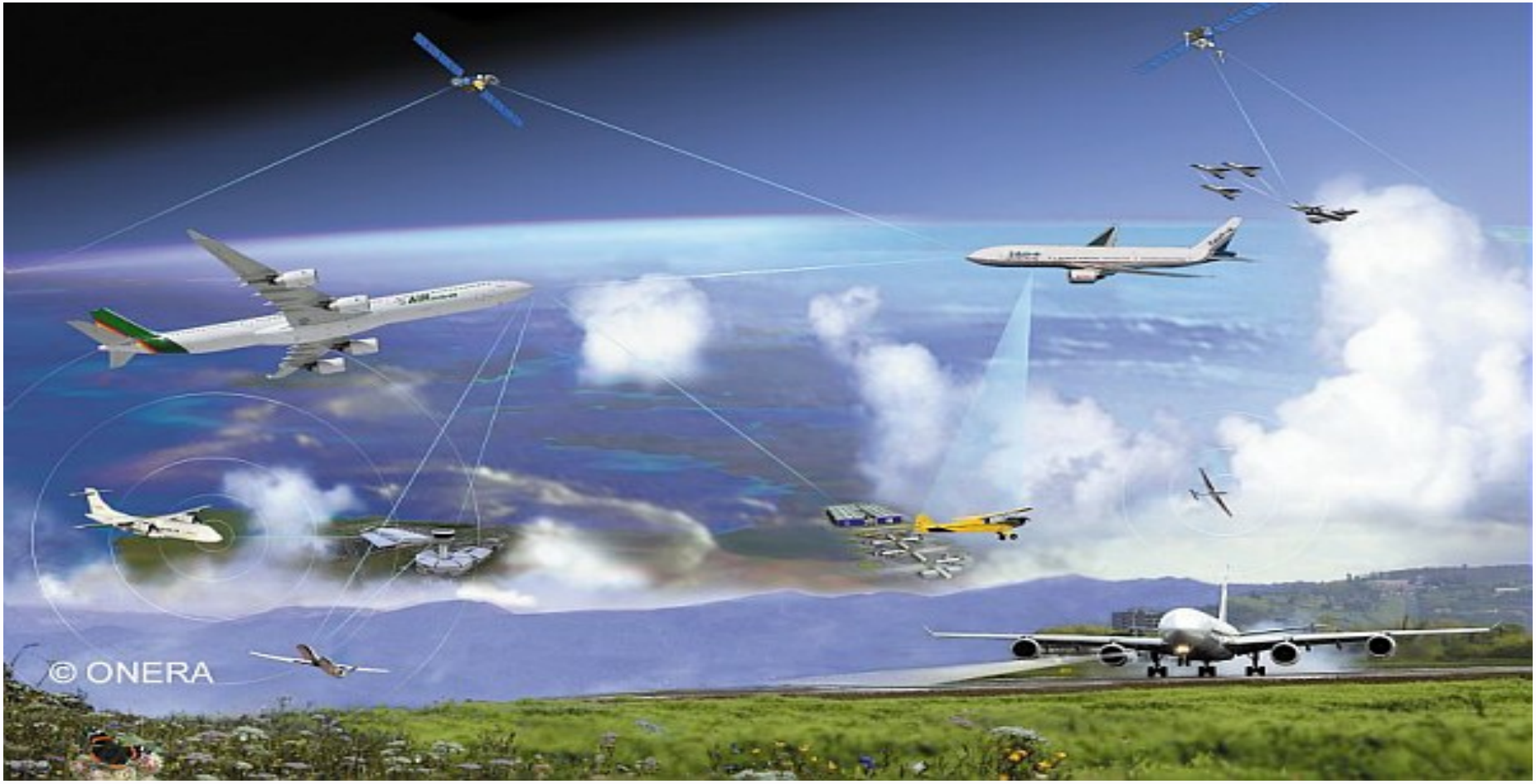


Verifying controlers: from system to software

the GEORGIA TECH-IRIT-NASA-NIA-ONERA team

06/28/2013

Current process

Differential Equations (plant)

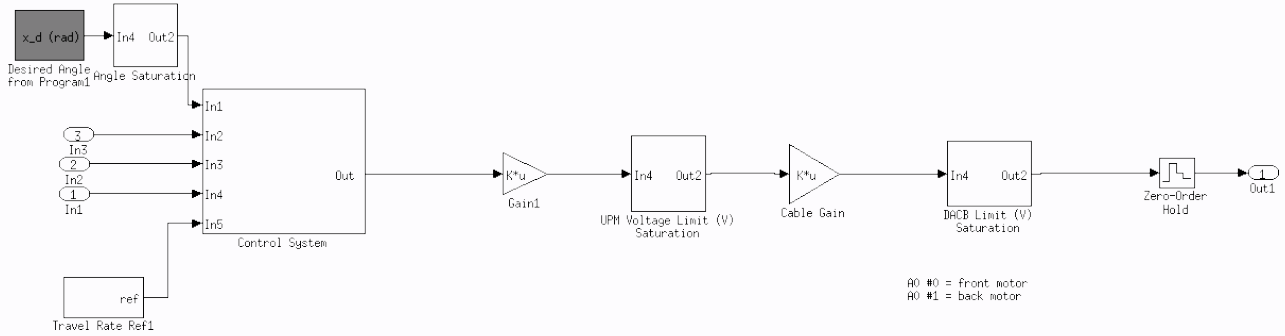
Control theorists

Current process

Differential Equations (plant)

↳ Continuous controller

Control theorists



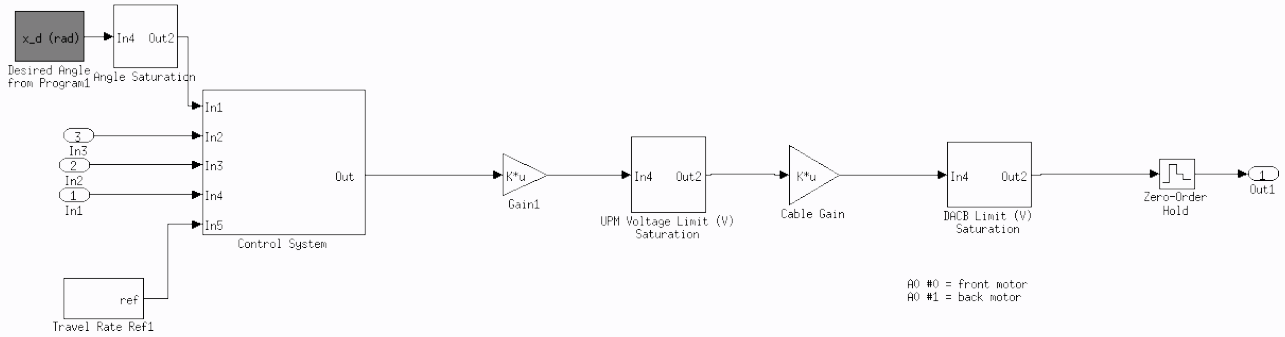
Current process

Differential Equations (plant)

→ Continuous controller

→ Discrete version

Control theorists



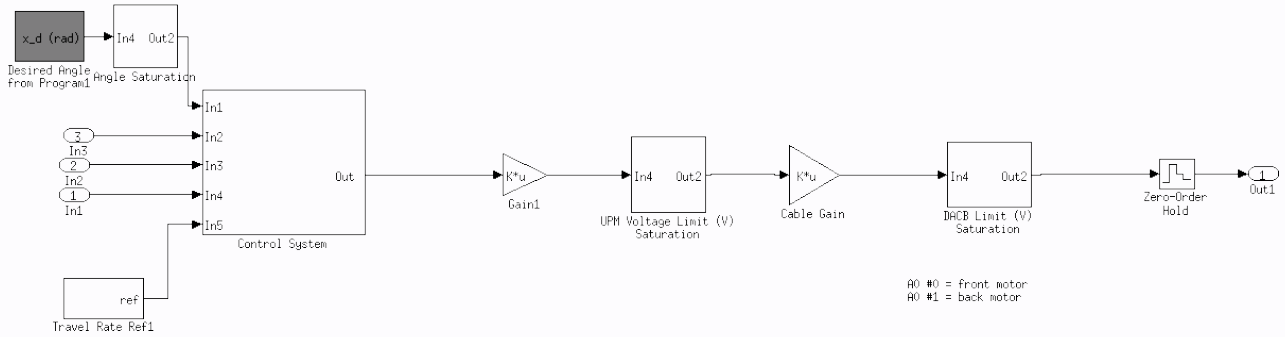
CURRENT PROCESS

Differential Equations (plant)

→ Continuous controller

→ Discrete version

Control theorists



- Control laws design:
  - usually simplification of the plant around specific points and controlers proposed for these

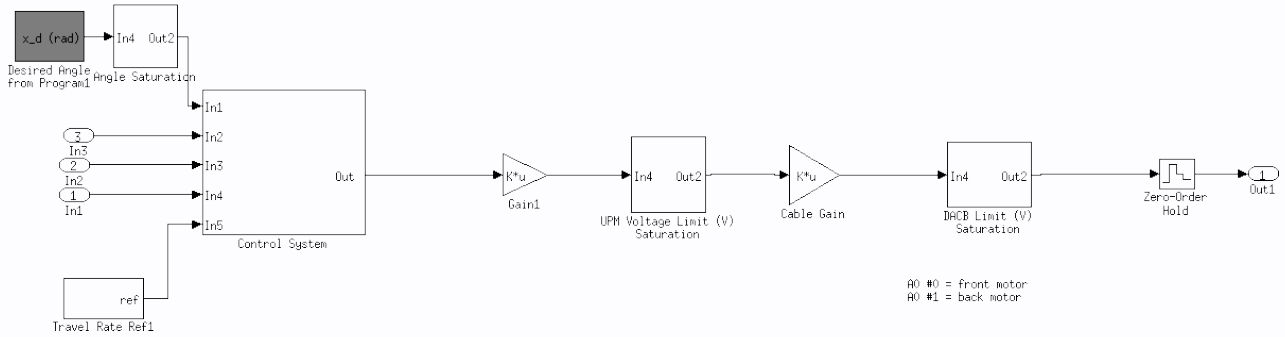
CURRENT PROCESS

Differential Equations (plant)

→ Continuous controller

→ Discrete version

Control theorists



- Control laws design:
  - usually simplification of the plant around specific points and controlers proposed for these
  - lots of arguments/evidences on those simple cases

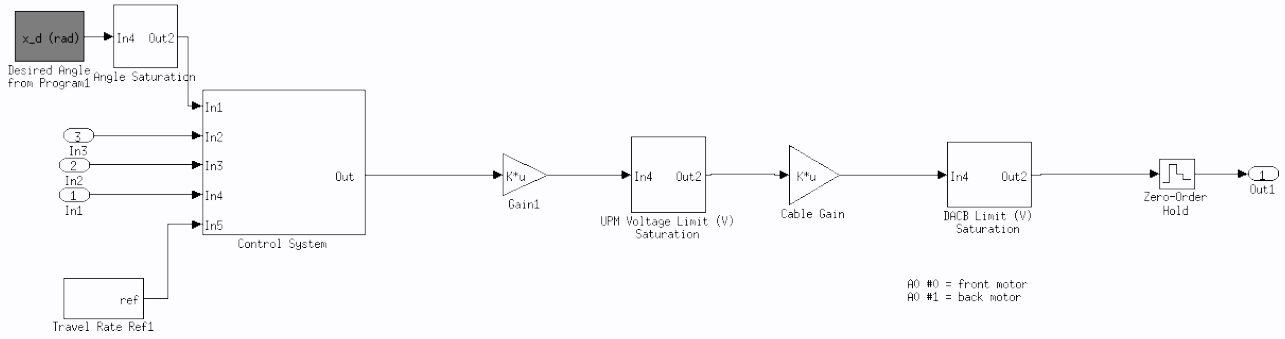
CURRENT PROCESS

Differential Equations (plant)

→ Continuous controller

→ Discrete version

Control theorists



- Control laws design:
  - usually simplification of the plant around specific points and controlers proposed for these
  - lots of arguments/evidences on those simple cases
  - are these good controlers individualy? when composed?

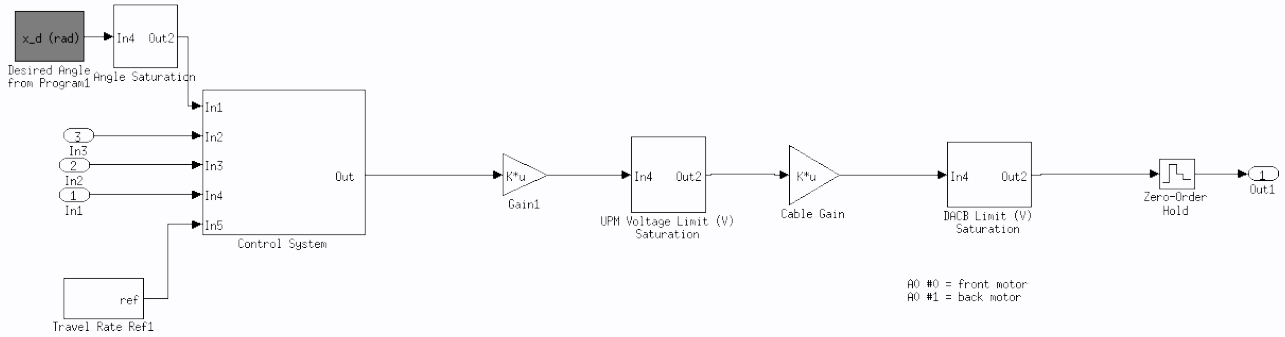
CURRENT PROCESS

Differential Equations (plant)

→ Continuous controller

→ Discrete version

Control theorists



- Control laws design:
  - usually simplification of the plant around specific points and controlers proposed for these
  - lots of arguments/evidences on those simple cases
  - are these good controlers individualy? when composed?
  - which property? stability, robustness, performances (need the plant!)

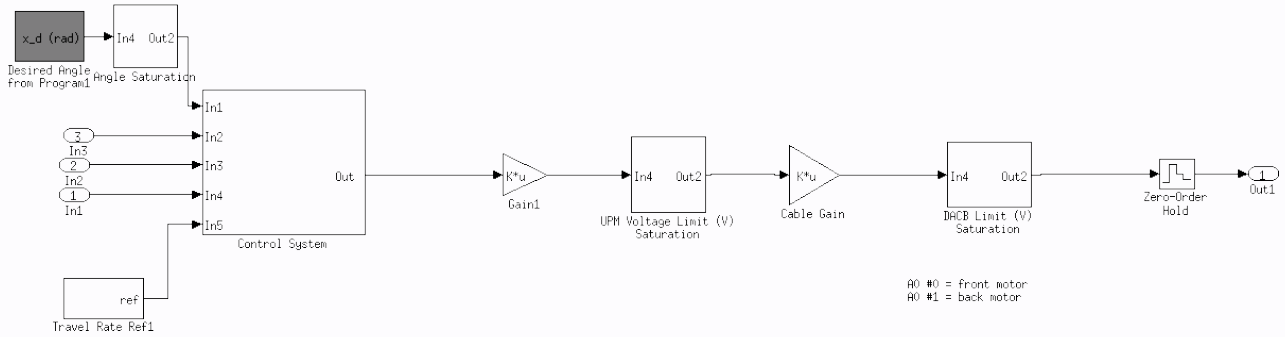
Current process

Differential Equations (plant)

→ Continuous controller

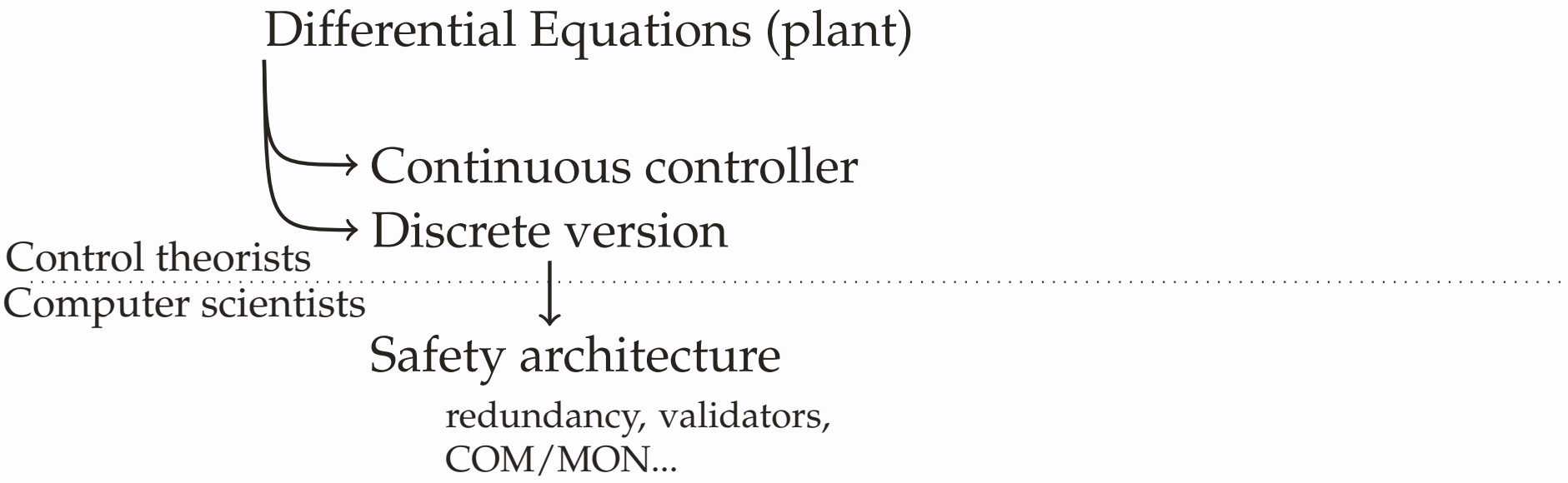
→ Discrete version

Control theorists

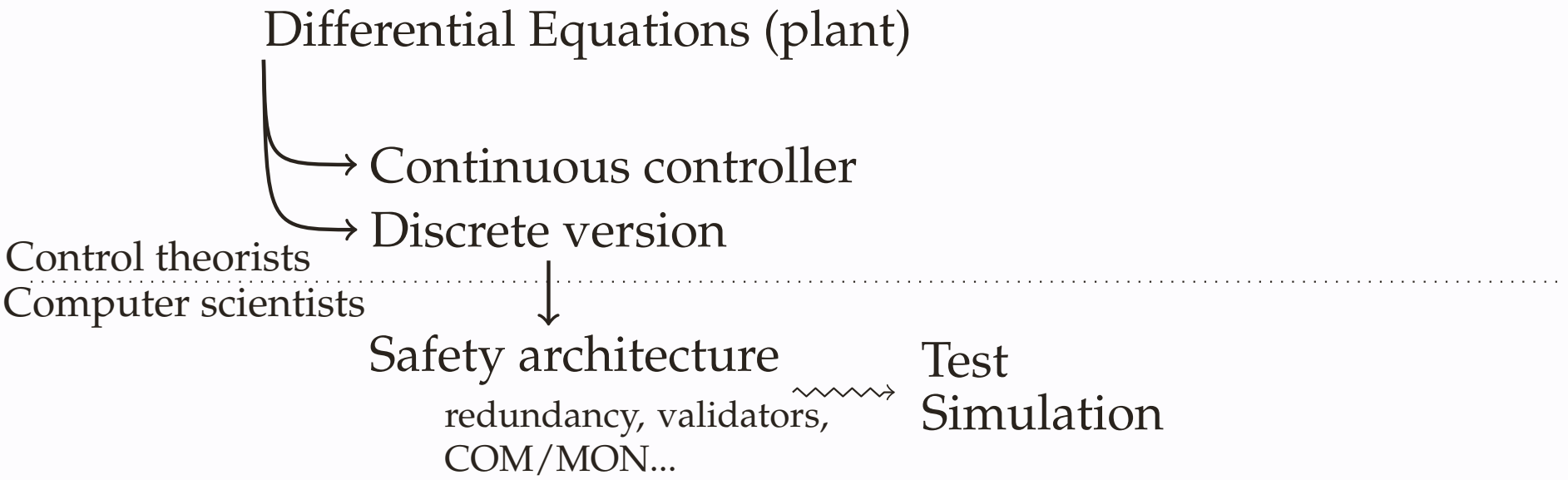


- Control laws design:
  - usually simplification of the plant around specific points and controlers proposed for these
  - lots of arguments/evidences on those simple cases
  - are these good controlers individualy? when composed?
  - which property? stability, robustness, performances (need the plant!)
  - frequency domain proof argument vs state space domain (ie. Lyapunov functions)

Current process



CURRENT PROCESS



- Fault tolerance: set of constructs to recover from system/hardware failures
  - is this architecture sound (ie. when there is less than n simultaneaous error, the output is still valid or there will still be a working controler)

Current process

Differential Equations (plant)

→ Continuous controller

→ Discrete version

Control theorists

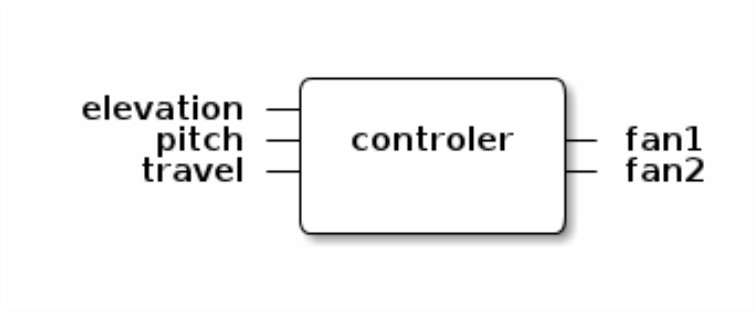
Computer scientists

↓

Safety architecture

redundancy, validators, COM/MON...

⇝ Test Simulation



Current process

Differential Equations (plant)

→ Continuous controller

→ Discrete version

Control theorists

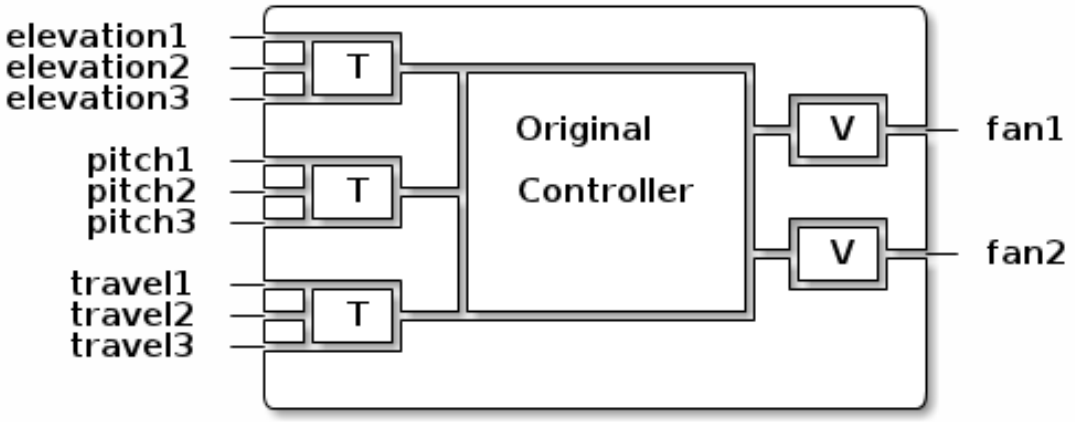
Computer scientists

↓

Safety architecture

redundancy, validators, COM/MON...

⇝ Test Simulation



Current process

Differential Equations (plant)

→ Continuous controller

→ Discrete version

Control theorists

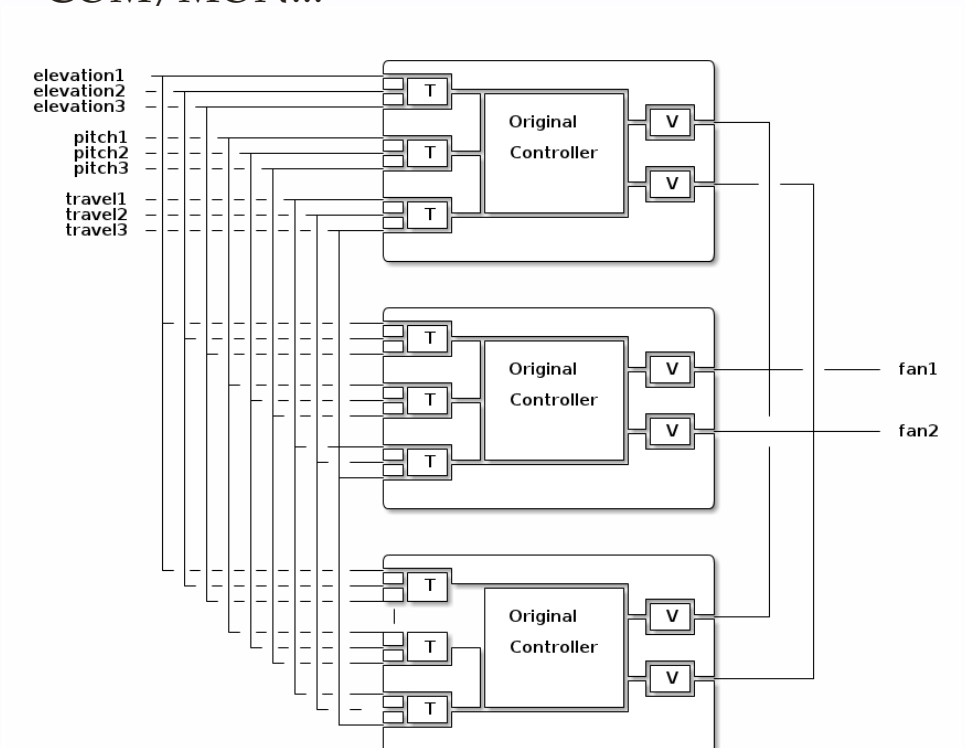
Computer scientists

↓

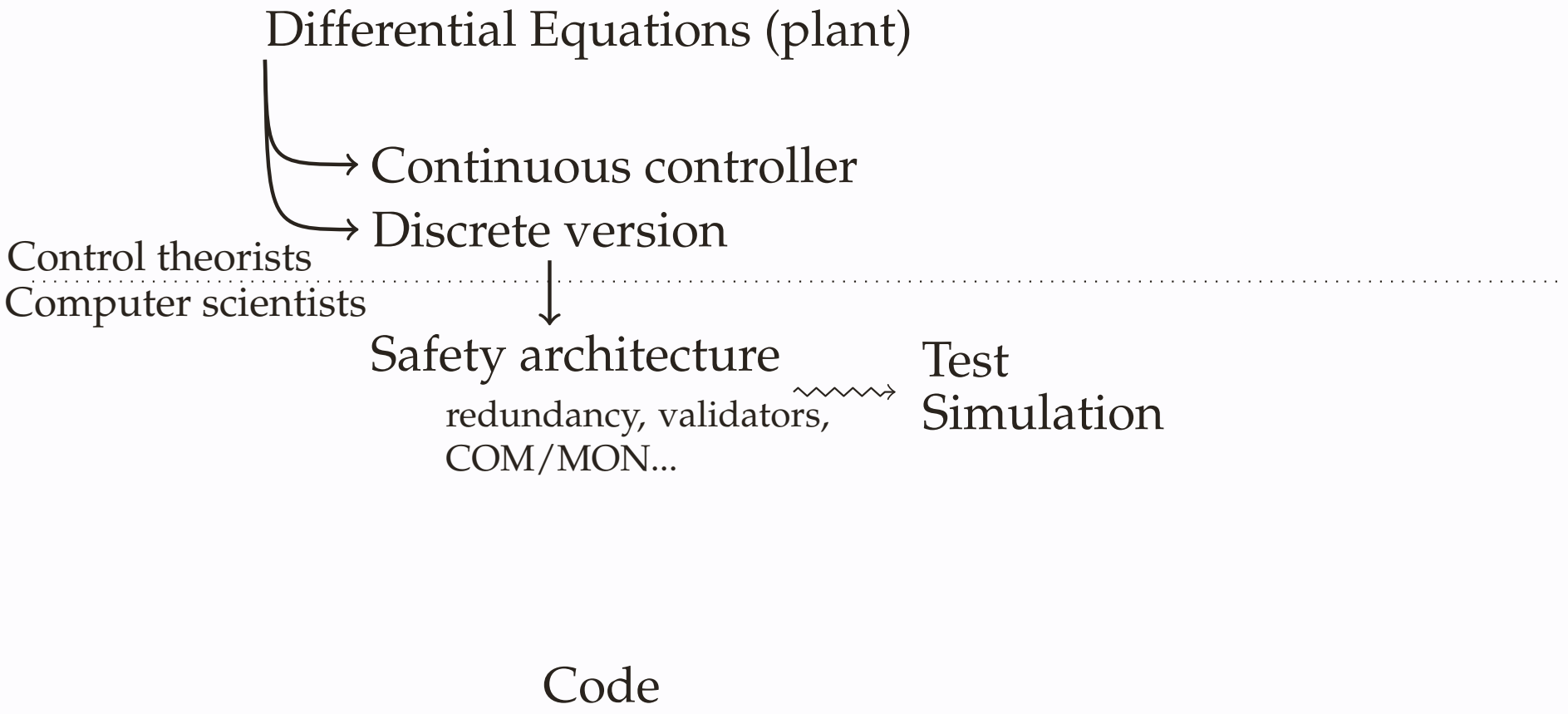
Safety architecture

redundancy, validators, COM/MON...

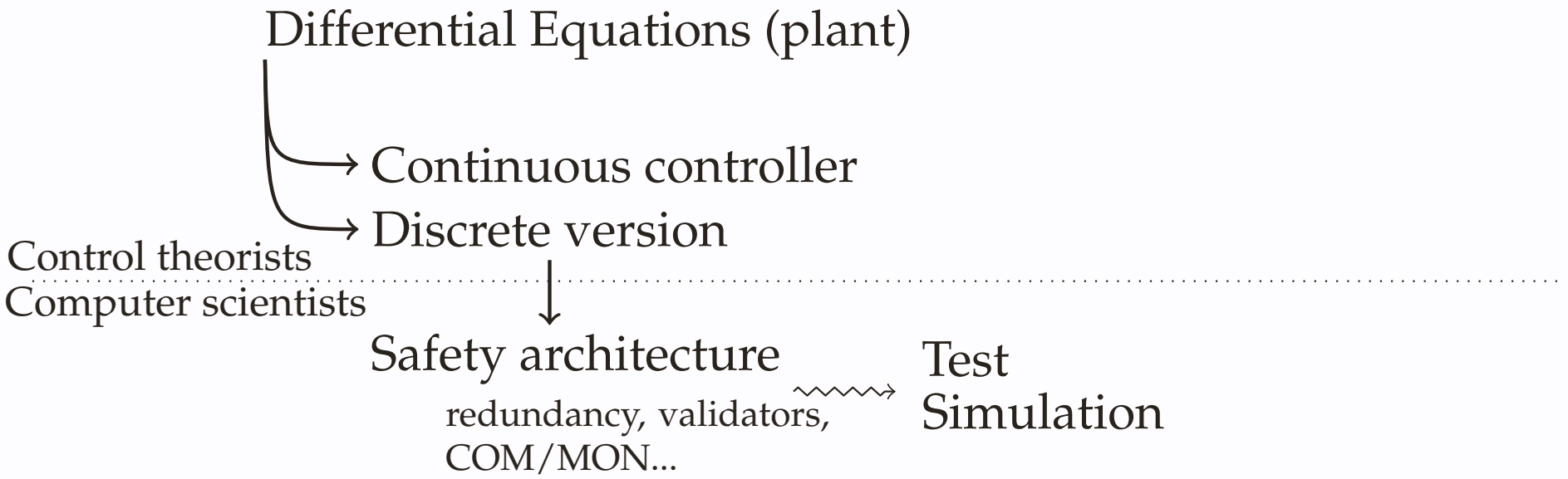
⇝ Test Simulation



Current process



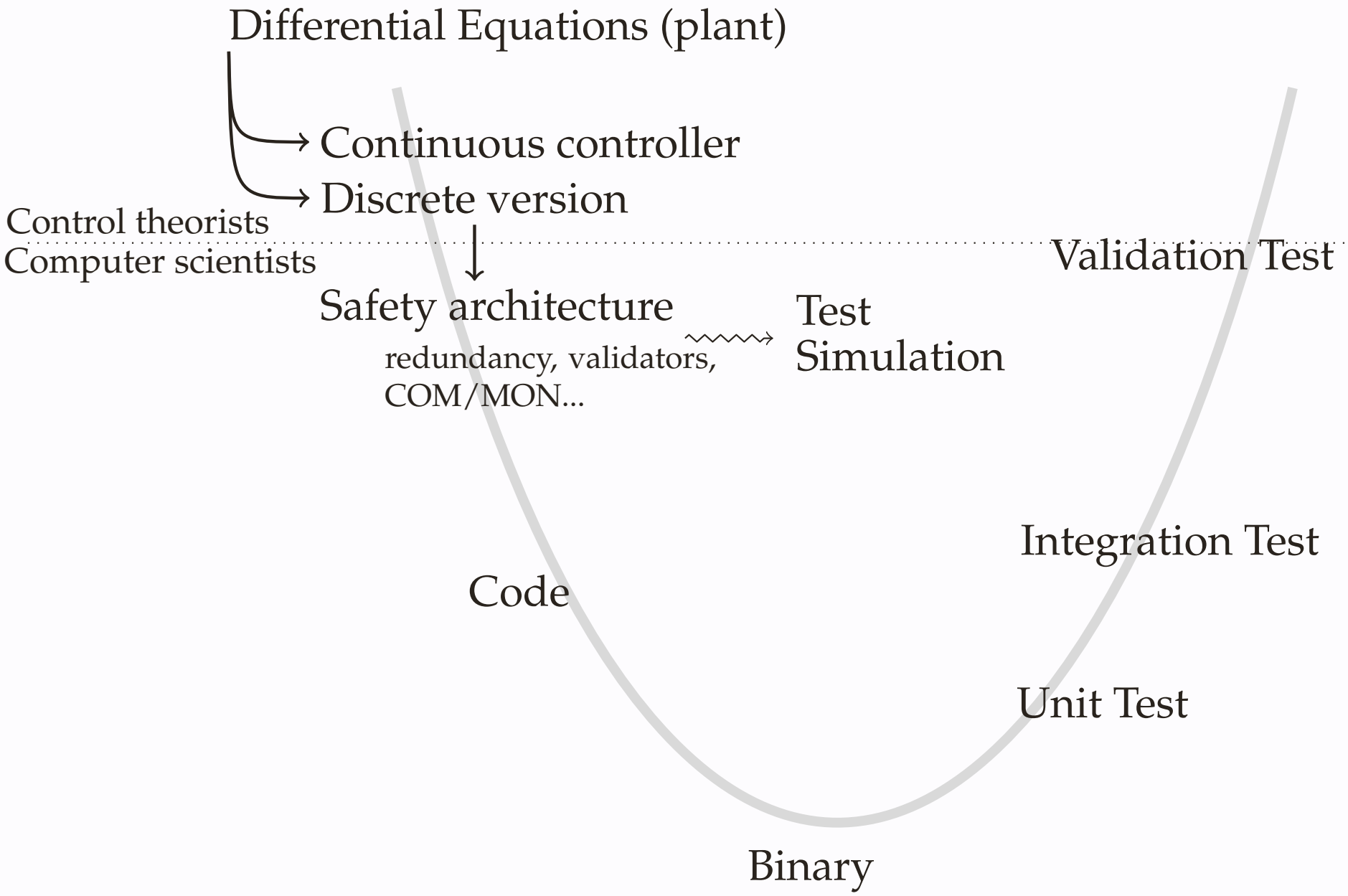
CURRENT PROCESS



Code

- Actual implementation:
  - floats not reals
  - pointers, arrays, memory access $\rightarrow$ potential failure
  - real world: overflows

Current process



Key elements of our approach

trust the system/software, not the process
BUT integrate V&V within the process through

Key 1

Autocoding (modular compilation)

Key 2

Formal specification

Key 3

Formal verification

Our proposal

- initial model in Simulink
- intermediate model in Lustre
  - compile simulink to lustre
  - express the control level properties and the fault-tolerance ones as synchronous observers
  - use tools to very such properties
- C code
  - compile lustre to C
  - compile synchronous observers as code annotations
  - use tools to very such properties and the absence of RTE

Ongoing work

- developing the complete compilation tool chain from Simulink to C
- building realistic examples: UAV controlers, FADEC, etc.
- extend our tool to prove the mentionned properties
  - automatic synthesis of Lyapunov function
  - proving stability at C code level
  - proving soundness of floating point implementation (wrt to algorithms with Reals)
- everything open-source $\rightarrow$ starting a repository to share examples
  - Control System Analysis Library - https://cavale.enseeiht.fr/redmine/projects/c-sal
  - bring challenges to the community
  - compare tools results on airspace software
  - should eventually provide the compilation toolchain
  - just started!

$\Longrightarrow$ feel free to contribute