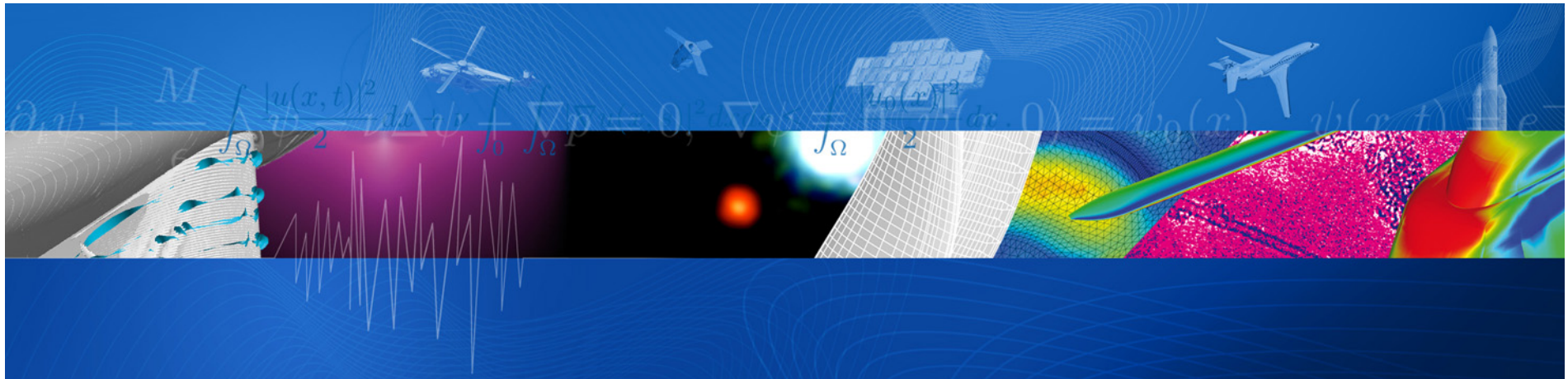


FORC^{3}ES

Contributors Forces 3: Pierre-Loïc Garoche, Thomas Loquen, Eric Noulard, Claire Pagetti, Clément Roos, Pierre Roux

FMF

October 10th 2017

ONERA

THE FRENCH AEROSPACE LAB

return on innovation

Outline

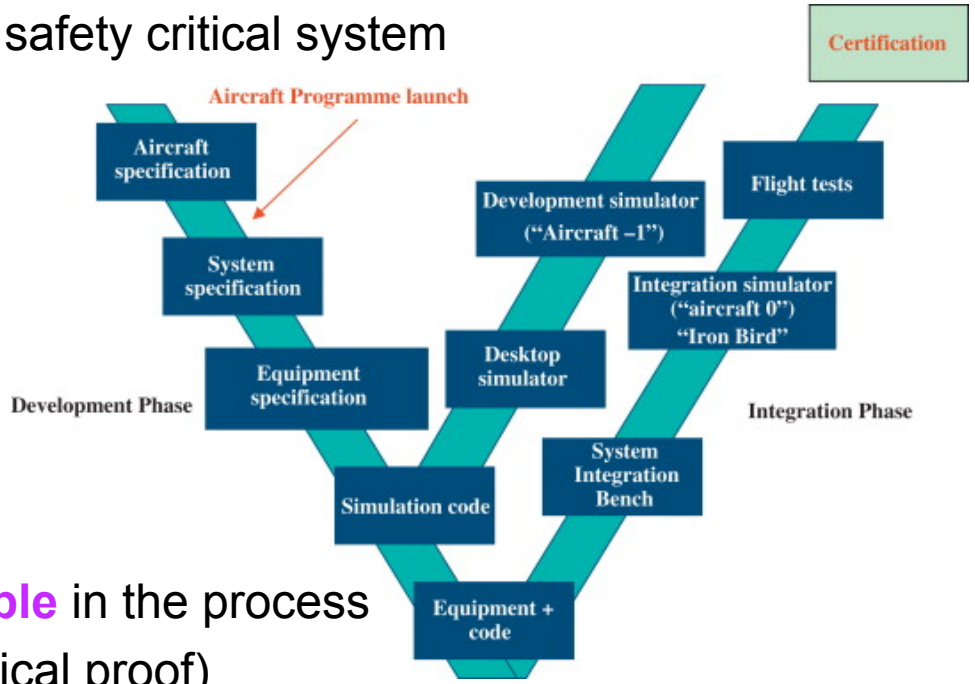
- Introduction
  1. Project description
  2. Example “avion jaune”
- Safe code generation
- Verification
- Conclusion

Context

- **Certified** embedded systems
  - Constraints on the development process / Verification objectives
  - Control-command application: safety critical system

- Issues
  - Safety
  - At limited costs

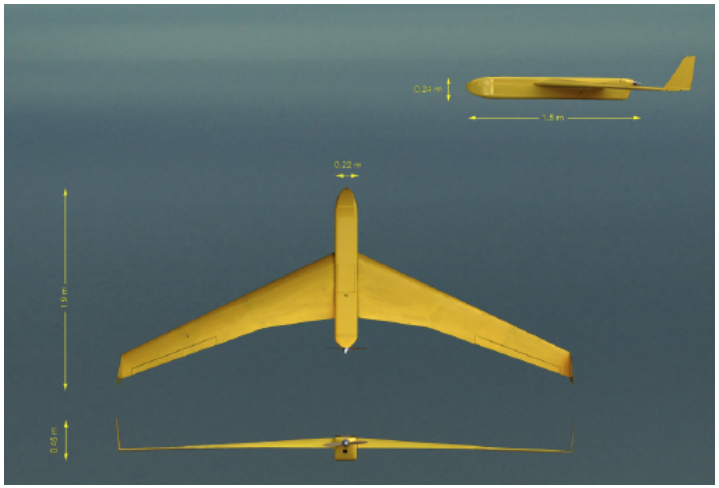
- Technical means
  - **Automatisation**
  - Verification **as early as possible** in the process
  - **Formal** verification (mathematical proof)



Application target

- Targeted for civil aircrafts
  - With certification constraints

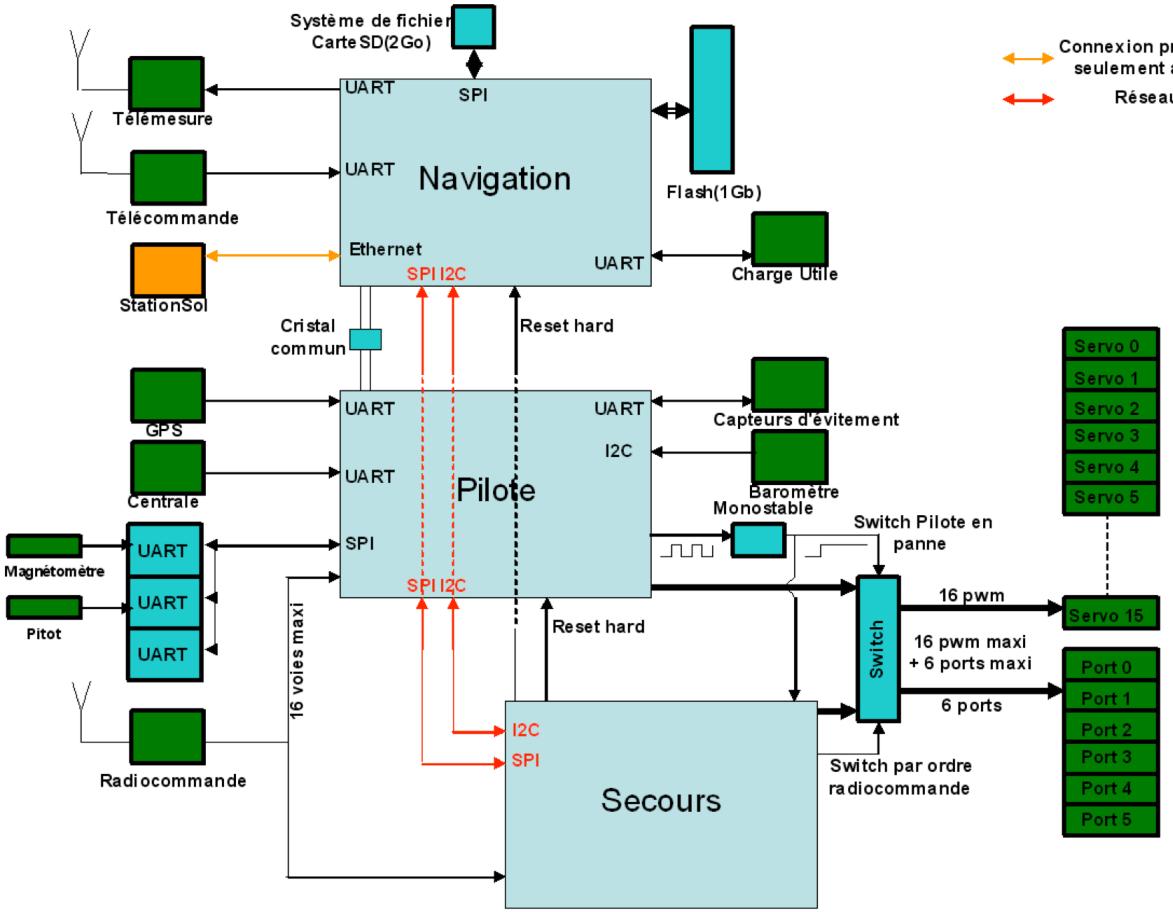
- Experimented on a UAV: « Avion jaune »
  - 11kg (empty)
  - 4m span
  - 4h autonomy



Architecture

Based in 3 micro-controllers ARM Cortex M3
(80MHz, Flash : 512Ko, internal RAM : 96Ko)

1 function ↔ 1 micro-controller



NAVIGATION : flight management

PILOTE : commande

SECOURS : hand over the manual pilot in case of problem

ONERA
THE FRENCH AEROSPACE LAB

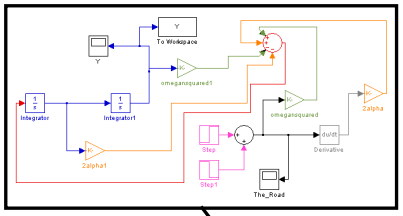
Objective of Forc^{3}es

- Formal development chain
  - Automated development
  - With verification at all stages
  - Dedicated to control-command systems
  - From control laws to embedded code

- Global Optimisation
  - Bridging the gap between system and software
  - Carrying information to ease formal verification all along the way

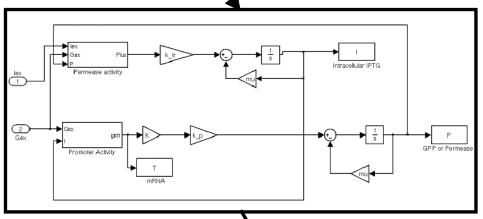
- Incrementality
  - Manage evolutions efficiently

Trends in avionic domain



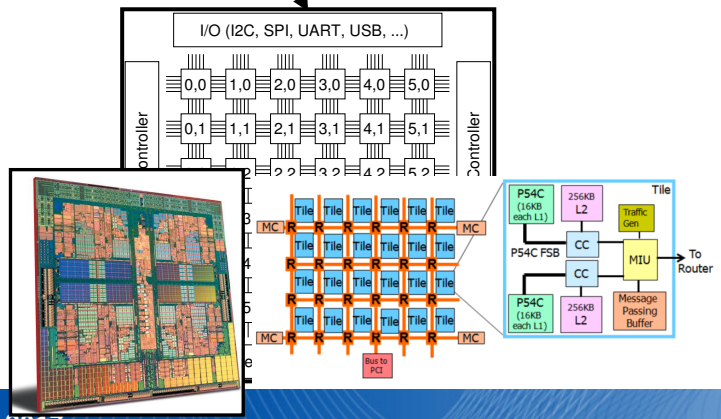
Control design level

- Tools: Matlab / Simulink
- High level requirements



Implementation level

– Coding of elementary blocks: Scade …
– Coding of multi-periodic assembly: home made language, manual coding, …

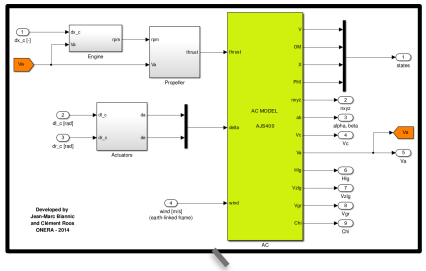


COTS hardware integration

– Automatic code generator
– Manual code
– Low level services
– WCET assessment

ONERA
THE FRENCH AEROSPACE LAB

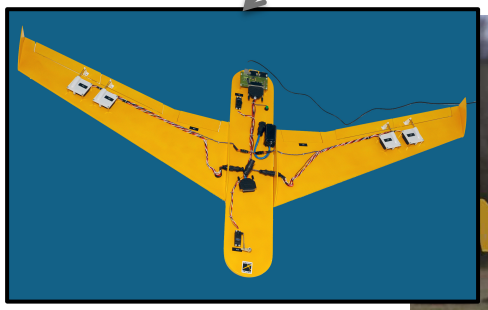
Objective of the Forc³es Integrated Development Framework



verification

```
node voter (s1, s2, s3: real)
returns (sensor: real);
let
   : :
tel
```

```
imported node syst ((s1, s2, s3: real)
returns (sensor: real) wcet 5;
node ex (i1 : rate(8,0); i2 : rate(20,0))
returns (01, 02);
let
   : : :
tel
```



Control design level

- Steps:
  - discrete models → choice of sampling periods
  - performance properties → **definition of simulation tests**
  - Integer properties→ **identification of invariants**
- Tools: Matlab / Simulink

Implementation level

- Steps:
  - Coding: **automatic code generation**
    - elementary blocks: Lustre
    - multi-periodic assembly: Prelude
  - **Verification**
    - Performances : simulation with SchedMcore
    - Invariants analysis

Integration level

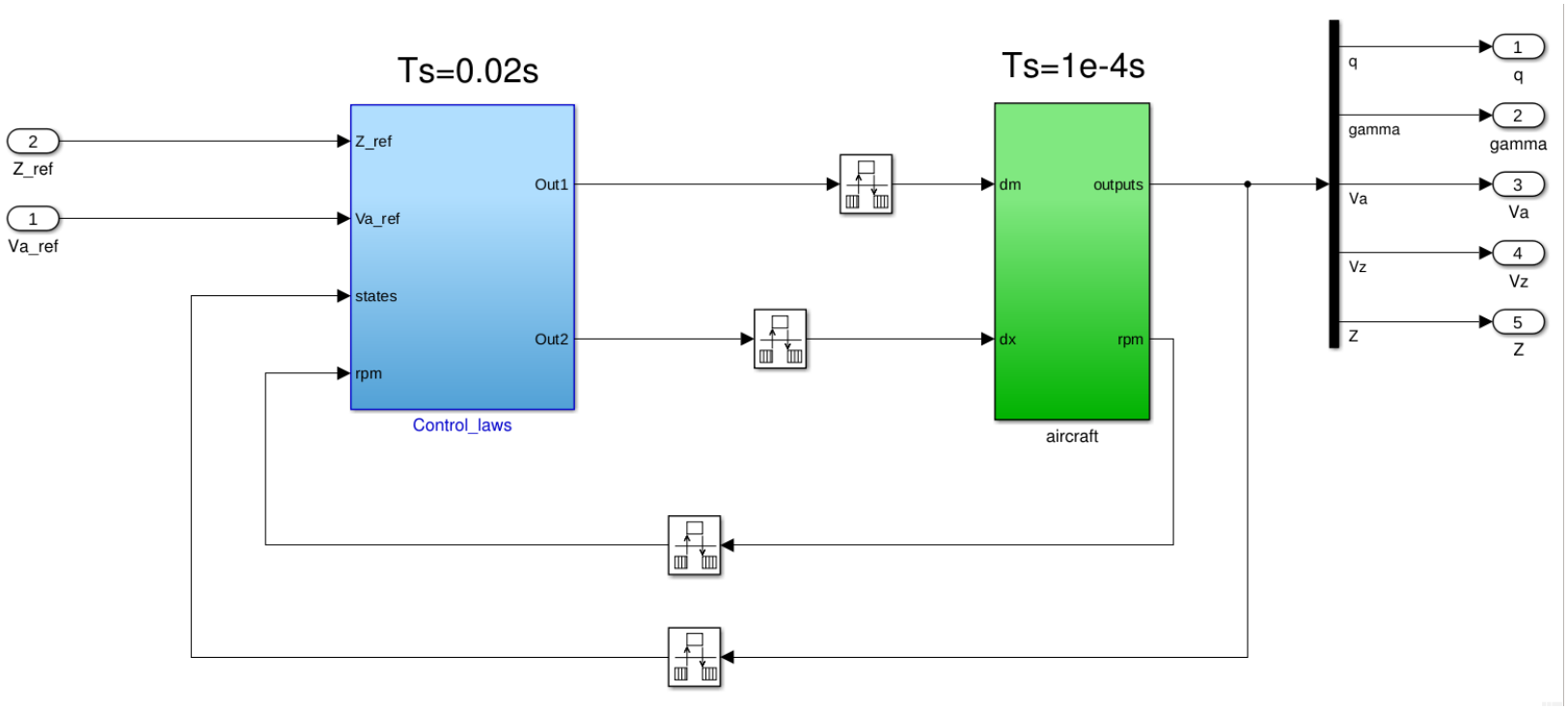
- Steps:
  - Integration + test on the iron bird
  - Integration on the real UAV

Outline

- Introduction
  1. Project description
  2. Example “avion jaune”
- Safe code generation
- Verification
- Conclusion

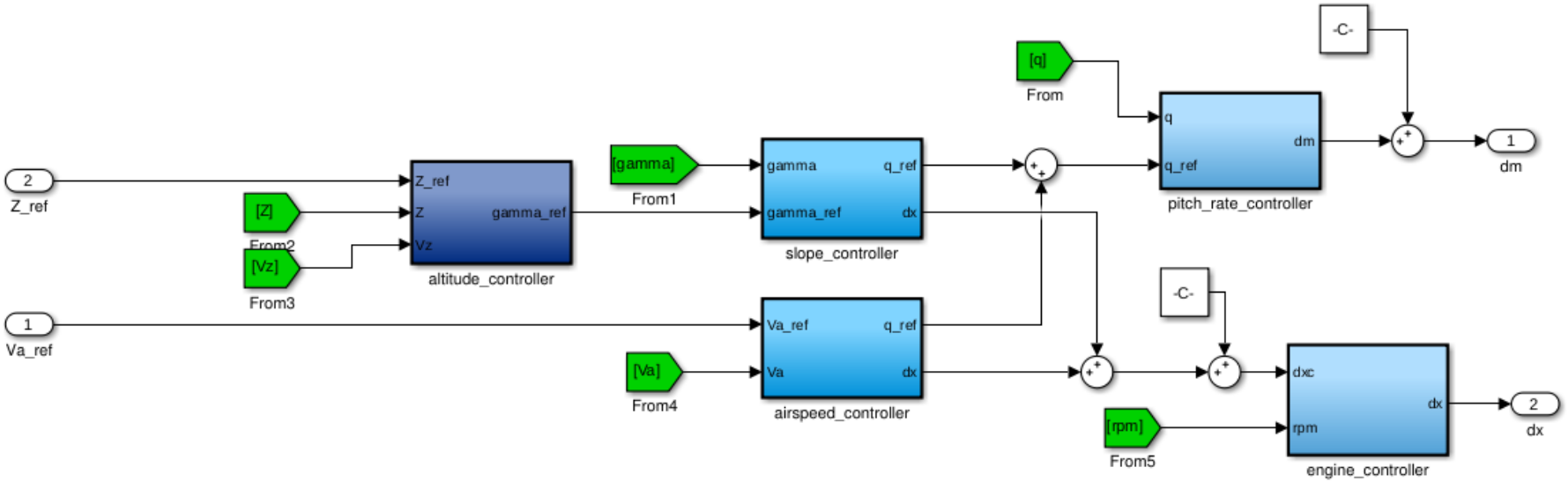
« Avion jaune » longitudinal control part

- Nonlinear aircraft model (dynamic model + actuator)
- Linearization around a flight point
- Controller synthesis



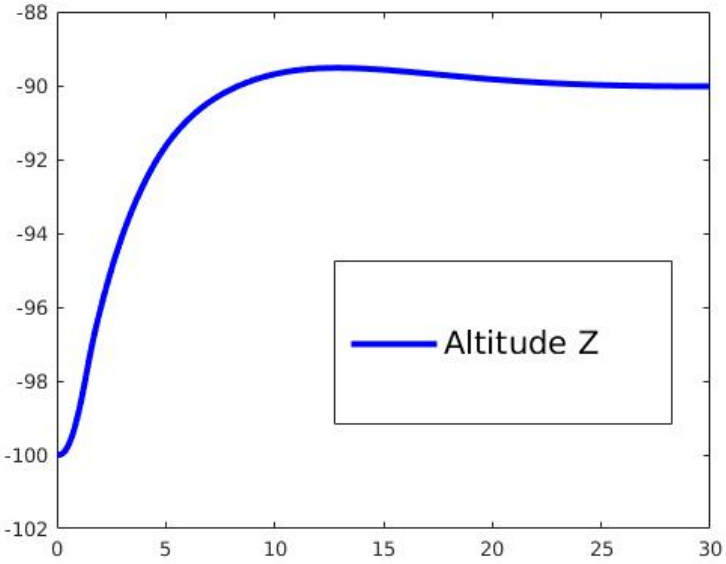
« Avion jaune » longitudinal control part

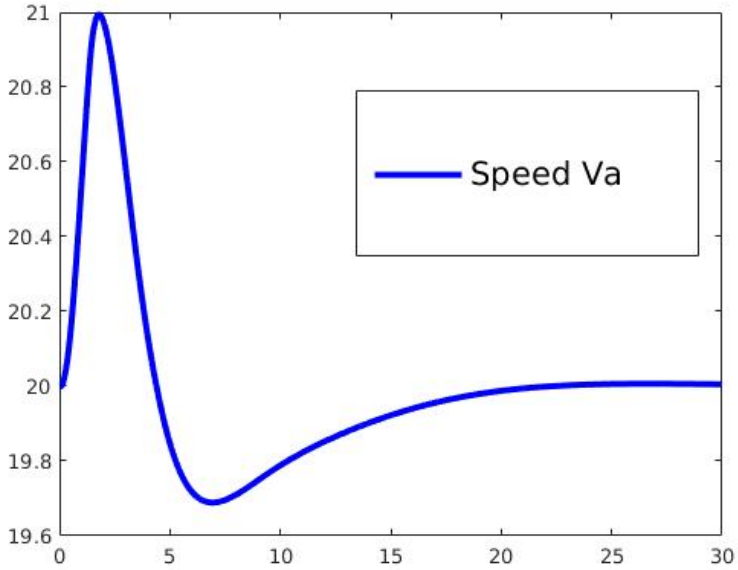
- control laws allowing *speed control* and *altitude control*



Performance properties

- Altitude change request



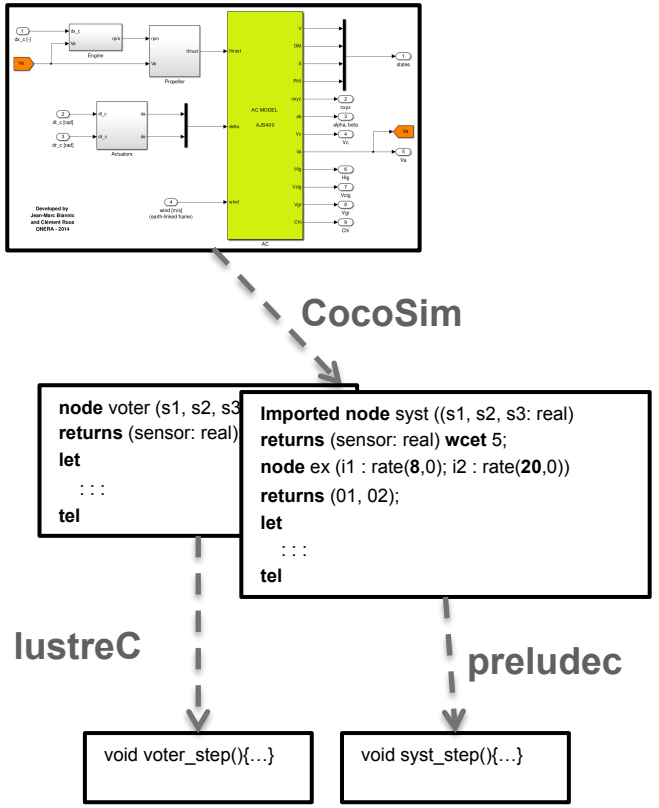


Outline

- Introduction
- Safe code generation
- Verification
- Conclusion

Safe code generation

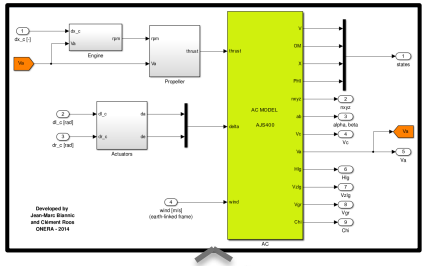
- Principle: several tools
  - Simulink → synchronous languages: ONERA/NASA toolbox (CocoSim/ Sim2LustrePrelude)
  - Lustre → C: lustreC compiler (ONERA/ ENSEEIHT)
  - Prelude→C: preludec compiler (ONERA/ LIFL)



1. Simulink → synchronous language code generation

- Principles:
  - Elementary blocks (gain, integrator, …) and mono-periodic assemblies → Lustre
  - Multi-periodic assemblies → Prelude
    - Software architecture language
    - With real-time primitives

  1) Exploration of the Simulink architecture
  2) Pretty-printing



Lustre

```
node voter (s1, s2, s3: real)
returns (sensor: real);
let
   : : :
tel
```

Prelude

```
Imported node voter ((s1, s2, s3: real)
returns (sensor: real) wcet 5;
node ex (i1 : rate(8,0); i2 : rate(20,0))
returns (01, 02);
let
   : : :
tel
```

- ONERA/NASA toolbox (CocoSim)
  - Open source Matlab library
  - **Input**: Simulink
  - **Outputs**: Lustre / Prelude
  - **Advantages**: optimized for and specific to ONERA/NASA controllers
  - **Drawback**: does not apply to any Simulink system

Translation tool: example – UAV control design

```
%file main.m
% Data loading
% UAV data include in AJ.mat
load AJ
T_env=1e-4; % UAV sample-time ('continuous' time)
%load controller data
controller_data;
Fs=50;    % Auto-pilot frequency [Hz]
Ts=1/Fs;  % Auto-pilot sample time [s]
%name of the Simulink model
simulink_model='bf_nl_discret';

%main routine; main node in Prelude, no comment
cocoSim(simulink_model, [], T_env, [], [],true, [],false);
```

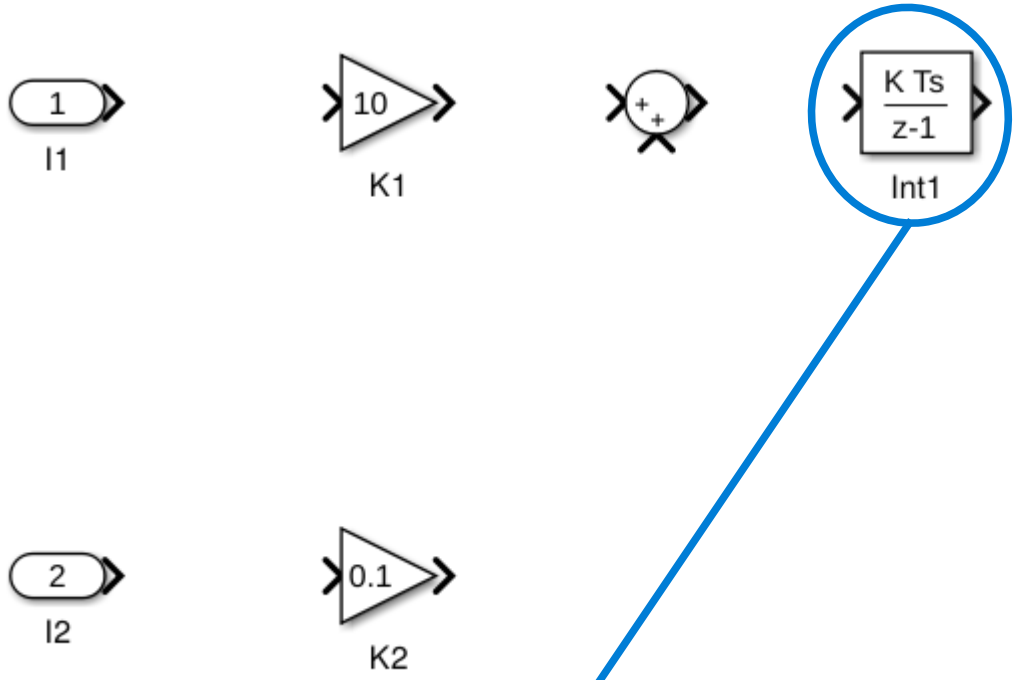
Results: synchronous language files

bf_nl_discret_bloc.lus Lustre elementary nodes and mono-rate assembly

bf_nl_discret_assemblage.plu Prelude assembly of multi-rate nodes

Simulink ➔ synchronous languages (main steps)

Simulink ➔ synchronous languages (main steps)

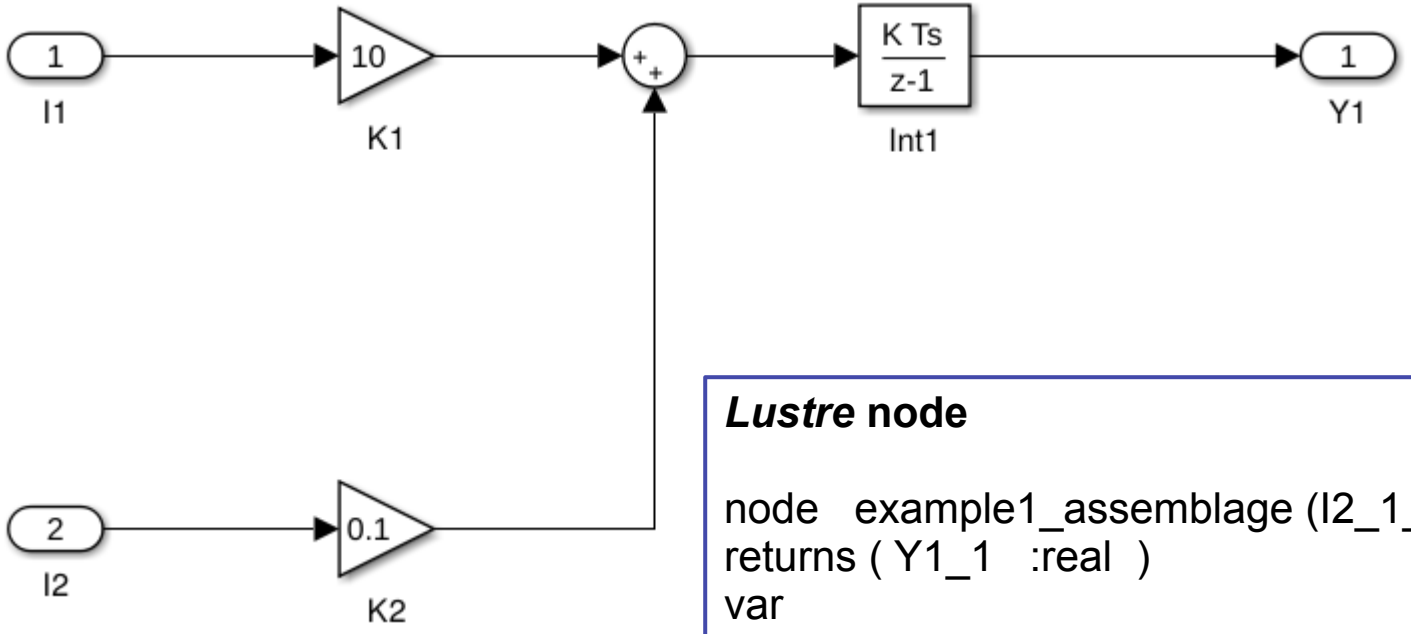


***Lustre* node with state**

```
node  example1_Int1 (Sum_1_1 :real )
returns ( Int1_1_1  :real )
let
      Int1_1_1 = 0.200 -> (1.00 * 0.0100 * pre Sum_1_1)
                          + pre Int1_1_1;
tel
```

```
origin_name: {'example1/Int1'}
 name: {'example1/Int1'}
 annotation: 1.5430e+03
  type: {'Integrator'}
  num_input: 1
  num_output: 1
  pre: {[1.5400e+03]}
  srcport: {[0]}
  post: {[1.5450e+03]}
  dstport: {[0]}
  rounding: 'Floor'
  prename: {{1x1 cell}}
  postname: {{1x1 cell}}
  sample_time: '0.01'
  srcport_size: 1
  dstport_size: 1
  inports_dt: {'double'}
  outports_dt: {'double'}
  inports_dim: [1 1]
  outports_dim: [1 1]
  conversion: {'double'}
  ismulti: 0
  [...]
```

Simulink ➔ synchronous languages (main steps)

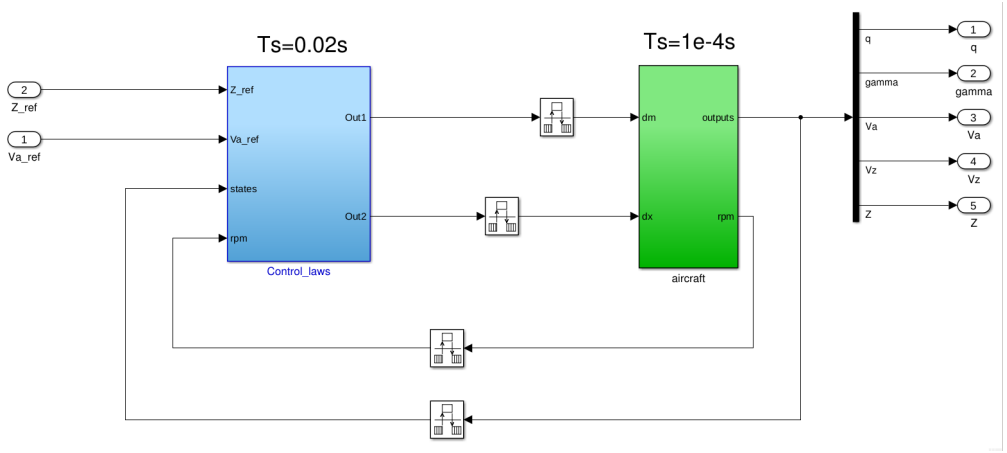


***Lustre* node**

```
node   example1_assemblage (I2_1_1, I1_1_1  :real)
returns ( Y1_1   :real  )
var
        Int1_1_1, K1_1_1,  K2_1_1 :real ;

 let
       (Int1_1_1) = example1_Int1( K1_1_1+K2_1_1) ;
       (K1_1_1) = 10*I1_1_1;
       (K2_1_1) = 0.1*I2_1_1;
        Y1_1 =  Int1_1_1 ;
tel
```

Simulink ➔ synchronous languages (main steps)



***Prelude* node**

```
node   example1_assemblage (Z_ref_1_1, Va_ref_1_1  :real rate (200,0))
returns ( q_1_1, gamma_1_1, Va_1_1, Vz_1_1, Z_1_1   :real  )
var
        out1_1_1, out2_1_1:real ;

 let
       (out1_1_1, out2_1_1) = control_laws(Z_ref_1_1, Va_ref_1_1 , (i1 fby (q_1_1,
                                        gamma_1_1, Va_1_1, Vz_1_1, Z_1_1))/^200,
                                      (i2 fby rpms) /^200) ;
       (q_1_1, gamma_1_1, Va_1_1, Vz_1_1, Z_1_1, rpms) =
                          aircraft (out1_1_1 *^200, out2_1_1 *^200)
```

Related work

- Many company internal code generators

- GENE-AUTO
  - open-source code generation toolset (ITEA project)
  - **input**: Simulink, Stateflow or Scicos
  - **output**: C or Ada; lustre (collaboration ONERA, IRIT, NASA)
  - **limits:**
    - Matlab Simulink r2008b, no fancy blocks, no complex z-expressions, just unit delays
    - Translate from an output generated file (not directly playing with the internal Simulink representation)

- Stavros Tripakis, Christos Sofronis, Paul Caspi, Adrian Curic. *Translating discrete-time Simulink to Lustre*. ACM Transaction Embedded Computing Systems 4(4): 779-818 (2005)

2. Lustre compiler (LustreC)

- Output: C or Java

- Open source implementation of Biernacki et al.

[Dariusz Biernacki, Jean-Louis Colaço, Grégoire Hamon, and Marc Pouzet. Clock-directed modular code generation for synchronous data- flow languages. In Kriszti an Flautner and John Regehr, editors, LCTES, pages 121–130. ACM, 2008]

- Collaboration with IRIT-ENSEEIHT (X. Thirioux)
  Available at cavale.enseeiht.fr/redmine/projects/lustrec

- Compilation strategy
  - Same as Esterel Scade certified compiler
  - Less efficient than other compilation scheme but … used in industry and certifiable
  - Provide traceability (and verification means)

3. Compilateur Prelude (Preludec)

- Output: C

- Open source implementation
  - Collaboration with LIFL (J. Forget) and Airbus Defence and Space (D. Lesens)
  - Available at https://svn.onera.fr/Prelude

- Compilation strategy
  - Static communication buffers
  - Shared memory or message passing

Outline

- Introduction
- Safe code generation
- Verification
- Conclusion

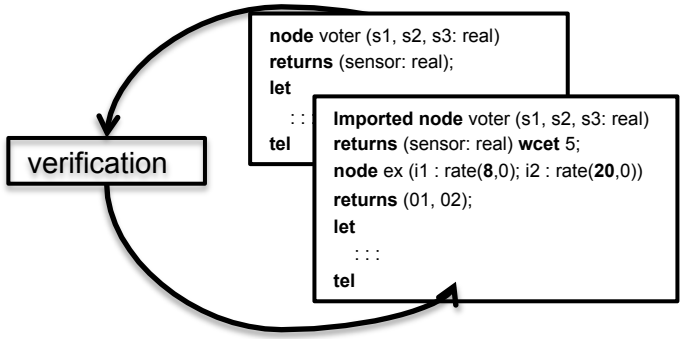
Verification activities

1 Property 1:

- **Performance** analysis
  - simulation

2 Property 2:

- **Discrete properties** verification
  - formal verification techniques



Property 1. Simulation at implementation level

- SchedMcore
  open source ONERA tool http://sites.onera.fr/schedmcore/
- Objective
  - Simulation of the specification
  - Real-time constraints are assumed correct
- Input
  - Lustre and Prelude programs
    (several assemblies can be encoded)
- Results
  - Temporal simulations according to the scenario tests defined at control design level
  - if simulations are compliant,
    - high level properties are still satisfied
    - It simplifies the integration level

Implantation level: example – UAV control design

Define simulation parameters

Complete the file bf_nl_discret_assemblage_include_sensors_actuators.c to select outputs to observe (actuators) and the value of inputs (sensors).

Nominal flight point of the aircraft

- Airspeed Va=20m/s
- Altitude Z=100m.

Requirements

- Airspeed =20m/s
- Altitude = 90m

Use provided makefile to compile with Lustrec and Preludec.

Results :

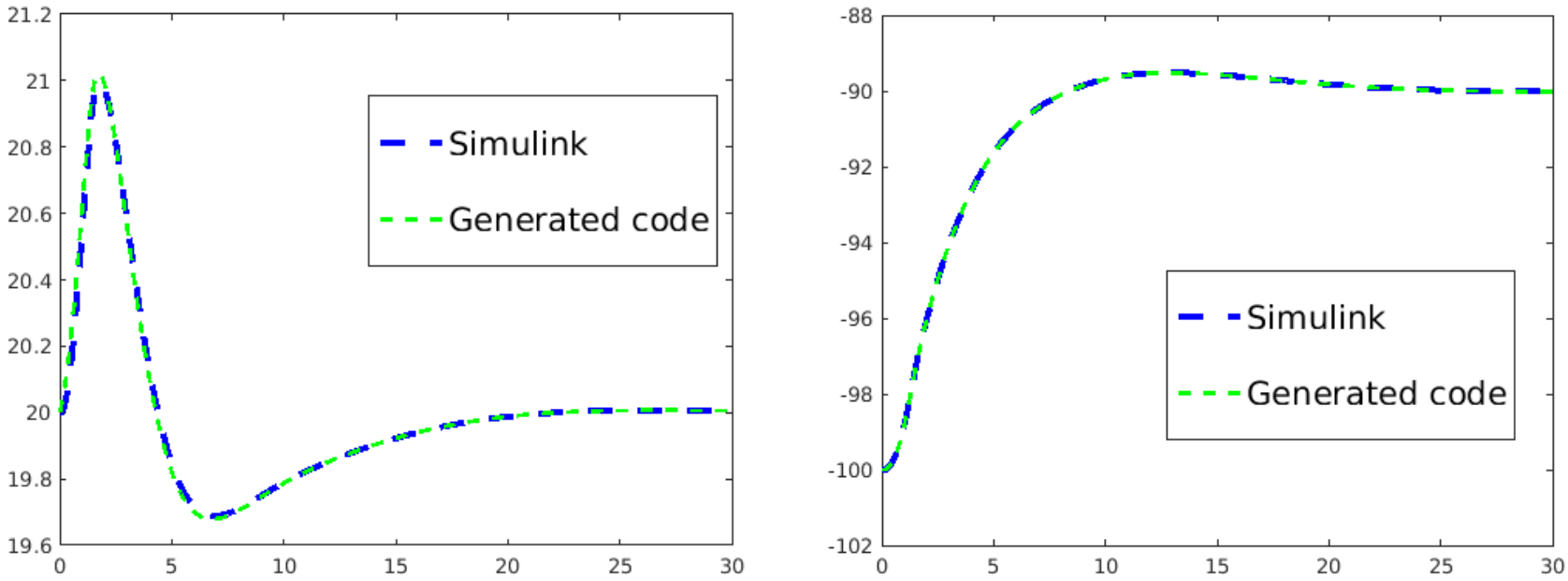
binary loadable library lib**filename**.so (or generated C) usable by various tools :

- scheduling analysis tool
- functional simulator
- real-time executor
- target runtime (OS, supervisor, in-house runtime)

Simulation level: example – UAV control design

Simulation on 30s with a step time of 1e-4s

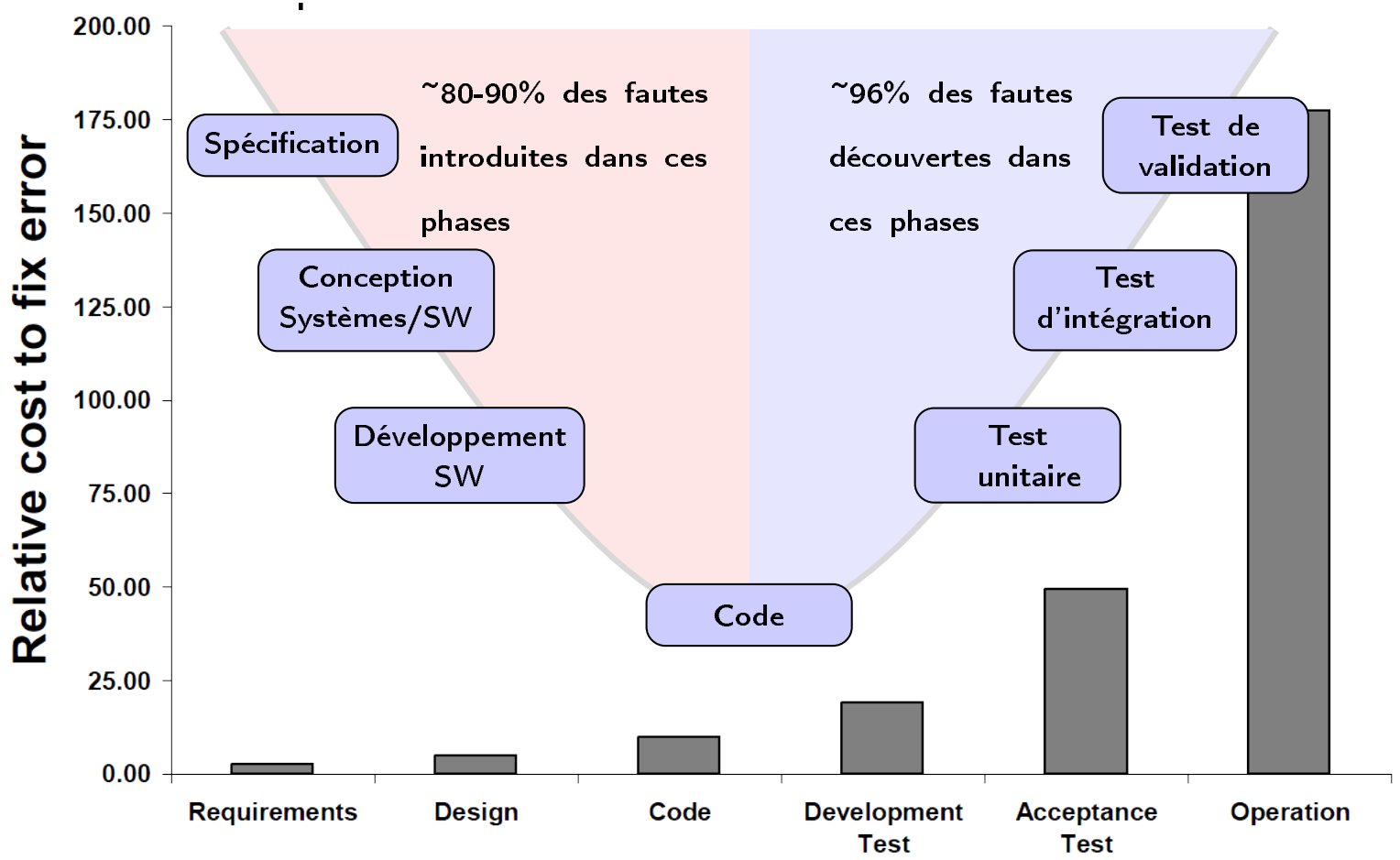
lsmc_sim -l ./libtest_bf_nl_discret_assemblage-noencoding.so -m 300000



Evolution of speed (left) and altitude (right)

Property 2: Automated Analysis framework for Simulink/Stateflow

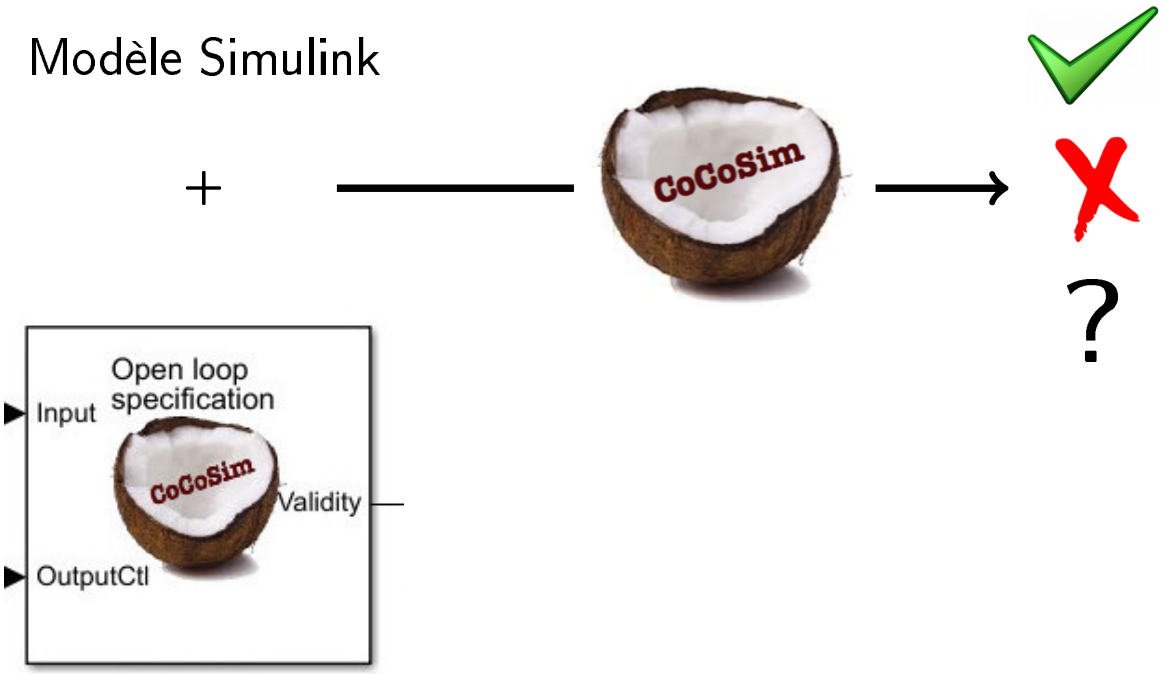
Motivation: CoCoSim: Automated Analysis and Compilation framework for Simulink/Stateflow



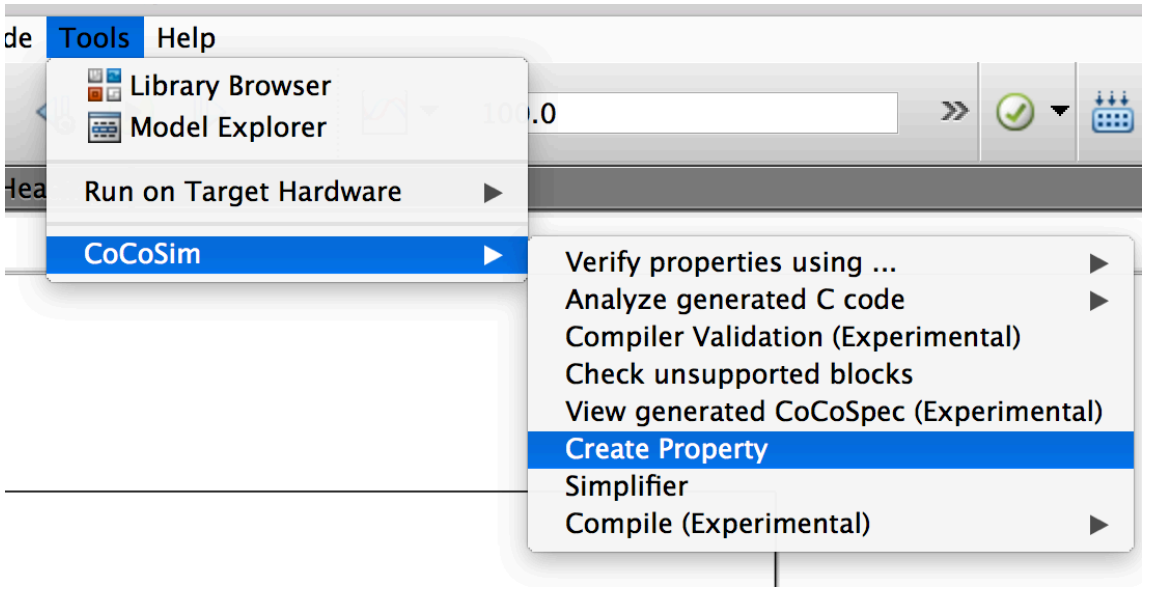
ONERA
THE FRENCH AEROSPACE LAB

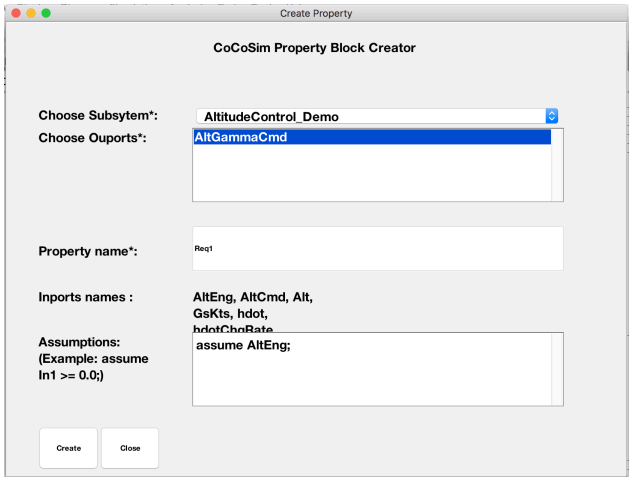
Safety properties analysis

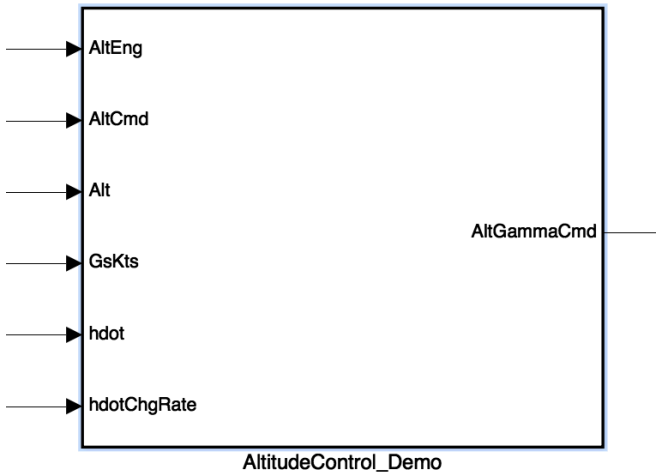
- Valid: model satisfies the specification
- Invalid: model does not satisfy the specification
- Counter-example: execution that violates the specification



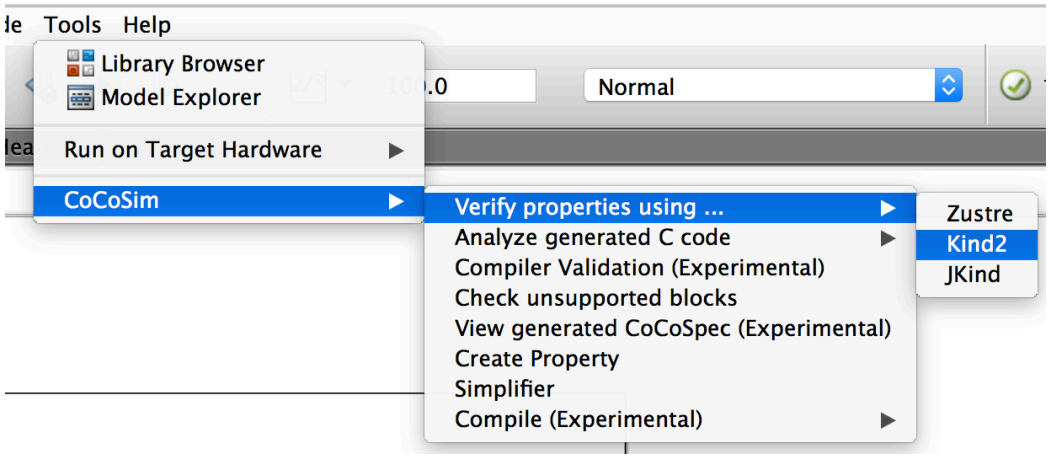
Functioning

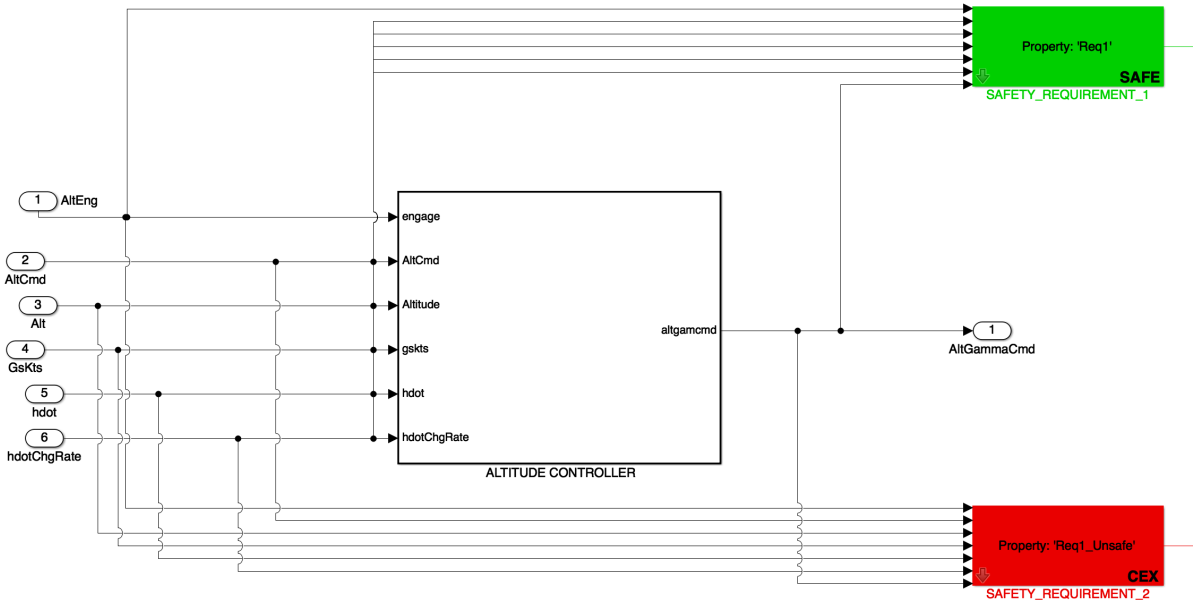




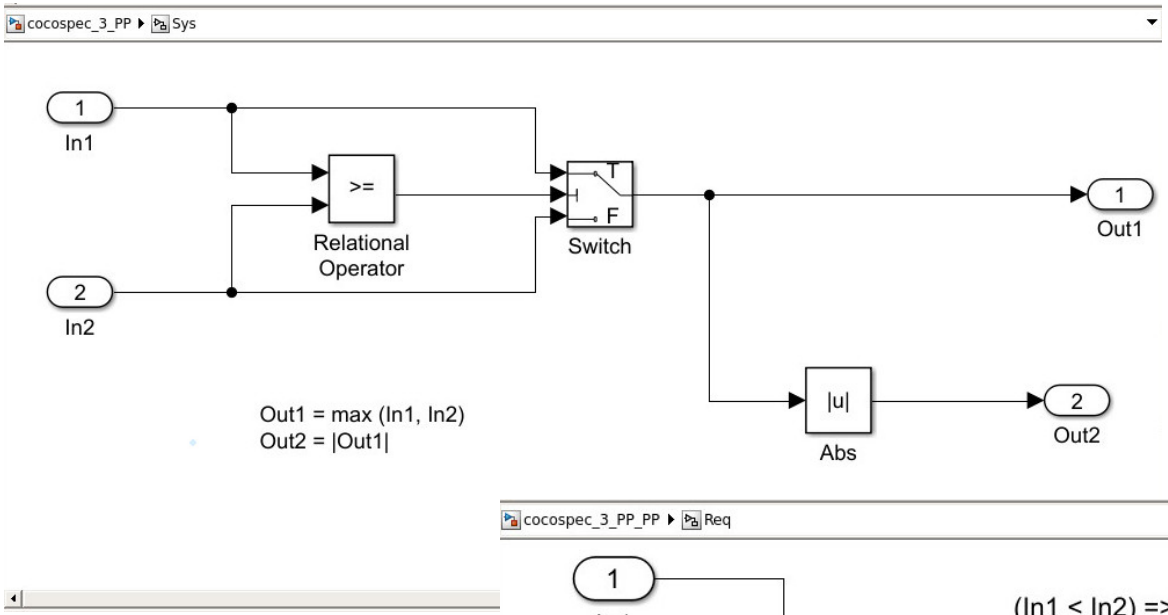


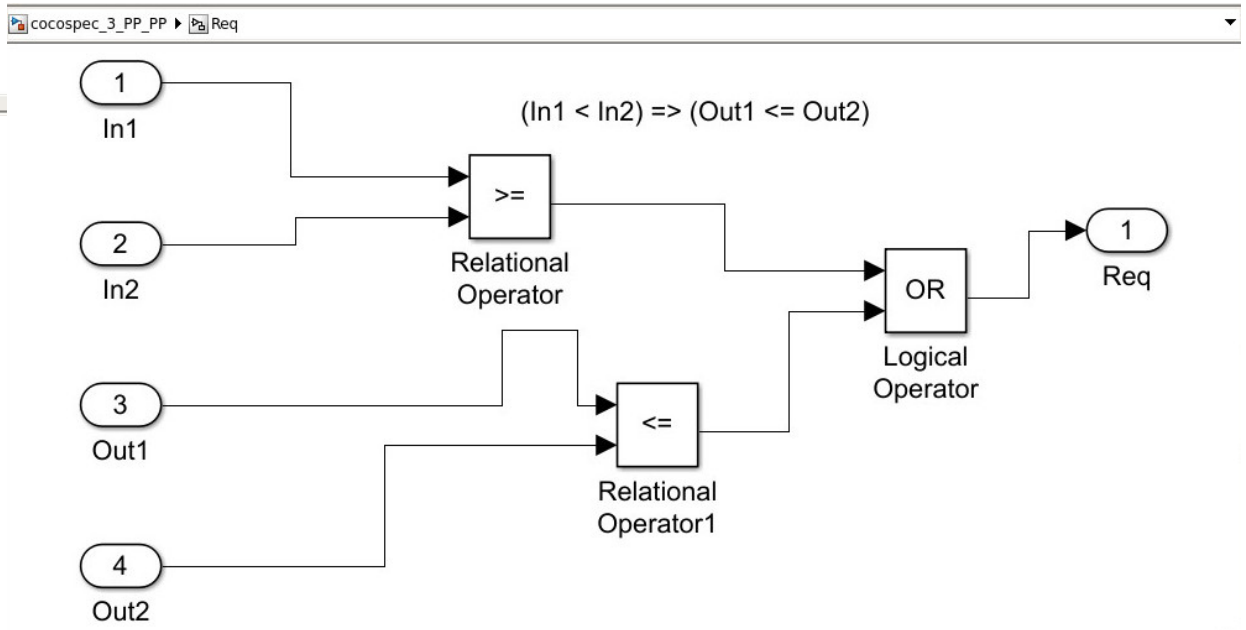
Functioning





Example





ONERA
THE FRENCH AEROSPACE LAB

Example

On SYS, Horn clause:

(In1 >= In2 => Switch = In1) and (In1 < In2 => Switch = In2) and
(Switch > 0 => abs = Switch) and (Switch < 0 => abs = -Switch) and
(Out2 = Abs) and (Out1 = Switch)

On SPEC, Horn clause:

Req = ((In1 > In2) or (Out1 <= Out2))

On global:

main and not Req => ERR

Use of Z3 to query ERR

Example (real generated clause for SYS)

```
(rule (=>
  (and (and (or (not (= (>= cocospec_3_PP_Sys.In1_1_1 cocospec_3_PP_Sys.In2_1_1)
          true))
             (= cocospec_3_PP_Sys.Switch_1_1 cocospec_3_PP_Sys.In1_1_1))
           (or (not (= (>= cocospec_3_PP_Sys.In1_1_1 cocospec_3_PP_Sys.In2_1_1) false))
             (= cocospec_3_PP_Sys.Switch_1_1 cocospec_3_PP_Sys.In2_1_1))
      )
      (and (or (not (= (>= cocospec_3_PP_Sys.Switch_1_1 0) true))
             (= cocospec_3_PP_Sys.Abs_1_1 cocospec_3_PP_Sys.Switch_1_1))
           (or (not (= (>= cocospec_3_PP_Sys.Switch_1_1 0) false))
             (= cocospec_3_PP_Sys.Abs_1_1 (- cocospec_3_PP_Sys.Switch_1_1)))
      )
      (= cocospec_3_PP_Sys.Out2_2_1 cocospec_3_PP_Sys.Abs_1_1)
      (= cocospec_3_PP_Sys.Out1_1_1 cocospec_3_PP_Sys.Switch_1_1)
      )
  (cocospec_3_PP_Sys_fun cocospec_3_PP_Sys.In1_1_1 cocospec_3_PP_Sys.In2_1_1
          cocospec_3_PP_Sys.Out1_1_1 cocospec_3_PP_Sys.Out2_2_1)
))
```

Outline

- Introduction
- Safe code generation
- Verification
- Conclusion

Conclusion and future work

- Consolidation of ONERA/NASA toolbox
- Execution on the iron bird and « avion jaune »

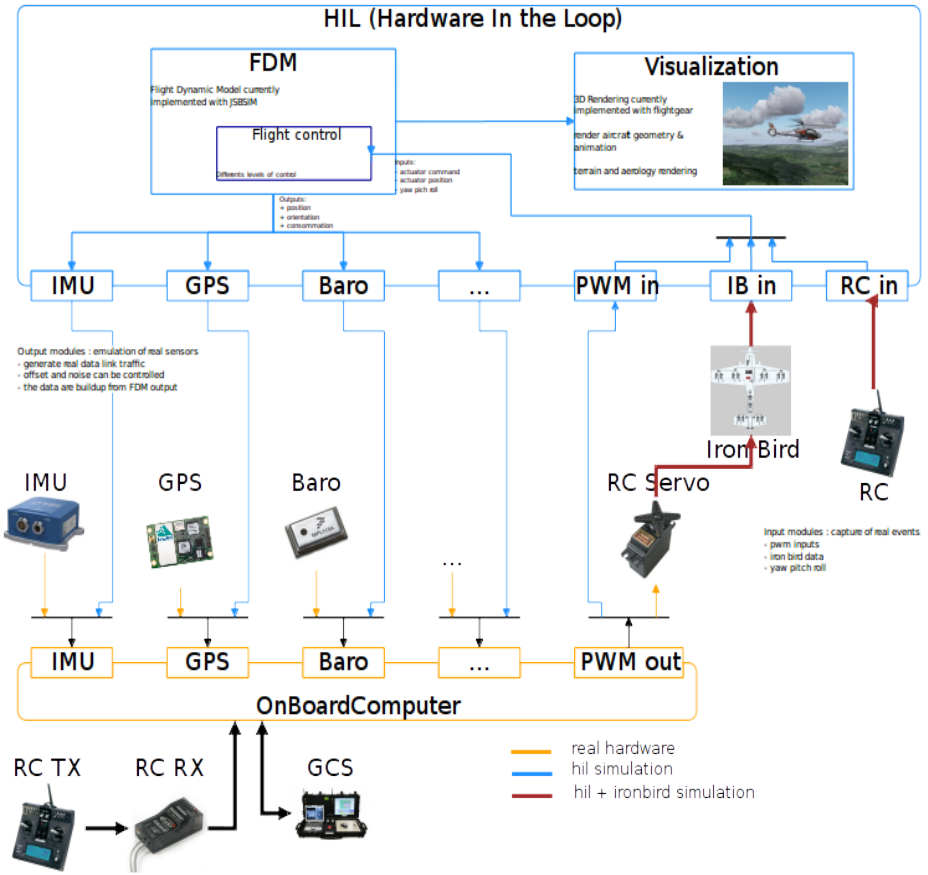
IronBird « avion jaune »

- Iron Bird : une maquette à l'échelle 3/8 du véritable Altimum (Société L'avion jaune)
  - Capteurs, actionneurs
  - calculateur réel
  - (sauf moteur)

- Autoriser le développement rapide sur matériel réel *et* des essais en vol virtuel

IronBird – HIL simulation

- Boucle fermée avec matériel réel (HIL)
- L'environnement (JSBSim) et le modèle avion (ONERA) sont **simulés** et **couplés à l'IronBird**
- La radio-commande, le calculateur bord et les servos moteur de l'IronBird sont **réels**
- **Logiciel de vol embarqué issu de la chaine de production ONERA pour FORCES3**



Conclusion

Thank you for your attention!