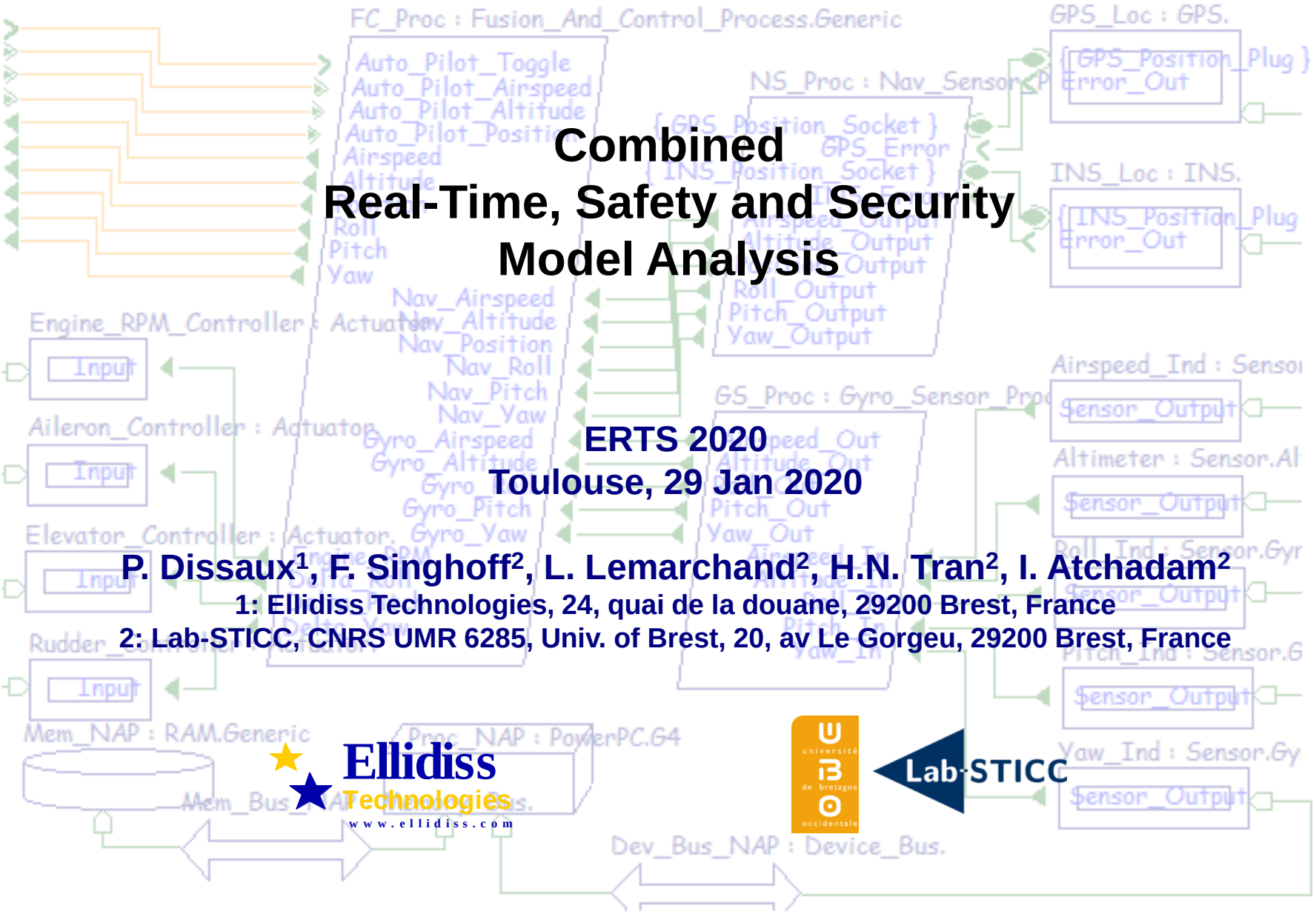


Combined Real-Time, Safety and Security Model Analysis

ERTS 2020

Toulouse, 29 Jan 2020

P. Dissaux[1], F. Singhoff[2], L. Lemarchand[2], H.N. Tran[2], I. Atchadam[2]

1: Ellidiss Technologies, 24, quai de la douane, 29200 Brest, France

2: Lab-STICC, CNRS UMR 6285, Univ. of Brest, 20, av Le Gorgeu, 29200 Brest, France

Ellidiss Technologies
www.ellidiss.com

Université de Bretagne Occidentale

Lab-STICC

Ellidiss Technologies
www.ellidiss.com

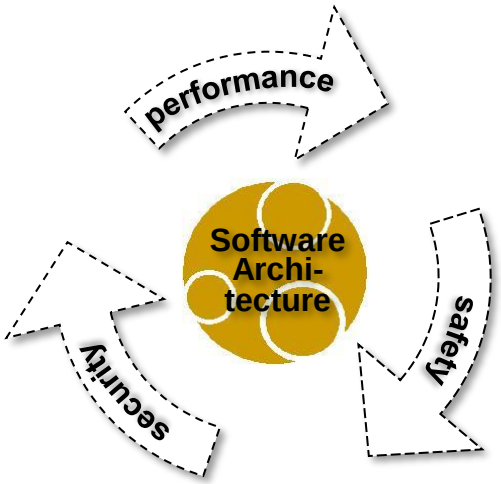
Scope

Model Driven Engineering for systems with embedded software

- *Step 1*: Modeling: designing before coding
- *Step 2*: Model analysis: early detection of defects
- *Step 3*: Model optimization: finding the best trade-offs

Multi-criteria model analysis

- Real-Time performance (response time, dataflow latency,…)
- Safety (mean time between failures,…)
- Security (confidentiality, integrity,…)
- Others: power consumption, weight, cost, …
- Possible conflicts:
  - Safety vs. Security
  - Safety and Security vs. Performance
- Using a single architectural model:
  - Reduces modeling effort
  - Increases the chances to find trade-offs



Experiment based on existing technologies

- Illustrative example: generic control-command system
- Pre-selected technologies and tools
- Current presentation focuses on *Steps 1 and 2*.
- *Step 3* is research work and has not been integrated yet

Ellidiss Technologies
www.ellidiss.com

Selected analysis topics

Real-Time performance analysis:

Scheduling Aware end to end Flow Latency Analysis (SAFLA):

- Select end to end data flows to be analysed
- Identify time-consuming data flow elements (threads, bus messages)
- Compute individual response times from scheduling analysis
- Sum up to estimate maximum flow latency

Safety analysis:

Fault Tree Analysis (FTA):

- Add error model information according to system composition
- Add error model information according to data flows
- Generate input file for specialized tools

Security analysis:

Common Criteria (CC):

- Availability:
  - Covered by performance and safety analysis
- Confidentiality & Integrity:
  - Define and implement data access control rules
  - Add security levels to data types
  - Run a rules checker

Ellidiss Technologies
www.ellidiss.com

Selected technologies

AADL modeling language

- Core language:
  - SW Architecture Description Language
  - Native support of Real-Time constructs
  - Can be enriched with Property Sets and Annexes (sub-languages)
- Behavior Specification annex (nominal behavior)
- Error Modeling annex (dysfunctional behavior)
- Security annex (still under development)

Tools

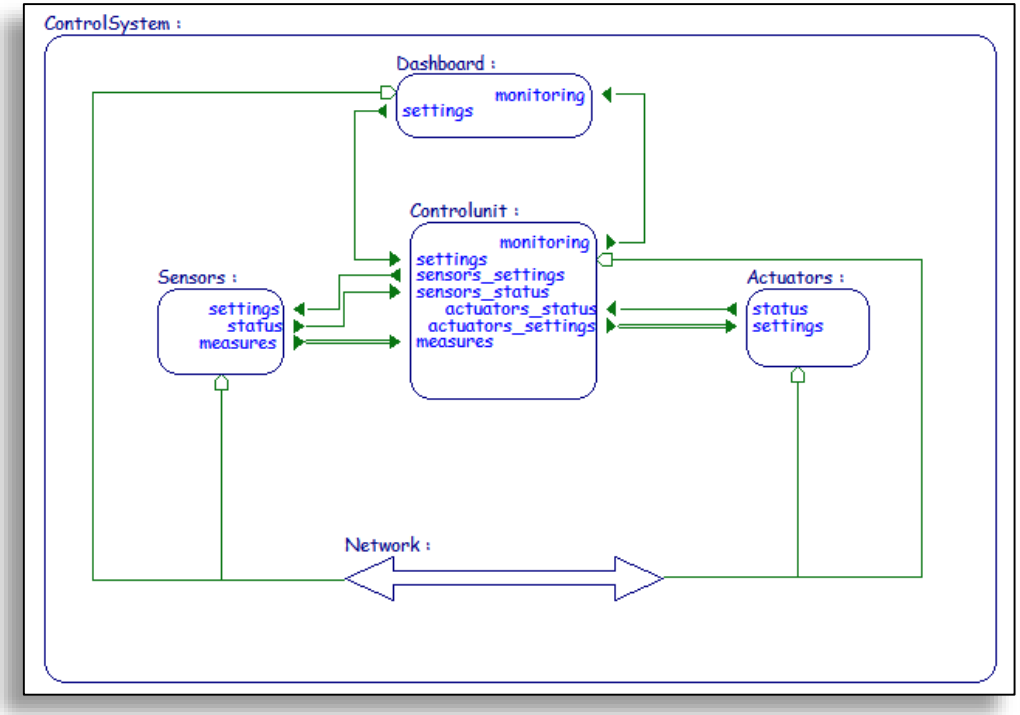
- Stood for AADL to build the model
  - Graphical editor for the architectural design phase
  - Detailed design structure to add properties and annexes
  - Automatic code generation of AADL source files
  - Design rules enforcement (HOOD)
- AADL Inspector to analyse it, including:
  - Cheddar: scheduling analysis (*beru.univ-brest.fr/~singhoff/cheddar*)
  - Marzhin: timing simulation
  - Arbre Analyste: fault tree analysis (*www.arbre-analyste.fr/en.html*)
  - LAMP: inline verification language

Illustrative example

AADL Architecture (1/3)

Generic control system:

- Sensors subsystem
- Control Unit subsystem
- Actuators subsystem
- Dashboard subsystem
- All subsystems distributed over a network



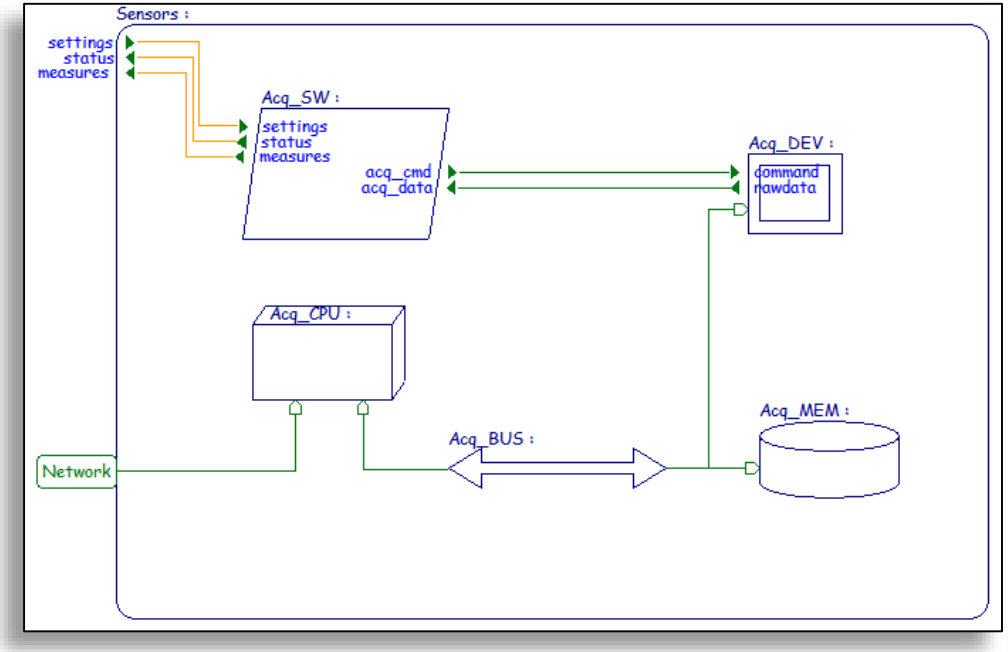

```
SYSTEM IMPLEMENTATION ControlSystem.others
SUBCOMPONENTS
 Sensors:      SYSTEM Sensors.others;
 Controlunit:  SYSTEM Controlunit.others;
 Actuators:    SYSTEM Actuators.others;
 Dashboard:    SYSTEM Dashboard.others;
 Network:      BUS Network;
CONNECTIONS
 cnx1:  PORT Dashboard.settings -> …
 cnx2:  PORT Controlunit.monitoring -> …
 cnx3:  PORT Controlunit.sensors_settings -> …
 cnx4:  PORT Sensors.status -> …
 cnx5:  PORT Sensors.measures -> …
 cnx6:  PORT Controlunit.actuators_settings -> …
 cnx7:  PORT Actuators.status -> …
 cnx8:  BUS ACCESS Network -> Dashboard.Nwk;
 cnx9:  BUS ACCESS Network -> Sensors.Nwk;
 cnx10: BUS ACCESS Network -> Actuators.Nwk;
 cnx11: BUS ACCESS Network -> Controlunit.Nwk;
PROPERTIES
 Actual_Connection_Binding => (reference(Network))
  applies to cnx1,cnx2,cnx3,cnx4,cnx5,cnx6,cnx7;
END ControlSystem.others;
```

Illustrative example

AADL Architecture (2/3)

Sensors subsystem:

- Acquisition software
- Acquisition processor
- Acquisition device
- Acquisition memory
- All distributed over a subnetwork

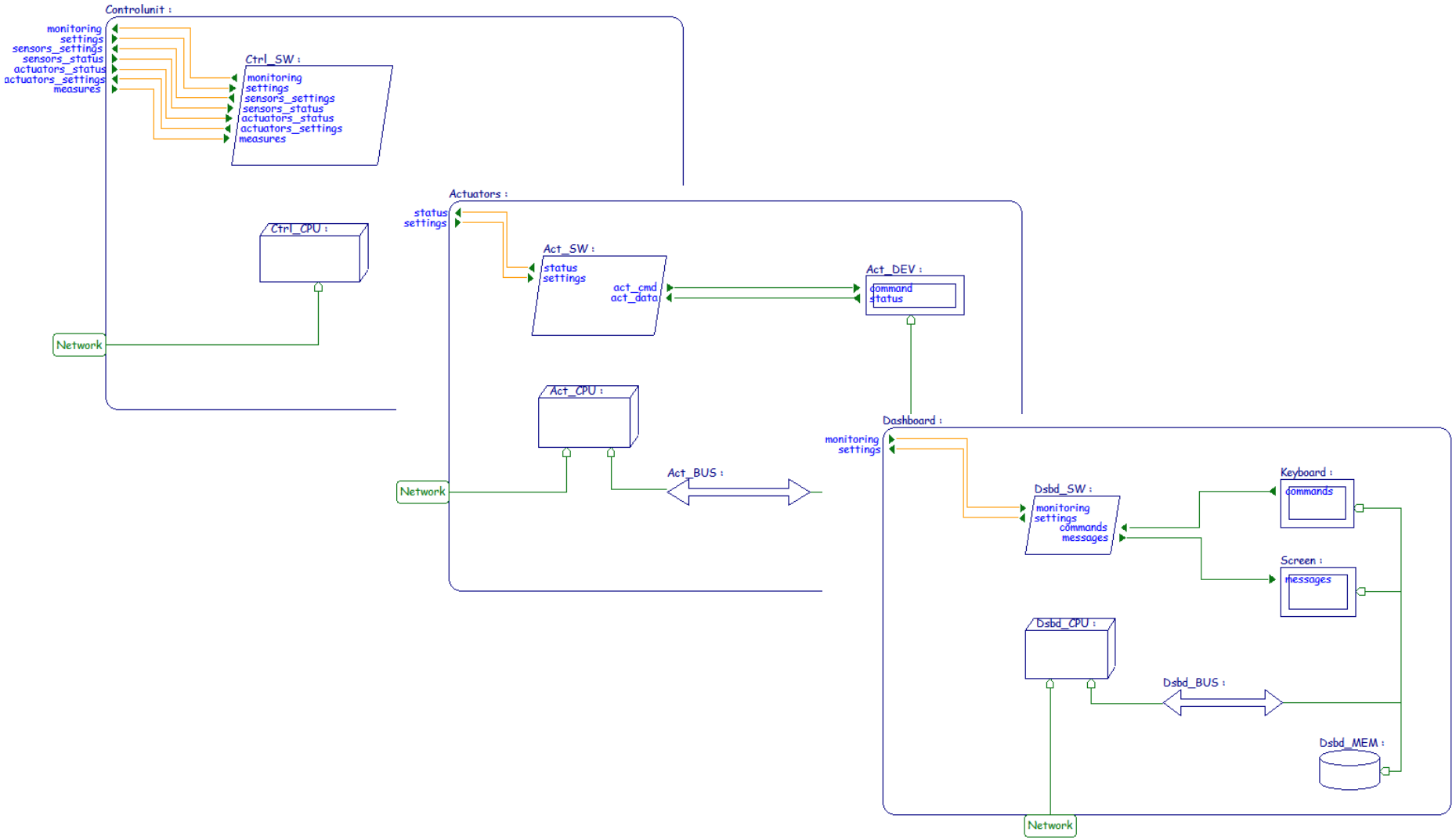



```
SYSTEM IMPLEMENTATION Sensors.others
SUBCOMPONENTS
  Acq_CPU : PROCESSOR Acq_CPU;
  Acq_MEM : MEMORY Acq_MEM;
  Acq_SW : PROCESS Acq_SW.others;
  Acq_DEV : DEVICE Acq_DEV;
  Acq_BUS : BUS Acq_BUS;
CONNECTIONS
  cnx1 : PORT settings -> Acq_SW.settings;
  cnx2 : PORT Acq_SW.status -> status;
  cnx3 : PORT Acq_SW.measures -> measures;
  cnx4 : PORT Acq_SW.acq_cmd -> Acq_DEV.command;
  cnx5 : PORT Acq_DEV.rawdata -> Acq_SW.acq_data;
  cnx7 : BUS ACCESS Acq_BUS -> Acq_CPU.Acq_BUS;
  cnx6 : BUS ACCESS Network -> Acq_CPU.Network;
  cnx9 : BUS ACCESS Acq_BUS -> Acq_MEM.Acq_BUS;
  cnx8 : BUS ACCESS Acq_BUS -> Acq_DEV.Acq_BUS;
PROPERTIES
  Actual_Processor_Binding => (reference(Acq_CPU))
applies to Acq_SW;
END Sensors.others;
```

Illustrative example

AADL Architecture (3/3)

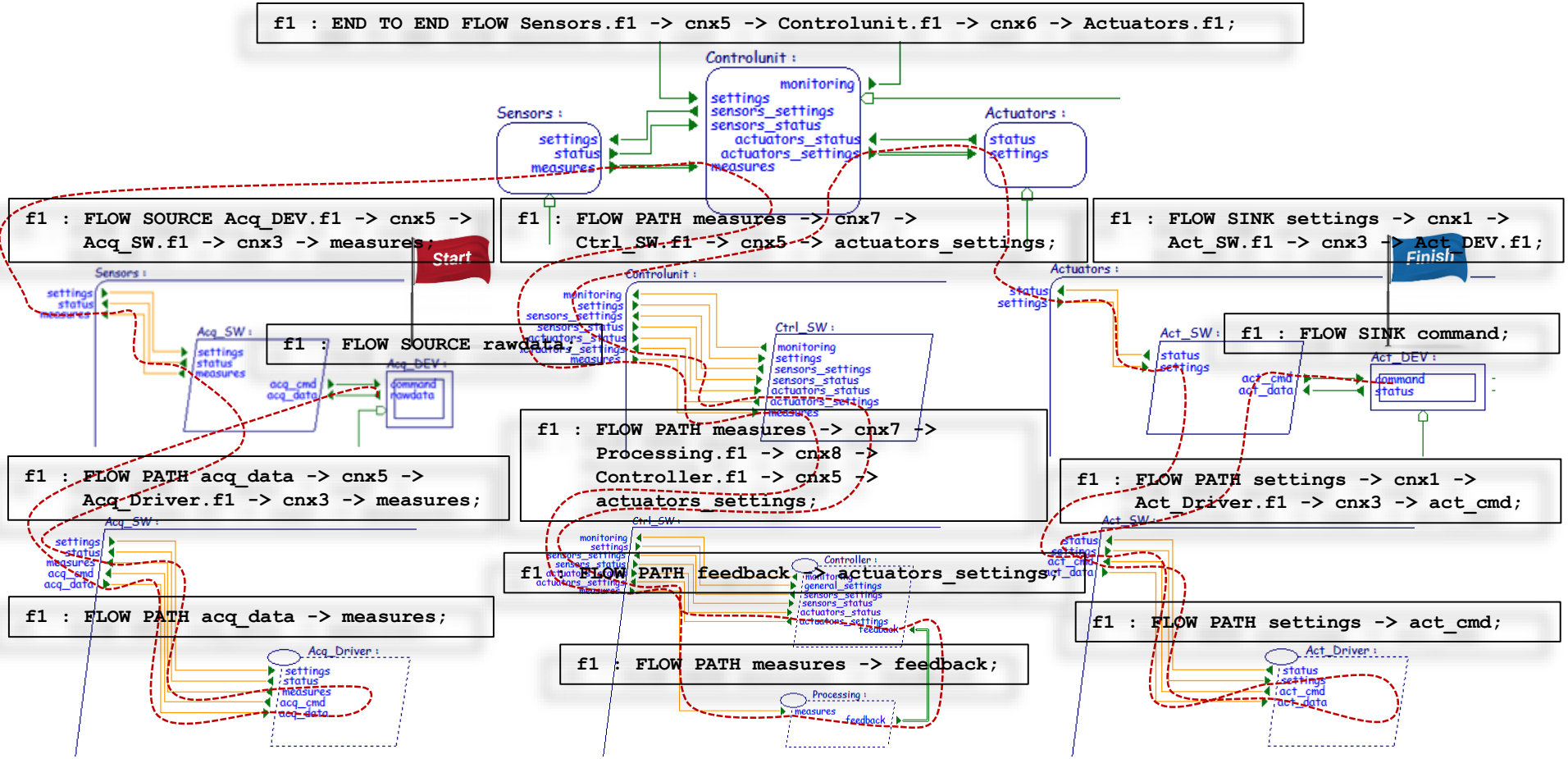
Similar decomposition for the other subsystems:



Illustrative example

Additional information for performance analysis:

Focus on *end-to-end flow latency*



Illustrative example

Additional information for safety analysis:

Focus on *composite error behavior* and *error propagation*

```
PACKAGE errorlibrary
PUBLIC
-- …
ANNEX EMV2 {**
 ERROR BEHAVIOR failstop
  EVENTS
    failure : ERROR EVENT;
  STATES
    operational : INITIAL STATE;
    failstop : STATE;
  TRANSITIONS
    failuretransition : operational -[ Failure ]-> failstop;
  END BEHAVIOR;
**};
-- …
END errorlibrary;
```

```
SYSTEM IMPLEMENTATION ControlSystem.others
-- …
ANNEX EMV2 {**
  use behavior errorlibrary::failstop;
  composite error behavior
  states
    [Dashboard.FailStop or Sensors.FailStop or
     ControlUnit.FailStop or Actuators.FailStop or
     Network.FailStop]-> FailStop;
  end composite;
**};
END ControlSystem.others;
```

```
SYSTEM IMPLEMENTATION Sensors.others
-- …
ANNEX EMV2 {**
  use behavior errorlibrary::failstop;
  composite error behavior
  states
    [Acq_CPU.FailStop or Acq_DEV.FailStop or
     Acq_BUS.FailStop]-> FailStop;
  end composite;
**};
END Sensors.others;
```

```
DEVICE Acq_DEV
FEATURES
  rawdata : OUT DATA PORT ControlSystemTypes::T_status;
-- …
ANNEX EMV2 {**
  use types errorlibrary;
  use behavior errorlibrary::failstop;

  error propagations
    rawdata : out propagation {NoValue};
  end propagations;

  component error behavior
  propagations
    p1 : FailStop -[]-> rawdata{NoValue};
  end component;

  properties
    EMV2::OccurrenceDistribution =>
      [ProbabilityValue => 1.0e-3; Distribution => Poisson;]
      applies to Failure;
**};
END Acq_DEV;
```

Ellidiss Technologies
www.ellidiss.com

Illustrative example

Additional information for security analysis:

Focus on: *data access control*

SW Engineering « good practices »:

- Modular decomposition with low residual coupling
- Data hiding: modeling restrictions, i.e. no AADL « provides data access » features
- Enforced by HOOD

Implement security rules, i.e.:

- *Sec_R1*: All components involved in a same end to end Flow must be at the same security level.
- *Sec_R2*: The security level of a component is the highest security level value associated with its Data ports.
- *Sec_R3*: When two components are connected via a shared Bus, they must comply with the No-Read-Up and No-Write-Down rules.

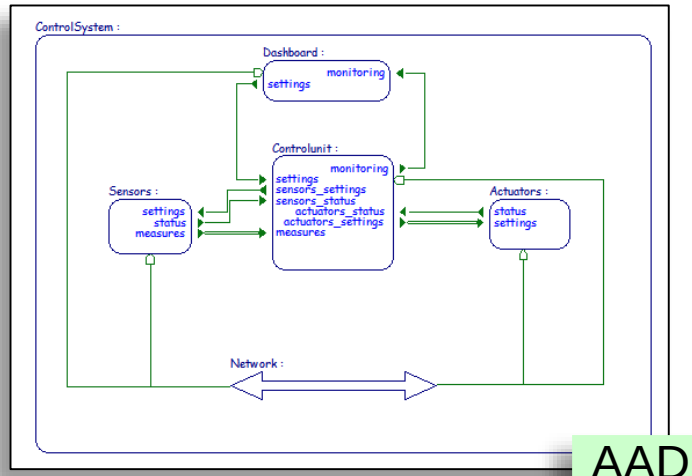
Add Security Level attribute to data:

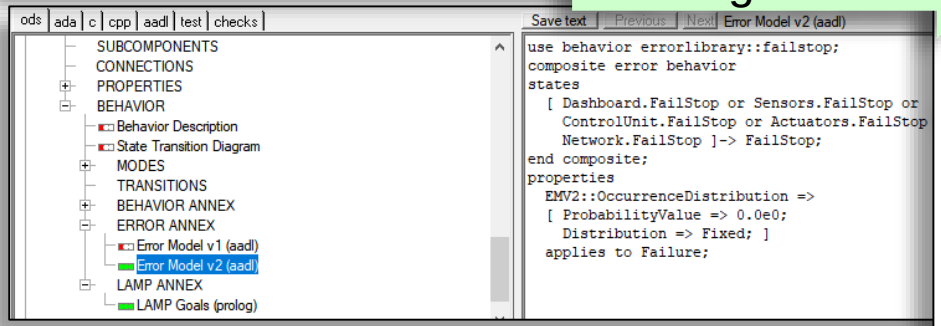
```
PROPERTY SET LAMP IS
-- …
Security_Level : AADLINTEGER
  APPLIES TO (Data, Data Access, Port, Parameter);
-- …
END LAMP;
```

```
PACKAGE ControlSystemTypes
PUBLIC

  DATA T_settings
  PROPERTIES
    LAMP::Security_Level => 3;
  END T_settings;

  DATA T_status
  PROPERTIES
    LAMP::Security_Level => 2;
  END T_status;

  DATA T_measures
  PROPERTIES
    LAMP::Security_Level => 5;
  END T_measures;

  DATA T_monitoring
  PROPERTIES
    LAMP::Security_Level => 2;
  END T_monitoring;


END ControlSystemTypes;
```

Running the experiment 1/4

AADL modeling with Stood





AADL generator

```
SYSTEM IMPLEMENTATION ControlSystem.others
SUBCOMPONENTS
 Sensors:      SYSTEM Sensors.others;
 Controlunit:  SYSTEM Controlunit.others;
 Actuators:    SYSTEM Actuators.others;
 Dashboard:    SYSTEM Dashboard.others;
 Network:      BUS Network;
CONNECTIONS
 cnx1:  PORT Dashboard.settings -> …
 cnx2:  PORT Controlunit.monitoring -> …
 cnx3:  PORT Controlunit.sensors_settings -> …
 cnx4:  PORT Sensors.status -> …
 cnx5:  PORT Sensors.measures -> …
 cnx6:  PORT Controlunit.actuators_se
 cnx7:  PORT Actuators.status -> …
 cnx8:  BUS ACCESS Network -> Dashboar    k;
 cnx9:  BUS ACCESS Network -> Sensor     wk;
 cnx10: BUS ACCESS Network -> Actu    rs.Nwk;
 cnx11: BUS ACCESS Network -> Co    rolunit Nwk;
FLOWS
 f1: END TO END FLOW
  Sensors.f1 -> cnx5 -> Controlunit.f1 ->    > Actuators.f1;
PROPERTIES
  tual_Connection_Binding => (refer    e(Network))
  applies to cnx1,cnx2,cnx3,cnx4    nx5,cnx6,cnx7;
 Timing => Immediate
  applies to cnx5,cnx6;
ANNEX EMV2 {**
 use behavior errorlibrary::fa
 composite error behavior
 states
  [ Dashboard.FailStop or
    Sensors.FailStop or
    ControlUnit.FailStop or
    Actuators.FailStop or
    Network.FailStop ]-> Fails
 end composite;
**};
END ControlSystem.others
```

Real Time

Safety

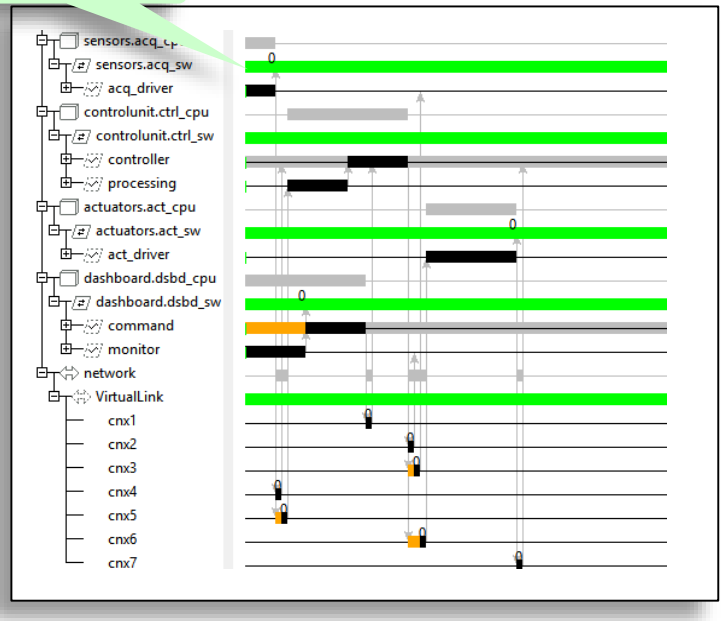
Security

```
PACKAGE ControlSystemTypes
PUBLIC

  DATA T_measures
  PROPERTIES
    LAMP::Security_Level => 5;
  END T_measures;

  DATA T_monitoring
  PROPERTIES
    LAMP::Security_Level => 2;
  END T_monitoring;

-- …

END ControlSystemTypes;
```

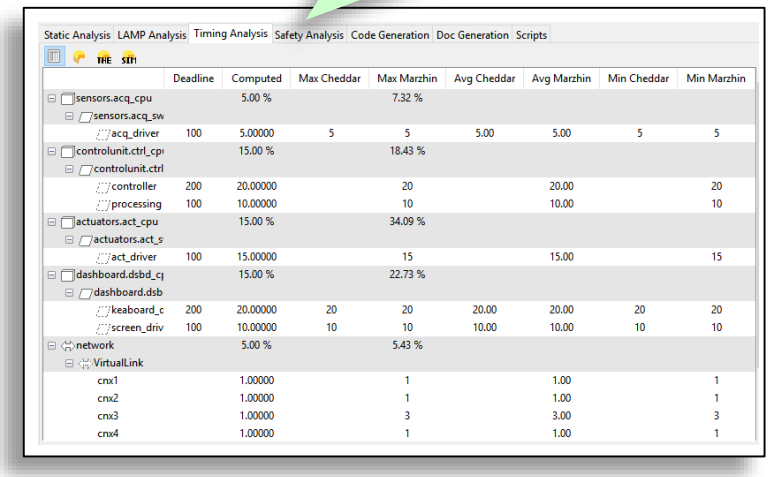
Running the experiment 2/4

Scheduling Aware end to end Flow Analysis with AADL Inspector (Marzhin and LAMP)

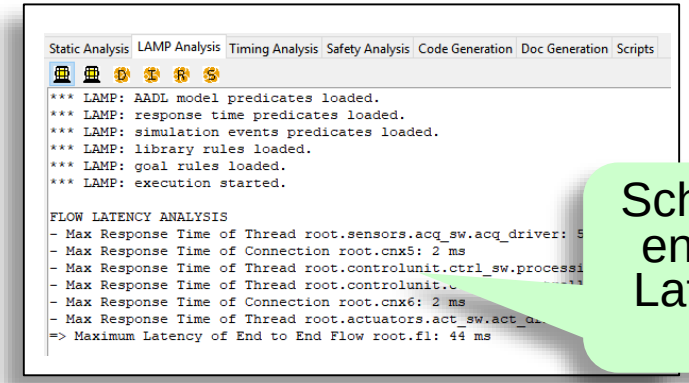
Simulation



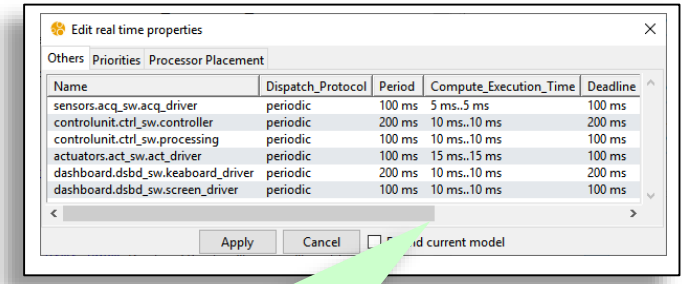
Response Time analysis



Scheduling Aware end to end Flow Latency Analysis with LAMP



Real-Time properties update



Ellidiss Technologies
www.ellidiss.com

Running the experiment 3/4

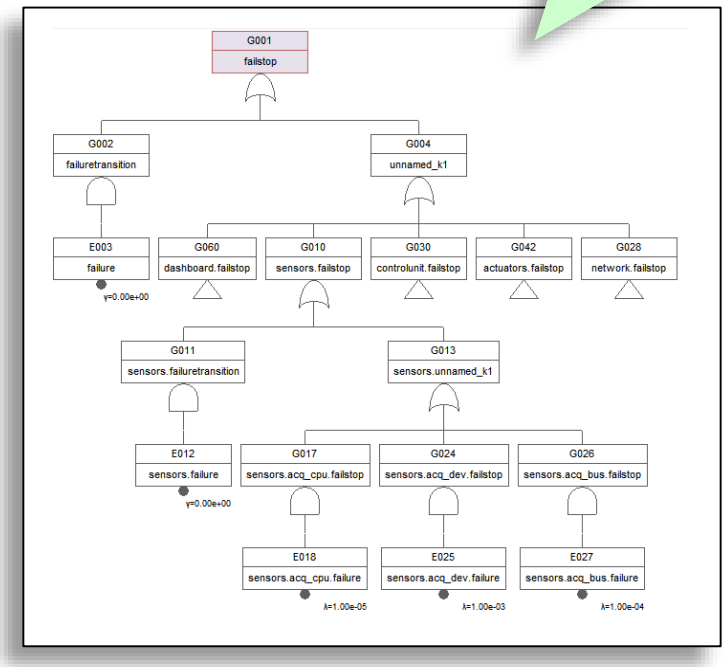
Open PSA generator

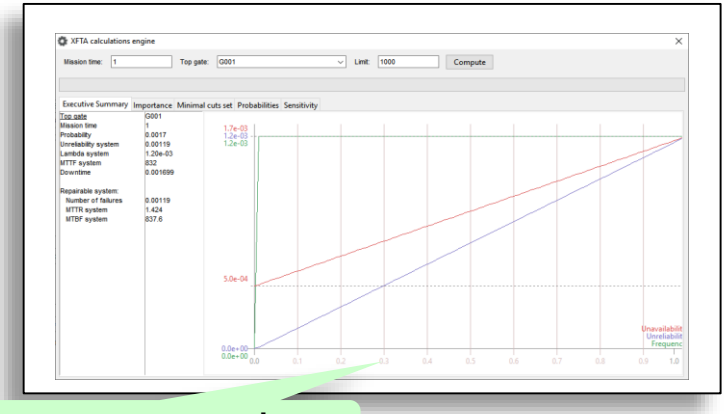
Static Analysis | LAMP Analysis | Timing Analysis | Safety Analysis | Code Generation | Doc Ge

```
<?xml version="1.0" encoding="UTF-8"?>
<open-psa author="Arbre Analyste" version="1.12">
  <label>from AADL model</label>
  <attributes>
    <attribute name="company" value="Ellidiss"/>
    <attribute name="author" value="AADL Inspector 1.7"/>
    <attribute name="creation-date" value="0"/>
    <attribute name="modification-date" value="0"/>
    <attribute name="version" value="1"/>
    <attribute name="performances" value="Q,F,T"/>
    <attribute name="page-id" value="1"/>
    <attribute name="page-1-name" value="A - failstop"/>
    <attribute name="page-1-description" value=""/>
    <attribute name="page-1-group" value="0"/>
    <attribute name="compute-id" value="1"/>
    <attribute name="compute-1-name" value="A - failstop"/>
    <attribute name="compute-1-gate" value="failstop"/>
    <attribute name="compute-1-time" value="1"/>
```

Fault Tree Analysis with AADL Inspector and Arbre Analyste (*)

Fault Tree Analysis





MTBF computation

(*) https://www.arbre-analyste.fr/en.html#

Running the experiment 4/4

Security Analysis with AADL Inspector (LAMP)

Security model

```
PROPERTY SET LAMP IS
-- …
Security_Level : AADLINTEGER APPLIES TO
  (Data, Data Access, Port, Parameter);
-- …
END LAMP;
```

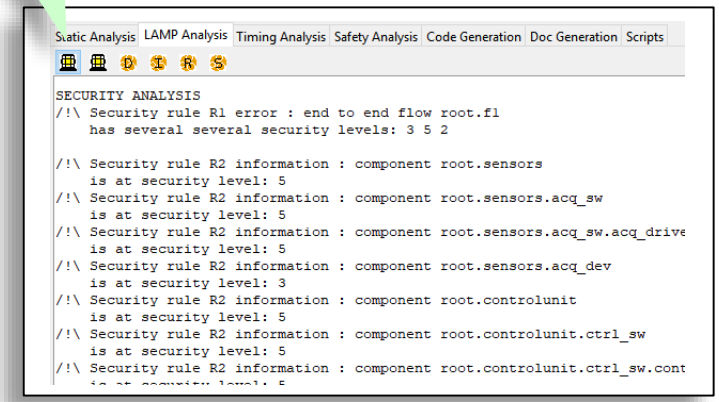
Security policy

- *Sec_R1*: All components involved in a same end to end Flow must be at the same security level.
- *Sec_R2*: The security level of a component is the higher security level value associated with its Data ports.
- *Sec_R3*: When two components are connected via a shared Bus, they must comply with the No-Read-Up and No-Write-Down rules.

Security rules implementation (LAMP)

```
PACKAGE ControlSystemAnalysis
PUBLIC

ANNEX LAMP {**
/* rule Sec_R1 */
checkFlowSecurity :-
 getRoot(R), getClassifier(R,P,T,I),
 getAncestorRec(P,T,I,Q,U,J),
 isFlowImplementation('END TO END',Q,U,J,E),
 concat('root.',E,F),
 getEndToEndFlow('root',E,M),
 getFlowSecurityLevels(M,[],L,0,N), N > 1,
 printMessageSec_R1(F,L).
checkFlowSecurity :- nl.

/* rule Sec_R2 */
checkMaxSecurityLevel :-
 getMaxSecurityLevel(X,L),
 printMessageSec_R2(X,L).
checkMaxSecurityLevel :- nl.

/* rule Sec_R3 */
checkNoWriteDown :-
 isAADLBusBinding(_,C,_),
 isAADLConnection(_,P,T,I,_,_,_,C,_,_,_,_),
 getConnectionEnds(P,T,I,C,Xs,Xd),
 getMaxSecurityLevel(Xs,Ls),
 getMaxSecurityLevel(Xd,Ld),
 Ls > Ld,
 printMessageSec_R3(C,Ls,Ld).
checkNoWriteDown :- nl.

-- …
END ControlSystemAnalysis;
```

Security assessment (LAMP)



Example of optimization (1/2): assign tasks to ARINC 653 partitions according to deadline and security objectives

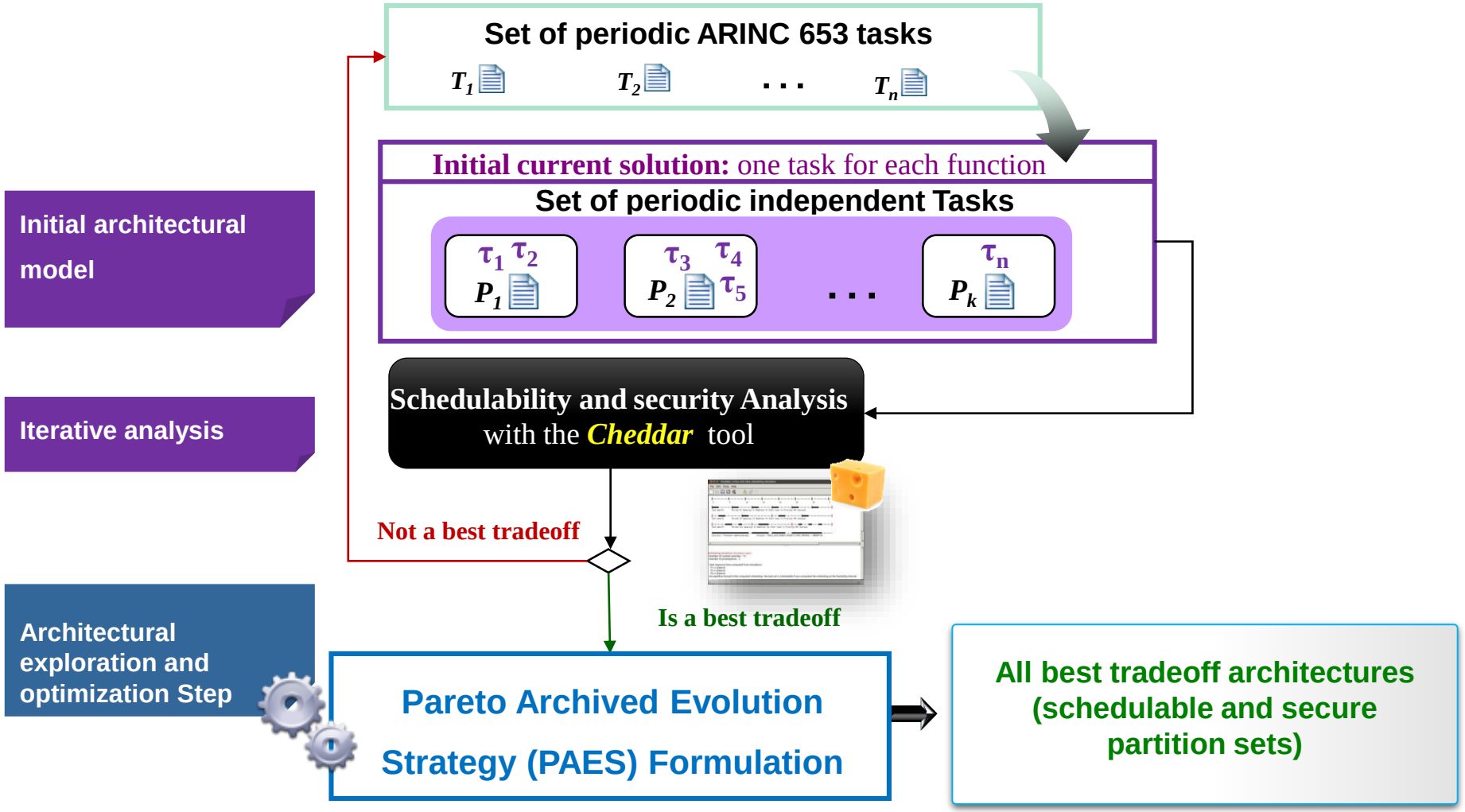
Problem Statement:

- Conflicting objective functions: security violations of the ARINC 653 communications and task deadline violations
- Tradeoffs between large number of candidate software architectures to assign tasks to partitions
- Numerous possible tradeoffs: cannot be computed by hand … need a heuristic

Contributions:

- Formulation based on PAES (Pareto Archived Evolution Strategy) to explore possible tasks to partition assignments
- Security verifications with Biba/La padula rules. Schedulability assessed by scheduling simulation
- Implemented into Cheddar

Example of optimization (2/2): assign tasks to ARINC 653 partitions according to deadline and security objectives



Ellidiss Technologies
www.ellidiss.com

Conclusion

Limitations of this first experiment

- Only addresses one pre-selected modeling language (AADL)
- Only address a few pre-selected analysis techniques (SAFLA, FTA, CC)
- Only considers pre-selected tools (Stood, AADL Inspector)
- Did not address multi-criteria model optimization yet

Foreseen future work:

- Add automatic generation error model when possible
- Add initial SysML to AADL transformation where meaningful
- Provide an analysis dashboard to ease results interpretation
- Integrate multi-criteria optimization developed by UBO/Lab-STICC
- Apply to other case studies

To learn more about this realization:

- Attend session Th.2.C.2 on Thursday 30 January at 2:30 pm in room Ariane 1
  « *LAMP: A new model processing language for AADL* »
- Visit our stand in the exhibition area for a live demo !