SAE Meeting – Anchorage – October 2007

MARTE: An OMG UML profile for Modeling and Analysis Real-Time and Embedded systems

Madeleine FAUGERE Ph.D
THALES Research & Technology, France
Madeleine.Faugere@thalesgroup.com

cea list THALES INRIA

UML MARTE
www.omgmarte.org

MARTE's sense of being

The rise of UML (1997)

- Too much Object-Oriented approaches, Object-Oriented languages and tools
  - Need to train engineers to a lot of different tools and languages
- Need of abstract views of software facilitating their design
- All these languages and requirements have been unified around a unique, common and shared language: **UML**

Requirements for RTE systems

- Too much specific Real-Time and Embedded approaches, RTE languages and tools
- Big Problem of tools, languages approaches interoperability.
  - Engineers need to understand the specification
- **NEED** of a unified modeling language for RTE Systems

UML is emerging as a solution to address the Real-Time and Embedded domain

- A large audience in the Software Engineering community
- Steady semantic foundations
- Extension capabilities through UML profiles (e.g. SysML)
- But lacks key notions to fully address RTE specifics (time, resource, scheduling)

cea list THALES INRIA

UML MARTE
www.omgmarte.org

In 2005, OMG called for a new UML profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE)

- Modeling and Analysis of Real-Time and Embedded systems
  - **1st UML based standard for Modeling RTE systems**

- MARTE specification adopted in June 2007 (www.omgmarte.org)
  - Beta document available: http://www.omg.org/cgi-bin/doc?ptc/2007-08-04
  - Finalization Task Force comments deadline: December 22nd 2007

cea list THALES INRIA

MARTE submitters

The PROMARTE TEAM

Tool vendors

- ARTiSAN Software Tools*
- International Business Machines (IBM)*
- Mentor Graphics Corporation*
- Softeam*
- Telelogic AB (I-Logix*)
- Tri-Pacific Software
- No Magic
- The Mathworks

Academics

- Carleton University
- Commissariat à l'Energie Atomique
- ESEO
- ENSIETA
- INRIA
- INSA from Lyon
- Software Engineering Institute (Carnegie Mellon University)
- Universidad de Cantabria

Industrial companies

- Alcatel*
- France Telecom
- Lockheed Martin*
- Thales*

* Submitters to the OMG UML for MARTE RFP

cea list THALES INRIA

The MARTE profile requirements

MARTE shall be generic

- "The UML profile for MARTE addresses modeling and analysis of real-time and embedded systems, including their software and hardware aspects" :
  - Provides support for non-functional property modeling
  - Adds rich time and resource models to UML
  - Defines concepts for software and hardware platform modeling
  - Defines concepts for allocation of applications on platforms
  - Provides support for quantitative analysis (e.g. scheduling, performance)

MARTE shall be compliant with other OMG standards

- **NO ALREADY existing concepts from other OMG profiles have been redefined**
  - The UML profile for Modeling QoS and FT Characteristics and Mechanisms - Addressed through MARTE NFP package
  - The UML profile for SoC - more specific than MARTE purpose
  - The Real-Time CORBA profile - Real-Time CORBA based architecture can be annotated for analysis with MARTE
  - The UML profile for Systems Engineering (SysML) - specialization of SysML allocation concepts and reuse of flow-related concepts

MARTE survey

NFPs and Time Modeling

Non Functional Properties (NFP) and Value Specification Language (VSL)

- New standard language for Non Functional Properties
  - Ex: value qualifier, measures, NFP libraries, annotation mechanism
- Extensible with the VSL
  - Ex: describing complex time expressions, variables, structured values…
- Good discussions to introduce this language in UML or SysML

Time Modeling

- Used to explicit any concept linked to a clock during different developpement phases (design, performance/scheduling analysis, etc…)
- Allow to express
  - Causal temporal time dependencies
  - Different time models : chronometric, logical, synchronous
  - Support distribution, clock uncertainties
  - Design vs. Runtime clocks

RT components and RT features

Support for RTE systems based on a component-based approach

- **No new component-based concepts have been introduced**
  - Has to cope with various component models: UML2, SysML, Spirit, AADL, Lightweight-CCM, EAST-ADL2, Autosar, …
- Relies mainly on UML structured classes, on top of which a support for SysML blocks has been added
- Support control flow and data flow architecture

High-level modeling concepts where RT/E concerns are embedded inside modeling artifacts (E.g Uml active/passive objects)

- Quantitative aspects (concurrency, behavior, and communication)
- Qualitative aspects (deadline and period)

Software and Hardware Modeling

Software Resource modeling

- Is not a new API standard dedicated to RTE domain
- provides an unified mean for describing existing / proprietary RTE sw execution APIs

Hardware Resource modeling

- Logical view (functional modeling)
  - Provides a description of functional properties
  - Based on a functional classification of hardware resources (Eg HwComputing resources, HwStorage resources, HwCommunication resources, HwTiming resources, HwDevice resource…)
- Physical view
  - Provides a description of physical properties (HwLayout, HwPower..)

Allocations

- Allocate application elements to an execution platform resources
- Refine a general element into specific elements
- Align with the SysML allocation definition

RTE Analysis

Scheduling Analysis

- Modeling for analysis techniques taking into account scheduling aspects
  - Provides high-level analysis constructs
    - Sensitivity analysis, parametric analysis
    - Observers for time constraints and time predictions at analysis context level
  - Supports most common scheduling analysis techniques
    - RMA-based, holistic techniques and modular techniques

Performance Analysis

- Supports most common performance analysis techniques
  - Queuing Networks and extensions, Petri Nets, simulation

Annexes

MARTE provide following libraries:

- OSEK/VDX OS
- ARINC-653

MARTE Guidance for AADL applications (next part of this presentation)

MARTE: Just a Langage

MARTE is just a Language !! (like UML)

- Offers you standard concepts
- You have to define your methodology

MARTE does not tell you:

- How to design your real-time and embedded systems
- How to model your RTE systems for Analysis
- How to model your RTE system for System C generation
- How to model in abstract way your RTE system Hardware

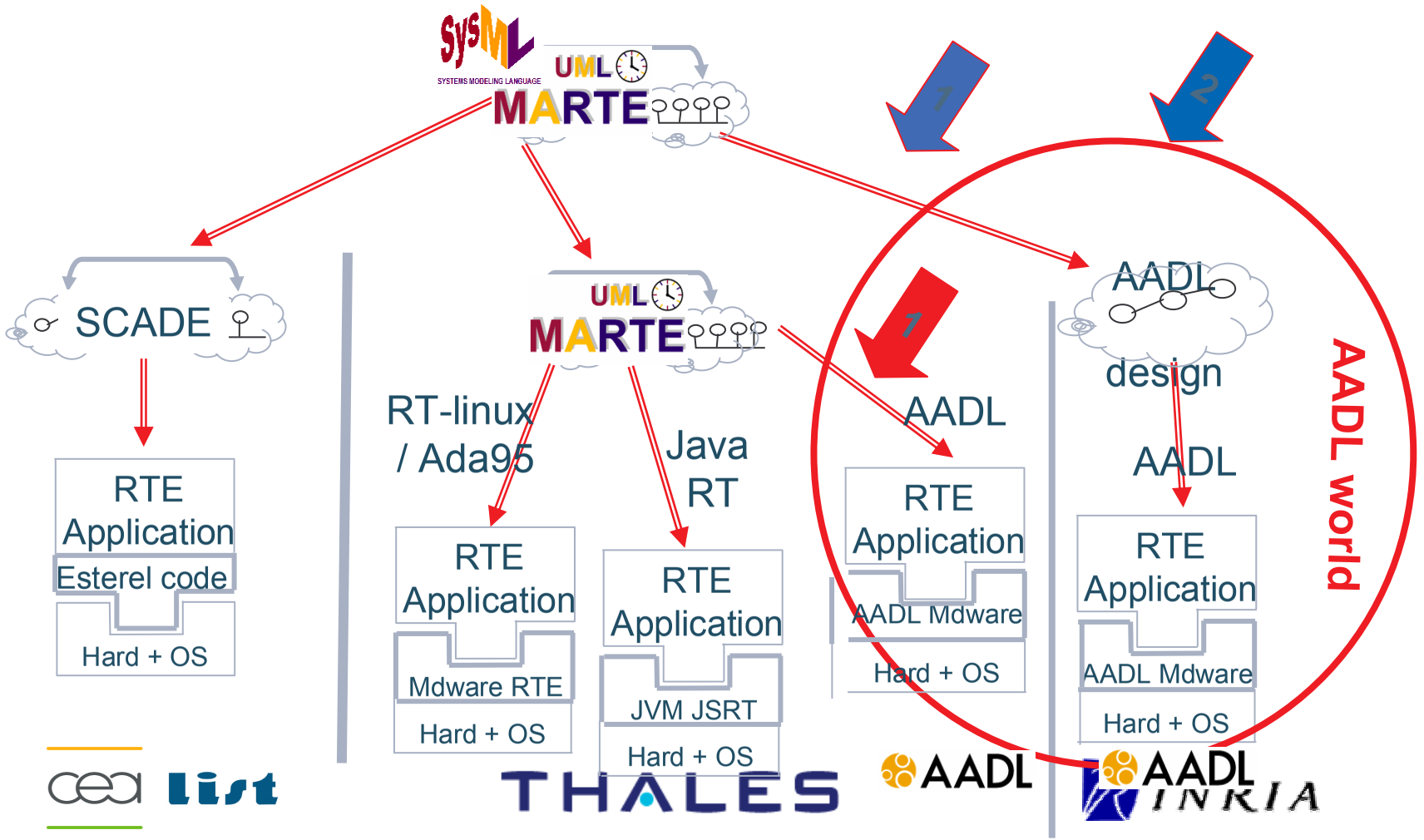
Guidelines for modeling AADL applications

cea list THALES INRIA

AADL guidelines sense of being

Allow UML/MARTE users to model AADL applications

- Consolidate MARTE on the “Unified” aspects
- Let end-users choose RT language and verification/validation tools at different development cycle stages



MARTE-AADL concepts mapping

- First AADL – MARTE alignment based on AADL constructs and features and MARTE artifacts.
- Next step: Deeper map AADL component semantics, AADL properties and implicit AADL platform semantics on MARTE concepts (MARTE concepts properties and NFP, VSL language)

AADL concepts	MARTE concepts
Software components	memoryPartition, swSchedulableRessource,.
Hardware components	hwProcessor, hwMemory,..
Binding	Allocated
AADL features	MARTE flowPorts, UML requ… interfaces…
Subcomponents	UML Parts
Port Connections	UML delegation/assembly
Flow specification	UML object flows
Modes	UML state machines
Properties	Not yet documented

AADL component Type and Implementation

AADL Component Types

- Specifies a functional interface in terms of features (ports, port groups, flow specifications, subcomponent access..), properties
- UML package containing AADL component declaration)

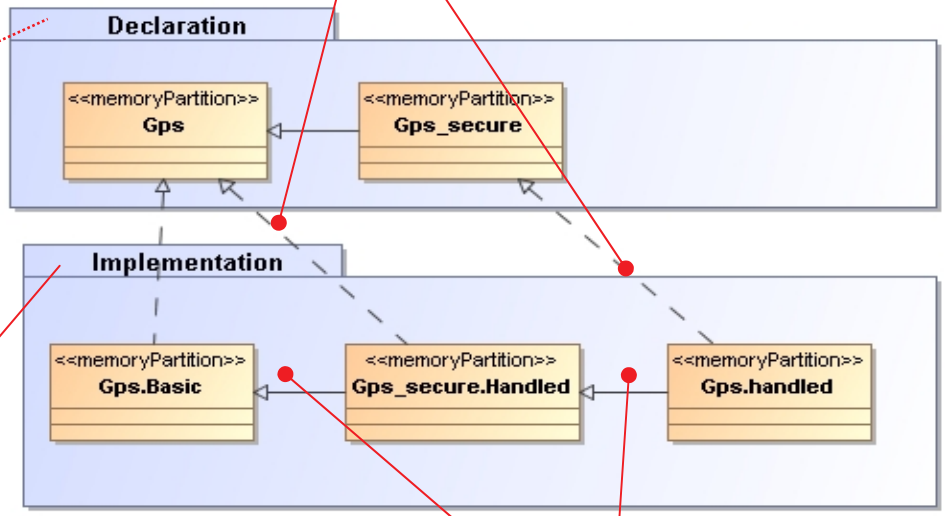
AADL Component Implementations

- describes the internal structure and behavior of that component in terms of subcomponents, connections and flows across them, and behavioral modes
- UML package containing AADL component implementations
- AADL implementation liked to AADL declaration through UML Realization

AADL Component Extension

- UML Generalization

Component Implementation link: UML Realization



Component Extension : UML Generalization

cea list THALES INRIA

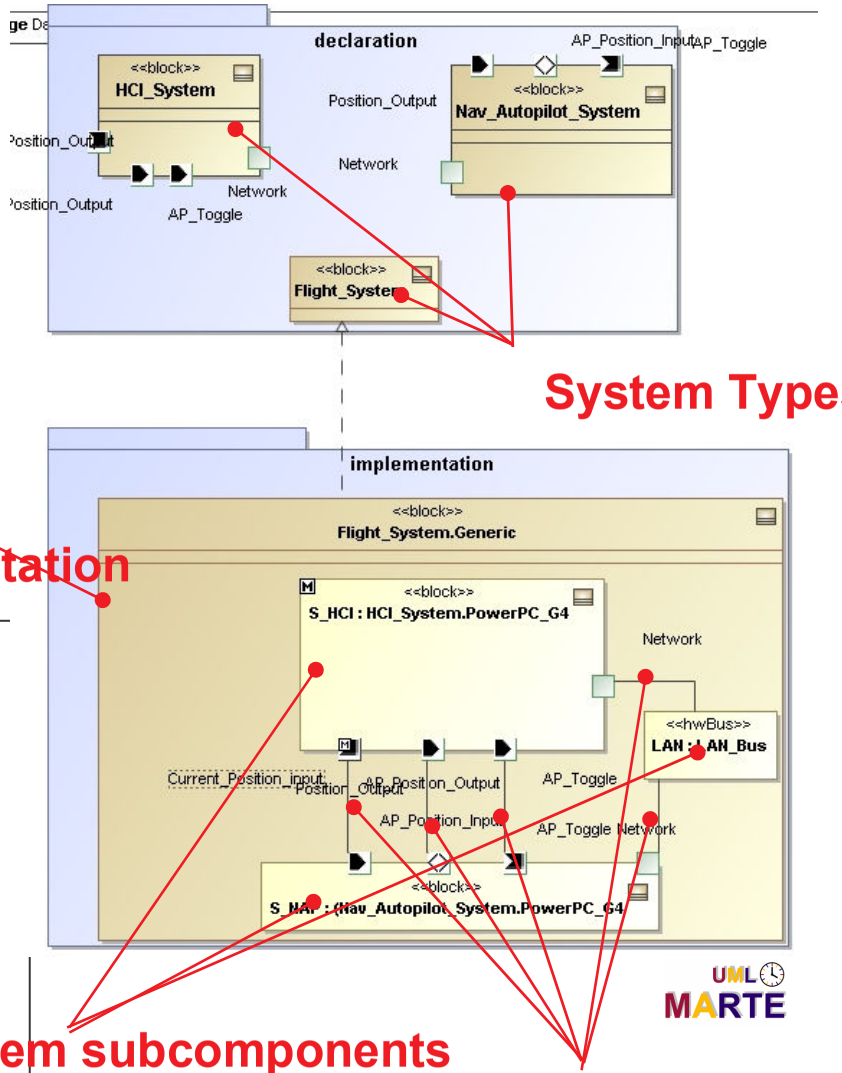
AADL System and AADL Subcomponents

AADL System

- Represents a composite software, execution platform or system components
- Represented by a “SysML” block

AADL Subcomponents

- represented by UML parts



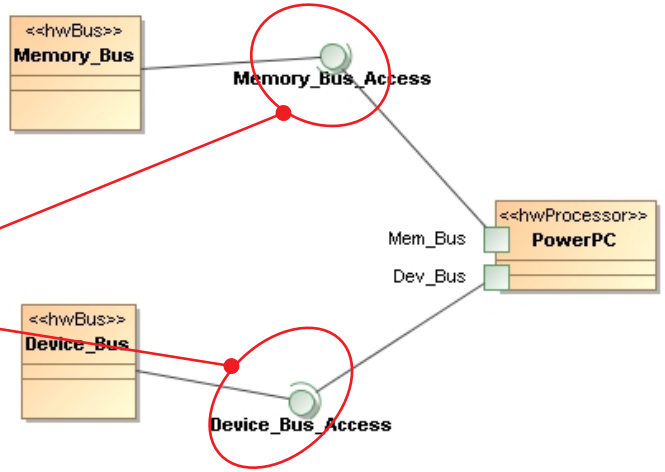

```
system Flight_System
end Flight_System;

system implementation Flight_System.Generic
subcomponents
        S_HCI : system HCI_System.PowerPC_G4;
        S_NAP : system Nav_Autopilot_System.PowerPC_G4;

        LAN : bus LAN_Bus;          ....
```

AADL

Required data/bus access

AADL Bus and Data components access

- Provide data/service to other AADL components
- Access required from other AADL components
- Declaration part : Provide/required access modeled in MARTE via provided / required UML Interfaces
- Implementation part :
  - access design by delegation / assembly connectors
  - Resources may be specialized with real time features or associated to services (synchronization, concurrency access …)

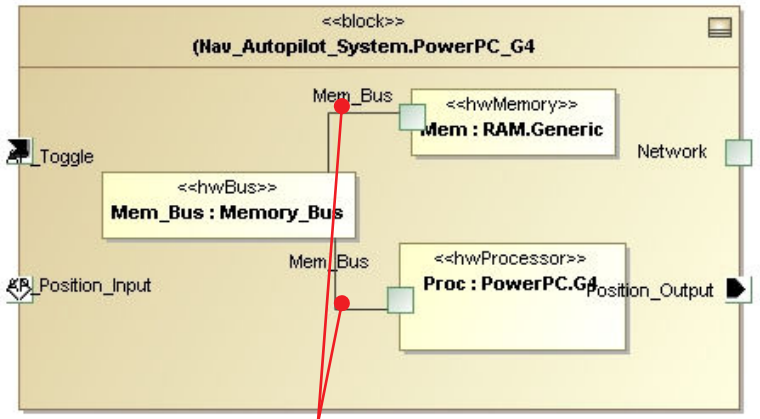


processor **powerPC**
Features
MemBus : requires **bus access Memory_Bus;**
Dev_Bus : requires **bus access Device_Bus**
End PowerPC;

bus **Memory_Bus**
end Memory_Bus;

AADL

Access declaration



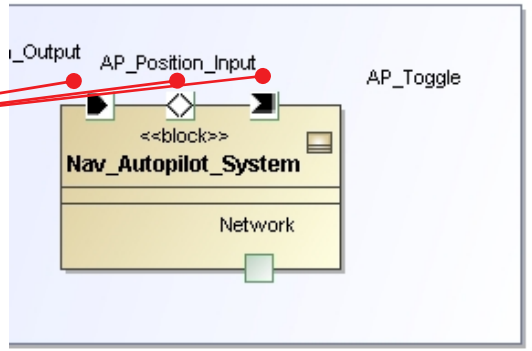
Access connections

cea list THALES INRIA UML MARTE

Ports and Connections

Ports

- Flow ports are represented by MARTE Flow Ports



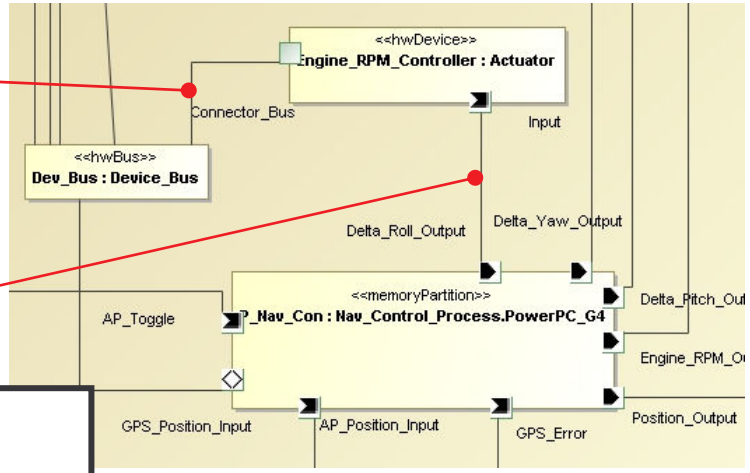

```
system Nav_Autopilot_Systemfeatures
        AP_Toggle : in event port;
        AP_Position_Input : in event data port Nav_Types::Position.GPS; …
```

AADL

Port, Bus and Memory Connections

- Bus/data access data are represented by UML connectors between the port providing the access and the device

- Flows ports connections are represented by UML connectors (delegation or assembly connectors)



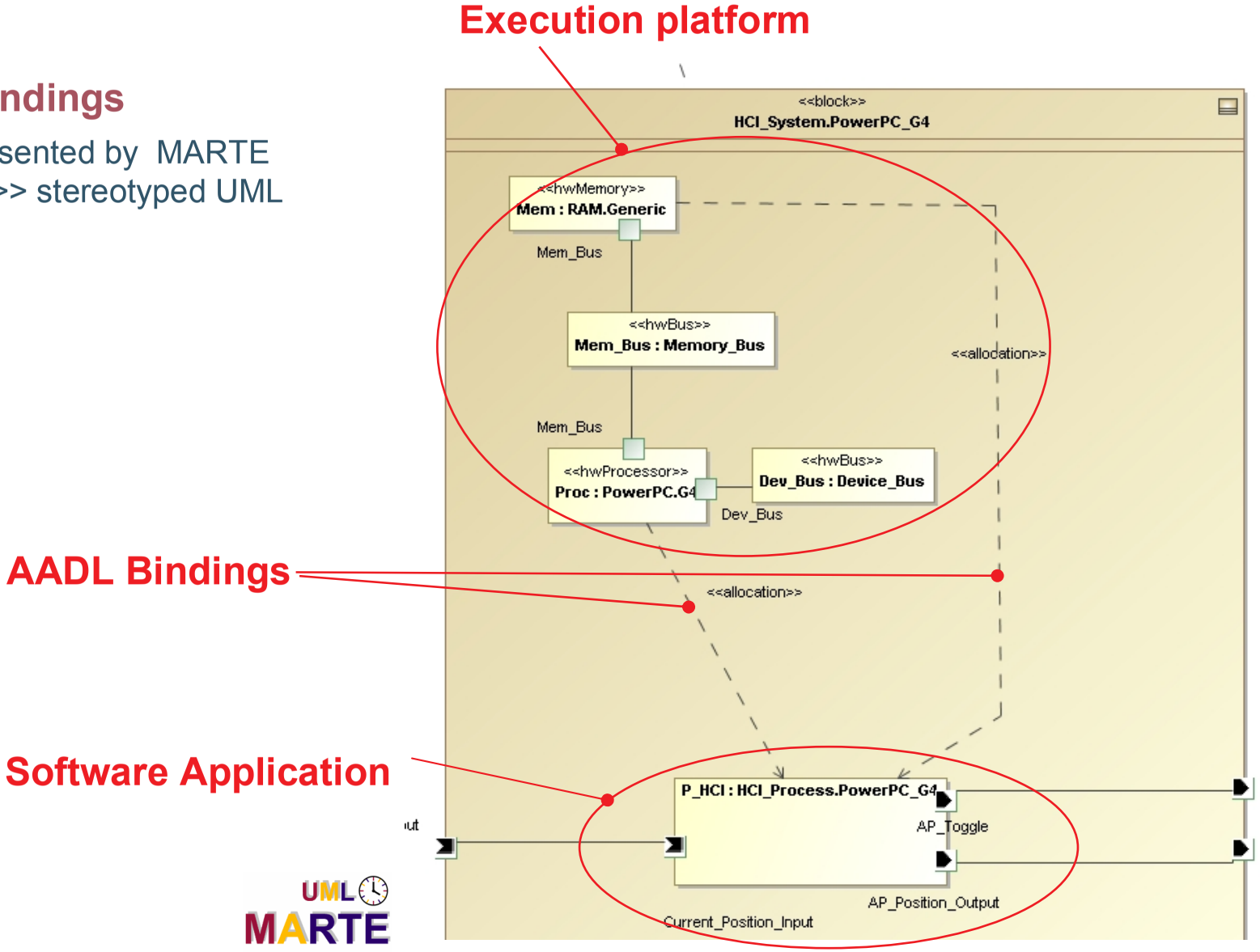

```
system implementation Nav_Autopilot_System.PowerPC_G4
….
bus access Dev_Bus -> Engine_RPM_Controller.Connector_Bus;
….
data port P_Nav_Con.Engine_RPM_Output -> Engine_RPM_Controller.Input ;
…
```

AADL

System binding

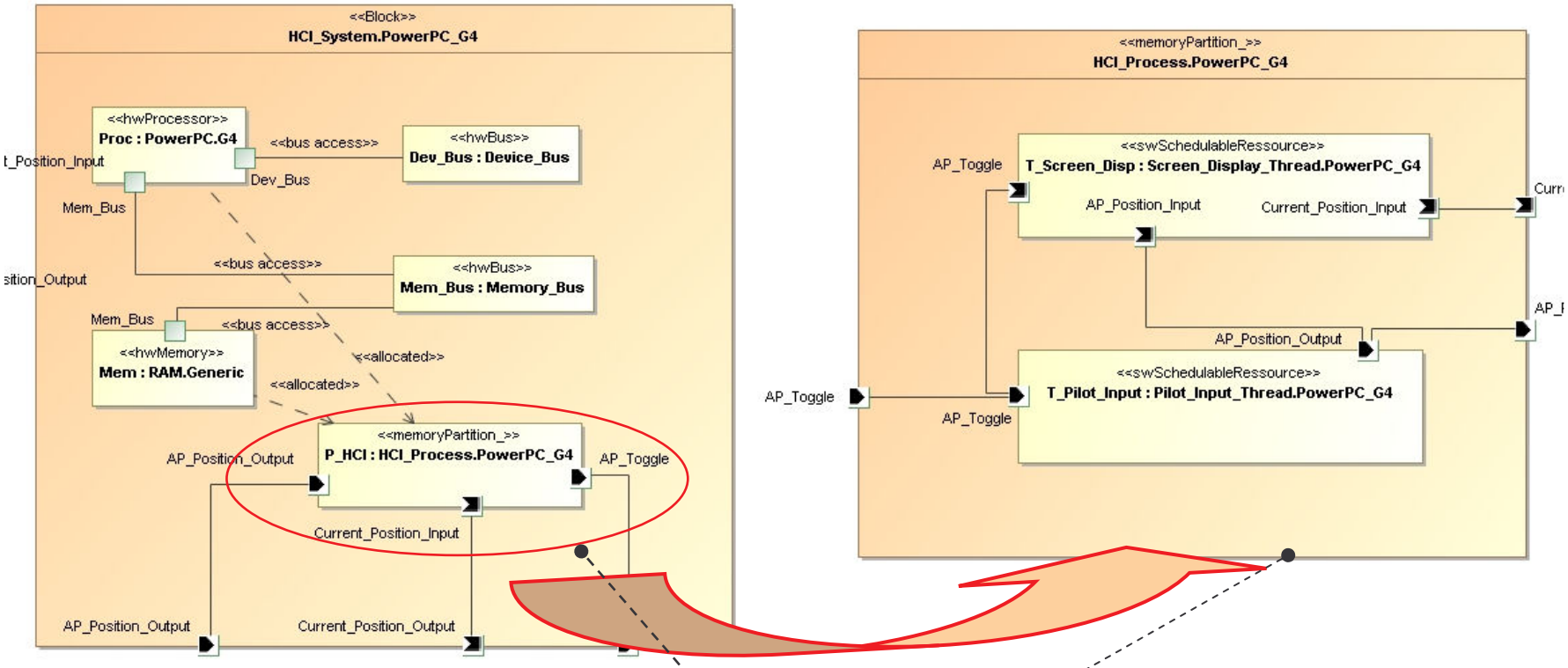
System bindings

- Are represented by MARTE <<allocation>> stereotyped UML Dependency



UML Hierarchical view

Nested hierarchical views in UML

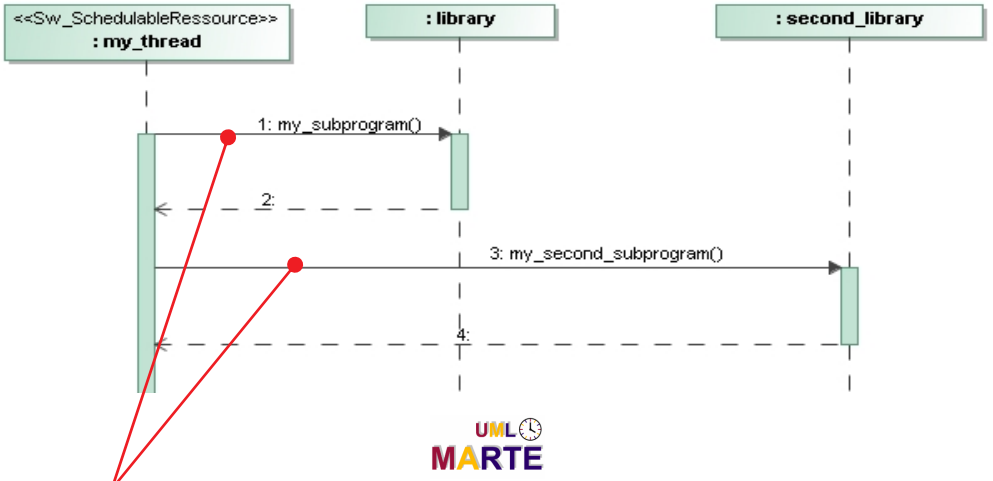


Software application

Subprogram call

Subprogram

- Are represented by UML Operations
- May be specialized with RT features (PpUnit,…)



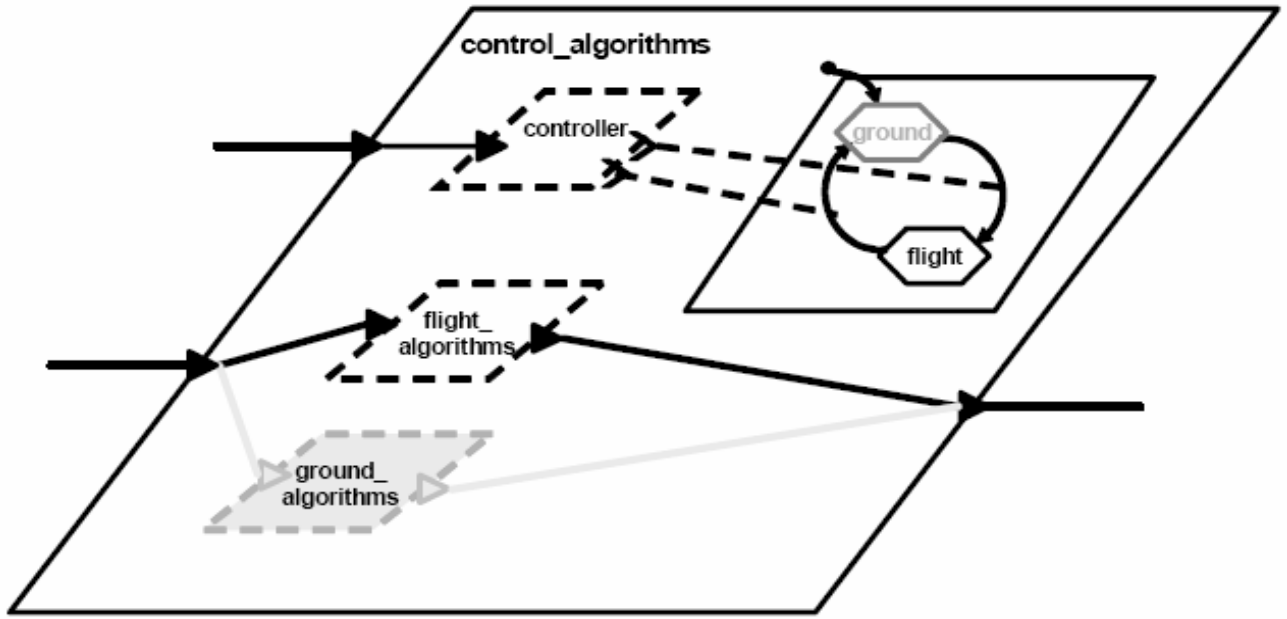
Subprogram calls

- Are represented through UML Messages on sequence diagrams

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

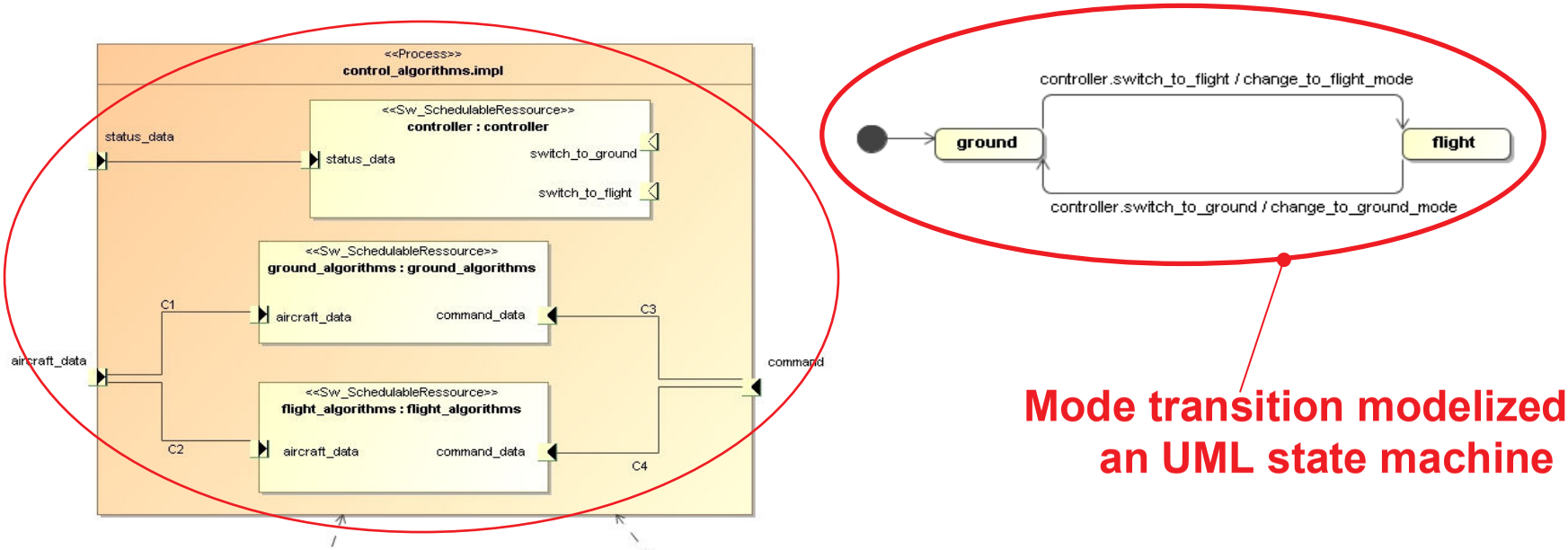
AADL

Modes in AADL

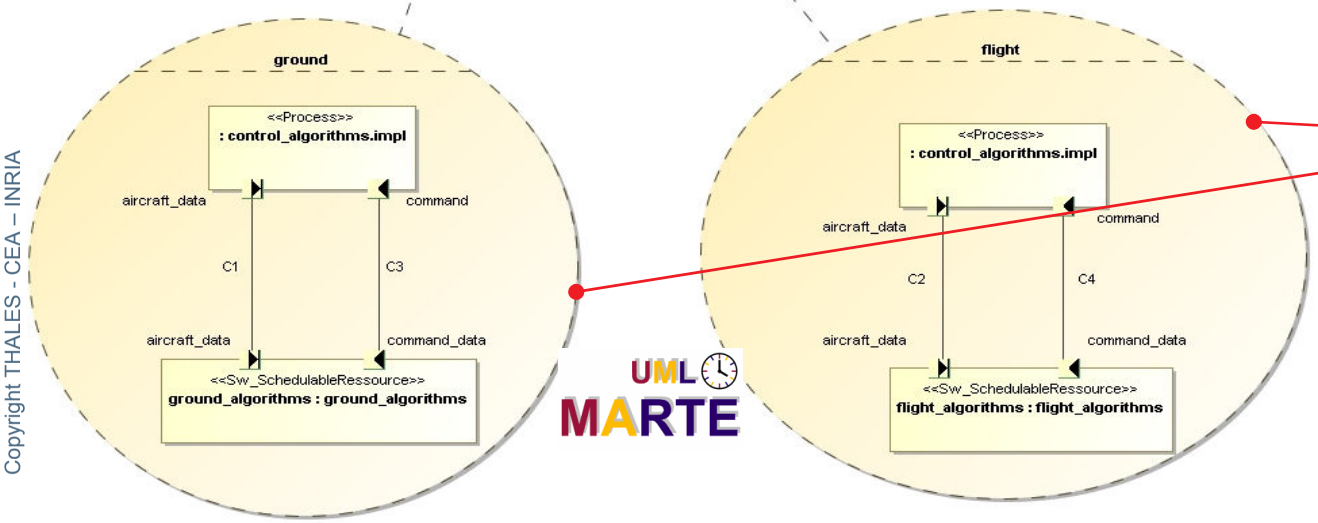


Flight simulator: a dynamic re-configuration thought mode switching

Modes with MARTE guidelines



Mode transition modelized by an UML state machine

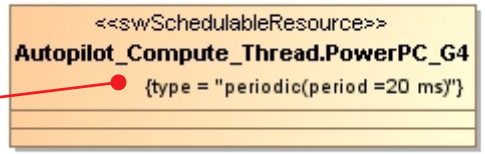


Different mode configuration through Collaboration diagrams

Properties

AADL Properties

- Are represented by <<AADL properties>> stereotyped UML comments
- Will be aligned using NFPs, VSL language and MARTE concepts properties



UML MARTE

cea list THALES INRIA

MARTE-AADL code generator

- MARTE to AADL code generator is already available
- Bridge between MARTE/AADL and Cheddar (Scheduling analysis) already tested

```
process implementation Nav_Control_Process.PowerPC_G4
subcomponents
                T_GPS_Reader : thread GPS_Sampling_Thread.PowerPC_G4 in modes (GPS_UP_AP_UP, GPS_UP_AP_DOWN);
                T_AP_Compute : thread Autopilot_Compute_Thread.PowerPC_G4 in modes (GPS_UP_AP_UP);  T_AP_Params : thread
Autopilot_Modify_Parameters_Thread.PowerPC_G4;
                D_AP_Destination : data Nav_Types::Position.Simple;
                D_AP_Airspeed : data Nav_Types::Integer;
                D_AP_Altitude : data Nav_Types::Integer;
connections
                data port GPS_Position_Input -> T_GPS_Reader.Position_Input in modes (GPS_UP_AP_UP, GPS_UP_AP_DOWN);
                data port T_GPS_Reader.Position_Output -> Position_Output in modes (GPS_UP_AP_UP, GPS_UP_AP_DOWN);
                data port T_GPS_Reader.Position_Output -> T_AP_Compute.Position_Input in modes (GPS_UP_AP_UP);
                T_AP_Compute.Delta_Roll_Output -> Delta_Roll_Output in modes (GPS_UP_AP_UP);
                data port T_AP_Compute.Delta_Yaw_Output -> Delta_Yaw_Output in modes (GPS_UP_AP_UP);
                data port T_AP_Compute.Delta_Pitch_Output -> Delta_Pitch_Output in modes (GPS_UP_AP_UP);
                data port T_AP_Compute.Engine_RPM_Output -> Engine_RPM_Output in modes (GPS_UP_AP_UP);
                event data port AP_Position_Input -> T_AP_Params.AP_Position_Input;          modes
                (GPS_UP_AP_DOWN : initial mode;
                GPS_UP_AP_UP : mode;
                -- <INITIAL_MODE> -[ <EVENT> ]-> <FINAL_MODE>
                GPS_UP_AP_DOWN -[ AP_Toggle ]-> GPS_UP_AP_UP;
                GPS_UP_AP_DOWN -[ GPS_Error ]-> GPS_DOWN;
                GPS_UP_AP_UP -[ GPS_Error ]-> GPS_DOWN;
end Nav_Control_Process.PowerPC_G4; …
…
process implementation HCI_Process.PowerPC_G4
subcomponents
                T_Screen_Disp : thread Screen_Display_Thread.PowerPC_G4;
                T_Pilot_Input : thread Pilot_Input_Thread.PowerPC_G4;
```

Conclusion

- Guidelines consolidation (+ AADL v2 alignment)
- Remarks, corrections, precisions are welcome
  - MARTE issues must to be sent before 22 December 2007
- MARTE v1.0 planned for the 8 July 2008

- Examples from the AADL info website have been modeled with these guidelines
- Reused AADL tools (TopCased, Cheddar,…)

MARTE Website: www.omgmarte.org

- Adopted Specification
- Tutorials, presentations
- implementations
- MARTE Models, examples …
- News