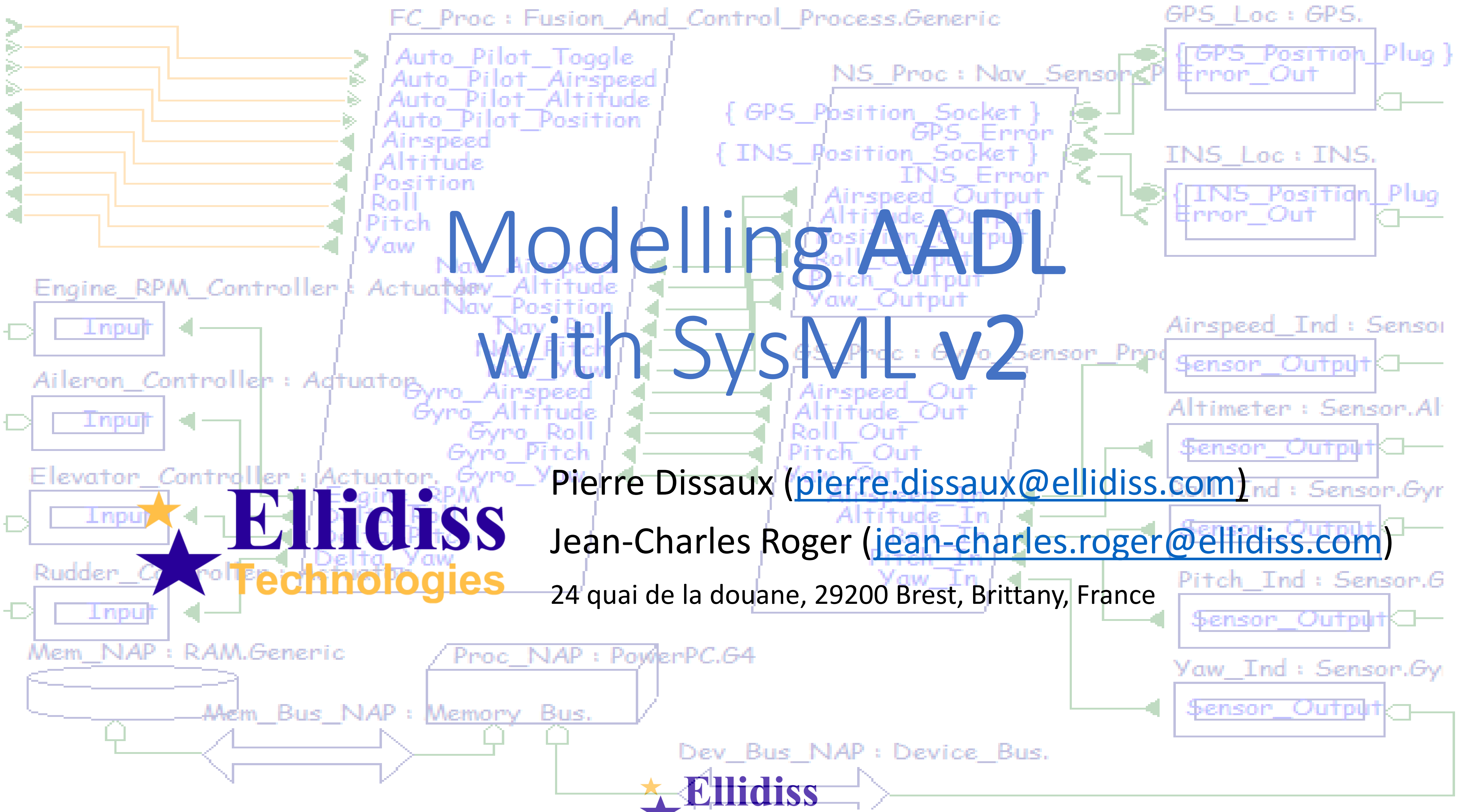


Modelling AADL with SysML v2

Ellidiss Technologies

Pierre Dissaux (pierre.dissaux@ellidiss.com)

Jean-Charles Roger (jean-charles.roger@ellidiss.com)

24 quai de la douane, 29200 Brest, Brittany, France

Ellidiss Technologies

Ellidiss Technologies

https:/www.ellidiss.com

Methods and Tools for critical software development

- HOOD design tools in use for major European aerospace projects:
  - Eurofighter/Typhoon; Airbus A380, A350, A400M, Tiger, European Robotic Arm for the ISS, …
- AADL Modeling and Verification tools:
  - Stood for AADL: design process and graphical editor
  - AADL Inspector: static, real-time, safety and security analysis
  - Visual Studio Code extension for AADL
- In-house or collaborative R&D projects
  - Model processing technologies (LMP)
  - European Space Agency (TASTE)
  - H2020 (Space Robotics)

Ellidiss Technologies

Stood for AADL

requirements coverage

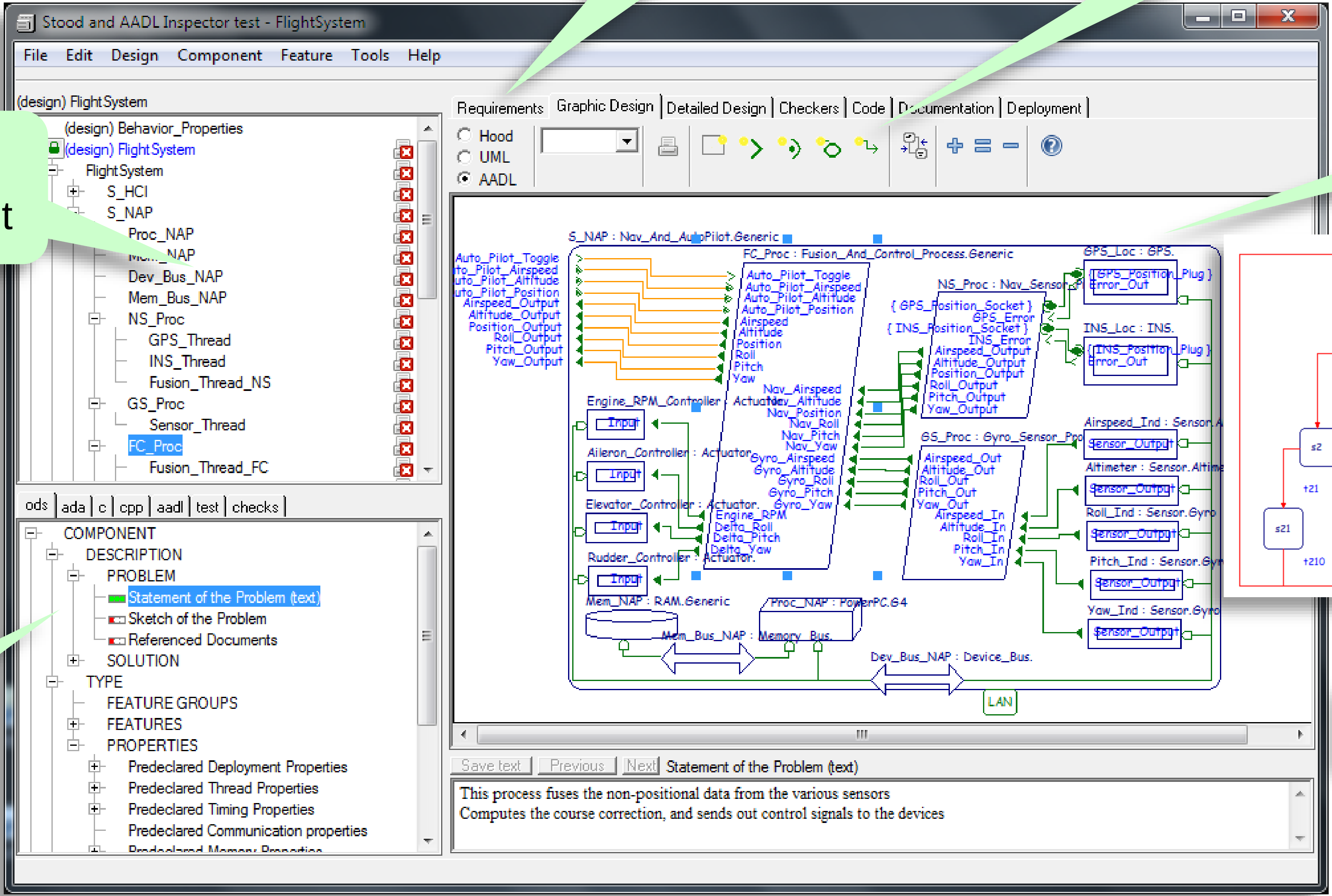
AADL text generator

multi-user project management

instance model graphical editor

structured design guidelines

behavior annex STD editor



Stood and AADL Inspector test - FlightSystem
File Edit Design Component Feature Tools Help
Requirements | Graphic Design | Detailed Design | Checkers | Code | Documentation | Deployment
Hood
UML
AADL
ods | ada | c | cpp | aadl | test | checks
Save text | Previous | Next
S_NAP : Nav_And_AutoPilot.Generic
FC_Proc : Fusion_And_Control_Process.Generic
NS_Proc : Nav_Sensor
GS_Proc : Gyro_Sensor_Proc
GPS_Loc : GPS.
INS_Loc : INS.
Airspeed_Ind : Sensor.
Altimeter : Sensor.Altimeter
Roll_Ind : Sensor.Gyro
Pitch_Ind : Sensor.Gyro
Yaw_Ind : Sensor.Gyro
Engine_RPM_Controller : Actuator.
Aileron_Controller : Actuator.
Elevator_Controller : Actuator.
Rudder_Controller : Actuator.
Mem_NAP : RAM.Generic
Proc_NAP : PowerPC.G4
Mem_Bus_NAP : Memory_Bus.
Dev_Bus_NAP : Device_Bus.
LAN

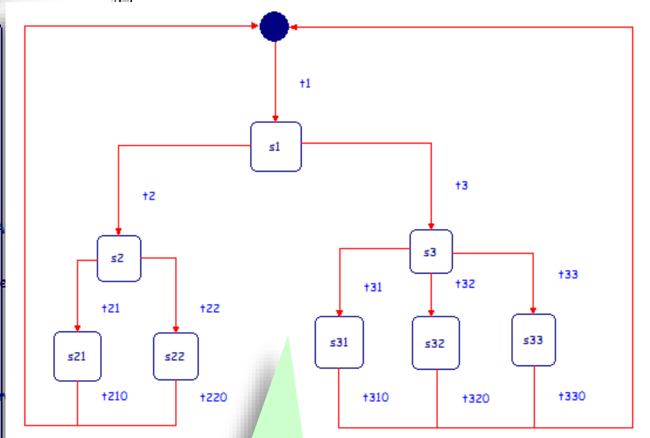
(design) FlightSystem

- (design) Behavior_Properties
- (design) FlightSystem
  - FlightSystem
    - S_HCI
    - S_NAP
      - Proc_NAP
      - Mem_NAP
      - Dev_Bus_NAP
      - Mem_Bus_NAP
      - NS_Proc
        - GPS_Thread
        - INS_Thread
        - Fusion_Thread_NS
      - GS_Proc
        - Sensor_Thread
      - FC_Proc
        - Fusion_Thread_FC

- COMPONENT
  - DESCRIPTION
    - PROBLEM
      - Statement of the Problem (text)
      - Sketch of the Problem
      - Referenced Documents
    - SOLUTION
  - TYPE
    - FEATURE GROUPS
    - FEATURES
    - PROPERTIES
      - Predeclared Deployment Properties
      - Predeclared Thread Properties
      - Predeclared Timing Properties
      - Predeclared Communication properties
      - Predeclared Memory Properties

Statement of the Problem (text)

This process fuses the non-positional data from the various sensors
Computes the course correction, and sends out control signals to the devices



Ellidiss Technologies

AADL Inspector

LAMP:
Flow analysis
Security analysis
Assurance cases

Scheduling Analysis (Cheddar)

Safety Analysis (FTA)

Import:
- SysML
- FACE
- Capella PA

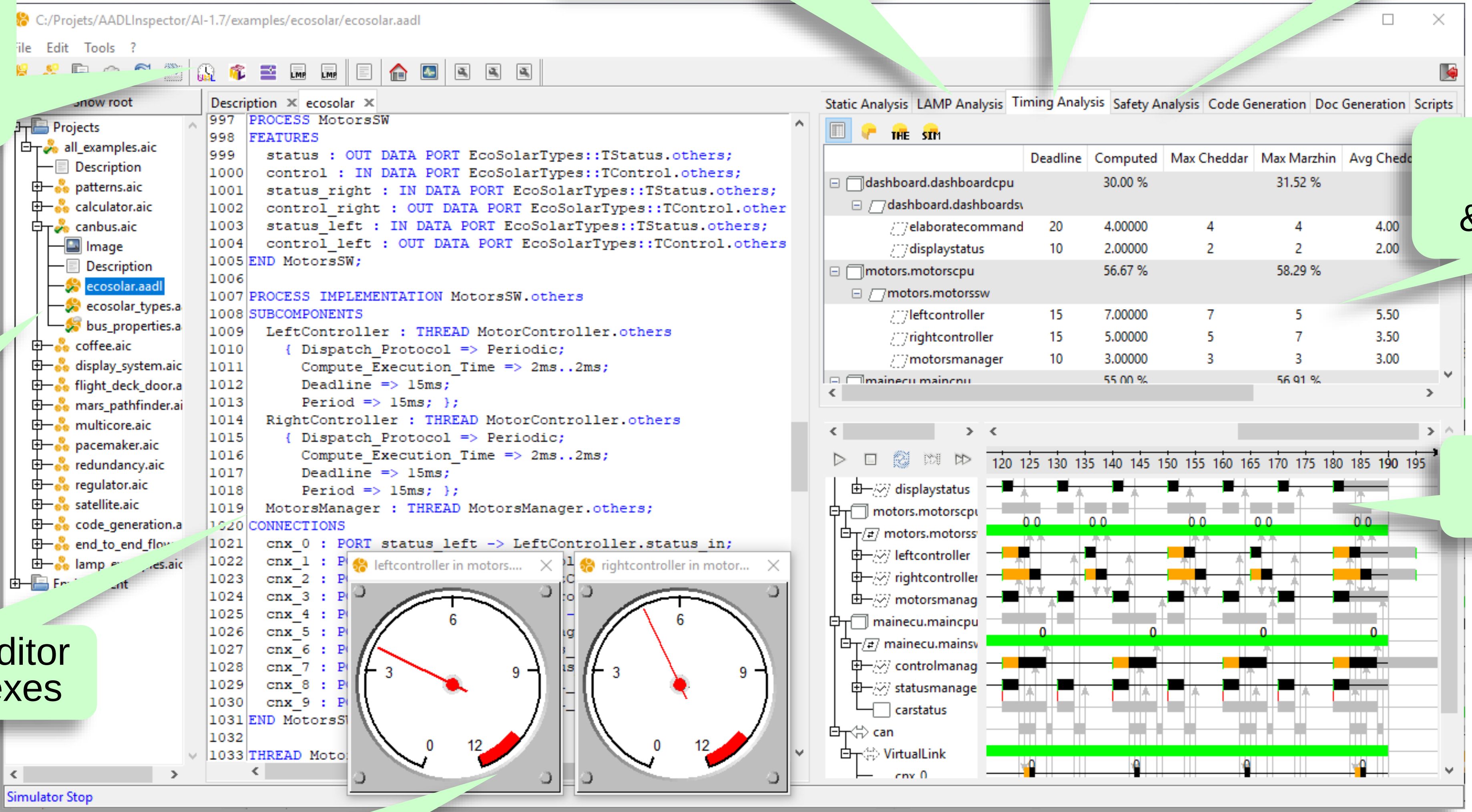
Projects manager

Response Time & CPU load

Simulation (Marzhin)

AADL text editor core + annexes

Simulation I/O



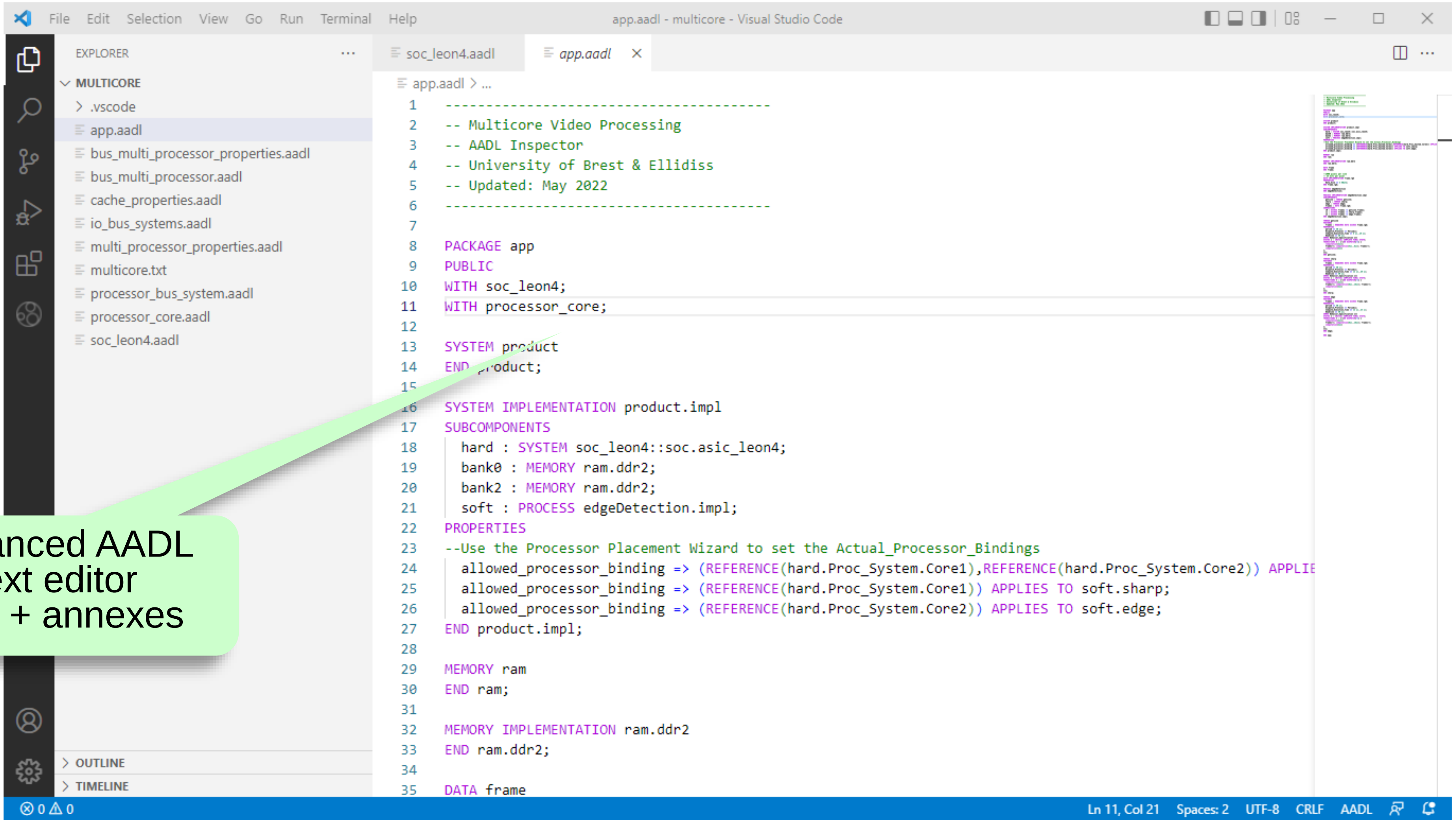
```
PROCESS MotorsSW
FEATURES
  status : OUT DATA PORT EcoSolarTypes::TStatus.others;
  control : IN DATA PORT EcoSolarTypes::TControl.others;
  status_right : IN DATA PORT EcoSolarTypes::TStatus.others;
  control_right : OUT DATA PORT EcoSolarTypes::TControl.other
  status_left : IN DATA PORT EcoSolarTypes::TStatus.others;
  control_left : OUT DATA PORT EcoSolarTypes::TControl.others
END MotorsSW;

PROCESS IMPLEMENTATION MotorsSW.others
SUBCOMPONENTS
  LeftController : THREAD MotorController.others
    { Dispatch_Protocol => Periodic;
      Compute_Execution_Time => 2ms..2ms;
      Deadline => 15ms;
      Period => 15ms; };
  RightController : THREAD MotorController.others
    { Dispatch_Protocol => Periodic;
      Compute_Execution_Time => 2ms..2ms;
      Deadline => 15ms;
      Period => 15ms; };
  MotorsManager : THREAD MotorsManager.others;
CONNECTIONS
  cnx_0 : PORT status_left -> LeftController.status_in;
  cnx_1 : P
  cnx_2 : P
  cnx_3 : P
  cnx_4 : P
  cnx_5 : P
  cnx_6 : P
  cnx_7 : P
  cnx_8 : P
  cnx_9 : P
END MotorsS

THREAD Moto
```

	Deadline	Computed	Max Cheddar	Max Marzhin	Avg Chedd
dashboard.dashboardcpu		30.00 %		31.52 %	
dashboard.dashboardsv					
elaboratecommand	20	4.00000	4	4	4.00
displaystatus	10	2.00000	2	2	2.00
motors.motorscpu		56.67 %		58.29 %	
motors.motorssw					
leftcontroller	15	7.00000	7	5	5.50
rightcontroller	15	5.00000	5	7	3.50
motorsmanager	10	3.00000	3	3	3.00
mainecu.maincpu		55.00 %		56.91 %	

Ellidiss Technologies

Visual Studio Code Extension for AADL

https://marketplace.visualstudio.com/items?itemName=Ellidiss.aadl-ellidiss


File Edit Selection View Go Run Terminal Help
app.aadl - multicore - Visual Studio Code
EXPLORER
MULTICORE
.vscode
app.aadl
bus_multi_processor_properties.aadl
bus_multi_processor.aadl
cache_properties.aadl
io_bus_systems.aadl
multi_processor_properties.aadl
multicore.txt
processor_bus_system.aadl
processor_core.aadl
soc_leon4.aadl
OUTLINE
TIMELINE
soc_leon4.aadl
app.aadl
app.aadl > ...
⊗ 0 ⚠ 0
Ln 11, Col 21 Spaces: 2 UTF-8 CRLF AADL

```
----------------------------------------
-- Multicore Video Processing
-- AADL Inspector
-- University of Brest & Ellidiss
-- Updated: May 2022
----------------------------------------

PACKAGE app
PUBLIC
WITH soc_leon4;
WITH processor_core;

SYSTEM product
END product;

SYSTEM IMPLEMENTATION product.impl
SUBCOMPONENTS
  hard : SYSTEM soc_leon4::soc.asic_leon4;
  bank0 : MEMORY ram.ddr2;
  bank2 : MEMORY ram.ddr2;
  soft : PROCESS edgeDetection.impl;
PROPERTIES
--Use the Processor Placement Wizard to set the Actual_Processor_Bindings
  allowed_processor_binding => (REFERENCE(hard.Proc_System.Core1),REFERENCE(hard.Proc_System.Core2)) APPLIE
  allowed_processor_binding => (REFERENCE(hard.Proc_System.Core1)) APPLIES TO soft.sharp;
  allowed_processor_binding => (REFERENCE(hard.Proc_System.Core2)) APPLIES TO soft.edge;
END product.impl;

MEMORY ram
END ram;

MEMORY IMPLEMENTATION ram.ddr2
END ram.ddr2;

DATA frame
```

Enhanced AADL
text editor
core + annexes

Ellidiss
Technologies

AADL 2.3

https://www.sae.org/standards/content/as5506d

- SW Architectural Analysis and Design Language.
  - Embedded and critical software (aerospace, transportation, medical, …).
  - Supports multi-thread, multi-partitions, multi-core, multi-processor architectures
  - Core language with optional annexes (behavior, safety, security,…)
  - Supported by a variety of tools (open-source and commercial)

- Digital workflow continuity for SW intensive systems.
  - Need to bridge AADL:
    - Upstream with system engineering (FACE©, SysML, Capella, …)
    - Downstream with source code and platform deployment (Ada, C, RTOS, middleware, …)
  - Existing connections between SysML v1 and AADL
    - AADL profiles for SysML v1
    - SysML v1 to AADL transformation

Ellidiss Technologies

SysML v2 (System Modeling Language)

https://www.omgsysml.org/SysML-2.htm
https://github.com/Systems-Modeling/SysML-v2-Release

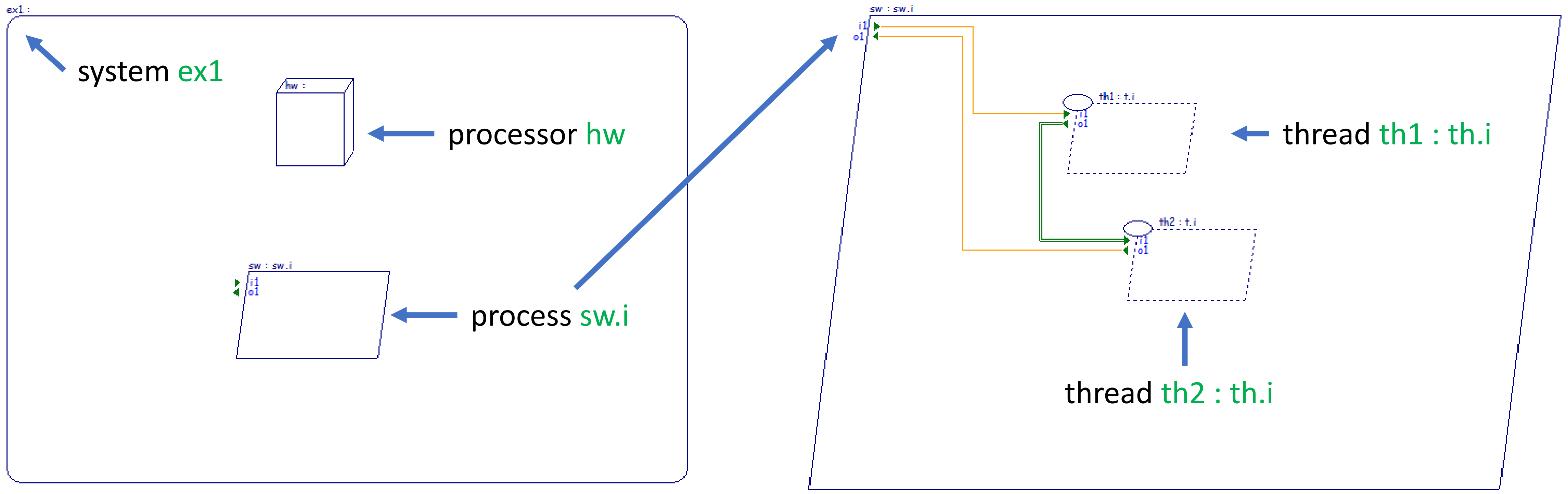
- A few interesting topics in SysML v2:
    - Standardized textual representation.
        - Better interoperability (between humans and tools).
        - Better scalability
        - Better integration within development environments (e.g. svn, git)
    - Support of instance models.
        - Deeper static and dynamic analysis.
    - Extensible by domain libraries (Instead of UML profiles)
        - Portable: no need for tool-specific extension.
        - Flexible: easy to configure and maintain.

- This work introduces a SysML v2 Domain Library for AADL (work in progress)

Illustrative AADL Example

Syntactic comparison of an AADL example and its SysML representation.

One system with 1 processor, 1 process and 2 communicating threads.



Ellidiss Technologies

System and processor

SysML Parts and part definitions for AADL components.

Use of specialization of pre-defined System, Processor, … part definitions.

AADL

```
package ex1_pkg public
 with Base_Types;
 renames data Base_Types::Integer;

system ex1 end ex1;
system implementation ex1.i
subcomponents
 hw : processor hw {
  Scheduling_Protocol => (Rate_Monotonic_Protocol);
 };
 sw : process sw.i;
properties
 Actual_Processor_Binding =>
  (reference(hw)) applies to sw;
end ex1.i;
processor hw end hw;
```

SysML

```
package ex1_pkg {
 import AADL::*;
 import ScalarValues::Integer;

part def ex1 specializes System;
part def 'ex1.i' specializes ex1,
SystemImplementation {
 part hw: Hw {
  redefines Scheduling_Prototol = Rate_Monotonic_Protocol;
 }
 part sw: 'sw.i’;

 allocation sw_to_hw: Actual_Processor_Binding
   allocate sw to hw;
}
part def Hw specializes Processor;
```

Ellidiss Technologies

Process first part

Use of SysML ports for AADL data port.

Redefinition of SysML properties for AADL properties.

```
process sw
features
 i1 : in data port Integer;
 o1 : out data port Integer;
end sw;

process implementation sw.i
Subcomponents

th1 : thread t {
  Dispatch_Protocol => Periodic;
  Period => 50ms;
  Deadline => 50ms;
  Compute_Execution_Time => 2ms..2ms;
};
```

AADL

```
part def sw specializes Process {

  in port i1: IntegerPort;
  out port o1: IntegerPort;
}

part def 'sw.i' specializes sw, ProcessImplementation
{

part th1: t {
  redefines Dispatch_Protocol = Periodic;
  redefines Period = 50 [ms];
  :>> Deadline = 50 [ms];
  :>> Compute_Execution_Time = 2[ms]..2[ms];
}
```

SysML

Ellidiss Technologies

Process final part and thread

Named connections with redefined properties.

AADL Thread as specialization of SysML Thread part definition.

```
th2 : thread t { … };
connections
 c1 : port i1 -> th1.i1;
 c2 : port th1.o1 -> th2.i1 {

  Timing => Sampled;
 };
 c3 : port th2.o1 -> o1;

end sw.i;
thread t
features
 i1 : in data port Integer;
 o1 : out data port Integer;
end t;
end ex1_pkg;
```

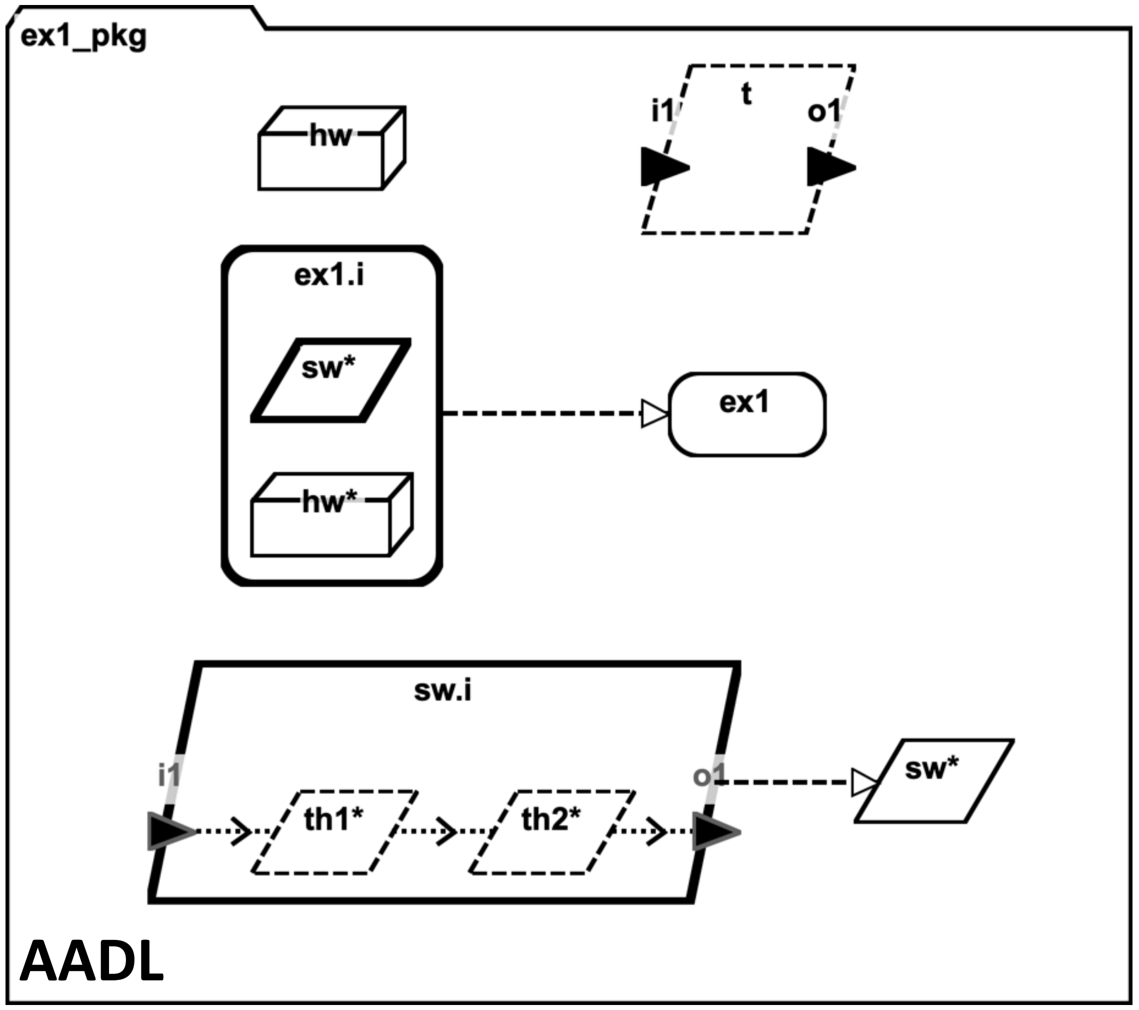
AADL

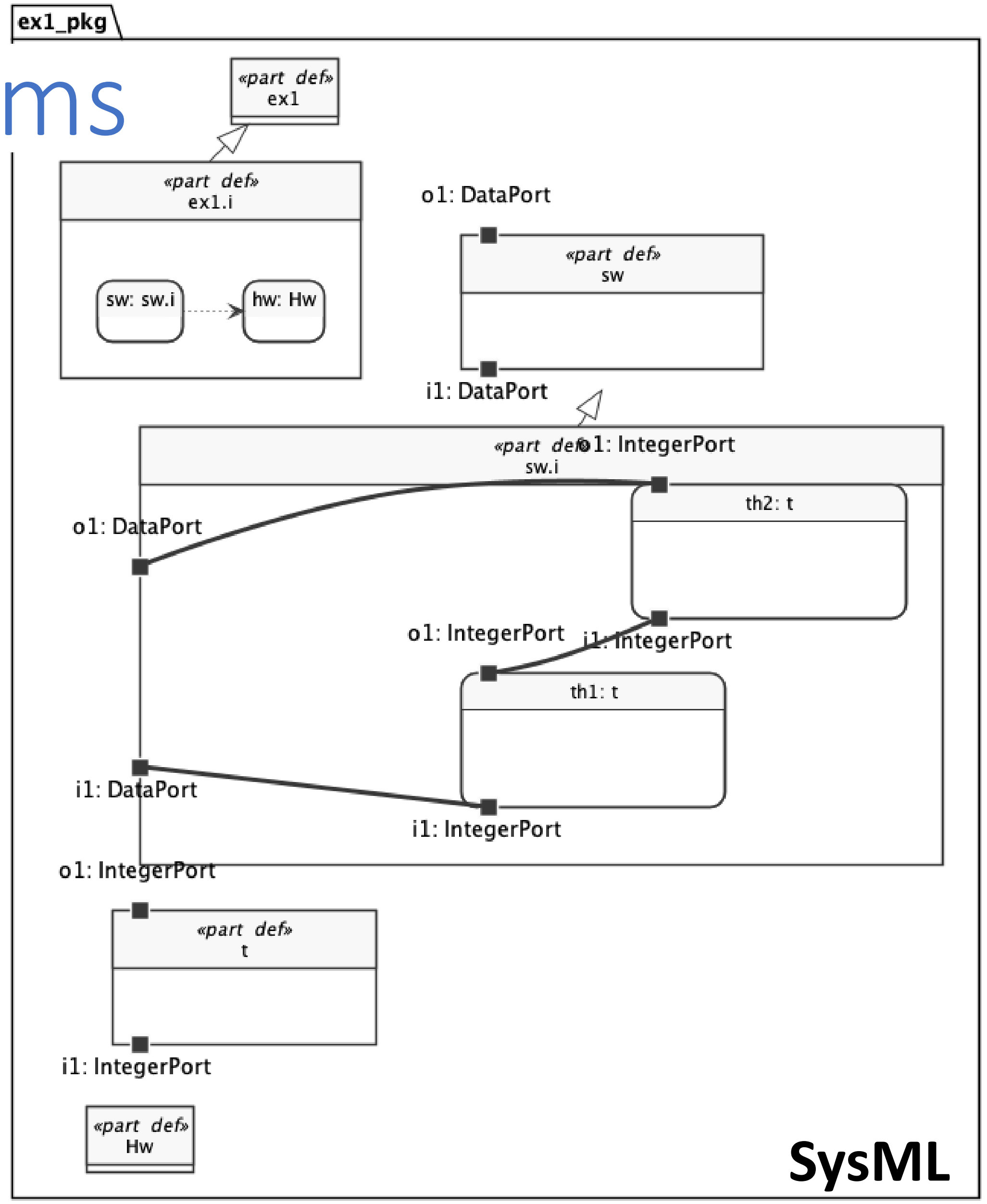
```
part th2: t { … }
connection c1: AADL::Connection
 connect i1 to th1.i1;
 connection c2: AADL::Connection
  connect th1.o1 to th2.i1 {
   redefines Timing = Sampled;
  }
 connection c3: AADL::Connection
  connect th2.o1 to o1;
}
part def t specializes Thread {

 in port i1: IntegerPort;
 out port o1: IntegerPort;
}
}
```

SysML

Ellidiss Technologies

AADL and SysML v2 Diagrams





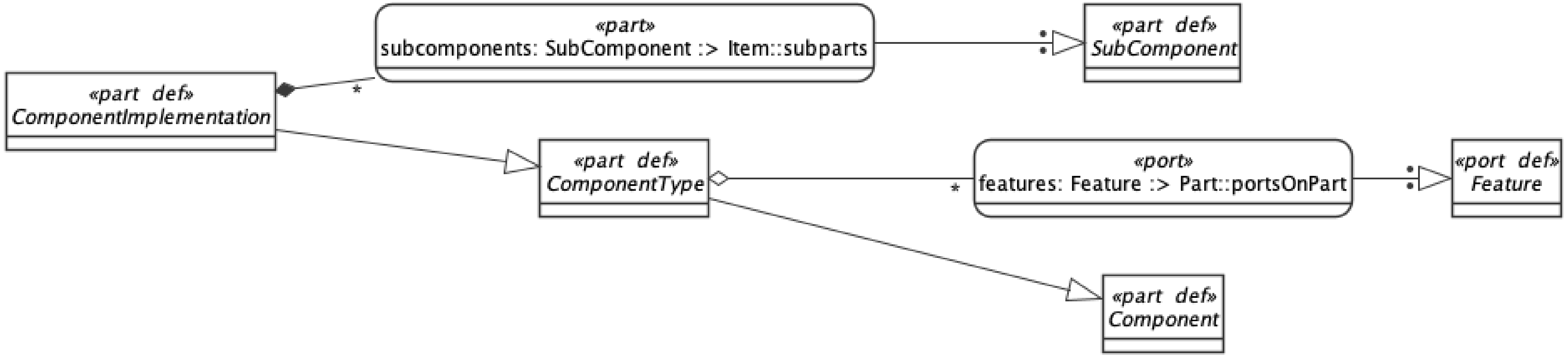
Ellidiss Technologies

AADL Domain Library (1): Components and Features

SysML part defs for AADL component types and implementations.

SysML

```
abstract part def Component;
abstract port def Feature;

abstract part def ComponentType  :> Component {
  port features: Feature[0..*] :> portsOnPart;
}

abstract part def SubComponent;

abstract part def ComponentImplementation  :> ComponentType {
  part subcomponents: SubComponent[0..*] :> subparts;
}
```



Ellidiss Technologies

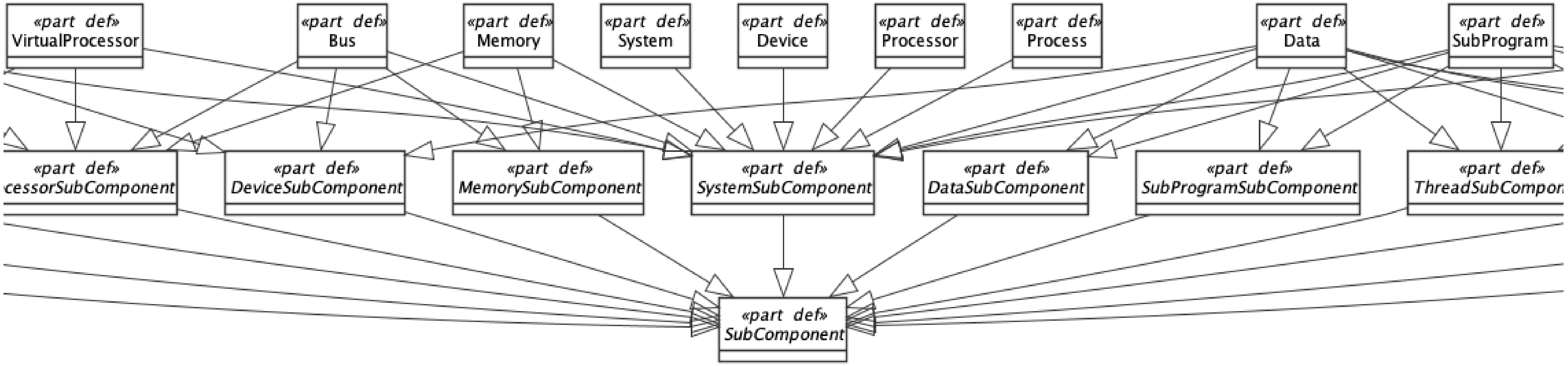
AADL Domain Library (2): Categories composition rules

Enforce composition constraints with SysML type system.

SysML

```
abstract port def ProcessFeature :> Feature;
part def Process :> ComponentType,
  SystemSubComponent, AbstractSubComponent  {
  port :>> features: ProcessFeature[0..*];
}
```

```
abstract part def ProcessSubComponent :> SubComponent;
part def ProcessImplementation :> Process,
  ComponentImplementation {
  part :>> subcomponents: ProcessSubComponent[0..*];
}
```



«part def» VirtualProcessor
«part def» Bus
«part def» Memory
«part def» System
«part def» Device
«part def» Processor
«part def» Process
«part def» Data
«part def» SubProgram
«part def» cessorSubComponent
«part def» DeviceSubComponent
«part def» MemorySubComponent
«part def» SystemSubComponent
«part def» DataSubComponent
«part def» SubProgramSubComponent
«part def» ThreadSubCompon
«part def» SubComponent

Ellidiss Technologies

AADL Domain Library (3): others

- AADL Connections

```
in port operand1 : FloatPort;
part calculator {
  out port i1: FloatPort;
}
connect (operand1, calculator.i1);
```

- AADL Properties

```
redefines Dispatch_Protocol = Sporadic;
redefines Deadline = 10 [ms];
```

- AADL Hardware/Software mapping

```
part cpu : CPU { redefines Scheduling_Prototol = AADL::RM; }
part app: App;
allocation processes: AADL::Actual_Processor_Binding
  allocate app to cpu;
```

- AADL Behaviour Specifications

```
state b: Behaviour {
  entry; then idle;
  state idle;
  accept Event via e1 do send 2 to o then idle;
}
```

Ellidiss Technologies

Work in progress

- Existing mapping elements should be evaluated considering both AADL and SysML v2 semantic rules.

- Evaluation of alternate mapping solutions:
  - Using attributes and attributes definitions for AADL Data instead of parts.
  - Using action and action definitions for AADL Subprograms instead of parts.

- AADL Abstract components and abstract Features have not been translated for now
  - Generic SysML v2 Parts and Ports may be well suited for that.

- Definition of a proper mapping for the other AADL constructs is in progress and subject to discussions.

- Development of SysML v2 parser/unparser components for the LMP toolbox to enable use of AADL Inspector processing tools for SysML v2 models

Conclusion

- Making the path between System Engineering and Software Engineering more seamless is a key issue to improve the continuity of the digital workflow.

- SysML v2 comes with a new modelling style (textual notation, domain libraries) that makes it more compatible with AADL.

- Current discussions within the standardization committees (SAE, OMG), as well as practical experiments with existing AADL tools, will determine the future of this approach.