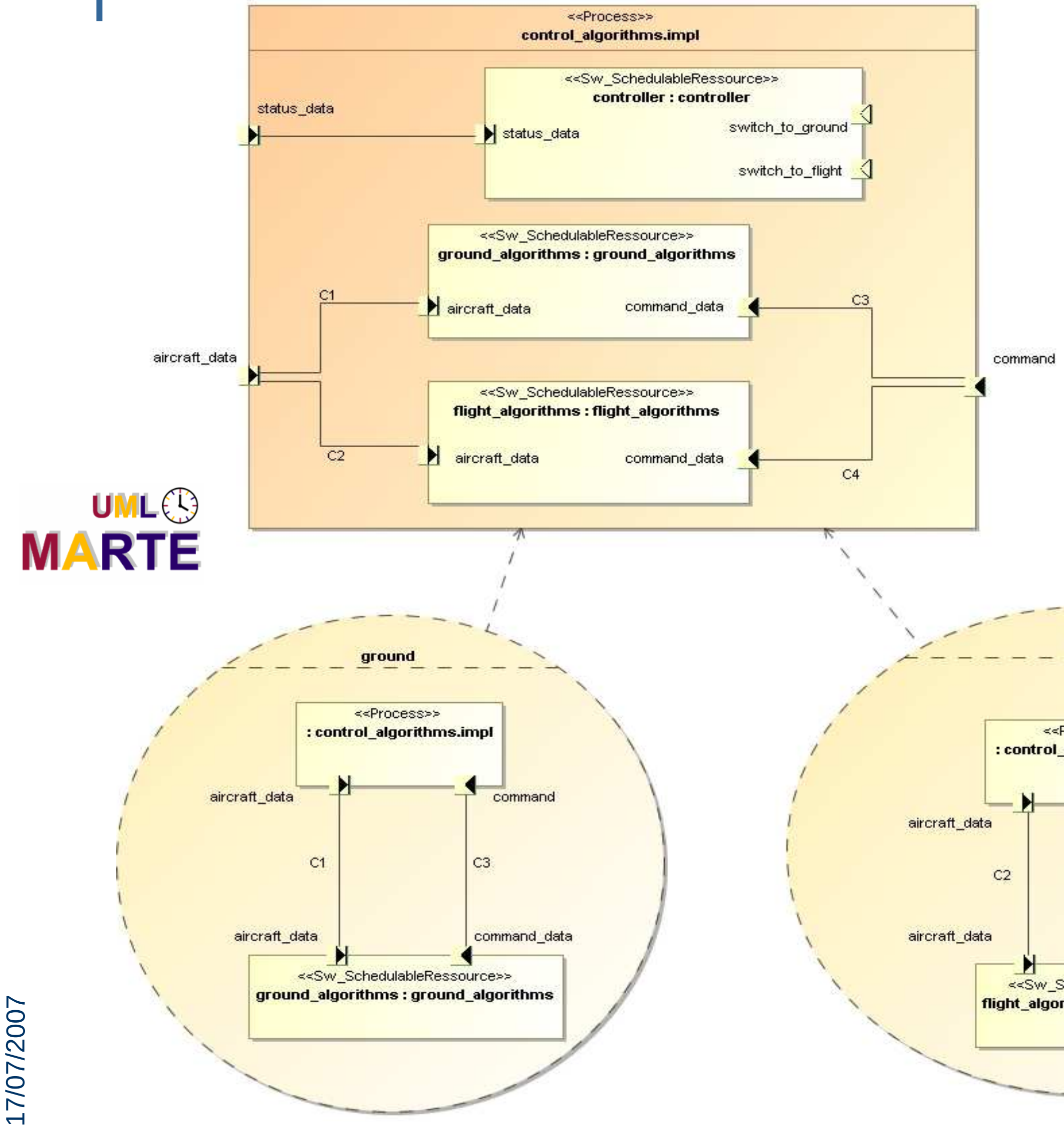
Subprogram are represented as Operation, and calls through UML Messages on sequence diagrams

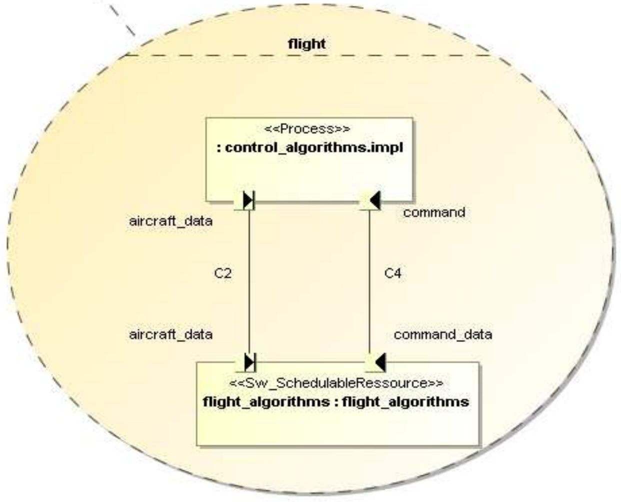
Modes in AADL



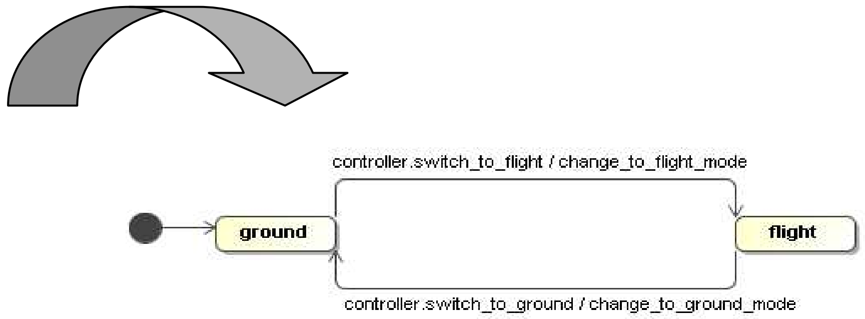
Flight simulator: a dynamic re-configuration thought mode switching

Modes in UML





flight
<<Process>>
: control_algorithms.impl
aircraft_data
command
C2
C4
aircraft_data
command_data
<<Sw_SchedulableRessource>>
flight_algorithms : flight_algorithms



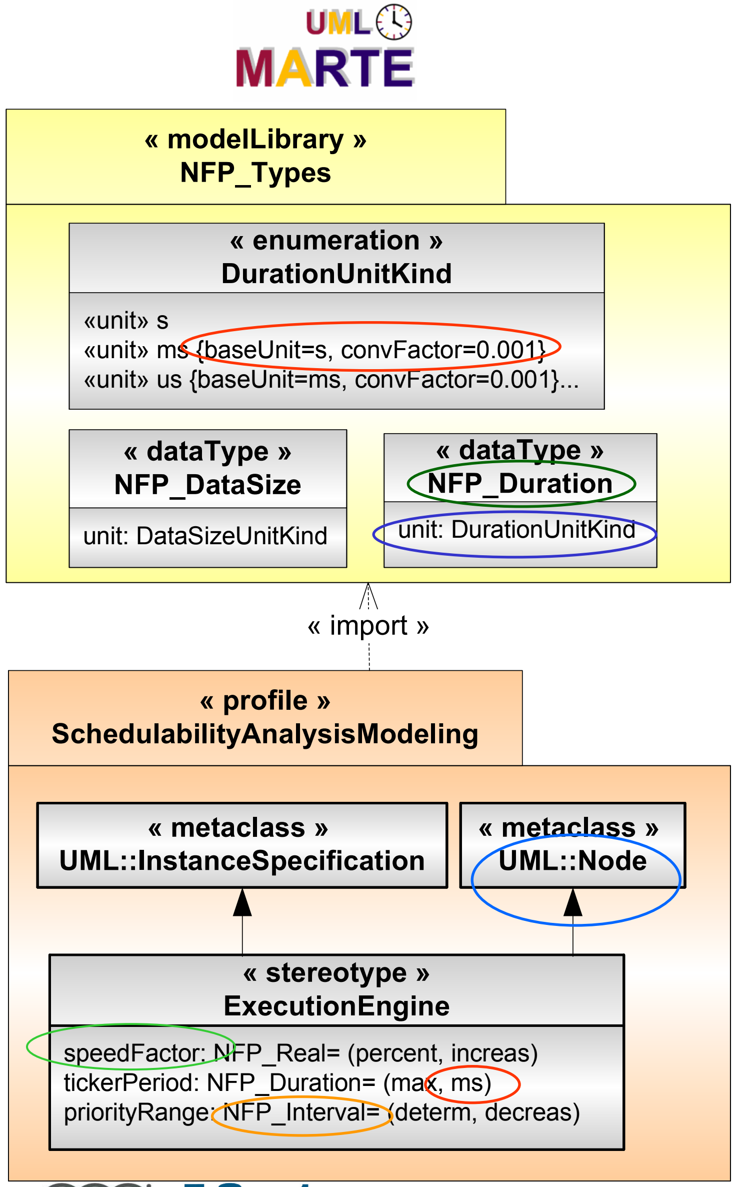
controller.switch_to_flight / change_to_flight_mode
ground
flight
controller.switch_to_ground / change_to_ground_mode

Mode transition modelized by an UML state machine

Different mode configuration through Collaboration diagrams

INRIA

AADL properties


UML MARTE
« modelLibrary »
NFP_Types
« enumeration »
DurationUnitKind
«unit» s
«unit» ms {baseUnit=s, convFactor=0.001}
«unit» us {baseUnit=ms, convFactor=0.001}...
« dataType »
NFP_DataSize
unit: DataSizeUnitKind
« dataType »
NFP_Duration
unit: DurationUnitKind
« import »
« profile »
SchedulabilityAnalysisModeling
« metaclass »
UML::InstanceSpecification
« metaclass »
UML::Node
« stereotype »
ExecutionEngine
speedFactor: NFP_Real= (percent, increas)
tickerPeriod: NFP_Duration= (max, ms)
priorityRange: NFP_Interval= (determ, decreas)

AADL

```
Length_Unit : type units ( mm, cm => mm
* 10, m => cm * 100, km => m * 1000 );

OnOff : type aadlboolean;

Speed_Range : type range of aadlreal 0
.. 250 units ( kph );

mass_t: type aadlreal units mass_u;

mass_u: type units (g, kg => g*1000, t
=> kg*1000);
```

```
Wheel_speed : aadlinteger 0 rpm .. 5000
rpm units ( rpm ) applies to (system);

allowed_mass: mass_range_t applies to
memory, processor, bus, device, system);

actual_mass: mass_t applies to (memory,
processor, bus, device, system);
```

cea list THALES INRIA

Conclusion

- First AADL – MARTE alignment based on AADL constructs and features and MARTE artifacts
- A MARTE2AADL code generator has been developed (Thales RT)

- Future work
  - AADL Profile will be extended by explicit AADL properties modelization