

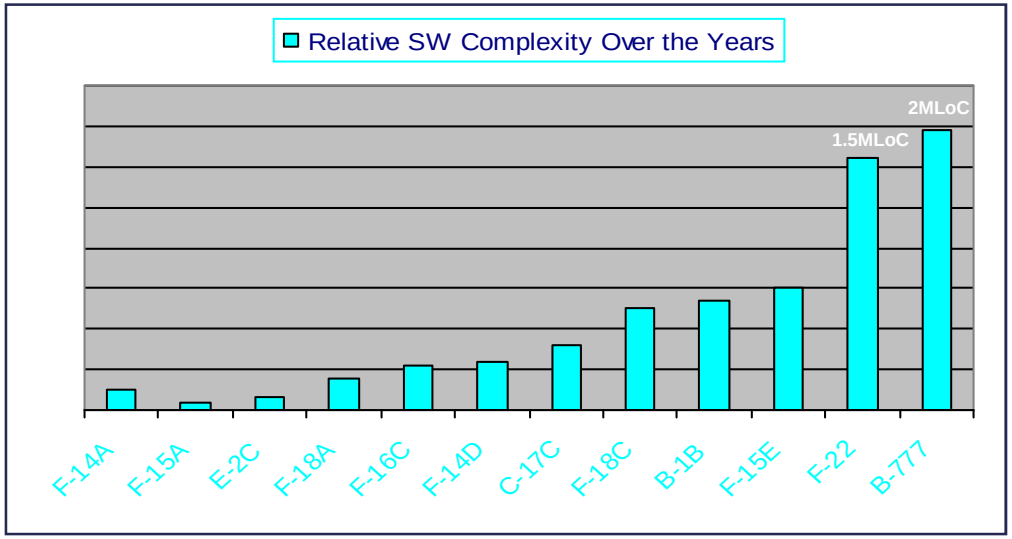


GEA Tianjin / 中国民航大学中欧航空工程师学院

INDUSTRIAL FEEDBACK
SCADE FOR EMBEDDED SYSTEMS

Specifics Avionics Market Trends

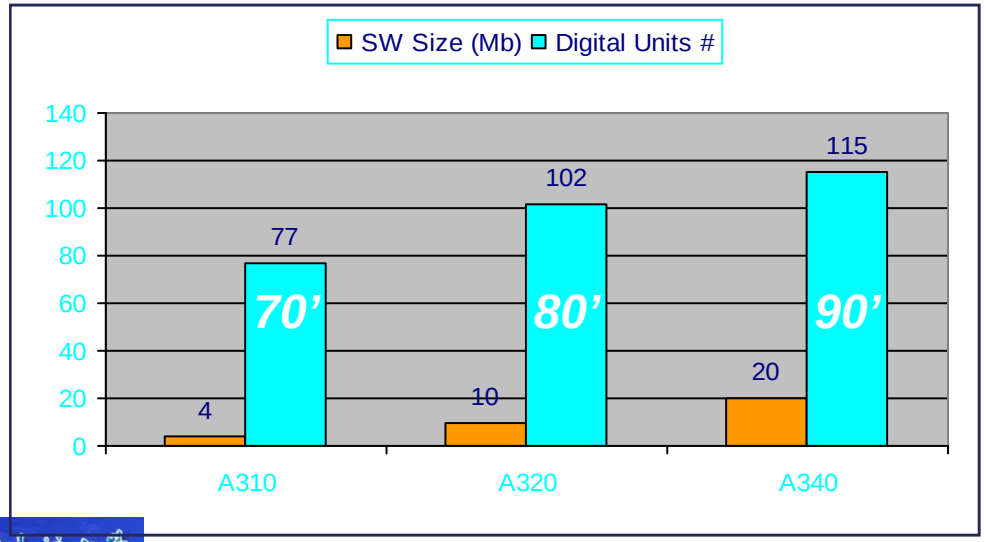
Boeing



Fact: Increase in Avionics Software Complexity …and So Do the Certification Costs!

(Up to 50% of Development Costs)

Airbus



GEA Tianjin / 中国民航大学中欧航空工程师学院

System & Software Workflow

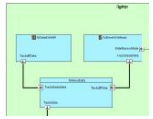
System Design

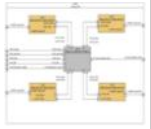
System Requirements

System Functions

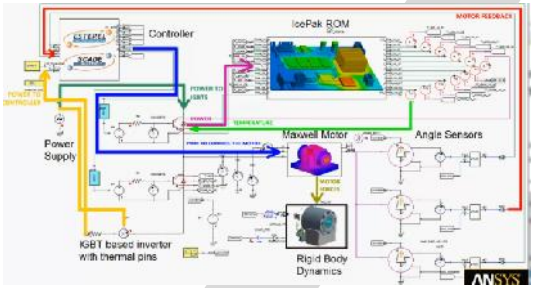
SCADE SYSTEM

System Architecture





System Validation



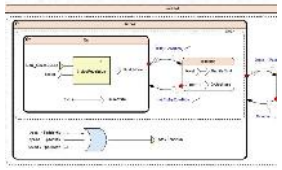
Software Development

Software Requirements

SW Requirements Validation

Software Design

SW Design Verification



SCADE SUITE

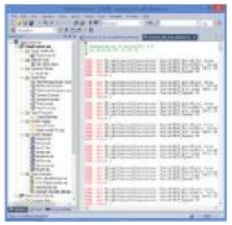
Executable Code

Target Testing

Software Validation

Requirements Validation

Requirements-based Test Creation



SCADE TEST

Test Execution on Host

Model & Code Coverage Analysis

Test Execution on Target

THALES AIRBUS EADS eurocopter an EADS Company SAFRAN

GEA Tianjin / 中国民航大学中欧航空工程师学院

SCADE in Aerospace & Defense Applications

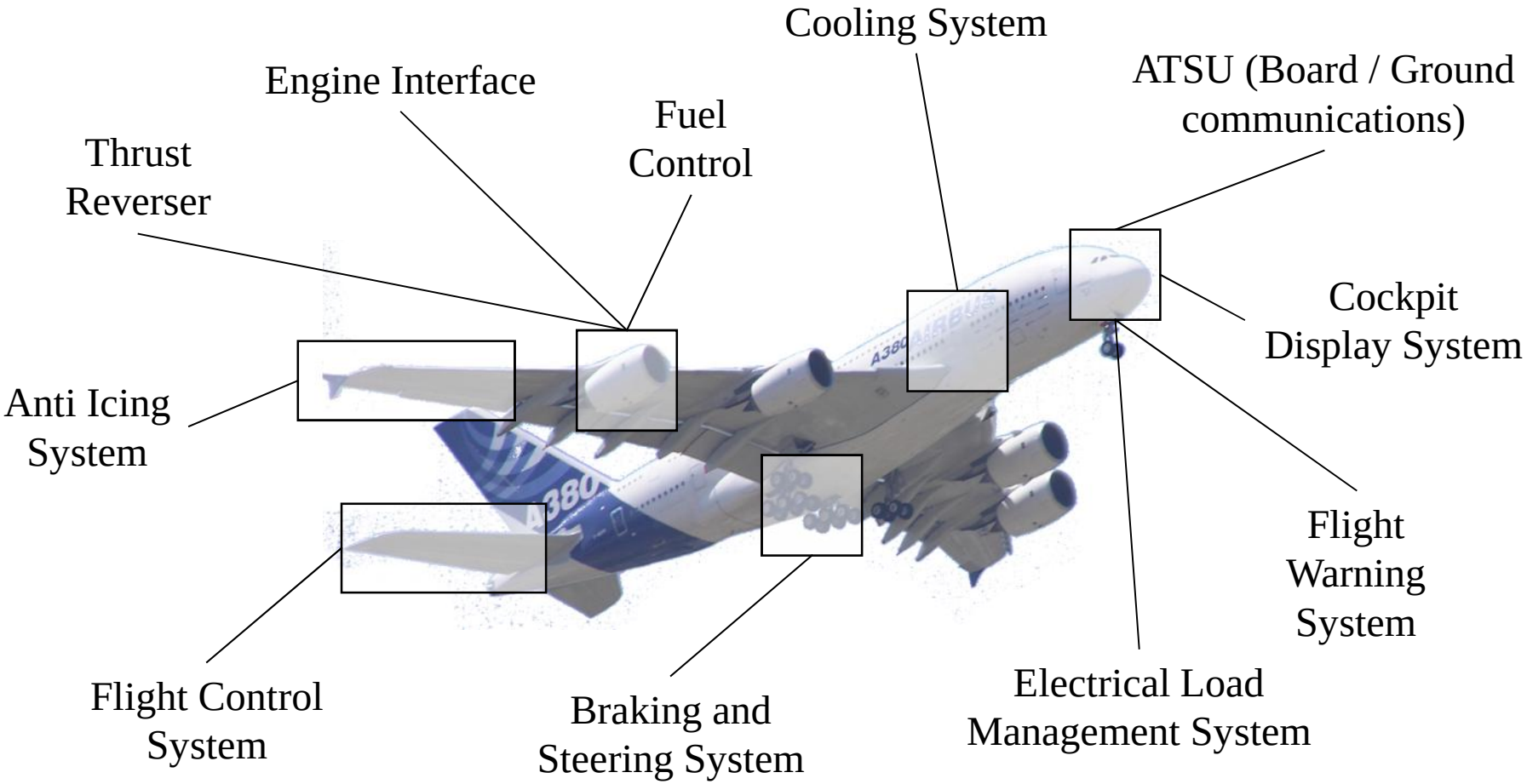
- Flight control systems
- Power management
- Reconfiguration management
- Autopilots
- Engine control systems Braking systems
- Cockpit display and alarm management
- Fuel management

(DO-178B EASA & FAA Qualified – up to level A)

THALES AIRBUS EADS eurocopter an EADS Company SAFRAN

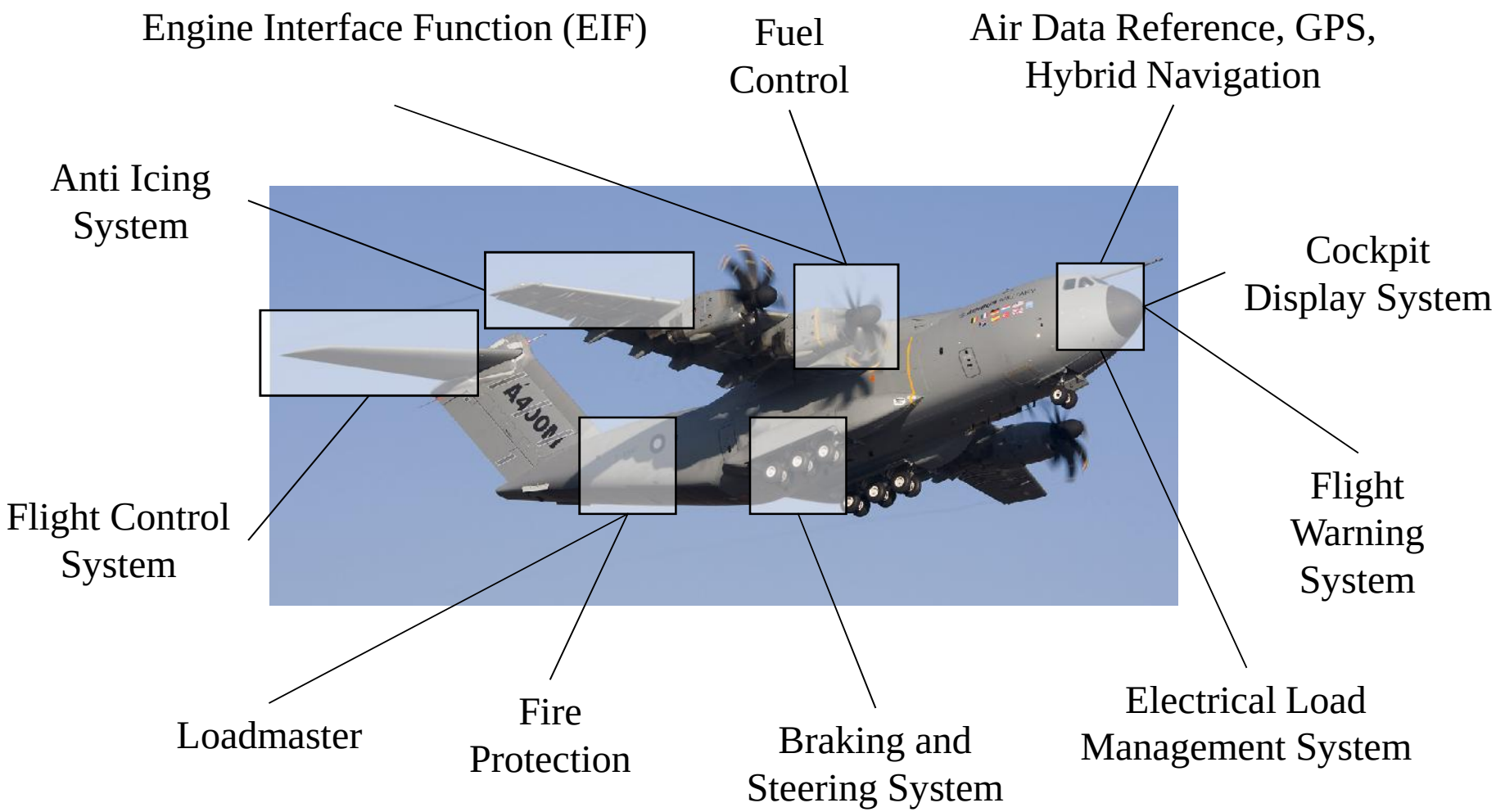
SCADE in the Airbus A380

8 Million Lines of Code Generated !



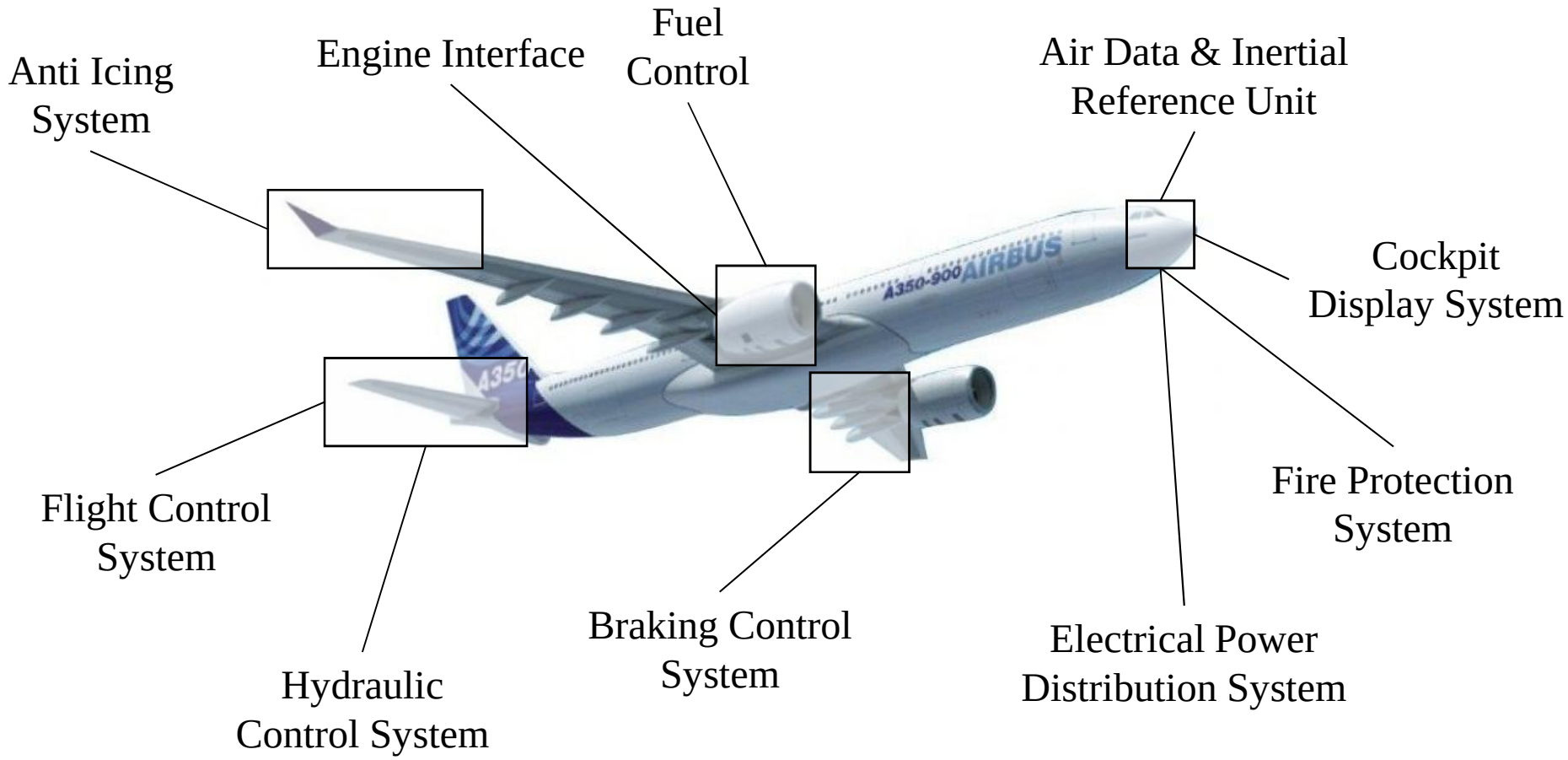
ANSYS

SCADE in the Airbus A400M



ANSYS

SCADE in the Airbus A350



SCADE @ CMC Esterline

- Program/Application:
  - Integrated Avionics Platform 7000

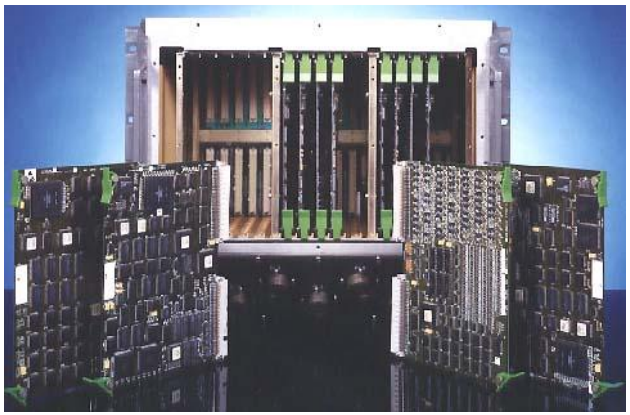
- Key Results:
  - Certification time and costs reduction
  - Time to market speed-up
  - Legacy flight display code migration onto a more modern technology platform, with SCADE Display

"One of the key aspects of our work with ANSYS is the ability to validate the design early in a project, which increases the level of confidence in the product and allows us to obtain early buy-in on key features or tradeoffs."

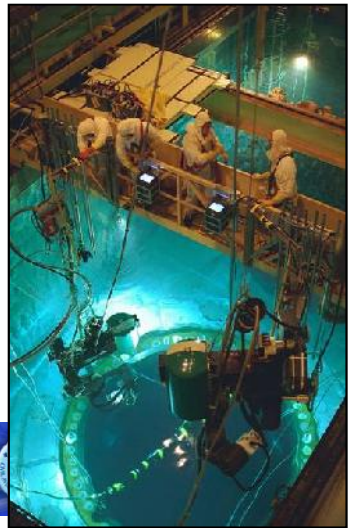
Marc Bouliane
Product Director, Integrated Avionics Platform 7000,
CMC Esterline

Other domains Applications

- Rail Automation
  - Train Control & Protection Systems
  - Systems Control



- Nuclear
  - Reactor Protection Systems
  - Nuclear Instrumentation Systems



SCADE Strategic Benefits in R&D

- Communication & Collaboration among system and software stakeholders
  - Model-based design is a powerful pre-sales and collaboration platform
- Product Line Development
  - Generation of application variants from the same models
  - Portability of generated code across hardware and RTOS
- Compliance with Software Safety Certification
  - Reduces risk, time and cost of DO-178 B/C Certification
  - Compliance with ARINC 653/IMA and FACE standards
  - Compliance with ARINC 661 architecture and standards, with model-based benefits

SCADE Technical Benefits in R&D

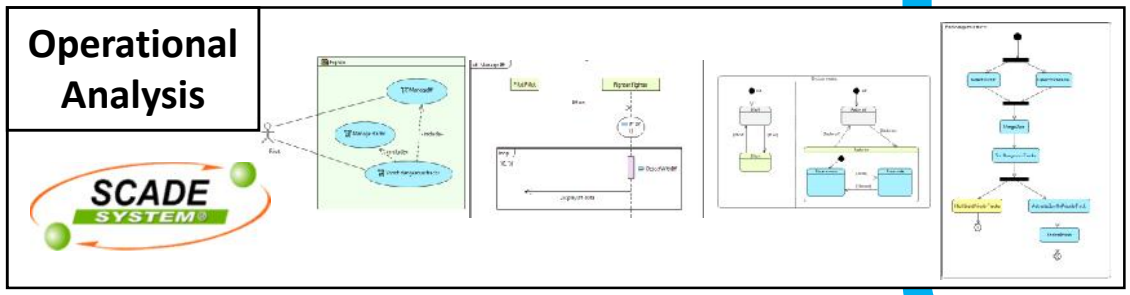
- Automated Production of readable, portable, high performance and high quality Code
  - Automatic and Certified Code Generators
  - Hardware, RTOS and platform independence
- Documentation Quality and Accuracy
  - Automatically produced and synchronized from models
- Early Detection of Design Flaws
  - Executable specifications increase understanding
  - Automatic model verification and simulation
- Long-term Maintainability of applications
  - Model maintenance simplified vs. manual code maintenance

SCADE Economical Benefits in R&D

- 50% Development and V&V Costs Reduction overall
  - Automatic production of 80% to 90% of the application software and related documentation
  - Suppression of code reviews
  - 80% low level testing costs reduction due to certification of automatic code generators

GEA Tianjin / 中国民航大学中欧航空工

Workflow with Scade

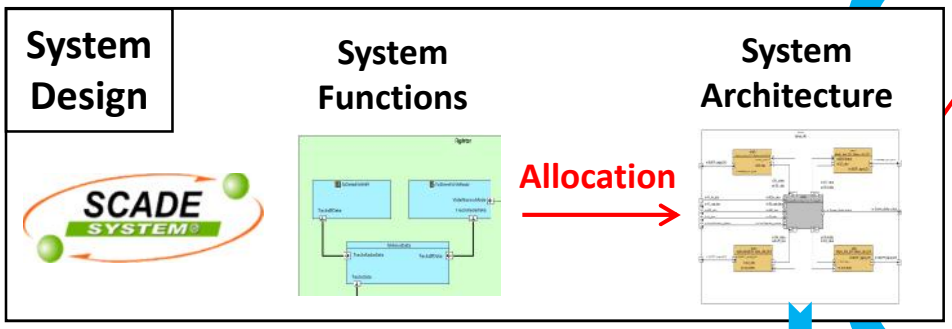


User Requirements

Traceability

Supports

System Requirements



Traceability

Supports

Software Requirements

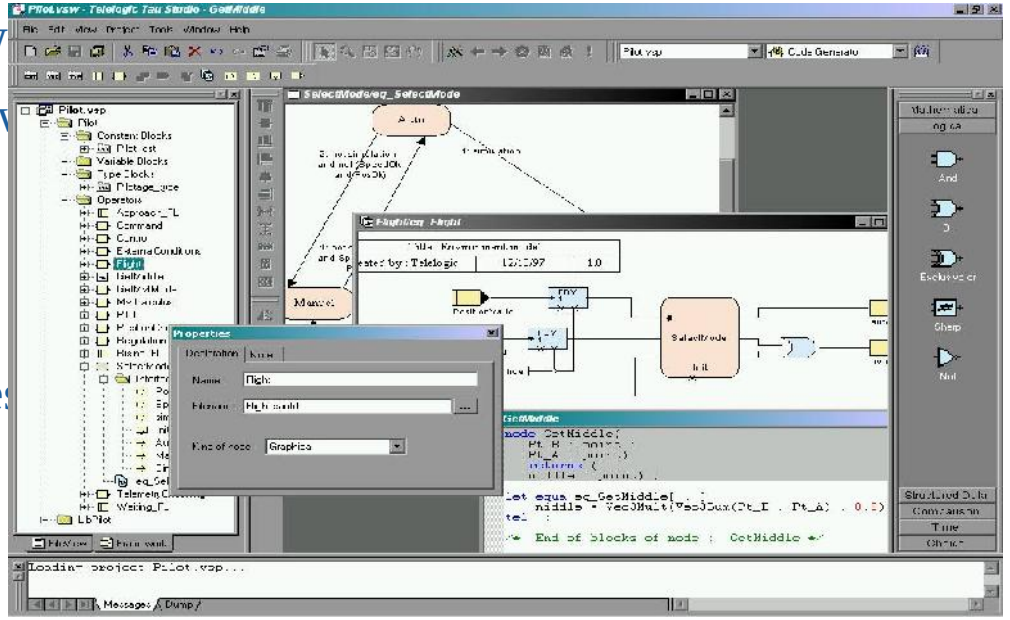
Synchronization
(detailed interfaces)

Traceability

THALES AIRBU

SCADE Tool Set

- Graphical editor
  - Available for UNIX and W
  - State Machine and data flo
  - Semantic check
  - Documentation generator
- Simulation
  - Interactive and batch mode
- Validation
  - Exhaustive proof (Prover)
- Code Generation
  - ADA and C
  - Qualified C for Do178A



Safety **C**ritical **A**pplications **D**evelopment **E**nvironment

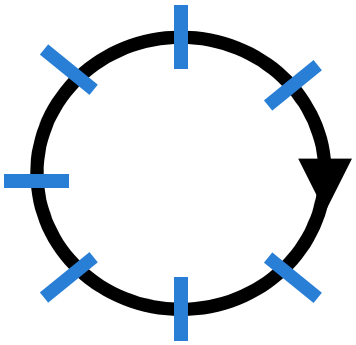
The SCADE language

- Modeling reactive systems
  - Answering their environment infinitely often

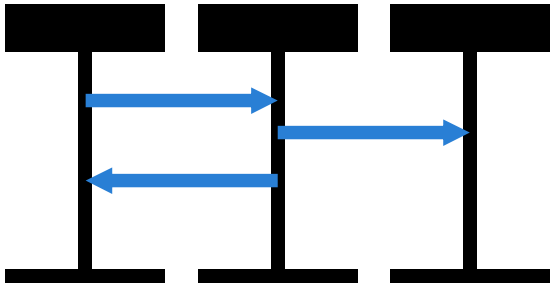
- Synchronous Data flow
  - Equations modellng the transformation of data
  - Cyclic computation

→best suited for the design of real-time controllers

SCADE positionnning



Time Driven
“Hard Real Time System”
SCADE



Event Driven
“I React on Events”
SDL, UML

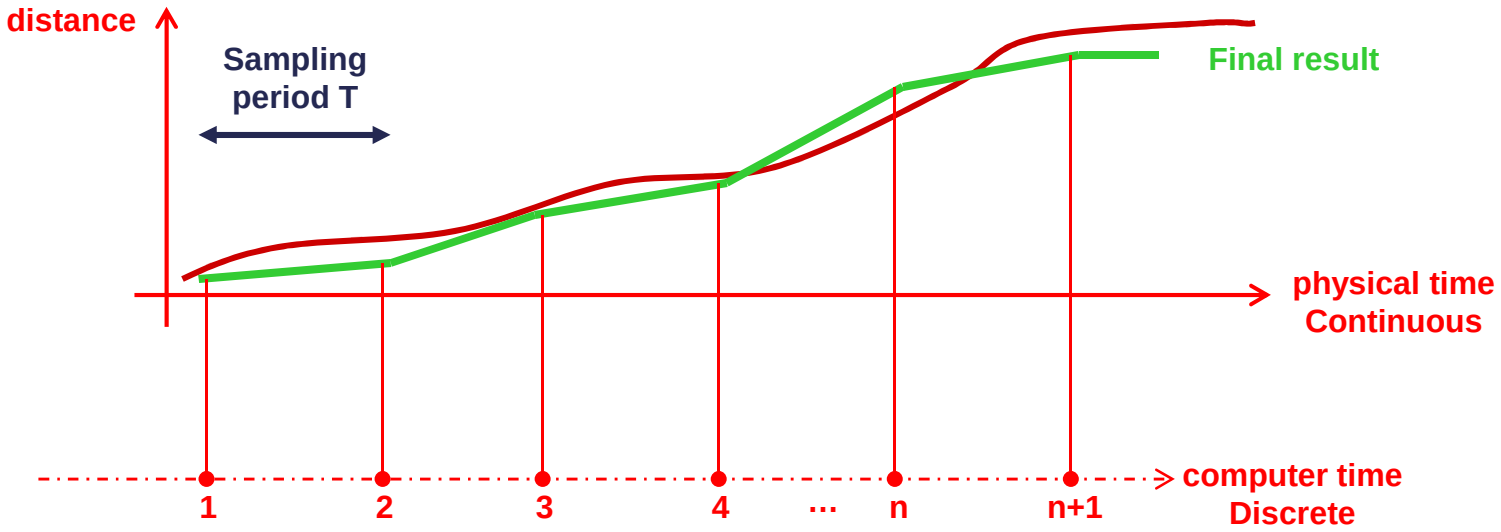
Language properties

- Deterministic
- Declarative and Structured (data and functions)
- Formal language
- Multi representational levels
- Security of execution
  - No loop, no dynamic allocation.
  - Time of computation maximum can be calculated.
- Based on the formal language LUSTRE
  - Synchronous dataflow approach
  - Syntax and semantic strictly defined (controls, formal proof)

Synchronous systems 1/2

- From continuous time to discrete time : sampling

  Example : speed computing System



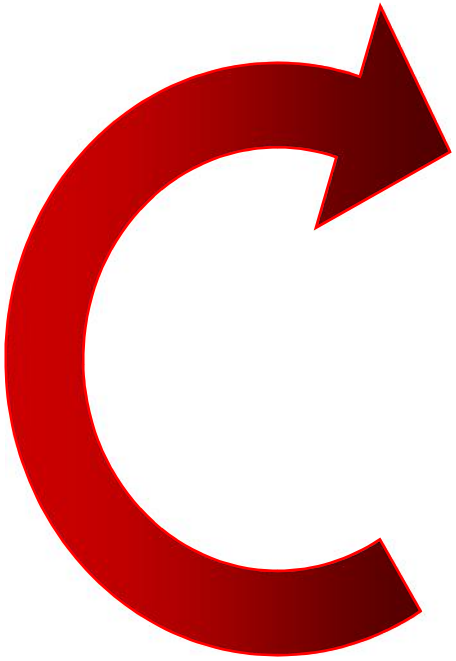
Real speed : d(distance)/dt

SCADE helps to define this

Computed speed : [distance(i) – distance (i-1)]/T

Synchronous systems 2/2

- The program examines its inputs and provides the outputs in a closed-loop process:



```
Loop forever
        Wait for clock
        …
        Read sensors
        Call SCADE main function
                Compute outputs
                Update states
        Write actuators
End loop
```

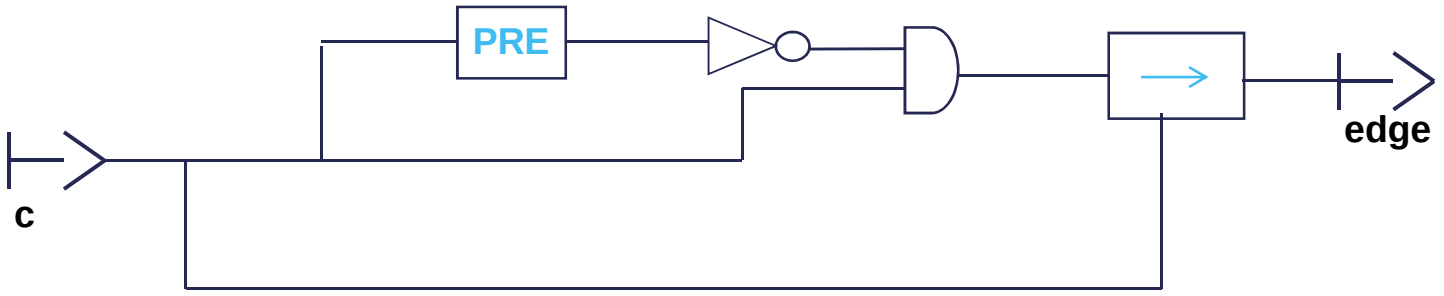
Scade Example: Rising Edge

→Detection of a true-false sequence

- Textual representation:

```
node RisingEdge (c: bool) returns (edge: bool);
let
edge = c → c and not pre(c);
tel;
```

- Graphical representation:



SCADE example: Fibonacci function

F_1 

F_2 

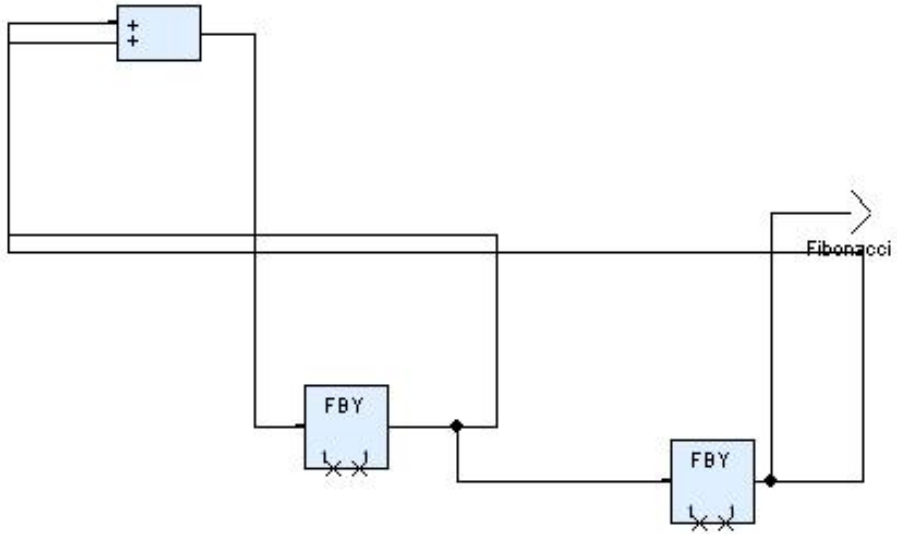
F_3  

F_4   

F_5     

F_6

$F_1=1$ • • •
$F_2=1$
$F_{n+2} = F_{n+1}+F_n$





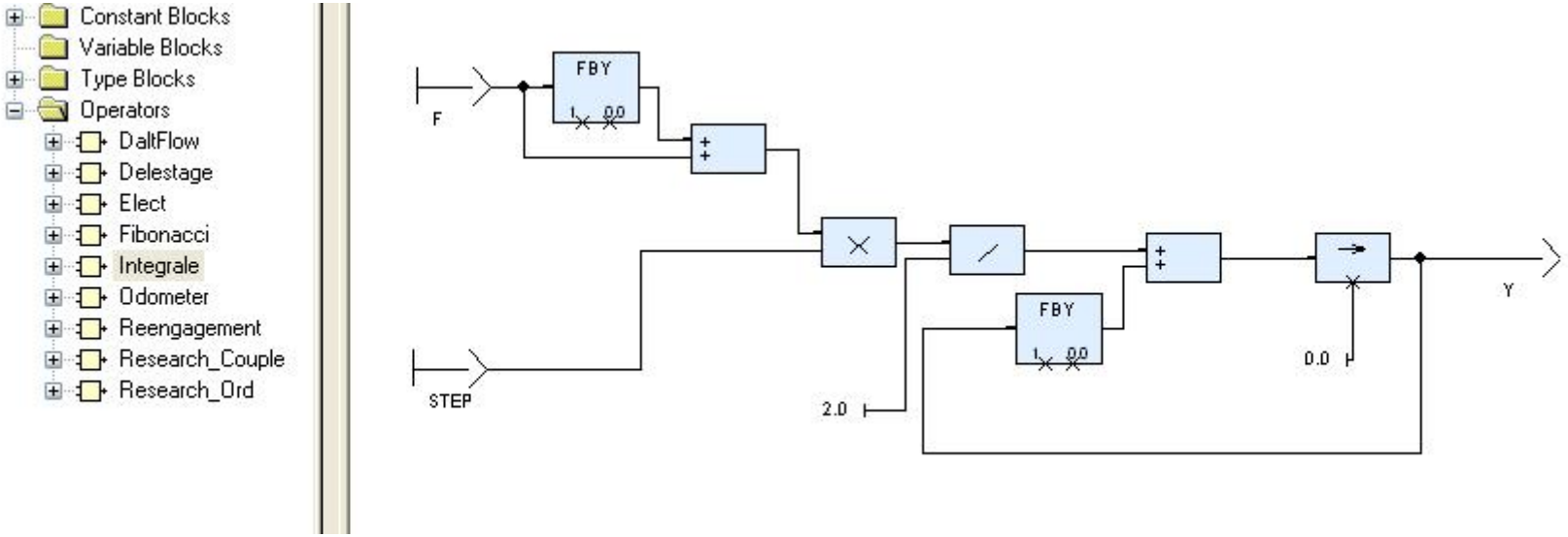


SCADE example: Compute the value of an integral

$$\int f(x)\,dx \sim \Sigma \left[\frac{(f(x_i) + f(x_{i+1}))\,(x_{i+1} - x_i)}{2} \right]$$

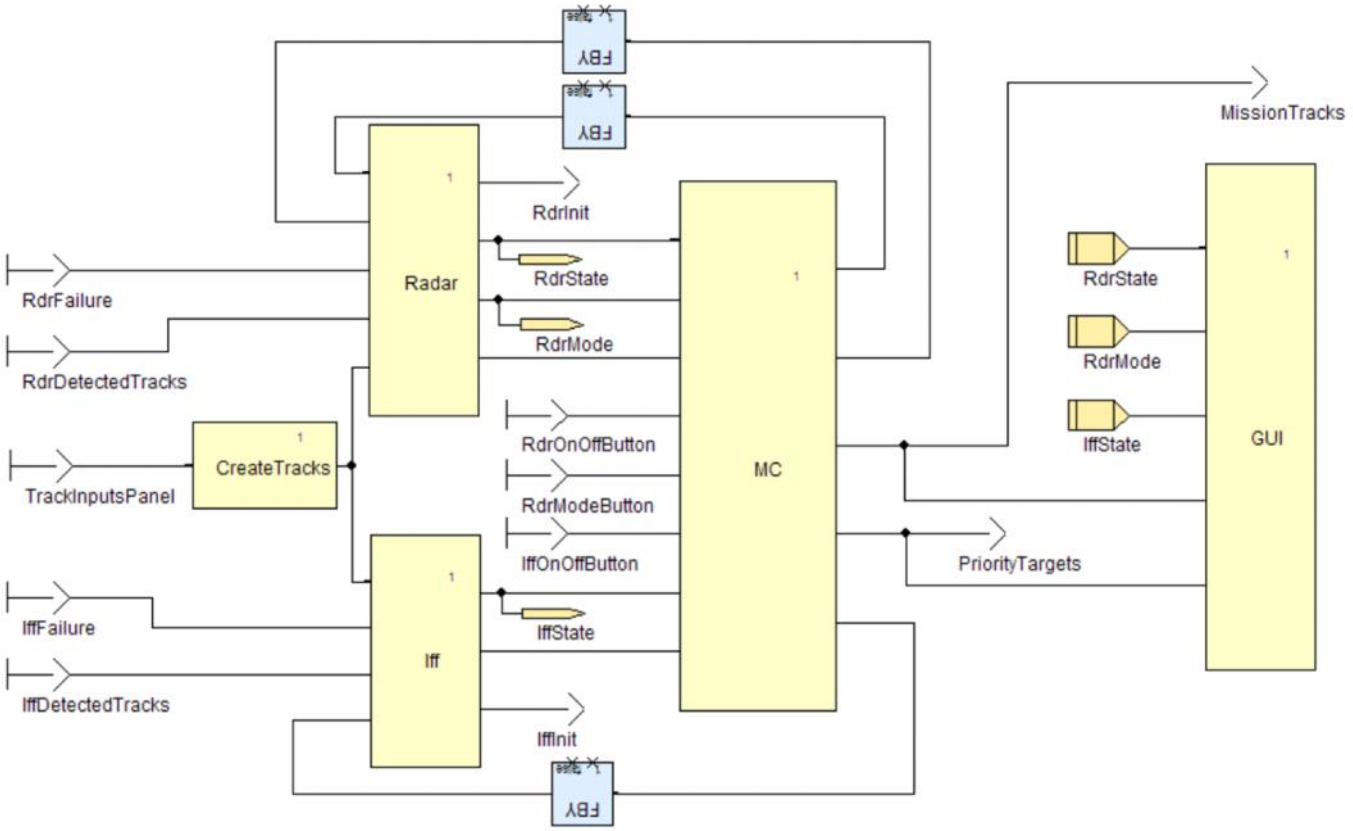
using the triangle method $\Rightarrow Y_{n+1} = Y_n + (F_n + F_{n+1}) * Step_{n+1} / 2$

with **F** : real flow of function f(x) values
Step : sampling interval



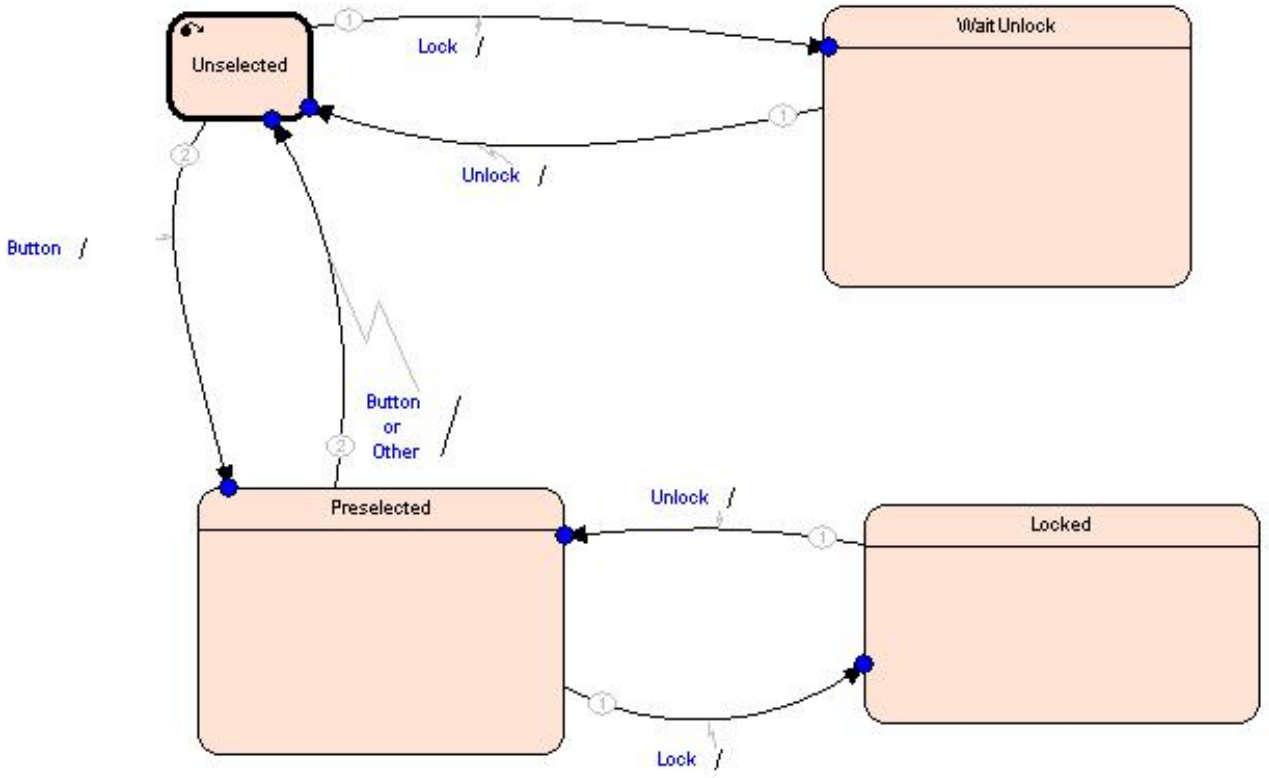
Dataflow Diagrams

- Boxes are computation units
- Wires are data-flows



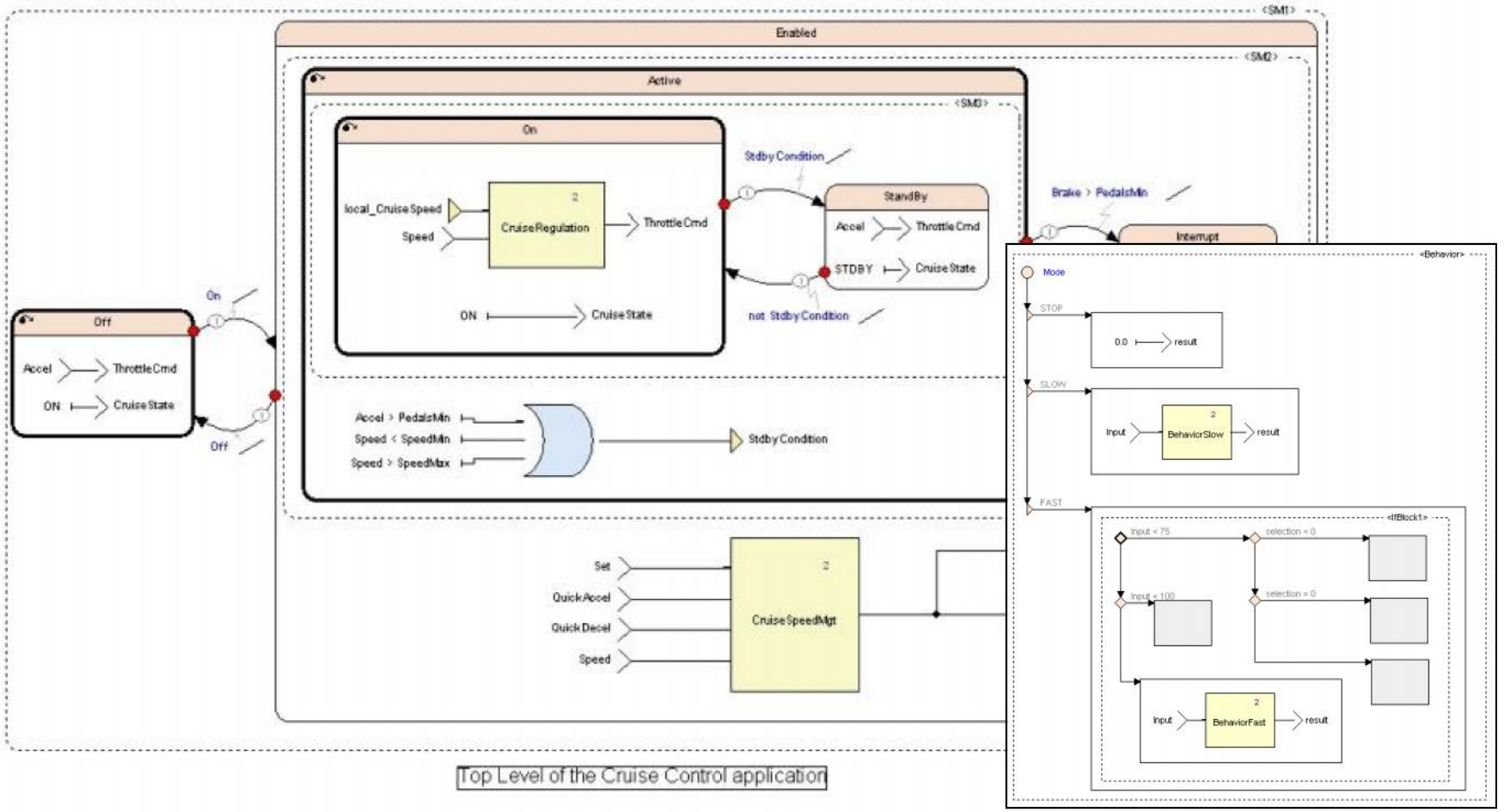
State Machines

- Boxes are states
- Arrows are state-transitions



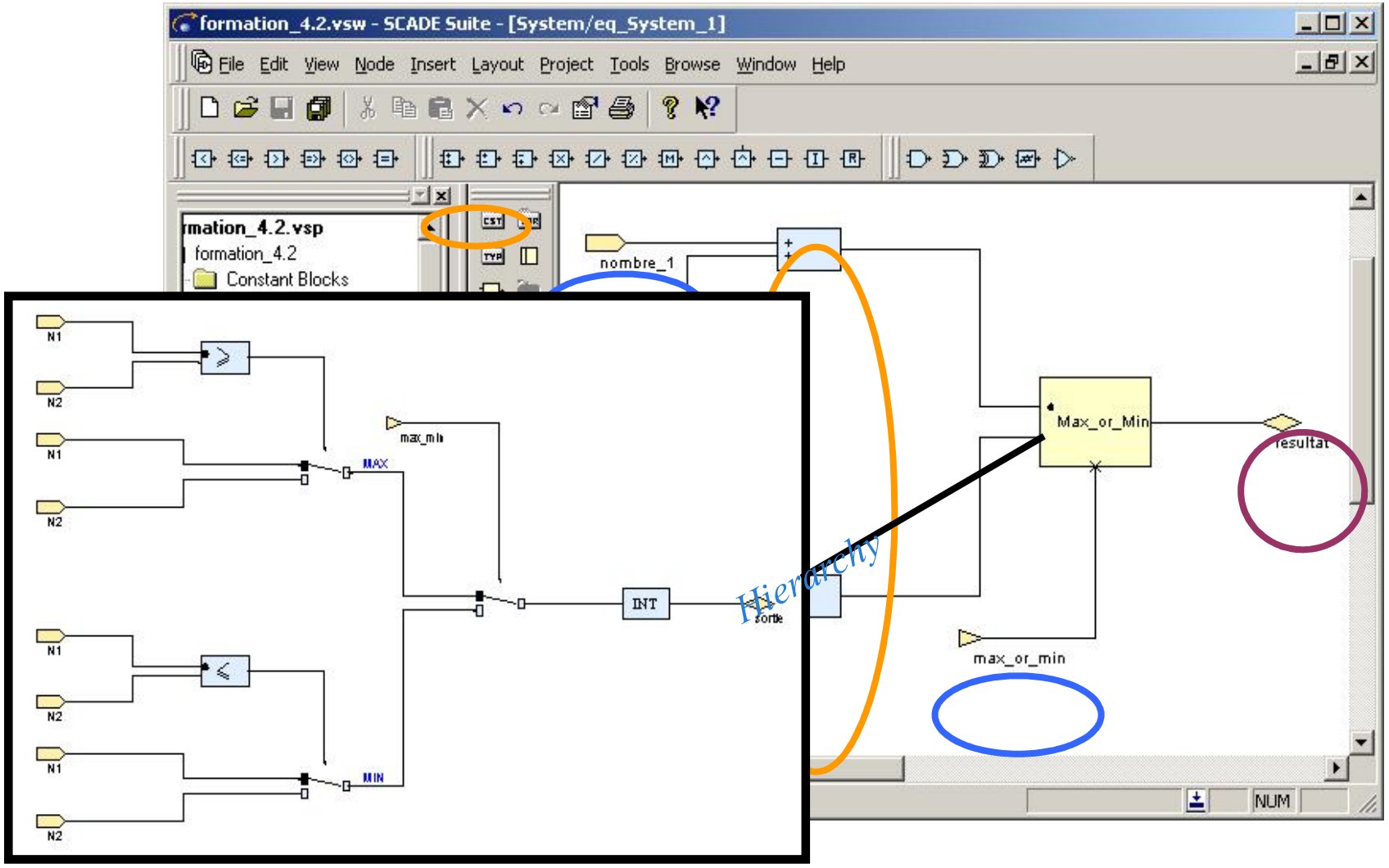
Integrated Data Flow & State Machines

- Modeling flexibility:
  Nested data flow & control flow



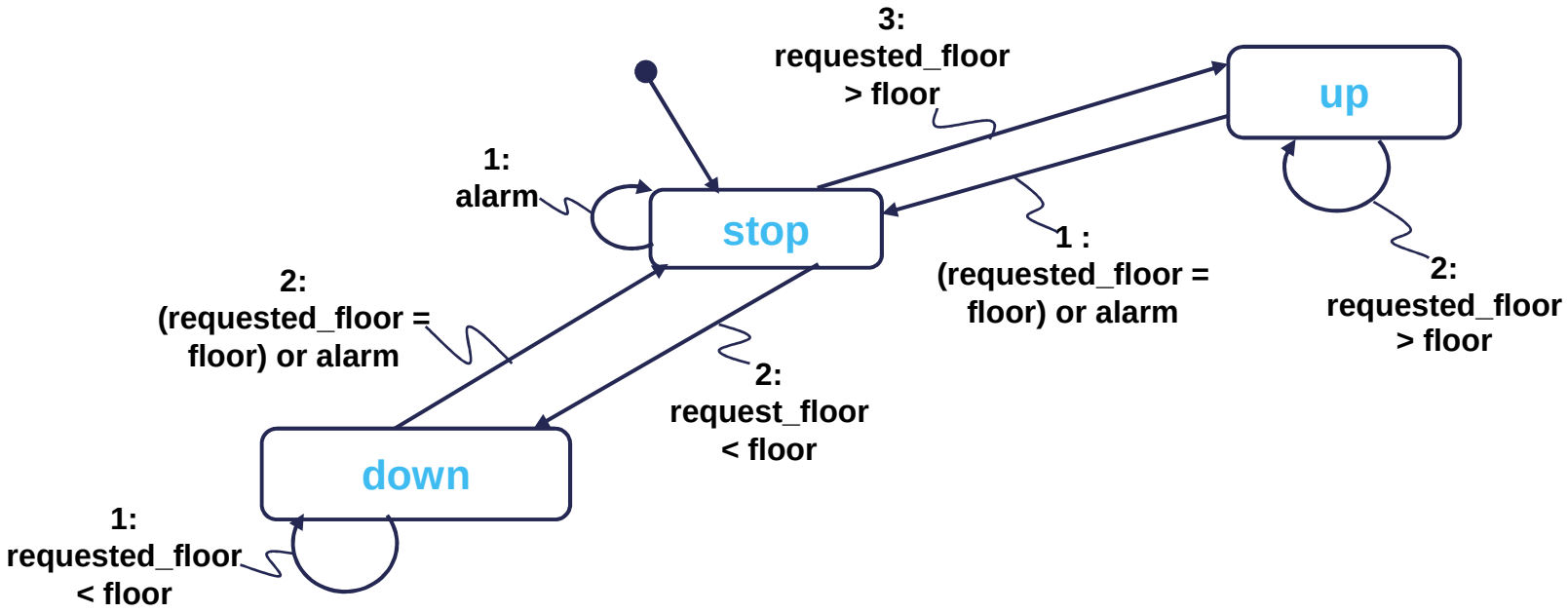
Top Level of the Cruise Control application

Hierarchy


formation_4.2.vsw - SCADE Suite - [System/eq_System_1]
File Edit View Node Insert Layout Project Tools Browse Window Help
formation_4.2.vsp
formation_4.2
Constant Blocks
nombre_1
Max_or_Min
resultat
max_or_min
NUM
N1
N2
MAX
MIN
INT
Hierarchy

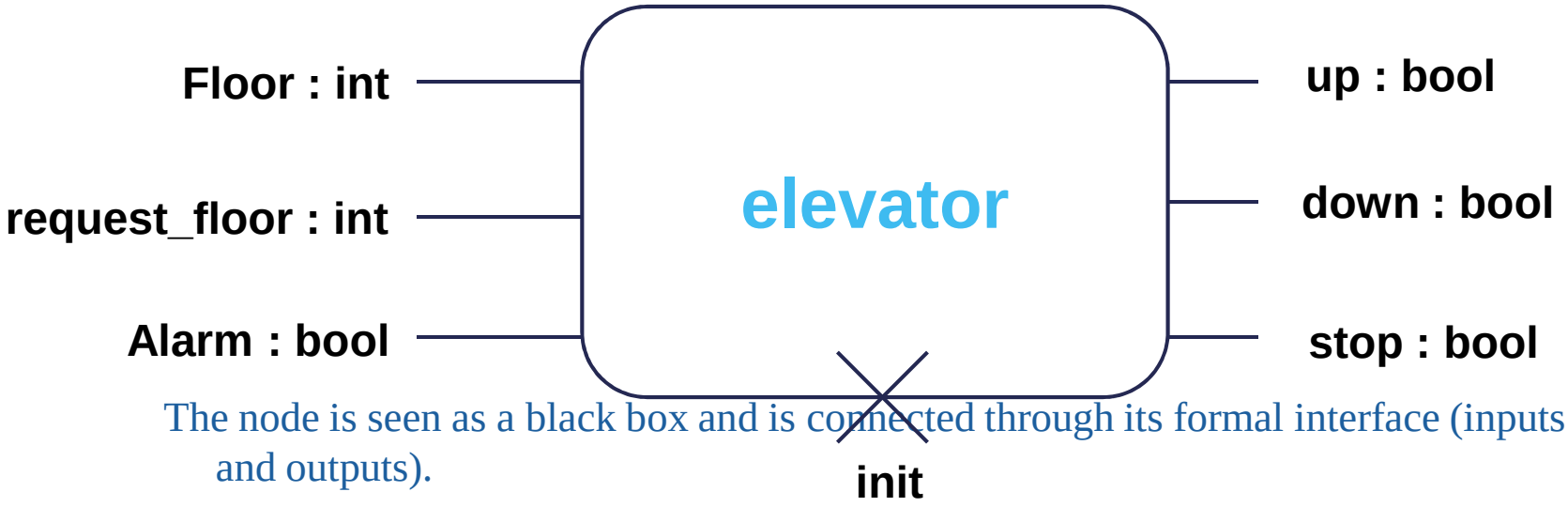
State Machine 1/2

- Transitions are orderly Boolean expressions :
  - Inputs are used with SCADE textual form.
  - Each transition has a explicit priority.



State Machine 2/2

- A state machine seen as a node:



The node is seen as a black box and is connected through its formal interface (inputs and outputs).





GEA Tianjin / 中国民航大学中欧航空工程师学院

SIMULATION WITH SCADE

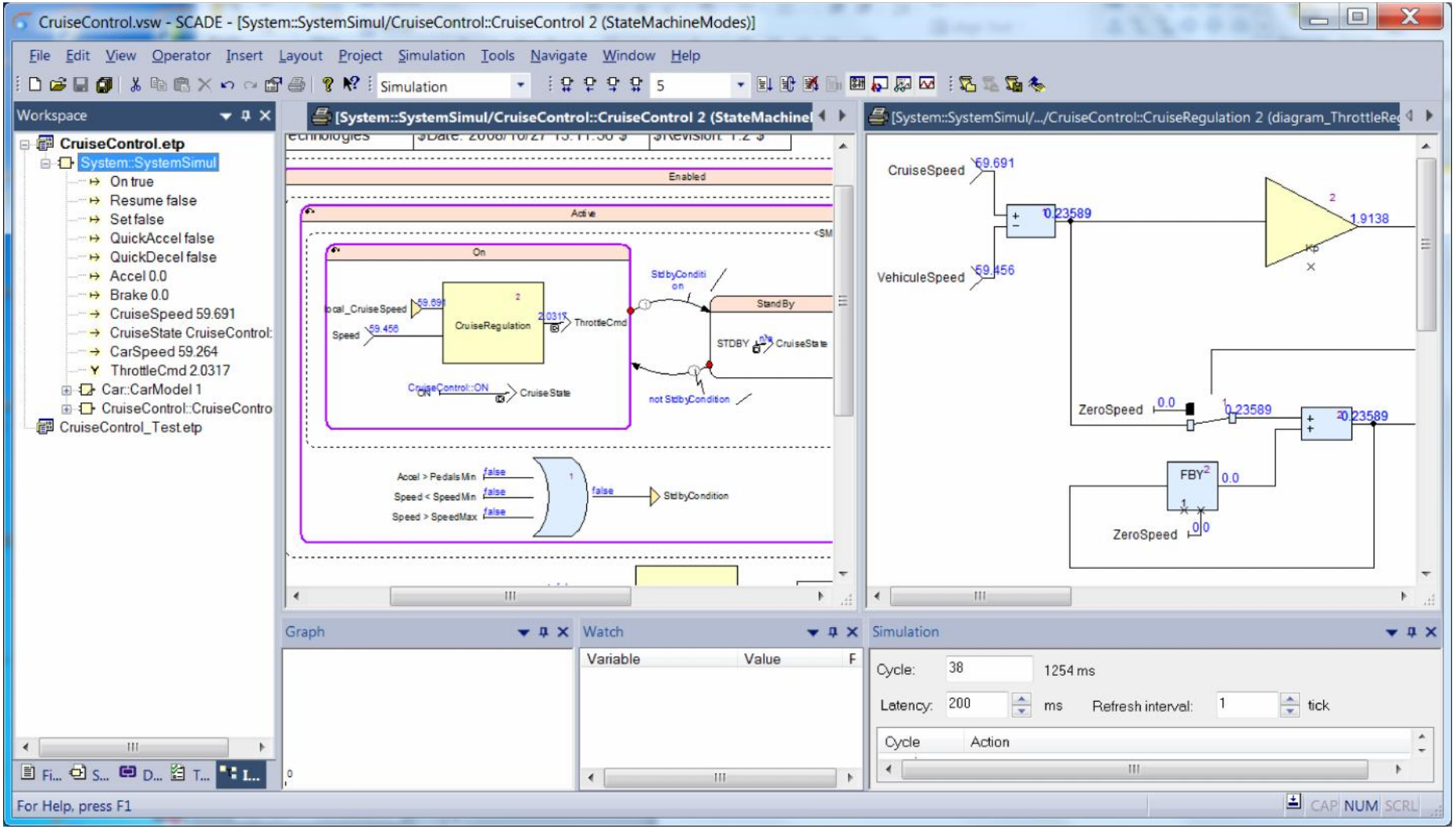




SCADE Simulator

- Functional testing
- Debug
- Stop conditions.
- Record and replay the test scenarios

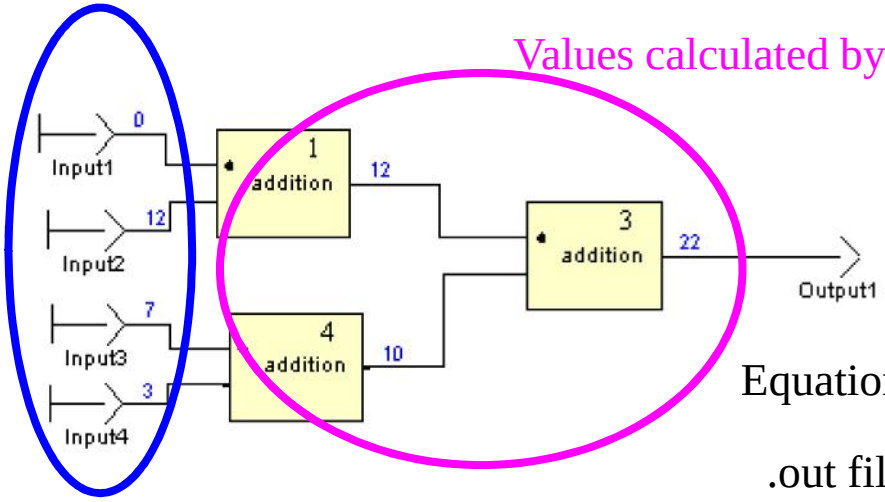
Debug & Simulation at Model Level



Simulation results

Values given by the user

Values calculated by the simulator



Watch window

Variable	Value
top_level/Output1	22
top_level/Input2	12

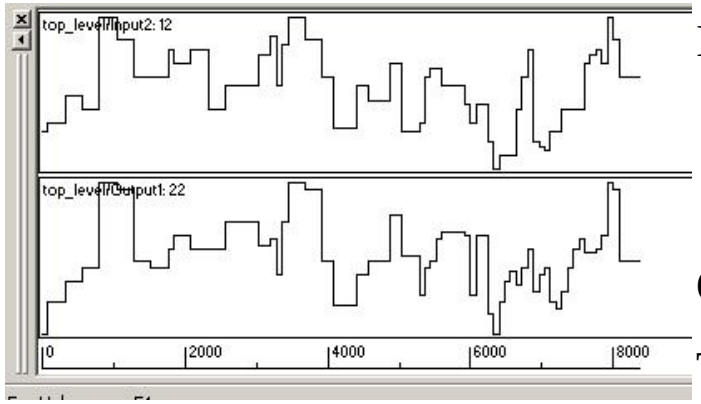
Equation block with variable

.out files

```
STEP 1
INPUT Input1 = 0
INPUT Input2 = 12
INPUT Input3 = 7
INPUT Input4 = 3
OUTPUT Output1 = 22

STEP 2
INPUT Input1 = 1
INPUT Input2 = 4
INPUT Input3 = 5
INPUT Input4 = 2
OUTPUT Output1 = 12
```

Graph view



Input2

Output1

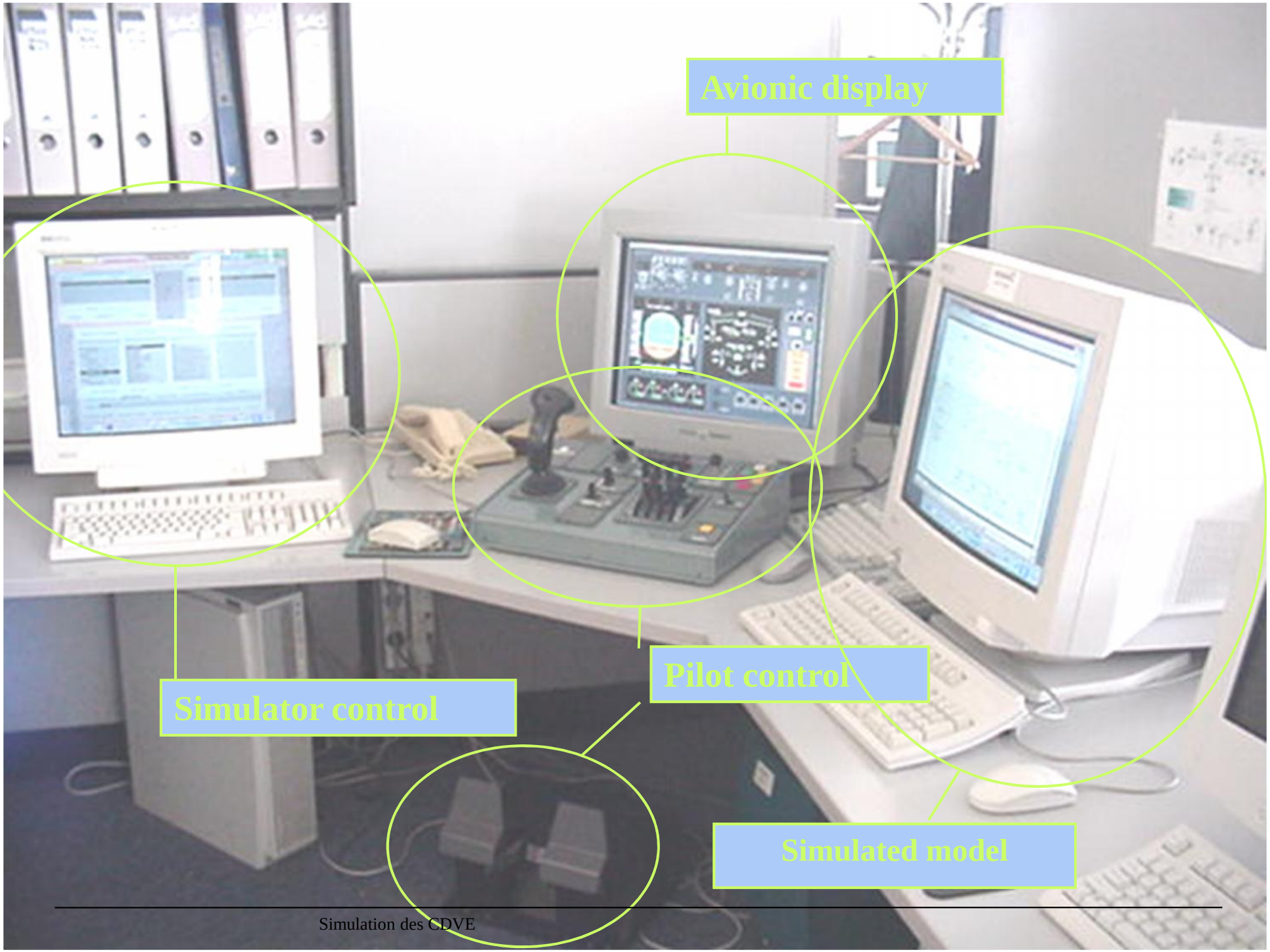
Time

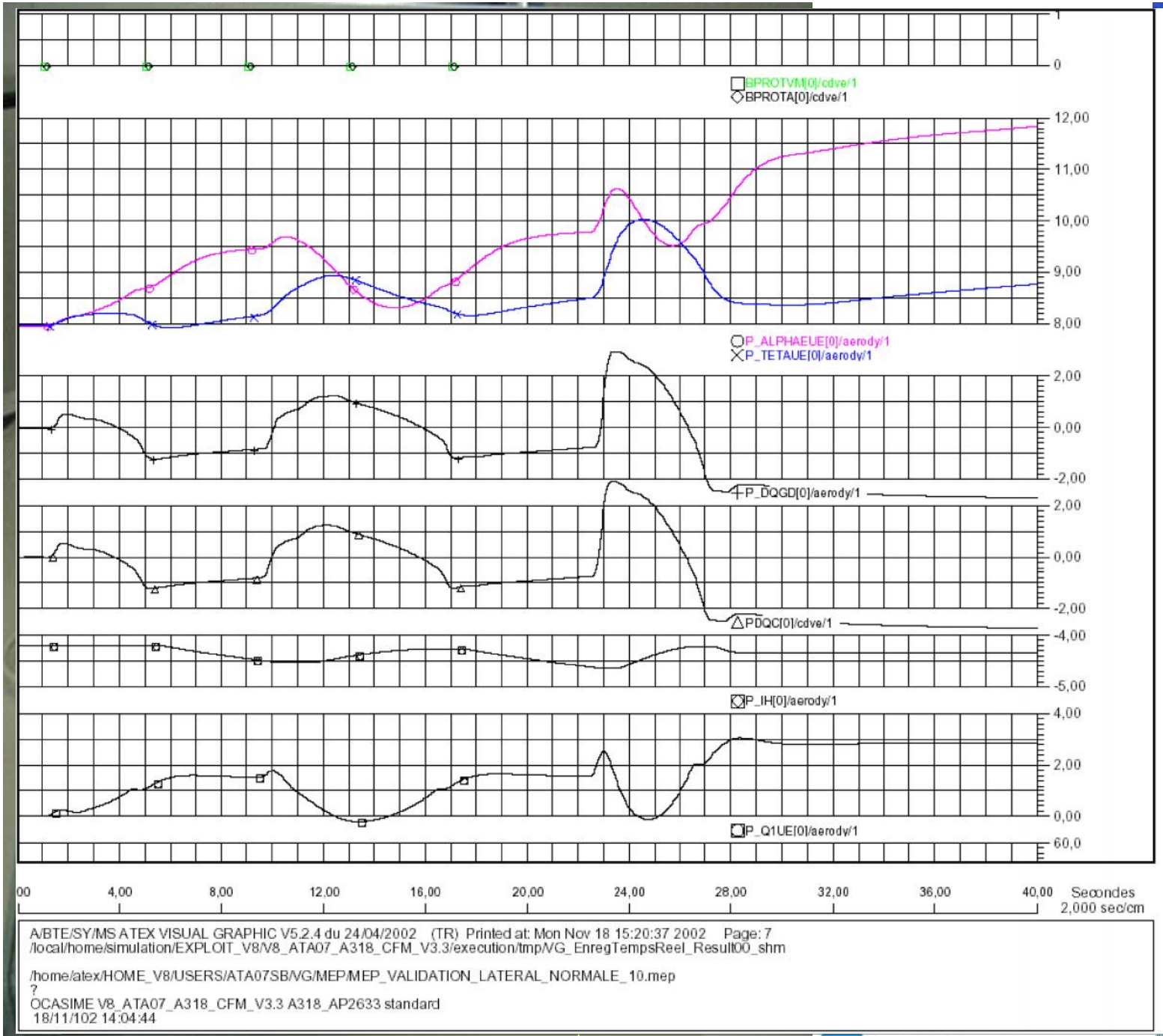
Simulation benefits

- Check of intended behaviour
  - By visual check
  - Through automated comparison to properties
- Identification of unintended behaviour
  - Through comparison to properties like « Unwanted event never appears »
- Non-regression tests
- Generation of test patterns

Simulation bench: OCASIME

- Outil de Conception Assisté par Simulation Multi-Equipement
  - Mixing real equipments with simulated parts (flight control laws)


Avionic display
Simulator control
Pilot control
Simulated model



Modele cdv
- Sous-modele cdve
- Init cdve
- Sous-modele adirsim_sersimu
- Init adirsim_sersimu
- Sous-modele simssm
- Init simssm

Modele bdvl
- Sous-modele aerody
- Init ...3/officiel/datappli/a318cfm40b
- Sous-modele mecvol
- Init mecvol
- Sous-modele radial
- Init radial
- Sous-modele rltfrn
- Init ...3.3/officiel/datappli/r318_40a1
- Sous-modele pomotr
- Init .../officiel/datappli/cfm565b9_40
- Sous-modele mochar
- Init ...3.3/officiel/datappli/c318_40a
- Sous-modele mcicar
- Init 1
- Sous-modele atmosf
- Init atmosf
- Sous-modele irs
- Init ...A318_CFM_V3.3/officiel/datap

Modele systeme
- Sous-modele diapmi
- Init diapmi
- Sous-modele cfm5b9
- Init cfm5b9
- Sous-modele cfmadc
- Init cfmadc
- Sous-modele bscumo
- Init bscumo
- Sous-modele slatfl
- Init slatfl
- Sous-modele lgcium
- Init lgcium
- Sous-modele etasimu
- Init 1
- Sous-modele syelec
- Init syelec

Modele env1
- Sous-modele vents
- Init donthermique.init
- Sous-modele turbul
- Init series.txt

Modele soreth40
- Sous-modele fgeint
- Init fgeint
- Sous-modele soreth5
- Init soreth5
- Sous-modele comtosgi
- Init 3|singlecast|12020|ocasi73
- Sous-modele entpos
- Init entpos

Modele soreth80
- Sous-modele comtosgi
- Init 3|singlecast|12020|ocasi73
- Sous-modele sorethvu
- Init 2|ocasi73|1091
- Sous-modele entethvu
- Init 4321
- Sous-modele entsorsys
- Init entsorsys
- Sous-modele syhydr
- Init syhydr

Lib pentpos L cfmadc libADI_cfmadc
Lib aerody libAER_cfm_3.0_11.1.4.s
Lib atmosf libATM_atmosf_12.0_11.

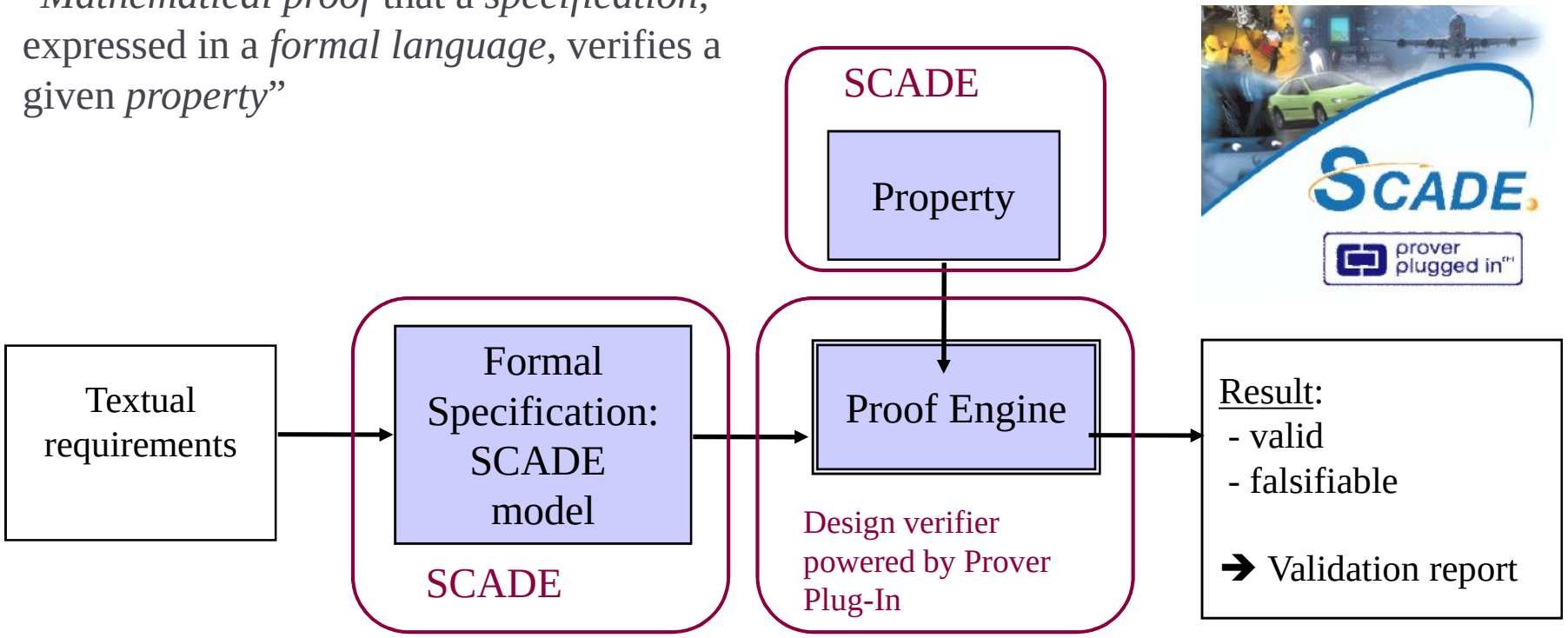
OCASIME benefits

- Early check of integration problems
- Verification of the complete fonction, on ground
- Validation by real users (= pilots)
- Comparison with in-flight data
  - → possibility to sync the model with the real world

Formal verification

What is formal verification?

"*Mathematical proof* that a *specification*, expressed in a *formal language*, verifies a given *property*"



GEA Tianjin / 中国民航大学中欧航空工程师学院

Design Verifier

Formal Verification Assistant

- **Formal Verification Assistant** to formally express and assess safety requirements
  - Find bugs early when they are less costly to fix
  - Produce counter-examples to help debugging
  - Perform exhaustive analysis
  - Identify special safety risks such as zero divide
- Limits
  - Standard mathematical limits: non linear problems



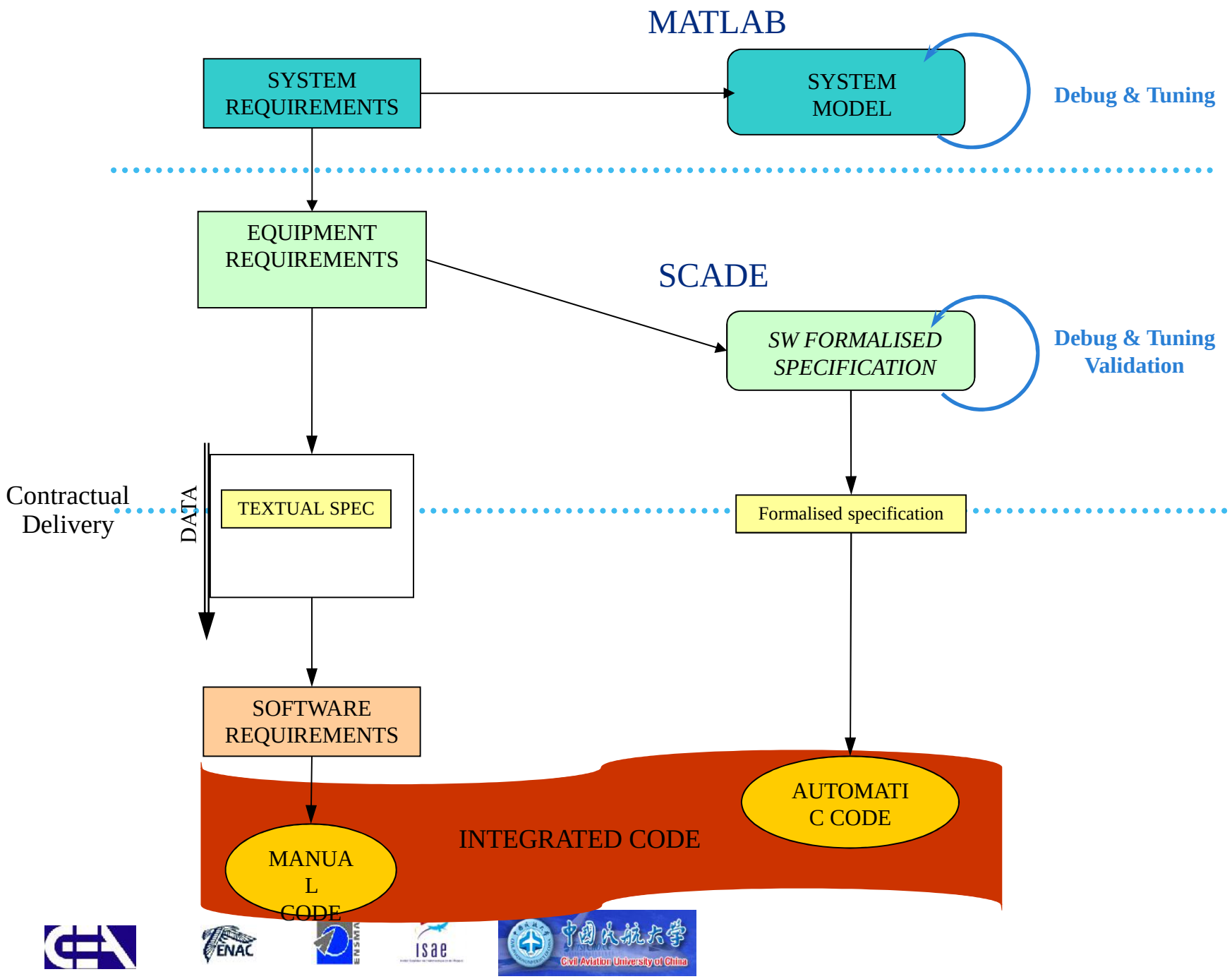


GEA Tianjin / 中国民航大学中欧航空工程师学院

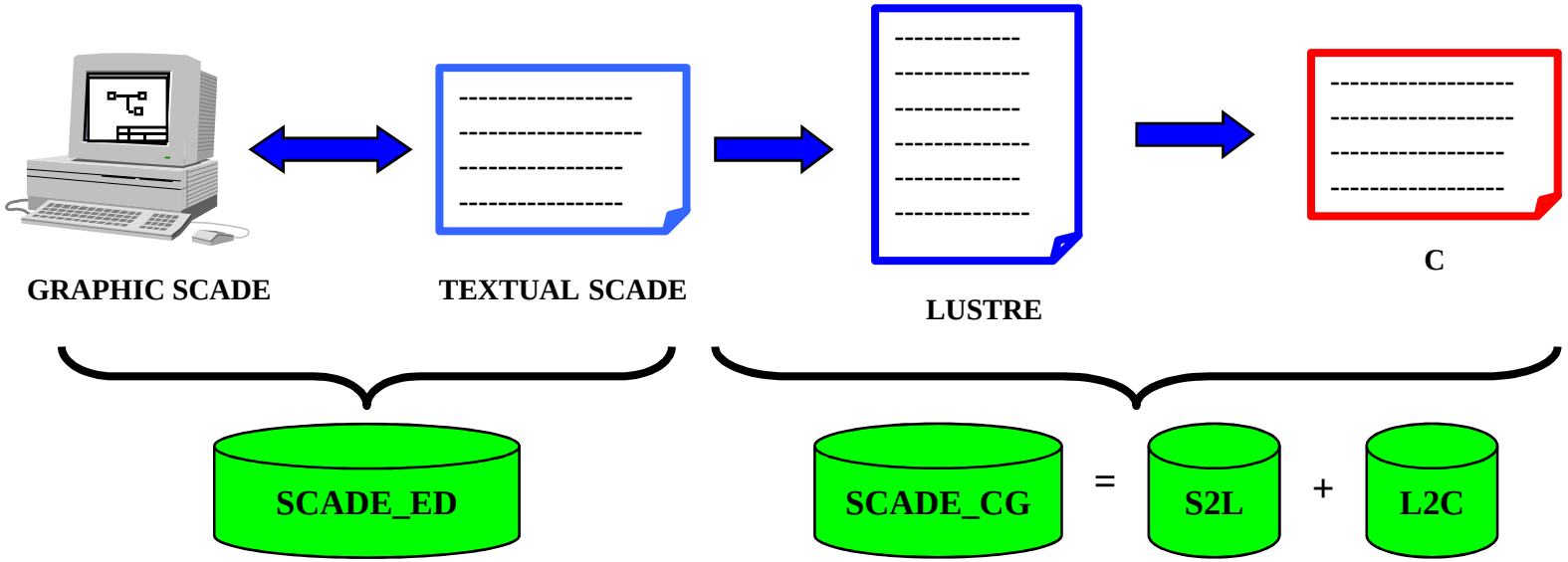
AUTOMATED CODE GENERATION






MATLAB
SYSTEM REQUIREMENTS
SYSTEM MODEL
Debug & Tuning
EQUIPMENT REQUIREMENTS
SCADE
SW FORMALISED SPECIFICATION
Debug & Tuning Validation
Contractual Delivery
DATA
TEXTUAL SPEC
Formalised specification
SOFTWARE REQUIREMENTS
AUTOMATI C CODE
INTEGRATED CODE
MANUA L CODE
ENAC
isae
Civil Aviation University of China

Code generation with SCADE



Generation for simulation

Instrumented C code

Generation for the final equipment

Qualified (DO178B) code generation

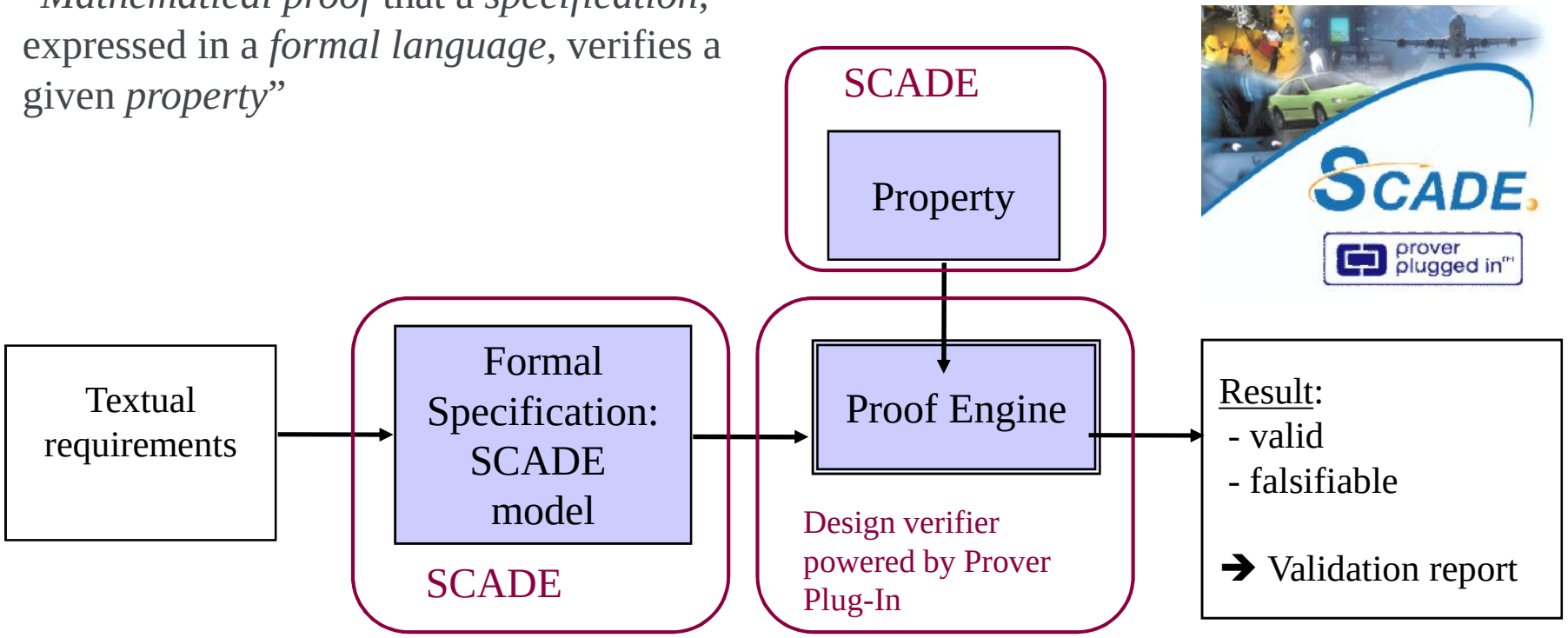
Generated code properties

- readable, traceable
- portable (independent from the target)
- Modular
- static memory allocation
- finite execution duration
- static size optimizations
  - Performance → execution time
  - Memory space → code volume

Formal verification

What is formal verification?

"*Mathematical proof* that a *specification*, expressed in a *formal language*, verifies a given *property*"



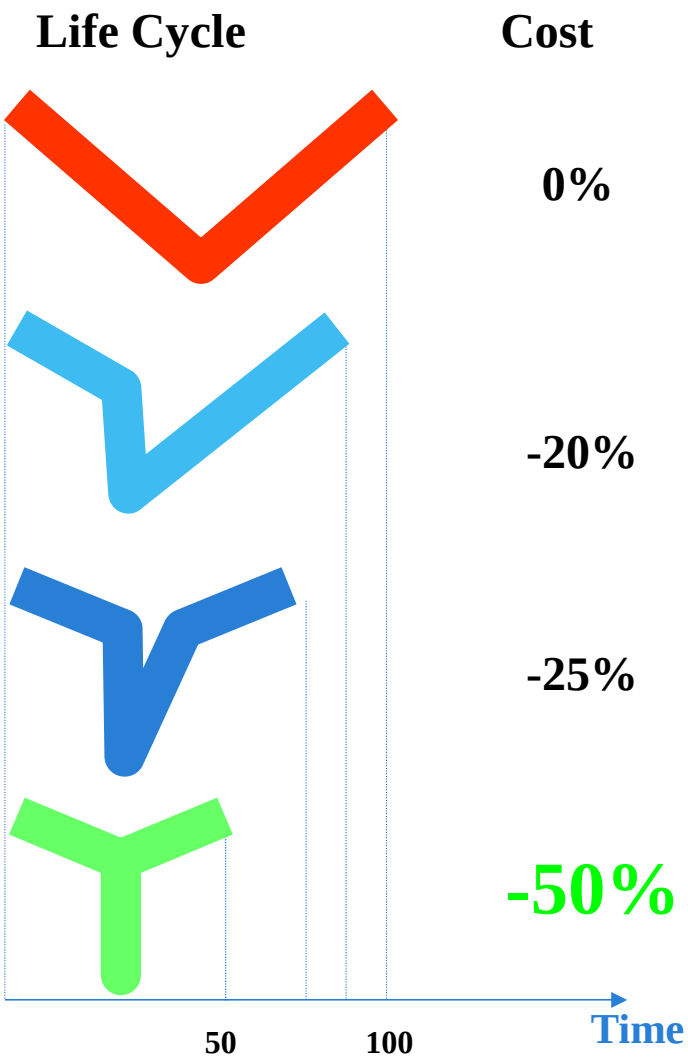
Project benefits : From V to Y Cycle

Manual coding

Use of a “regular” automatic code generator

on a validated model
Use of the qualifiable code generator as a verification tool

Use of the qualifiable code generator as a development tool

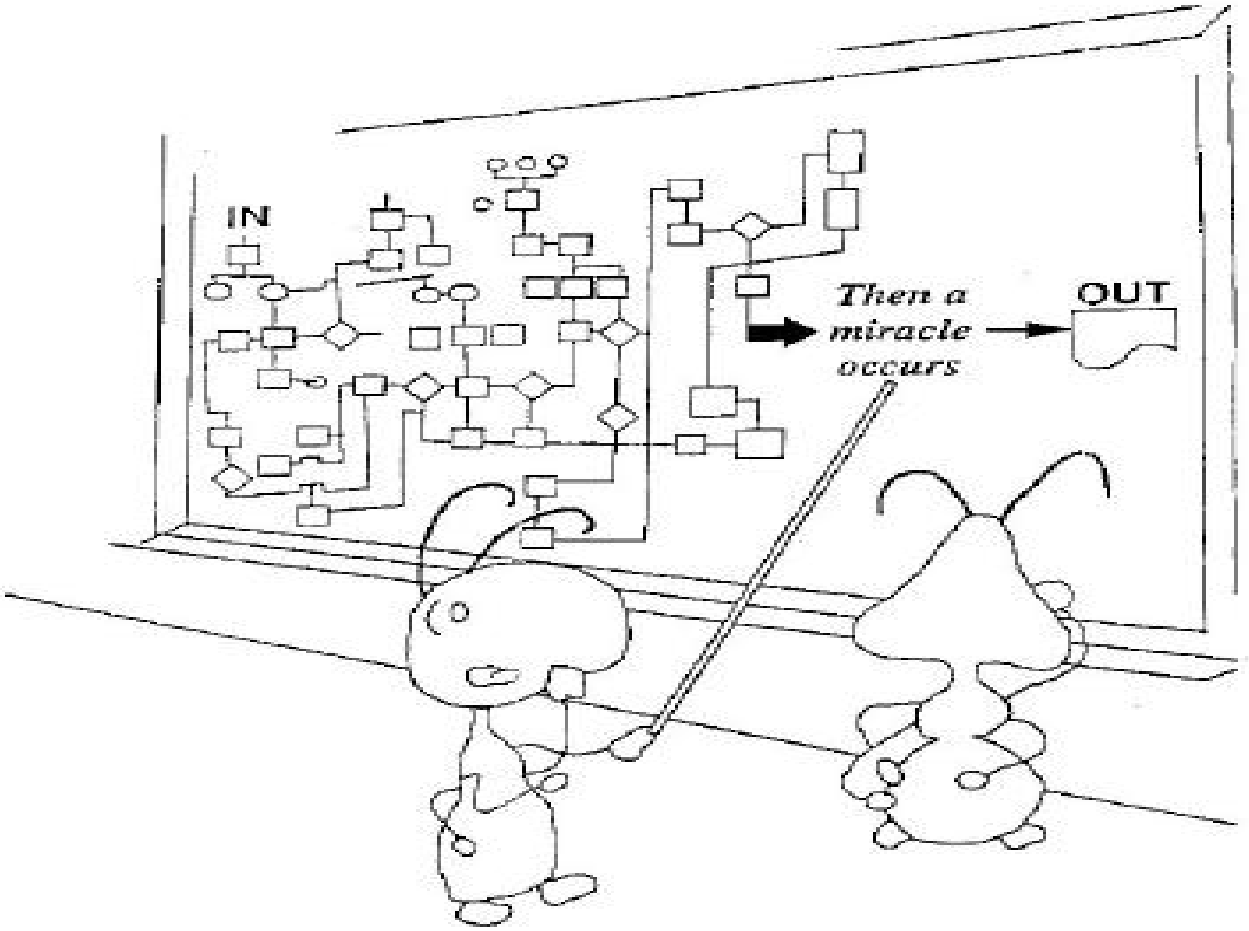


SCADE benefits

For DO-178B projects, SCADE™ provides:

- productivity and quality benefits
- automatic semantic verifications
- automatic code generation (40% to 70%)
- testing assistance
- automatic documentation (maintenance)
- reusability
- suppression of the code review and tests reduction (qualifiable code generator)

That's all folk's…



"Good work ..... but I think we need just a little more detail right here"