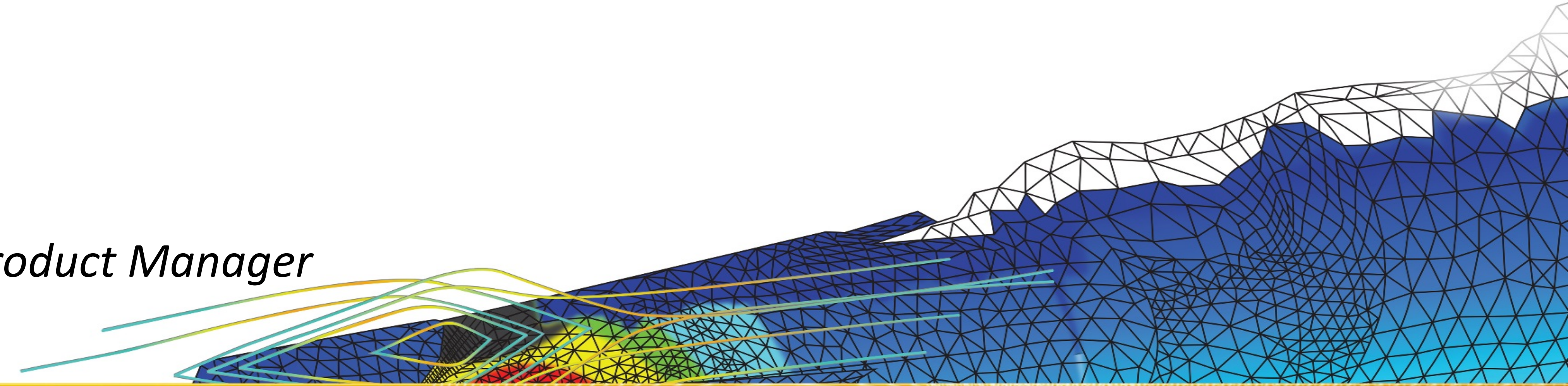
ANSYS®

SCADE AADL

Thierry Le Sergent
SCADE Architect Product Manager



SCADE

- Safety Critical Application Development Environment (SCADE) is a family of integrated tools:

  - SCADE Architect: SysML Engineering tool, extensible to support Domain Specific Languages (DSL) via a dedicated module named “Configurator”.

  - SCADE Suite: Industry-proven solution dedicated to the development of safety critical embedded software. The SCADE Suite code generator is qualified according to DO-178C/DO-330 at TQL-1.

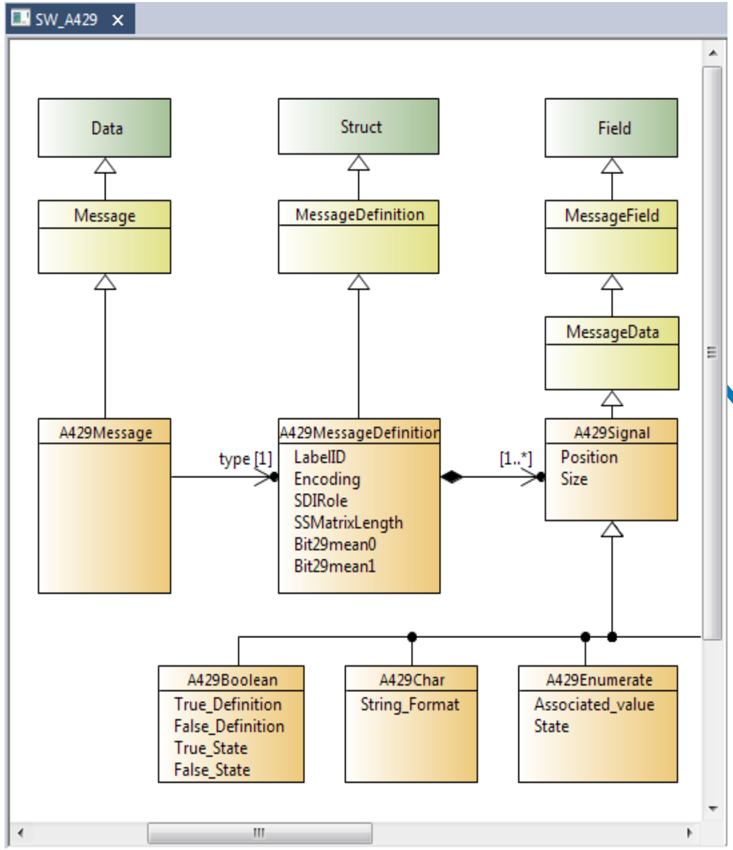
  - SCADE Display: Model-based HMI software design solution, designed for displays with safety objectives,

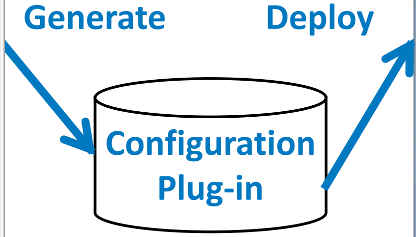
  - SCADE Test: Complete set of simulation, verification and validation tools.

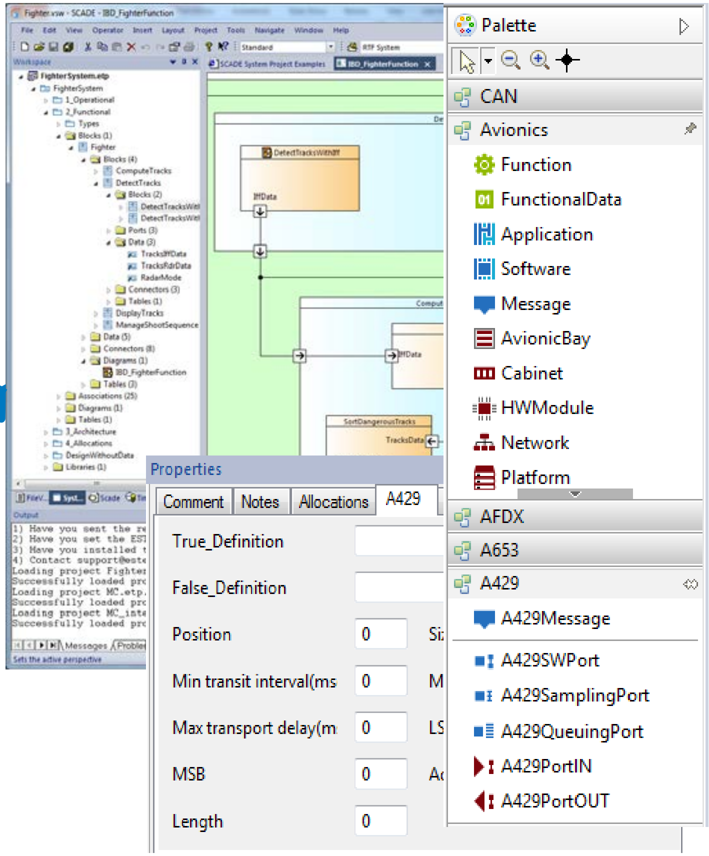
SCADE Architect Configurator Workflow

Specialist SCADE Architect Configurator

End-User SCADE Architect Modeler







True_Definition

False_Definition

Position 0

Min transit interval(ms 0

Max transport delay(m 0

MSB 0

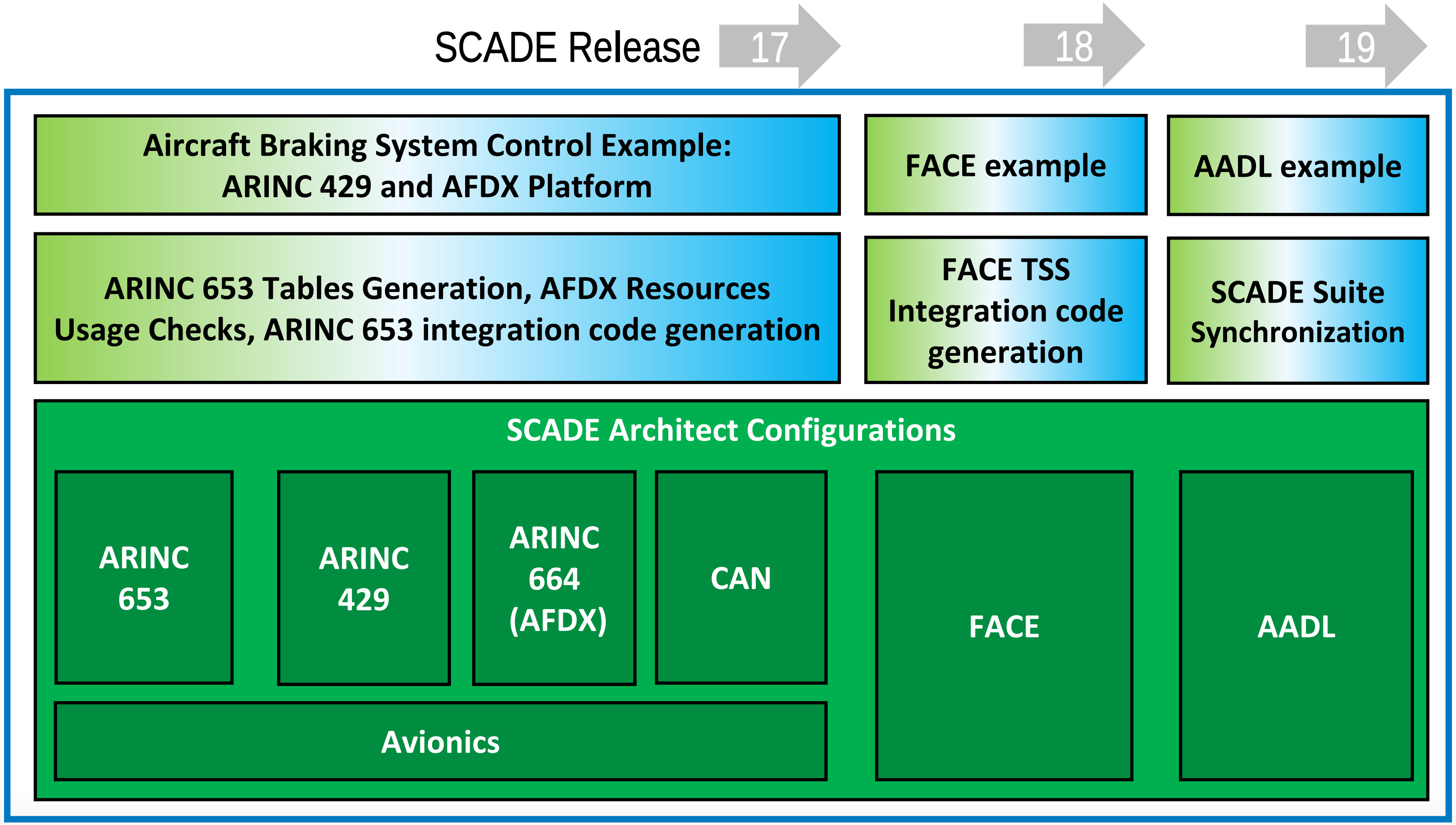
Length 0

Define customized object kinds, derived from SCADE Architect objects

Domain specific modeler

SCADE Avionics Package

- An extension of SCADE design capabilities for the aerospace industries



SCADE Release 17 18 19

Aircraft Braking System Control Example: ARINC 429 and AFDX Platform

FACE example

AADL example

ARINC 653 Tables Generation, AFDX Resources Usage Checks, ARINC 653 integration code generation

FACE TSS Integration code generation

SCADE Suite Synchronization

SCADE Architect Configurations

ARINC 653

ARINC 429

ARINC 664 (AFDX)

CAN

Avionics

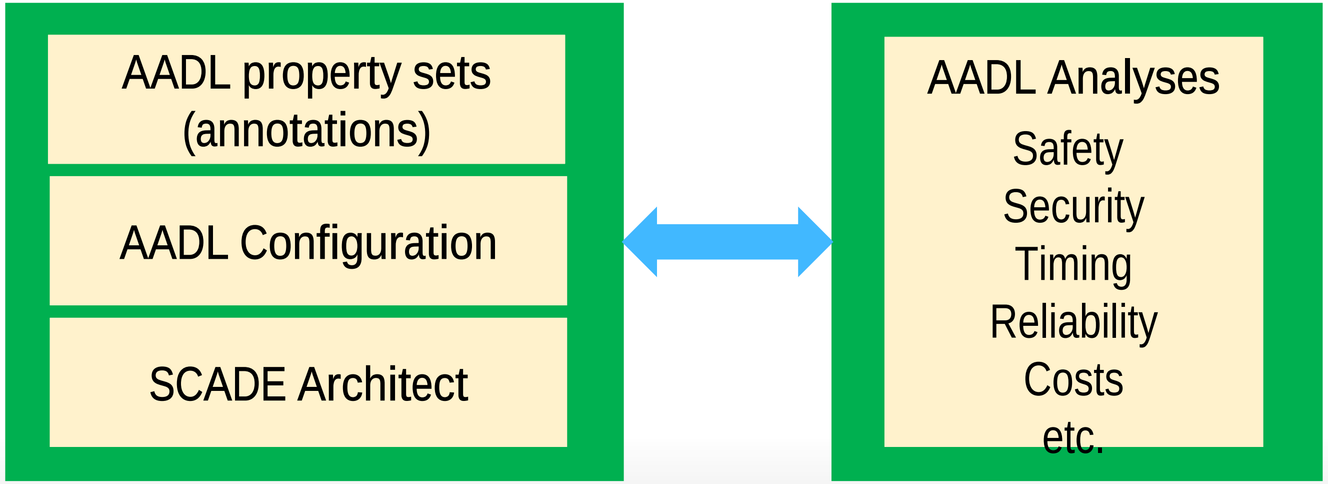
FACE

AADL

SCADE AADL

Objectives

- **Full compatibility with AADL v2.2 standard**
  - Allows for legacy models import; Allows for export to third party analyzers
- **Easy to use**
  - AADL expressiveness simplified: concrete components, merge type and implementation, etc.
  - Nice graphical interface & diagrams
- **Benefit from SCADE tools ecosystem**
  - Bi-directional synchro with SCADE Suite for SW component development, verification & certification
  - Same IDE as for SysML and FACE modeling (mixed design supported)


AADL property sets (annotations)

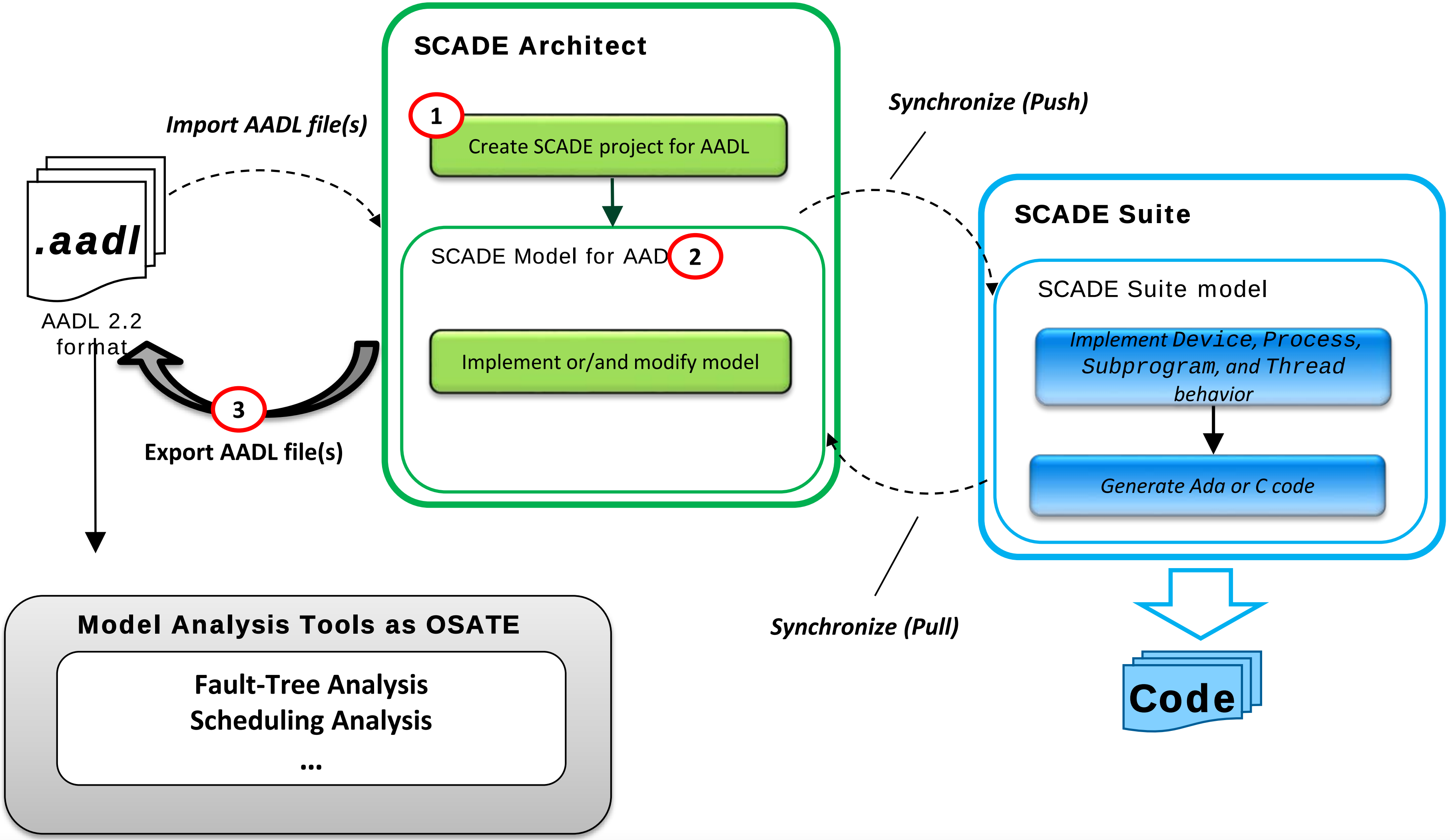
AADL Configuration

SCADE Architect

AADL Analyses

Safety
Security
Timing
Reliability
Costs
etc.

Workflows


SCADE Architect
Import AADL file(s)
1
Create SCADE project for AADL
.aadl
AADL 2.2 format
SCADE Model for AADL
2
Implement or/and modify model
3
Export AADL file(s)
Synchronize (Push)
SCADE Suite
SCADE Suite model
Implement Device, Process, Subprogram, and Thread behavior
Generate Ada or C code
Synchronize (Pull)
Code
Model Analysis Tools as OSATE
Fault-Tree Analysis
Scheduling Analysis
...

AADL language expressiveness & complexity

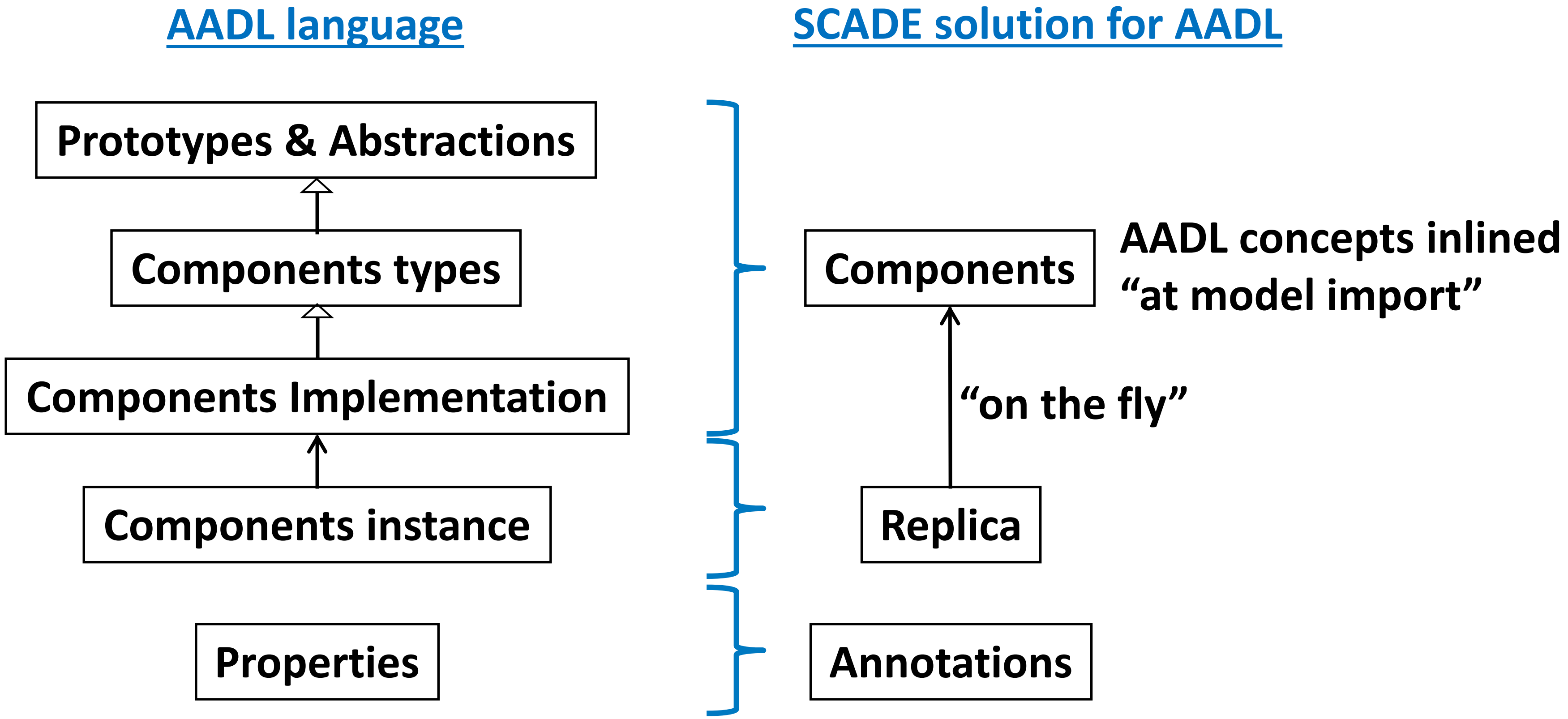
- **AADL language**
  - **Object-oriented inheritance mechanism:**
    - **Prototypes** and **Abstract** components
      - later extended and refined into concrete category
    - **Component types** and **Component implementation**
      - An interface definition can have multiple implementations
      - But definition mandatory before specifying implementation
  - **Instantiation:**
    - **Component instances** are references to **component implementation**, that must be inlined for analysis
    - Inlining done as an explicit tool action in OSATE to get an instantiated model

- **In SCADE: 2 simplifications**
  - AADL Abstraction & Inheritance inlining
  - AADL instance based modeling

SCADE solution for AADL

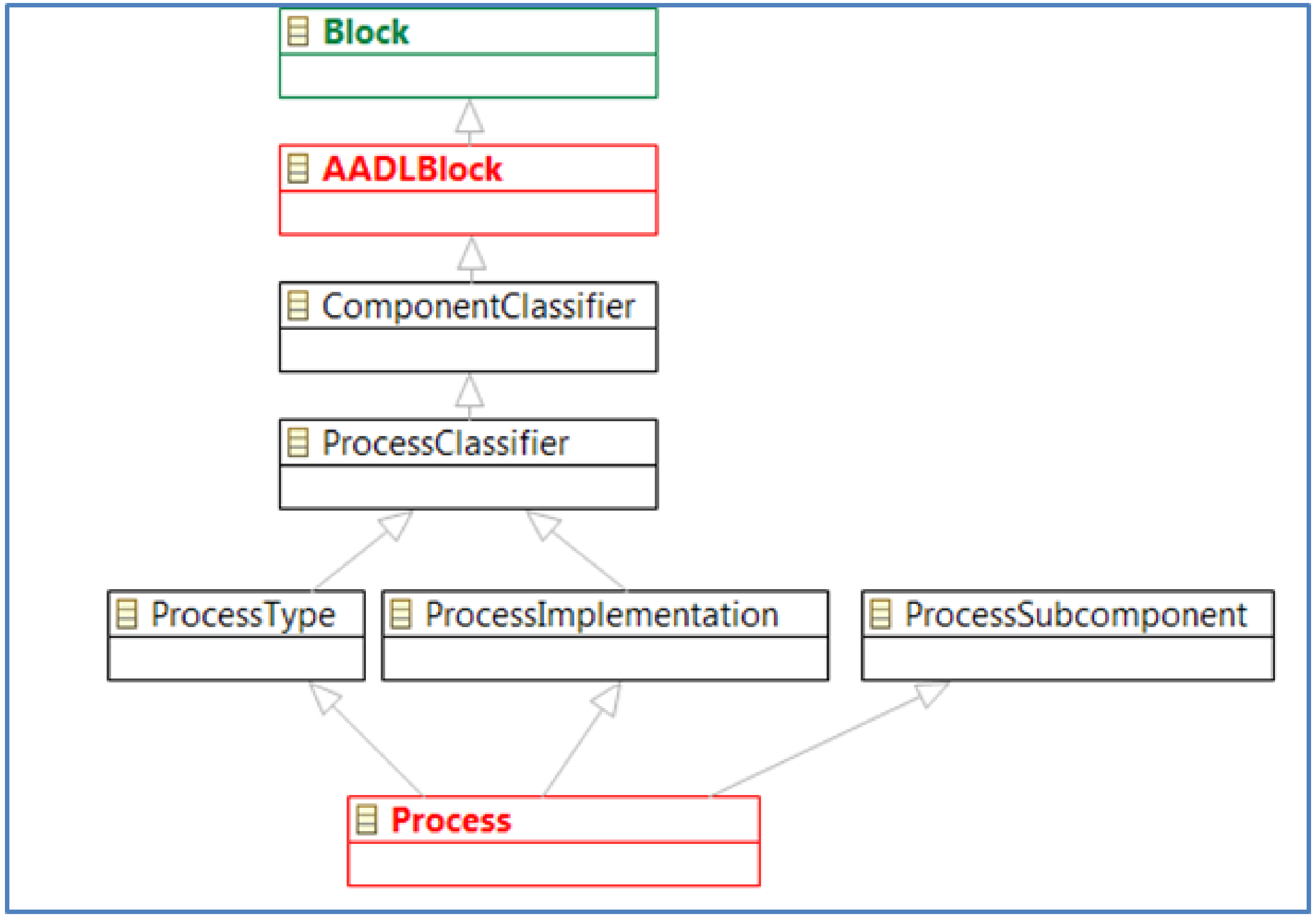
Instance based modeling

- **Support for AADL “instance based modeling”: much simpler model understanding**



SCADE Architect AADL configuration

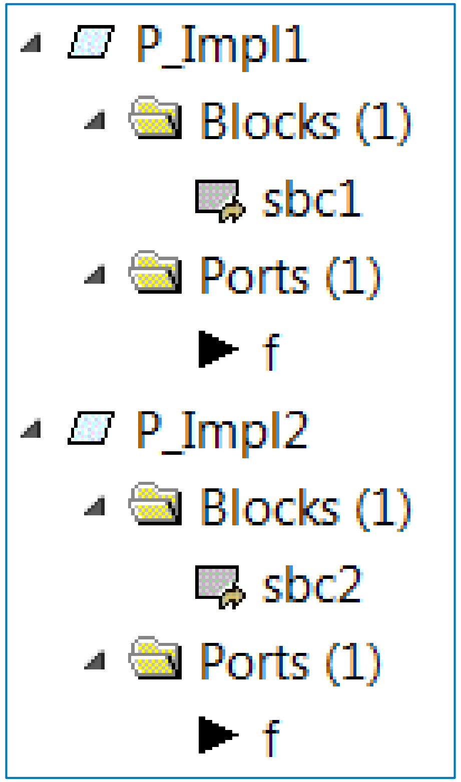
Process class (extract)



AADL - SCADE AADL Model transformation

1. Merge component type and implementation in a single object

```
process P
  features
    f: in data port Base_Types::Unsigned_16;
end P;

process implementation P.Impl1
  subcomponents
    sbc1: data Base_Types::Unsigned_16;
end P.Impl1;

process implementation P.Impl2
  subcomponents
    sbc2: data Base_Types::Float_64;
end P.Impl2;
```

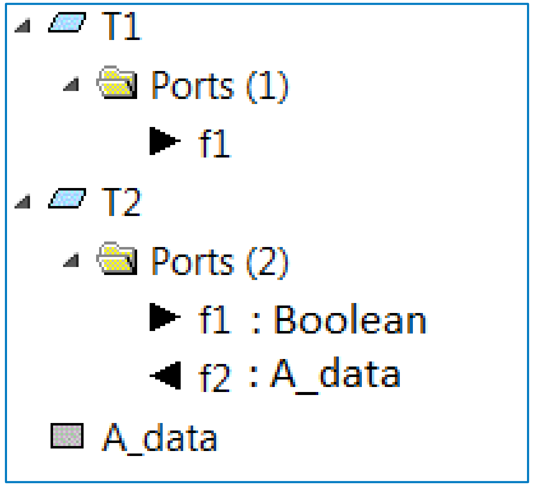


- P_Impl1
  - Blocks (1)
    - sbc1
  - Ports (1)
    - f
- P_Impl2
  - Blocks (1)
    - sbc2
  - Ports (1)
    - f

AADL - SCADE AADL Model transformation

2. Inline inheritance
3. Resolve prototypes (ignored if not bounded)
4. Import abstract elements when they are refined to concrete ones

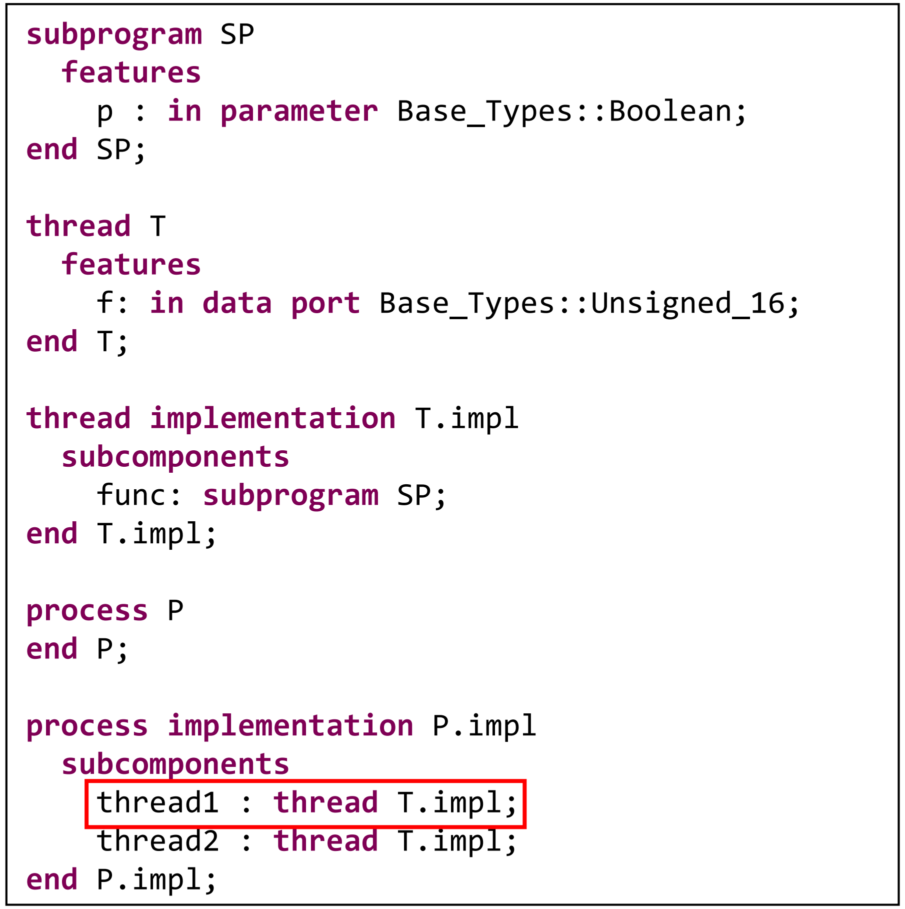
```
thread T1
  prototypes
    p: data;
  features
    f1: in data port p;
end T1;

thread T2 extends T1 (p => data Base_Types::Boolean)
  features
    f2 : out data port A;
end T2;

abstract A
end A;
```



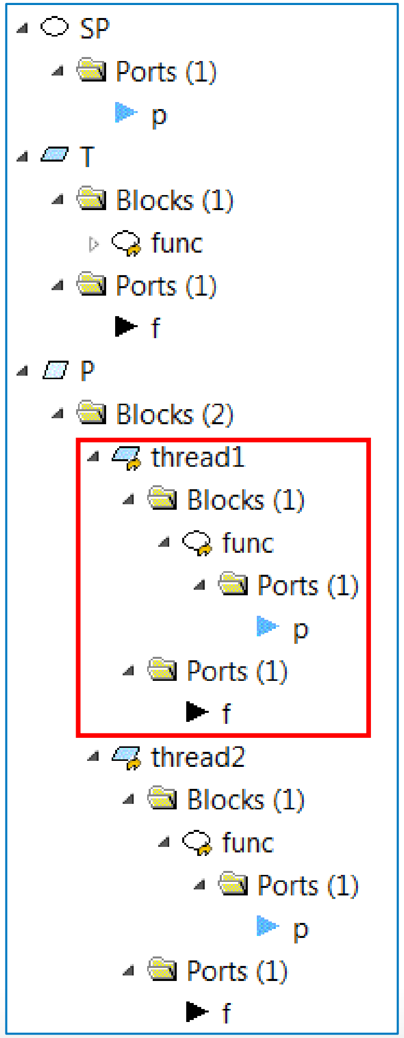
- T1
  - Ports (1)
    - ▶ f1
- T2
  - Ports (2)
    - ▶ f1 : Boolean
    - ◀ f2 : A_data
- ▨ A_data

AADL - SCADE AADL Model transformation

5. Usage of SCADE Architect replication mechanism for immediate instantiation of components.



```
subprogram SP
  features
    p : in parameter Base_Types::Boolean;
end SP;

thread T
  features
    f: in data port Base_Types::Unsigned_16;
end T;

thread implementation T.impl
  subcomponents
    func: subprogram SP;
end T.impl;

process P
end P;

process implementation P.impl
  subcomponents
    thread1 : thread T.impl;
    thread2 : thread T.impl;
end P.impl;
```



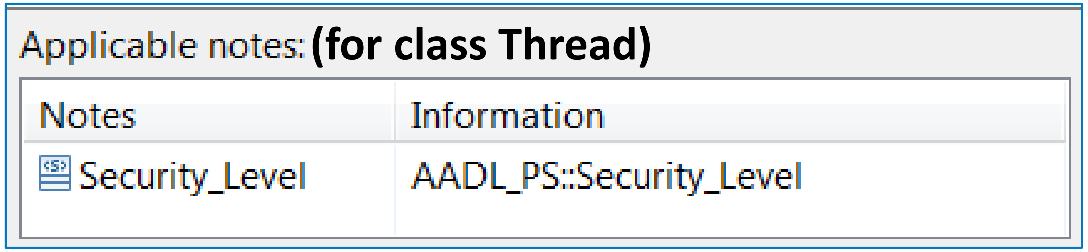
```
SP
  Ports (1)
    p
T
  Blocks (1)
    func
  Ports (1)
    f
P
  Blocks (2)
    thread1
      Blocks (1)
        func
          Ports (1)
            p
      Ports (1)
        f
    thread2
      Blocks (1)
        func
          Ports (1)
            p
      Ports (1)
        f
```

AADL - SCADE AADL Model transformation

6. Dynamic transformation of AADL Property sets into SCADE Architect Annotation type files

```
property set PS is
  Security_Level: aadlinteger  applies to (thread);
end PS;
```

```
Security_Level ::= SEQUENCE OF {SEQUENCE { annot_object
OID, name STRING, information { Security_Level INTEGER }}}


AADL-Thread ::= {
  {Security_Level F 0 1}
}
```

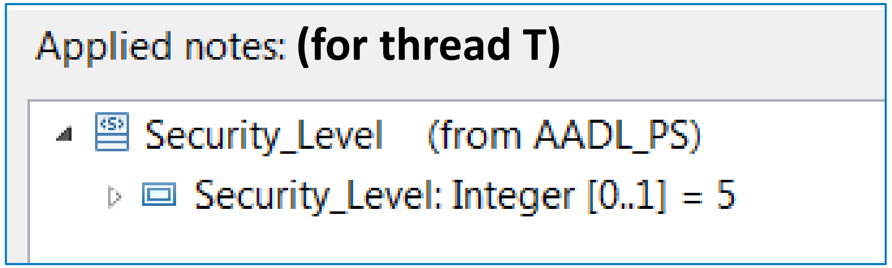


Applicable notes: (for class Thread)

Notes	Information
Security_Level	AADL_PS::Security_Level

7. Translate properties into annotations

```
thread T
  properties
    PS::Security_Level => 5;
end T;
```



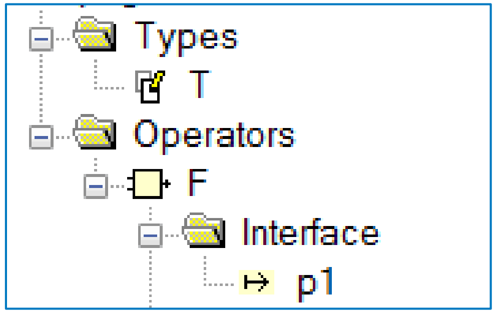
Applied notes: (for thread T)

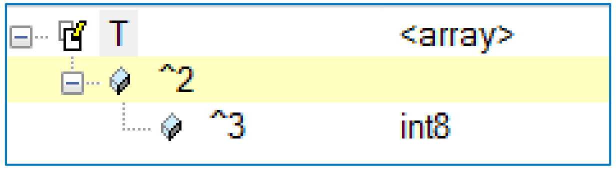
- Security_Level   (from AADL_PS)
  - Security_Level: Integer [0..1] = 5

Synchronization SCADE AADL – SCADE Suite

- Bi-directional synchronization of AADL threads and subprograms with SCADE Suite operators

```
subprogram F
  features
    p1: in parameter T;
end F;
data T
  properties
    Data_Model::Data_Representation => Array;
    Data_Model::Base_Type => (classifier (Base_Types::Integer_8));
    Data_Model::Dimension => (2, 3);
end T;
```

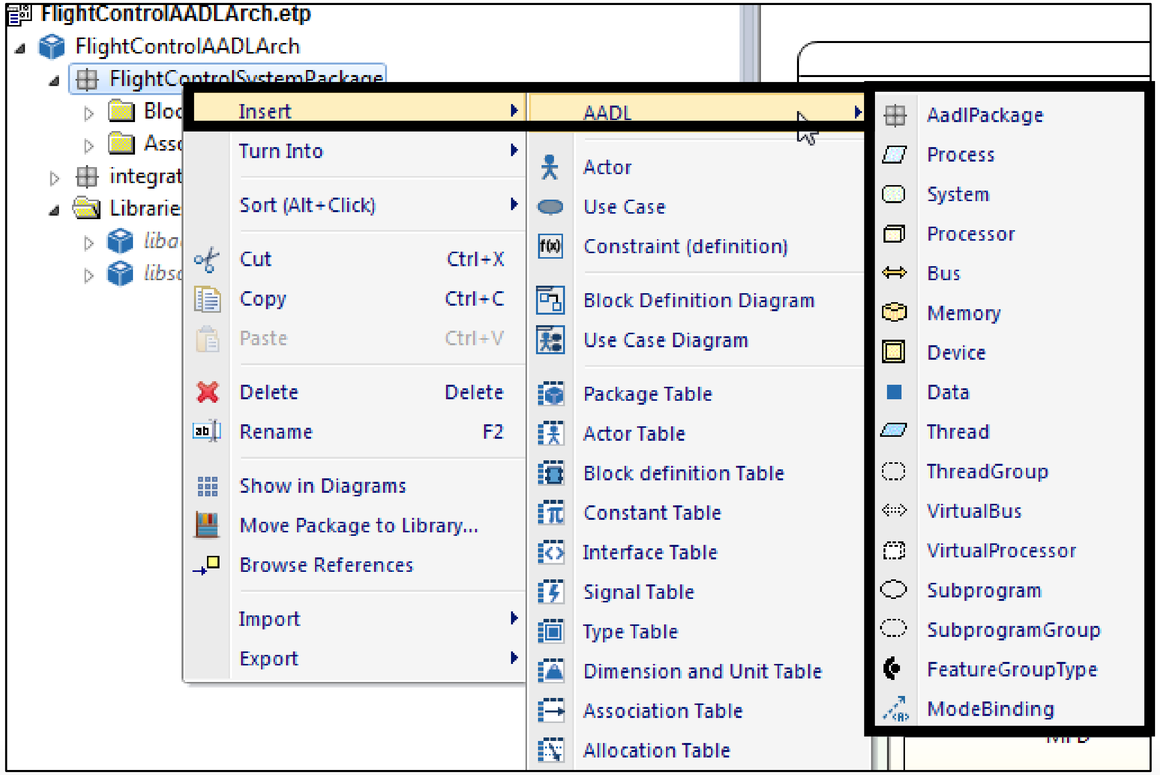




- Behavior implementation and simulation in SCADE Suite and certified C/Ada code generation

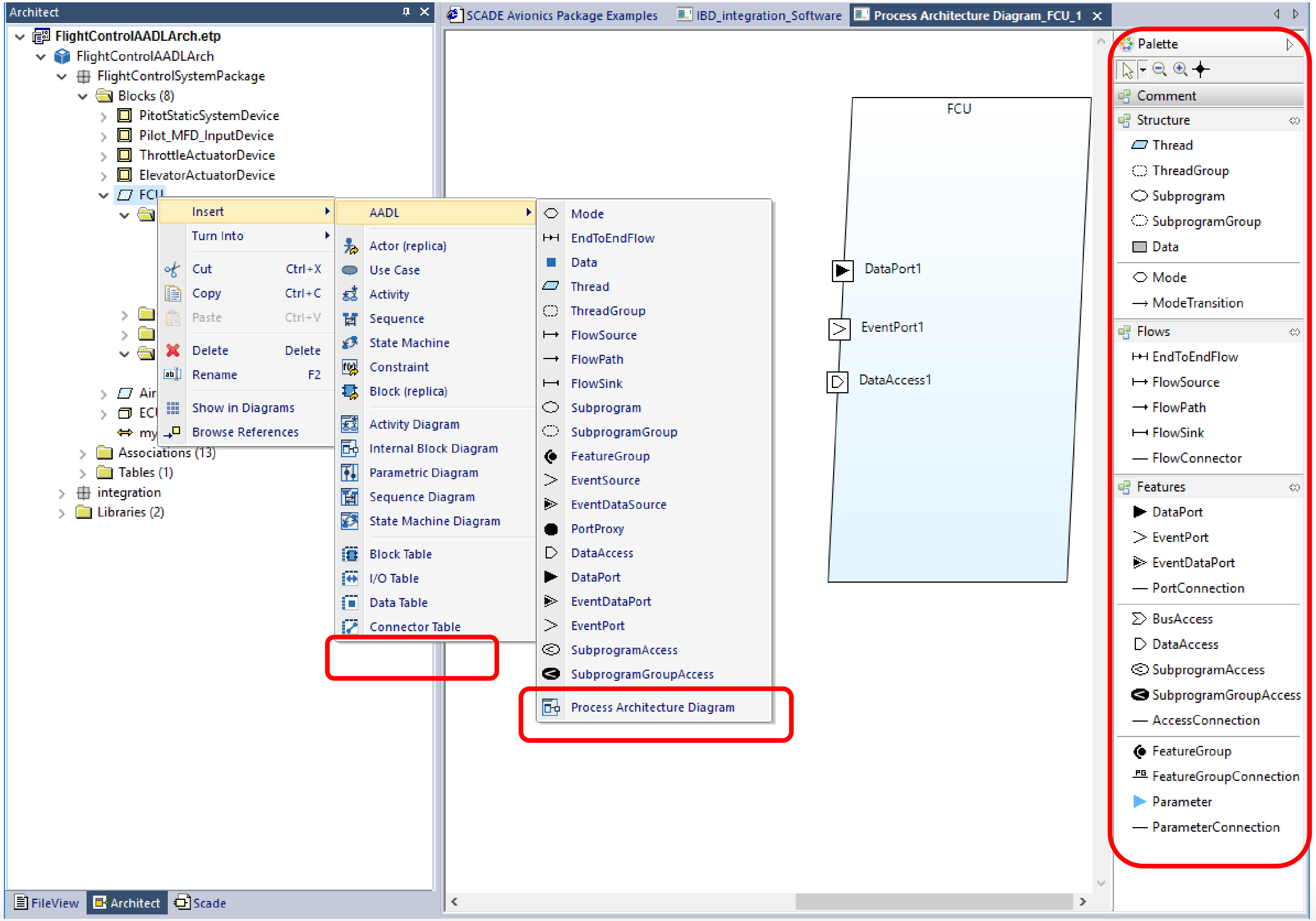
Edit SCADE Model for AADL

- Select a high level element such as `AadlPackage` or component and insert the required objects according to the context through the AADL configuration



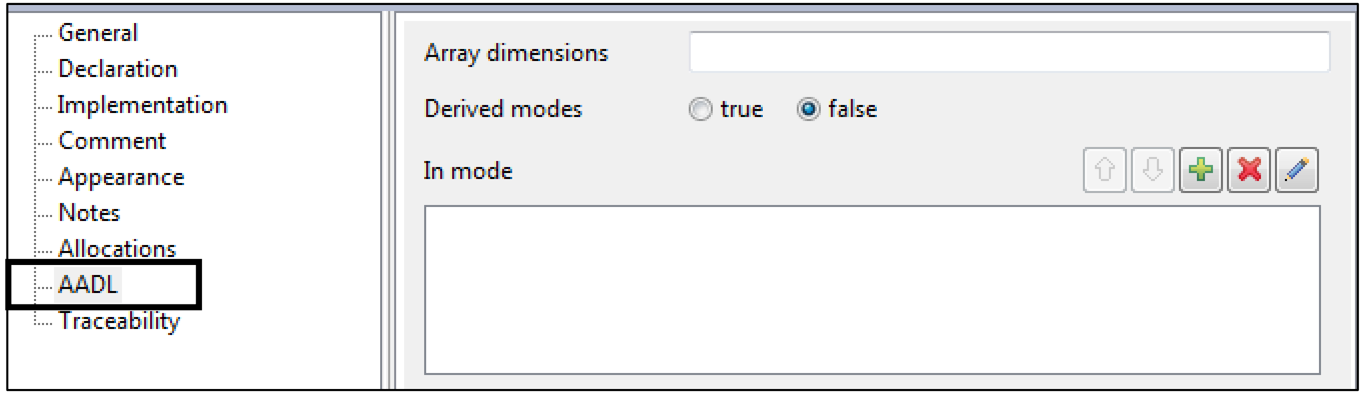
Dedicated palettes to each component

Process component



Edit SCADE Model for AADL

- **Select an element and go to the AADL tab to specify values of attributes (when it is required)**



Array dimensions

Derived modes ○ true ● false

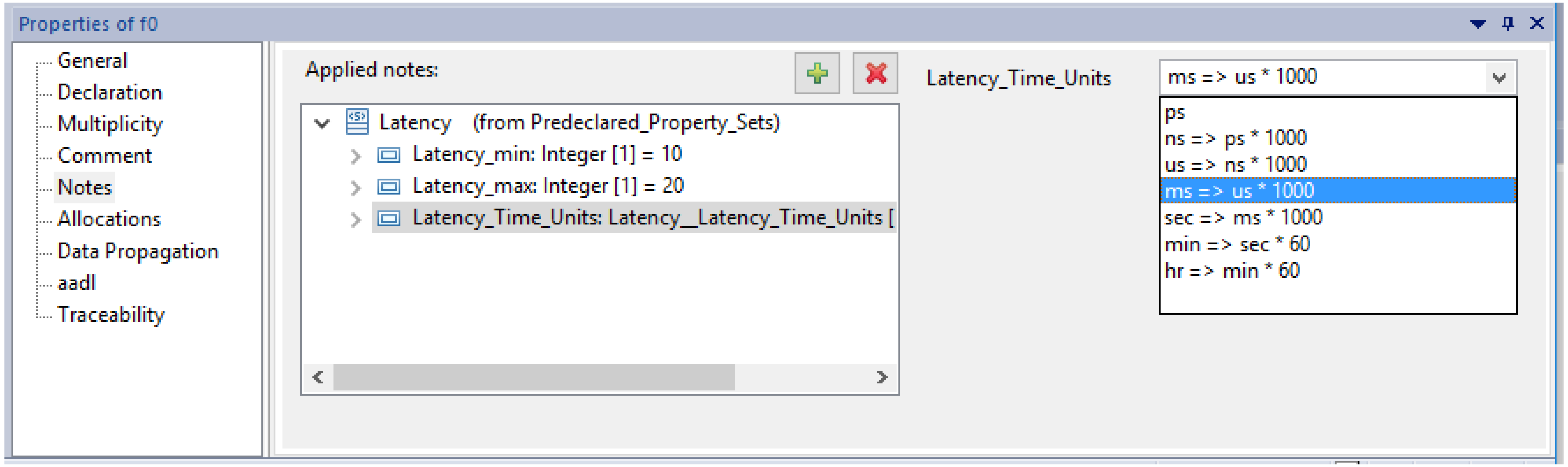
In mode

AADL Property sets

- **Automated conversion**

  <property set>.aadl → <SCADE note types>.aty

- **Benefits**
  - **Reused SCADE IDE matured technology**
  - **Automated GUI to set properties on objects in a model**



Latency (from Predeclared_Property_Sets)
Latency_min: Integer [1] = 10
Latency_max: Integer [1] = 20
Latency_Time_Units: Latency__Latency_Time_Units [

Latency_Time_Units: ms => us * 1000

ps
ns => ps * 1000
us => ns * 1000
ms => us * 1000
sec => ms * 1000
min => sec * 60
hr => min * 60

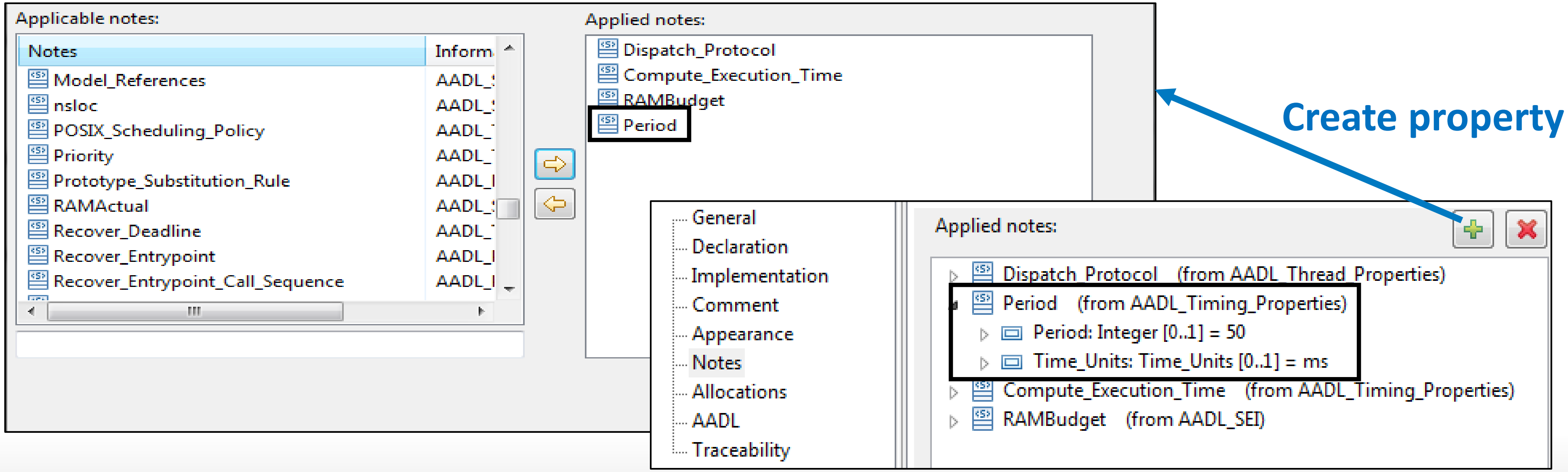
Predeclared Property Sets

- **The following predeclared property sets are modeled in predefined annotation files (.aty) inserted automatically during the creation of SCADE project for AADL:**

Property Set	SCADE Definition
Deployment_Properties	***AADL_Deployment_Properties.aty***: contains properties related to the deployment of the embedded application on the execution platform
Thread_Properties	***AADL_Thread_Properties.aty****:* contains properties that characterize threads and their features
Timing_Properties	***AADL_Timing_Properties.aty****:* contains properties related to execution timing
Communication_Properties	***AADL_Communication_Properties.aty****:* contains properties communication specify connection topology and queuing characteristics
Memory_Properties	***AADL_Memory_Properties.aty****:* contains properties related to memory as storage, data access, and device access
Programming_Properties	***AADL_Programming_Properties.aty****:* contains properties for relating AADL models to application programs
Modeling_Properties	***AADL_Modeling_Properties.aty****:* contains properties related to the AADL model itself

User-Defined Property Sets

- **Users can define customized property sets:**
  - **AADL property set files imported into a SCADE project for AADL through the AADL import mechanism**
    - Property set files are translated into annotation files (.aty)
    - Each property is translated into a note
    - Each note is applicable only to the AADL element listed by the “applies to” directive from property set file(s)



Applicable notes:

Notes	Inform
Model_References	AADL_
nsloc	AADL_
POSIX_Scheduling_Policy	AADL_
Priority	AADL_
Prototype_Substitution_Rule	AADL_
RAMActual	AADL_
Recover_Deadline	AADL_
Recover_Entrypoint	AADL_
Recover_Entrypoint_Call_Sequence	AADL_

Applied notes:

- Dispatch_Protocol
- Compute_Execution_Time
- RAMBudget
- Period

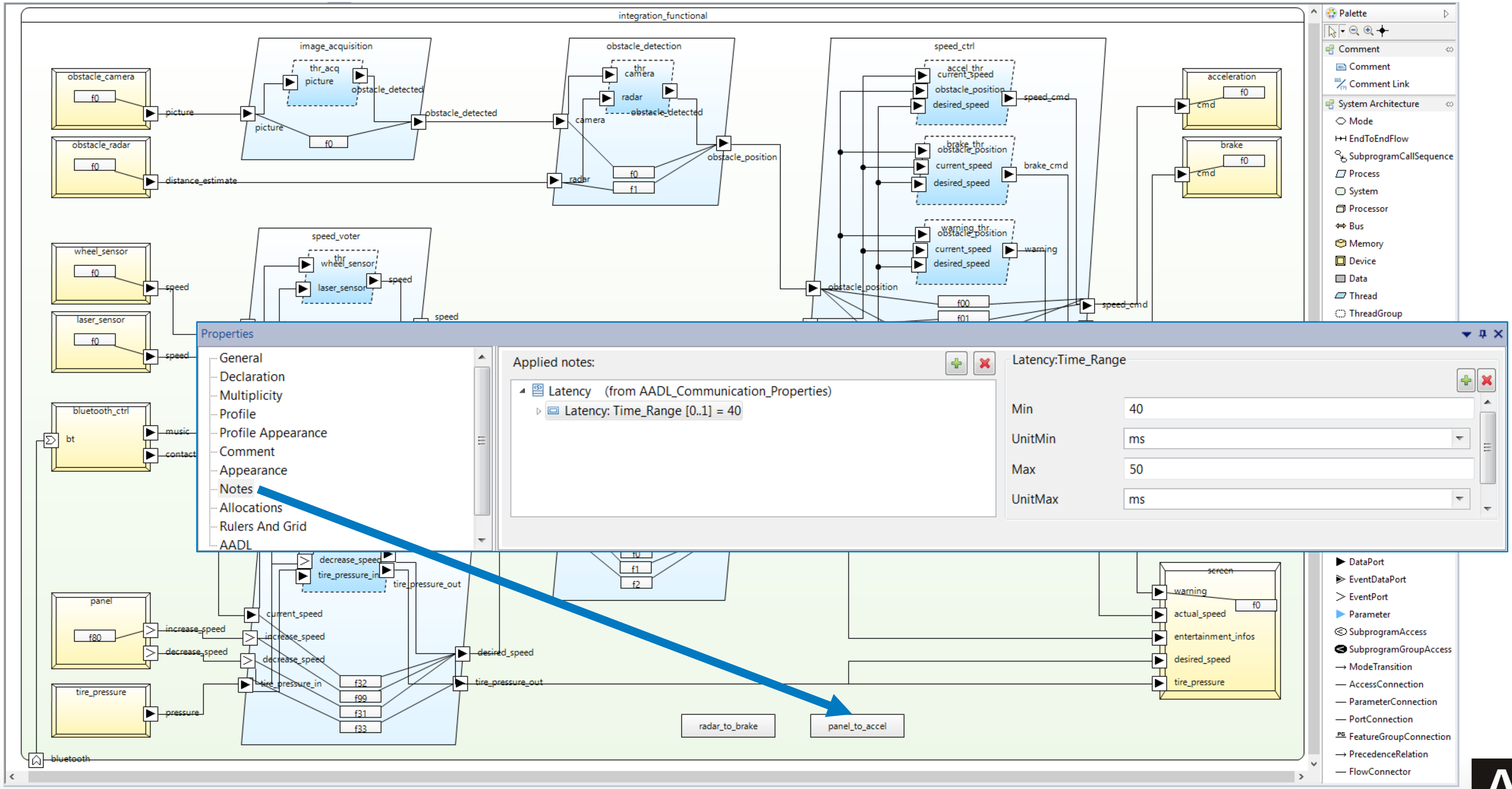
Applied notes:

- Dispatch_Protocol (from AADL_Thread_Properties)
- Period (from AADL_Timing_Properties)
  - Period: Integer [0..1] = 50
  - Time_Units: Time_Units [0..1] = ms
- Compute_Execution_Time (from AADL_Timing_Properties)
- RAMBudget (from AADL_SEI)

Create property

Case study

A simple self-driving car example. “AADL In Practice”, Julien Delange: http://www.aadl-book.com



Properties

- General
- Declaration
- Multiplicity
- Profile
- Profile Appearance
- Comment
- Appearance
- Notes
- Allocations
- Rulers And Grid
- AADL

Applied notes:

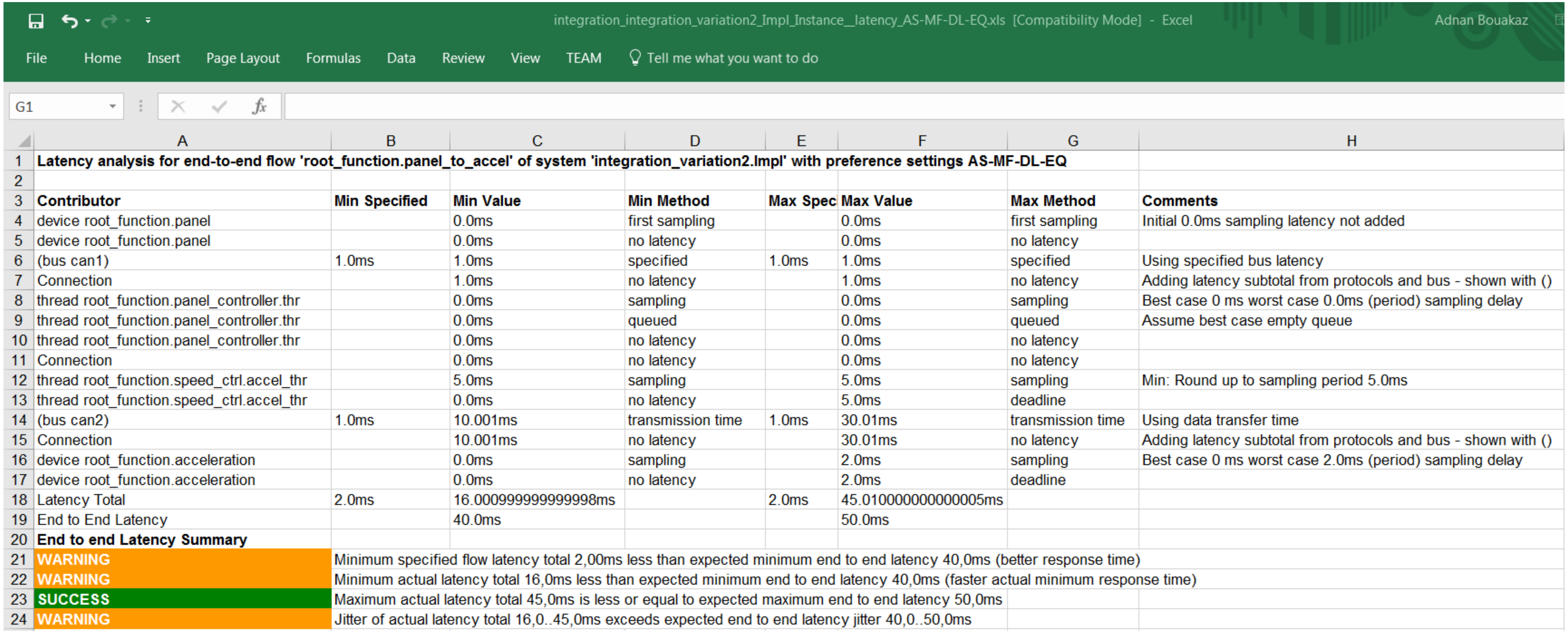
- Latency (from AADL_Communication_Properties)
  - Latency: Time_Range [0..1] = 40

Latency:Time_Range

Min	40
UnitMin	ms
Max	50
UnitMax	ms

Case study

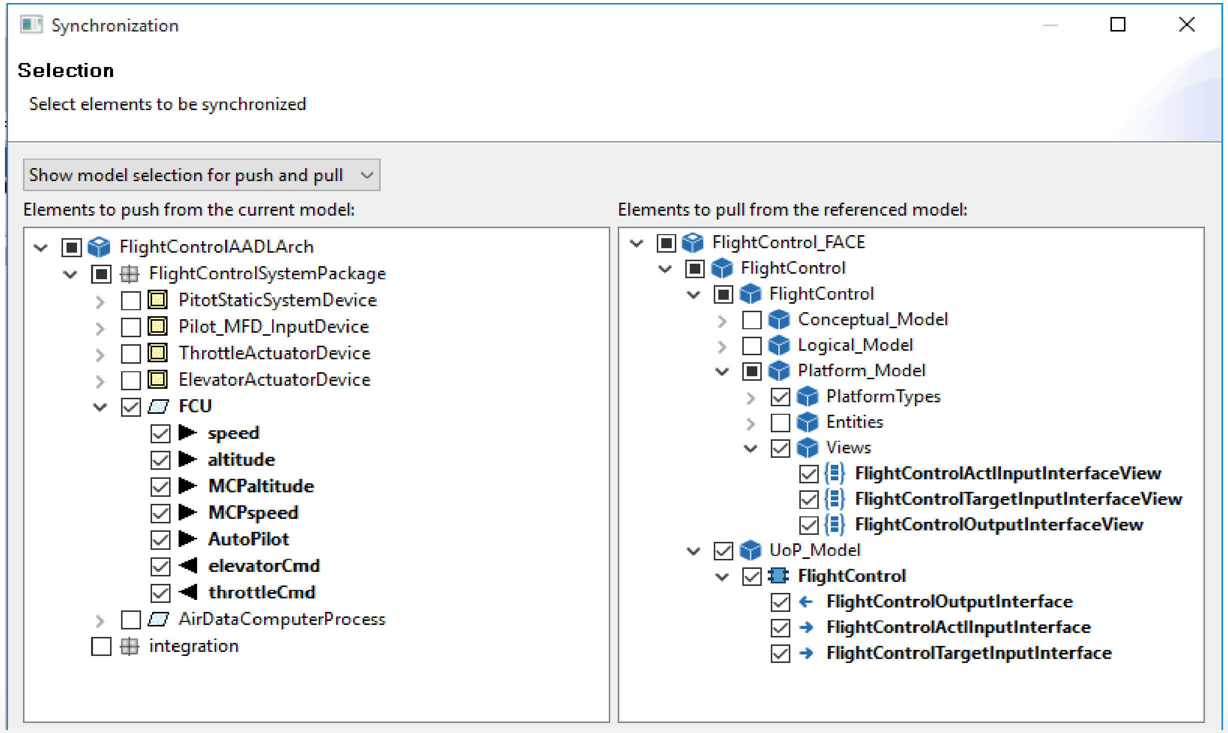
- **Analysis example**
  - Latency analysis result from Open Source tool OSATE


integration_integration_variation2_Impl_Instance__latency_AS-MF-DL-EQ.xls [Compatibility Mode] - Excel
Adnan Bouakaz
File
Home
Insert
Page Layout
Formulas
Data
Review
View
TEAM
Tell me what you want to do
G1

	A	B	C	D	E	F	G	H
1	**Latency analysis for end-to-end flow 'root_function.panel_to_accel' of system 'integration_variation2.Impl' with preference settings AS-MF-DL-EQ**							
2								
3	**Contributor**	**Min Specified**	**Min Value**	**Min Method**	**Max Spec**	**Max Value**	**Max Method**	**Comments**
4	device root_function.panel		0.0ms	first sampling		0.0ms	first sampling	Initial 0.0ms sampling latency not added
5	device root_function.panel		0.0ms	no latency		0.0ms	no latency	
6	(bus can1)	1.0ms	1.0ms	specified	1.0ms	1.0ms	specified	Using specified bus latency
7	Connection		1.0ms	no latency		1.0ms	no latency	Adding latency subtotal from protocols and bus - shown with ()
8	thread root_function.panel_controller.thr		0.0ms	sampling		0.0ms	sampling	Best case 0 ms worst case 0.0ms (period) sampling delay
9	thread root_function.panel_controller.thr		0.0ms	queued		0.0ms	queued	Assume best case empty queue
10	thread root_function.panel_controller.thr		0.0ms	no latency		0.0ms	no latency	
11	Connection		0.0ms	no latency		0.0ms	no latency	
12	thread root_function.speed_ctrl.accel_thr		5.0ms	sampling		5.0ms	sampling	Min: Round up to sampling period 5.0ms
13	thread root_function.speed_ctrl.accel_thr		0.0ms	no latency		5.0ms	deadline	
14	(bus can2)	1.0ms	10.001ms	transmission time	1.0ms	30.01ms	transmission time	Using data transfer time
15	Connection		10.001ms	no latency		30.01ms	no latency	Adding latency subtotal from protocols and bus - shown with ()
16	device root_function.acceleration		0.0ms	sampling		2.0ms	sampling	Best case 0 ms worst case 2.0ms (period) sampling delay
17	device root_function.acceleration		0.0ms	no latency		2.0ms	deadline	
18	Latency Total	2.0ms	16.000999999999998ms		2.0ms	45.010000000000005ms		
19	End to End Latency		40.0ms			50.0ms		
20	**End to end Latency Summary**							
21	WARNING	Minimum specified flow latency total 2,00ms less than expected minimum end to end latency 40,0ms (better response time)						
22	WARNING	Minimum actual latency total 16,0ms less than expected minimum end to end latency 40,0ms (faster actual minimum response time)						
23	SUCCESS	Maximum actual latency total 45,0ms is less or equal to expected maximum end to end latency 50,0ms						
24	WARNING	Jitter of actual latency total 16,0..45,0ms exceeds expected end to end latency jitter 40,0..50,0ms						

Conclusion

- **SCADE Solution for AADL**
  - AADL V2.2 extension on top of SCADE Architect targeting industrial usage
  - Model to Model transformations for Import/Export and Synchronization with SCADE Suite
- **Next features under study**
  - Multi-view AADL – FACE



Synchronization

Selection

Select elements to be synchronized

Show model selection for push and pull

Elements to push from the current model:

- ▣ FlightControlAADLArch
  - ▣ FlightControlSystemPackage
    - □ PitotStaticSystemDevice
    - □ Pilot_MFD_InputDevice
    - □ ThrottleActuatorDevice
    - □ ElevatorActuatorDevice
    - ☑ FCU
      - ☑ **speed**
      - ☑ **altitude**
      - ☑ **MCPaltitude**
      - ☑ **MCPspeed**
      - ☑ **AutoPilot**
      - ☑ **elevatorCmd**
      - ☑ **throttleCmd**
    - □ AirDataComputerProcess
  - □ integration

Elements to pull from the referenced model:

- ▣ FlightControl_FACE
  - ▣ FlightControl
    - ▣ FlightControl
      - □ Conceptual_Model
      - □ Logical_Model
      - ▣ Platform_Model
        - ☑ PlatformTypes
        - □ Entities
        - ☑ Views
          - ☑ **FlightControlActlInputInterfaceView**
          - ☑ **FlightControlTargetInputInterfaceView**
          - ☑ **FlightControlOutputInterfaceView**
    - ☑ UoP_Model
      - ☑ **FlightControl**
        - ☑ **FlightControlOutputInterface**
        - ☑ **FlightControlActlInputInterface**
        - ☑ **FlightControlTargetInputInterface**

ANSYS®

Thank You!