Agenda

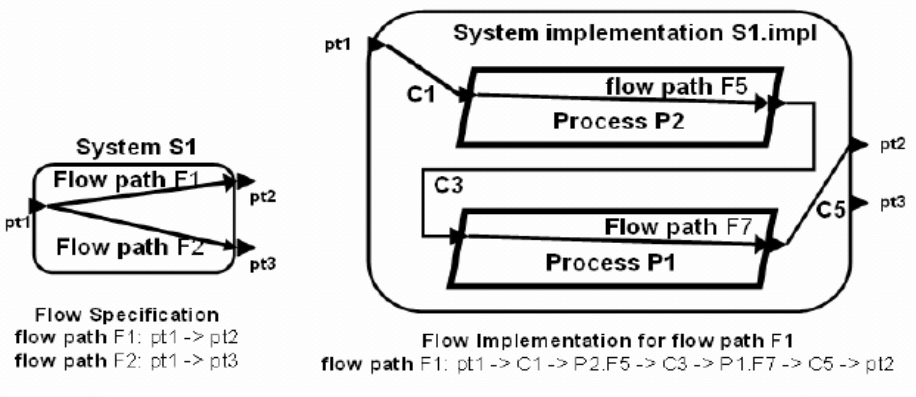
- Part 1
  - Introduction to MDD for RT/E systems & MARTE in a nutshell
- Part 2
  - Non-functional properties modeling
  - Outline of the Value Specification Language (VSL)
- Part 3
  - The timing model
- Part 4
  - A component model for RT/E
- Part 5
  - Platform modeling
- Part 6
  - Repetitive structure modeling
- Part 7
  - Model-based analysis for RT/E
- Part 8
  - **MARTE and AADL**
- Part 9
  - Conclusions

AADL Overview

AADL Architecture Analysis Description Language

- Architecture Description Language dedicated to RTES
- International standard at SAE (AS5506, 2004)
- Adapted for many critical computer system domain
  - Automotive, space, robotics, industrial control, medical, avionics, …
- Allow specification, analysis and automated integration of real-time performance critical
  - Timing,
  - Safety,
  - Schedulability,
  - Fault tolerant,
  - Security,
  - Distributed computing systems,….

What does AADL look like ?



AADL

A graphical representation

```
  compute12: data port compute1.outed ->> compute2.ined;
  compute23: data port compute2.outed -> compute3.ined;
  actuateconn: data port compute3.outed ->> actuate.ined;
  bus access db -> sense.devbus;
  bus access db -> actuate.devbus;
flows
  etelatency: end to end flow sense.flow1 -> senseconn -> com-
pute1.flow1
            -> compute12 -> compute2.flow1 -> compute23 -> com-
pute3.flow1
            -> actuateconn -> actuate.flow1 { latency => 153 ms;};
end application.twosamplesteps;
```

A textual representation

Guidelines for modeling AADL application in MARTE

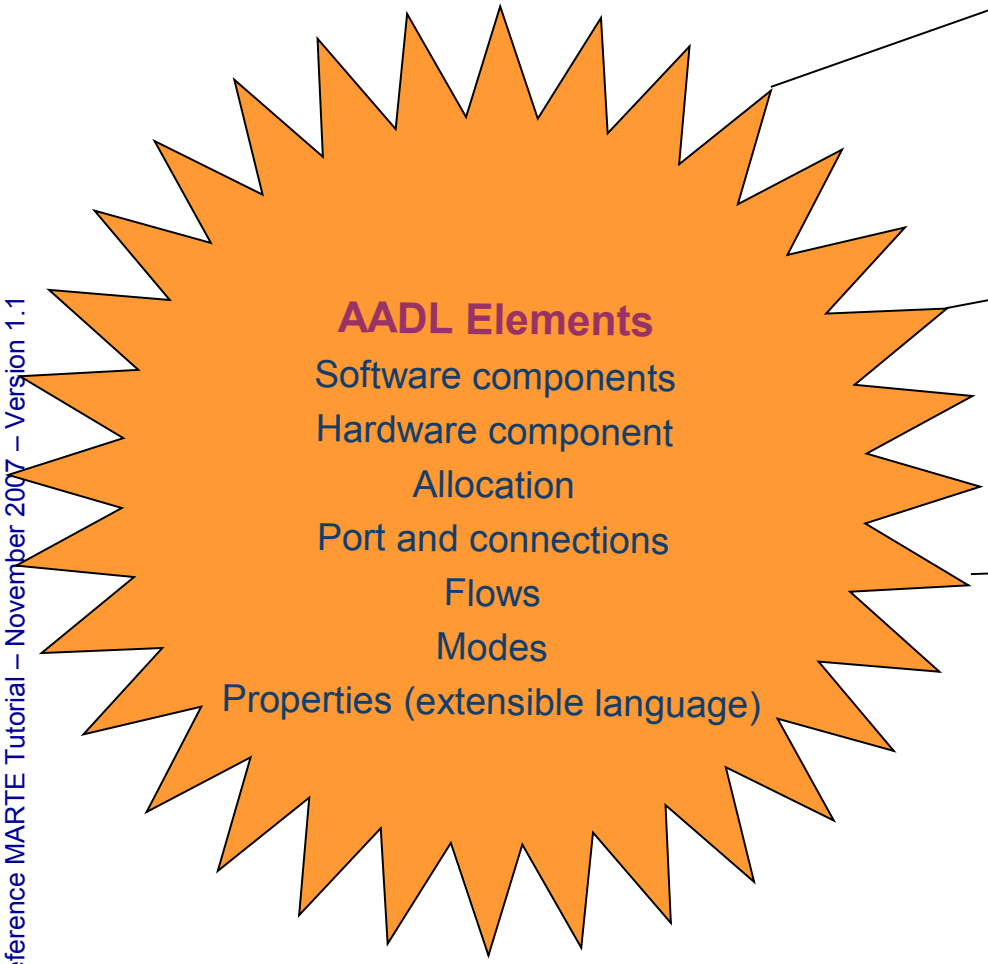
MARTE

- Generic for Real time Embedded System application modeling and analysis
- Address early and detailed design stages
- Complementary and consistent views make the model more understandable
  - Platform execution model can be explicitly modelized
  - Full integration of non-functional properties in the model (Time, performance, scheduling features…)

AADL

- Specific to synchronous data flow application modeling and analysis
- Address detailed design stages
- Based on an implicit execution platform model
  - specific thread execution automata
  - Applications and platform execution semantics have to be in line
- Lack of non-functional properties model integration

AADL useful for….



Non functional aspects through properties specification

- Temporal, safety, reliability,…

Architecture and design

- Components and interfaces, connections
- Data and control flows
- Run time architecture

Analysis and verification aspects

- End-to-end latency
- Age of signal data and jitter
- …

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, wSchedulableRessource,.
Hardware components	hwProcessor, hwMemory,.
Binding	Allocated
AADL features	MARTE flowPorts, UML required/provided 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

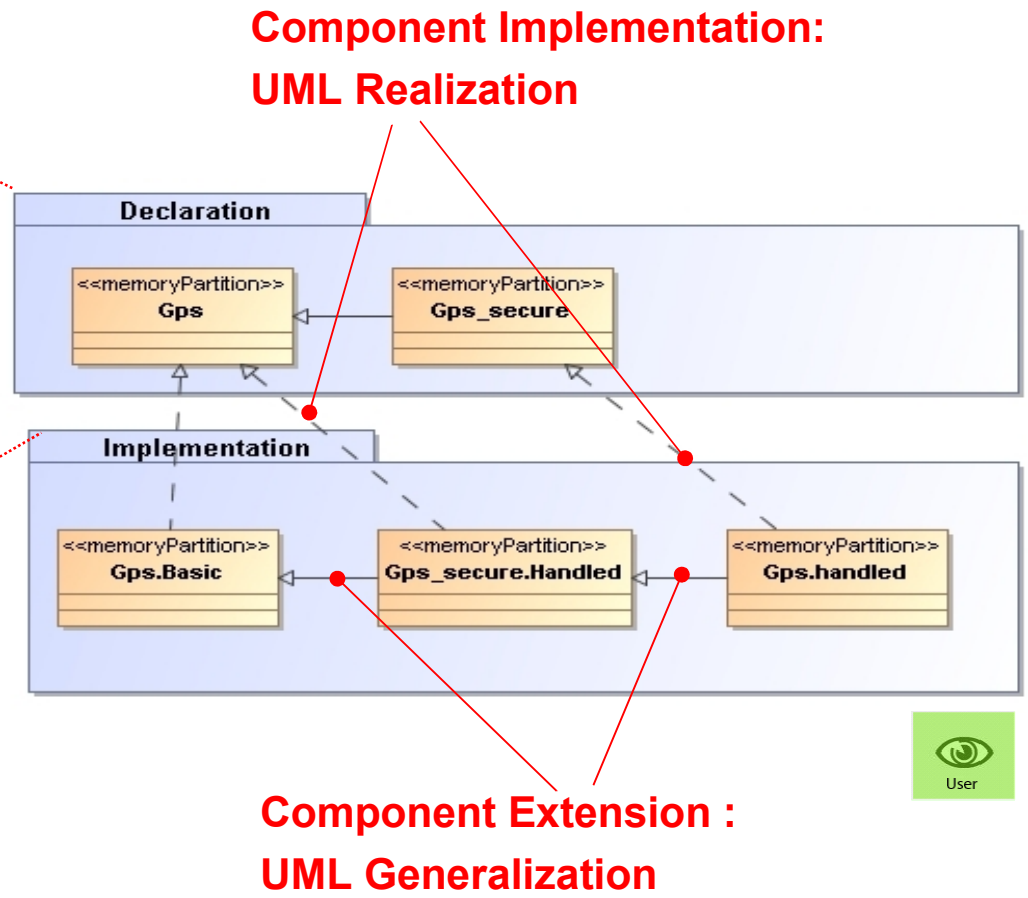
- Specifies a functional interface in terms of features (ports, port groups, flow specifications, subcomponent access..), properties
- UML package containing AADL component declaration)

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

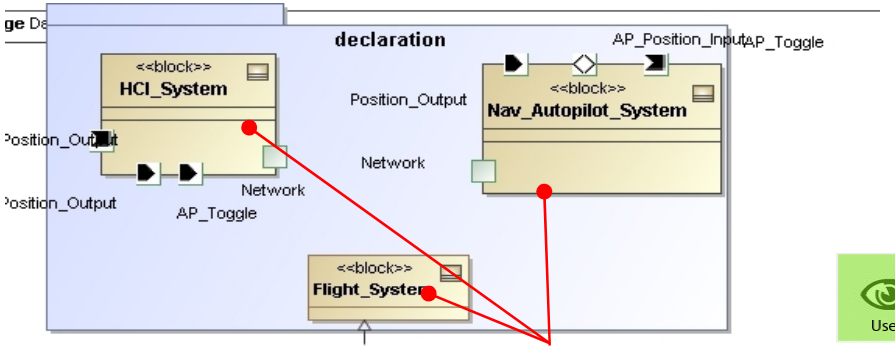
- UML Generalization



AADL System and AADL Subcomponents

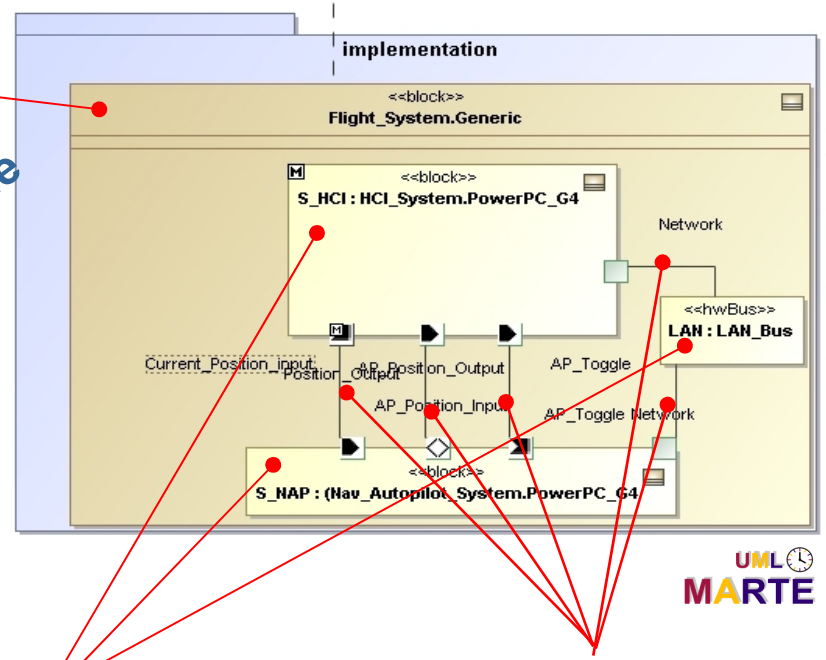
- **AADL System**
  - Represents a composite software, execution platform or system components
  - Represented by a "SysML" block

- AADL Subcomponents
  - represented by UML parts



System Types

System Implementation



Generated code

```
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;   ….
```

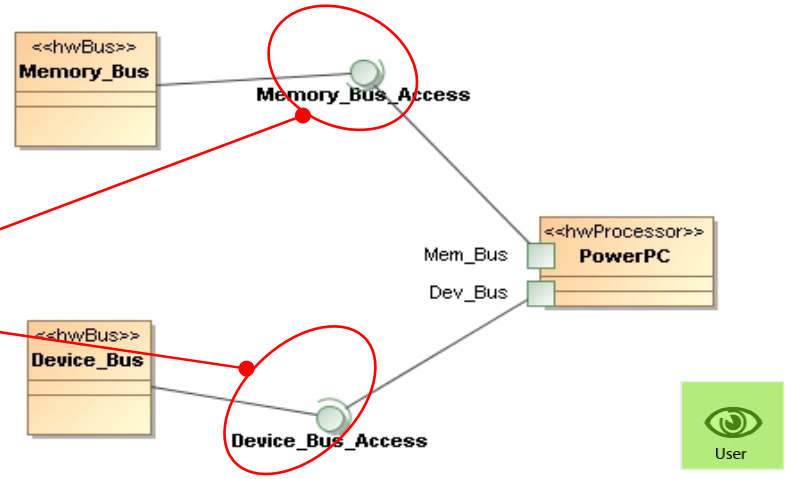
AADL

System subcomponents

Subcomponents relationships

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 …)



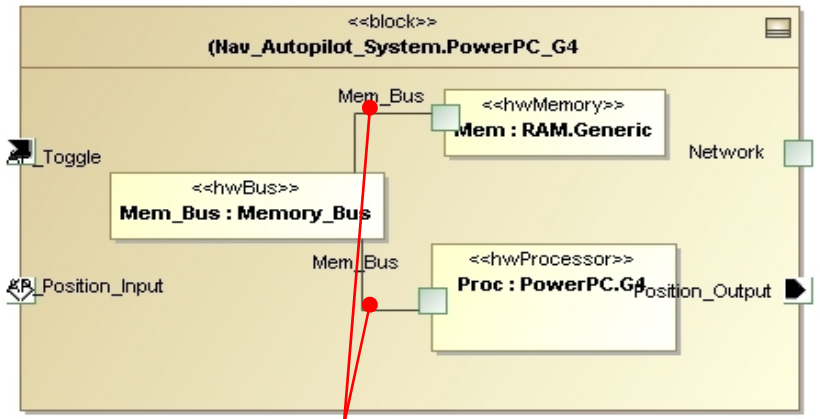
Access declaration

Generated code

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

bus Memory_Bus
end Memory_Bus;
```

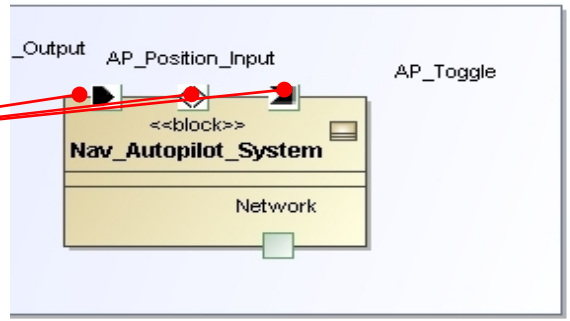
AADL



Access connections

Ports and Connections

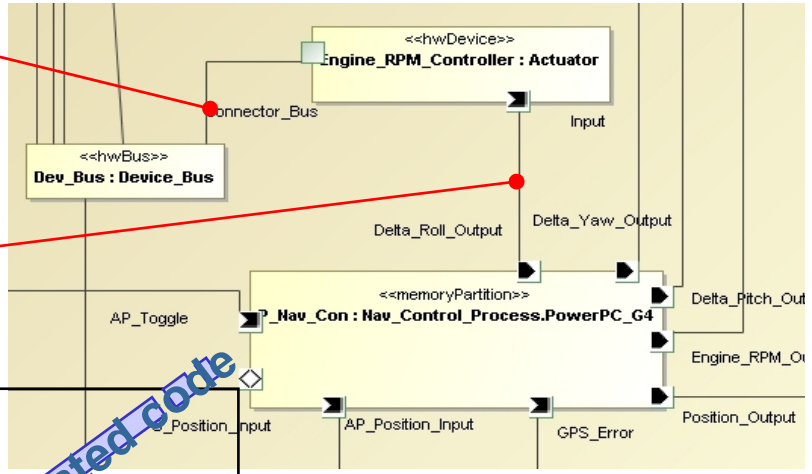
- **AADL Ports**
  - Flow ports are represented by MARTE Flow Ports




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

AADL

- **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)**

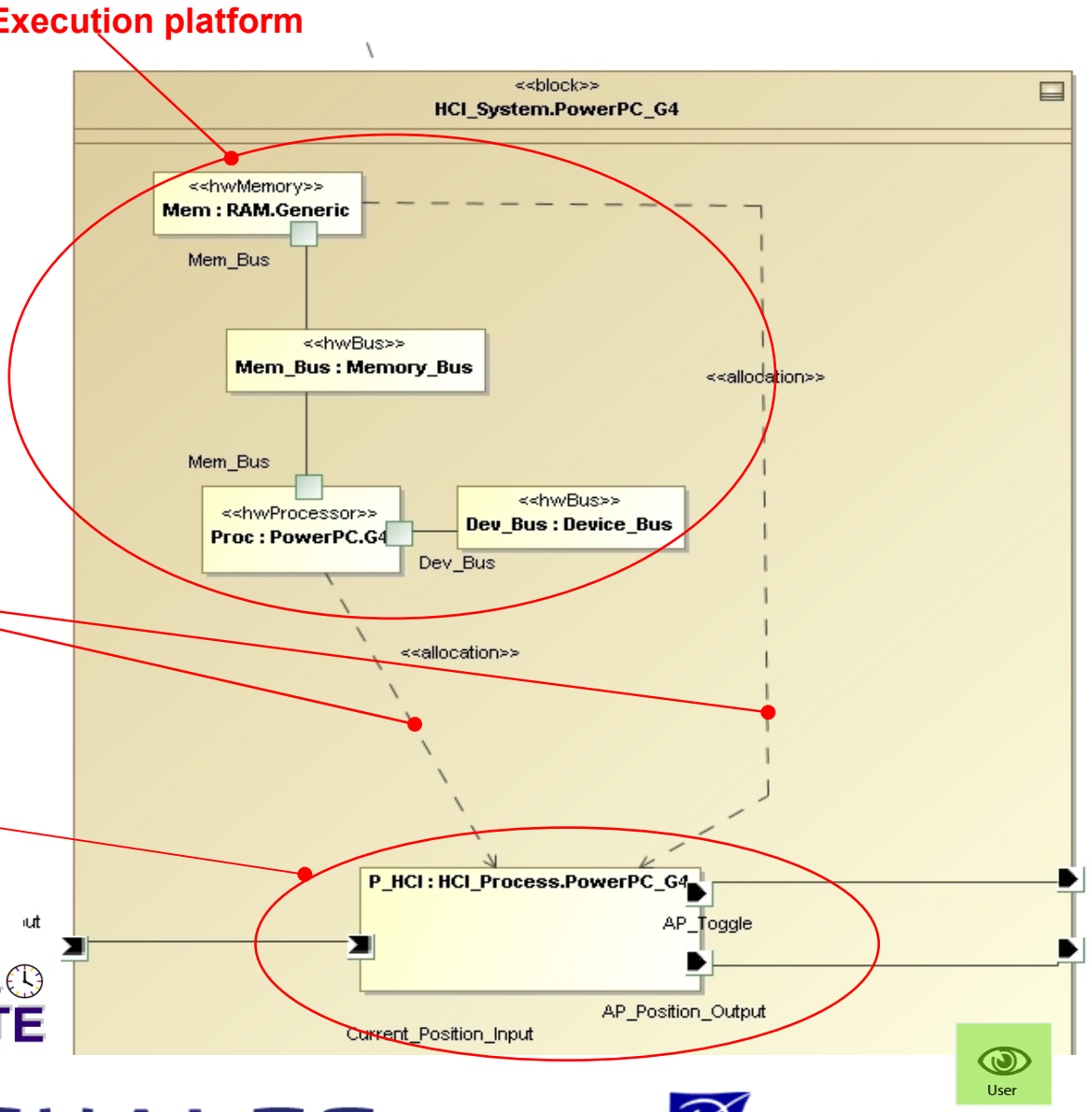
```
system implementation Nav_Autopilot_System.PowerPC_G4
….
bus access Dev_Bus -> Engine_RPM_Controller.Connector_Bus;
….
data port P_Nav_Con.Engine_RPM_Output -> Engine_RPM_Controller.Input ;
…
```

Generated code

AADL

System binding

- **AADL System bindings**
  - Are represented by MARTE <<allocation>> stereotyped UML Dependency



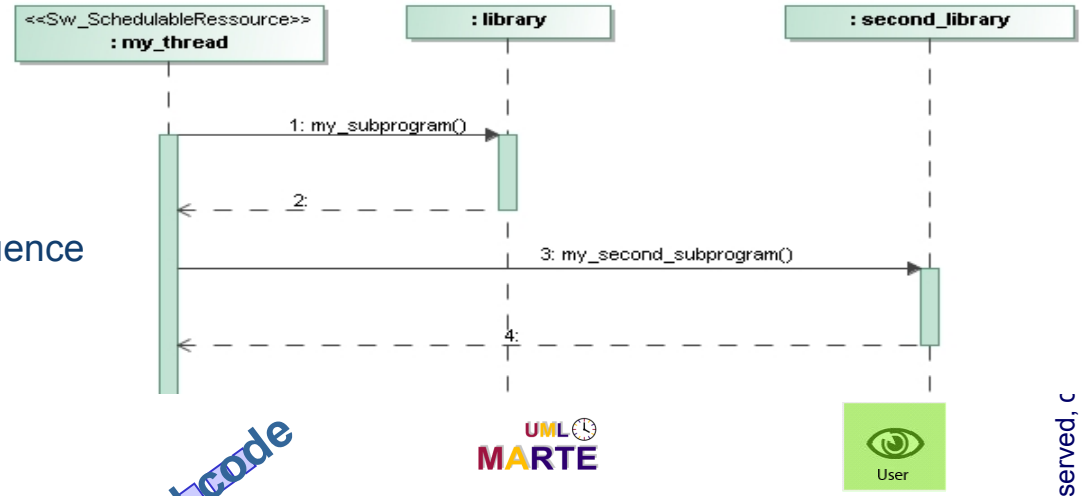
Subprogram call

Subprogram

- Are represented by Operation

Subprogram calls

- Are represented through UML Messages on sequence diagrams

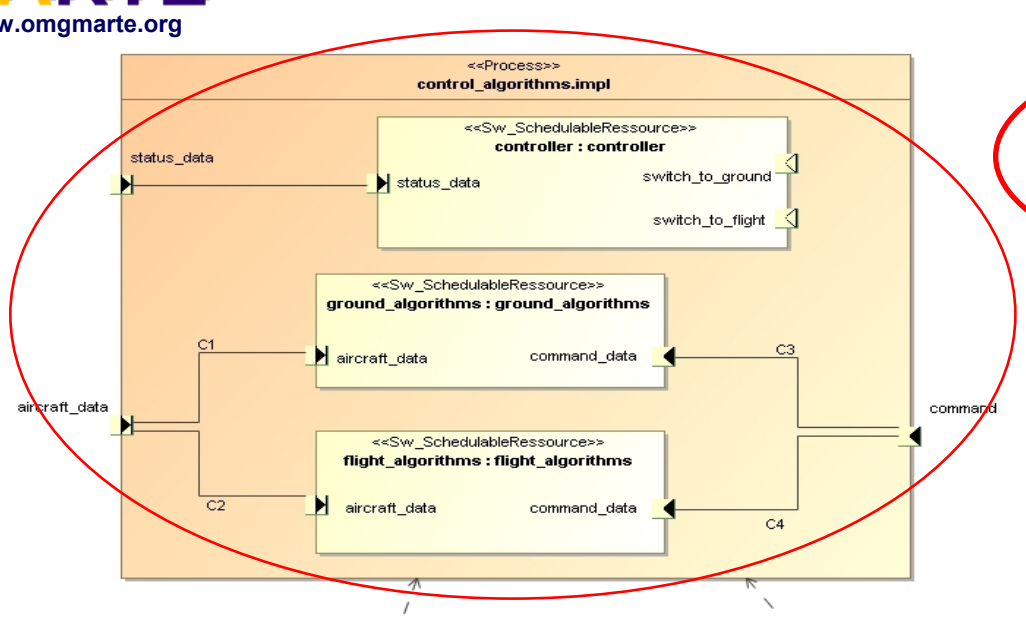


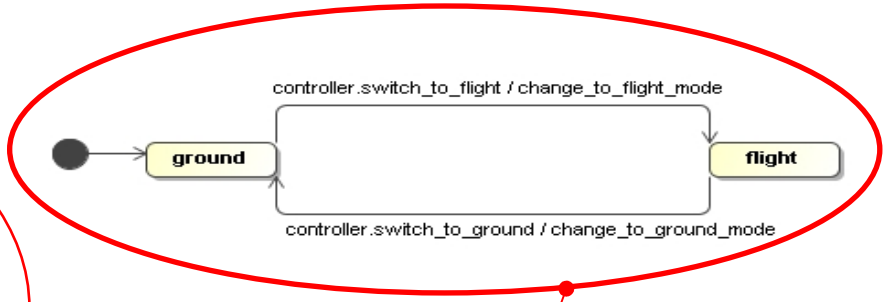
Generated code

AADL

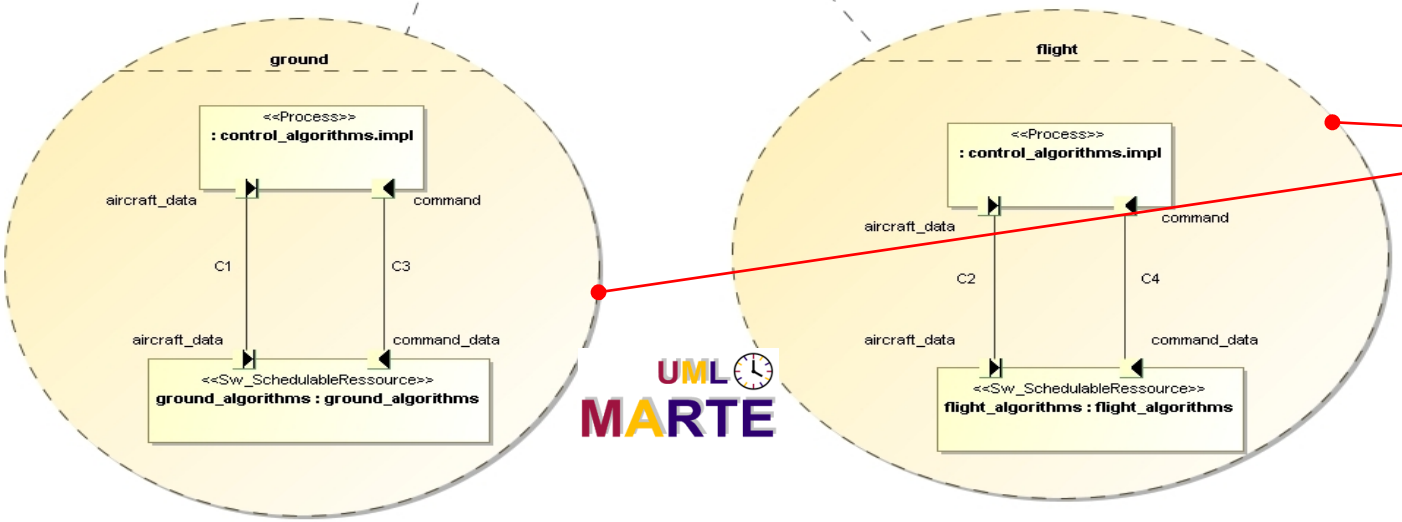
```
thread implementation my_thread.impl
calls {
        first_subpgr : subprogram my_subprogram;
        second_subpgr : subprogram my_second_subprogram;
    }
end my_thread.impl;
```

Modes with MARTE guidelines





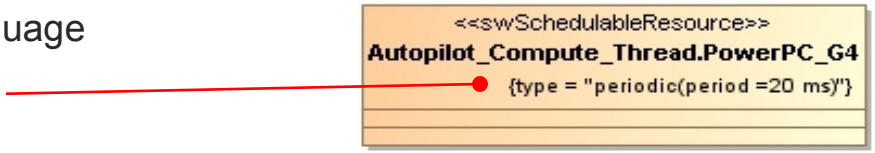
Mode transition modelized by an UML state machine



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

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_Display_Thread : thread Screen_Display_Thread.PowerPC_G4;
                T_Pilot_Input : thread Pilot_Input_Thread.PowerPC_G4;
connections
                event port T_Pilot_Input.AP_Toggle -> AP_Toggle;
                event data port T_Pilot_Input.AP_Position_Output -> AP_Position_Output;
                T_Pilot_Input.AP_Toggle -> T_Screen_Disp.AP_Toggle;
```

CODE GENERATED