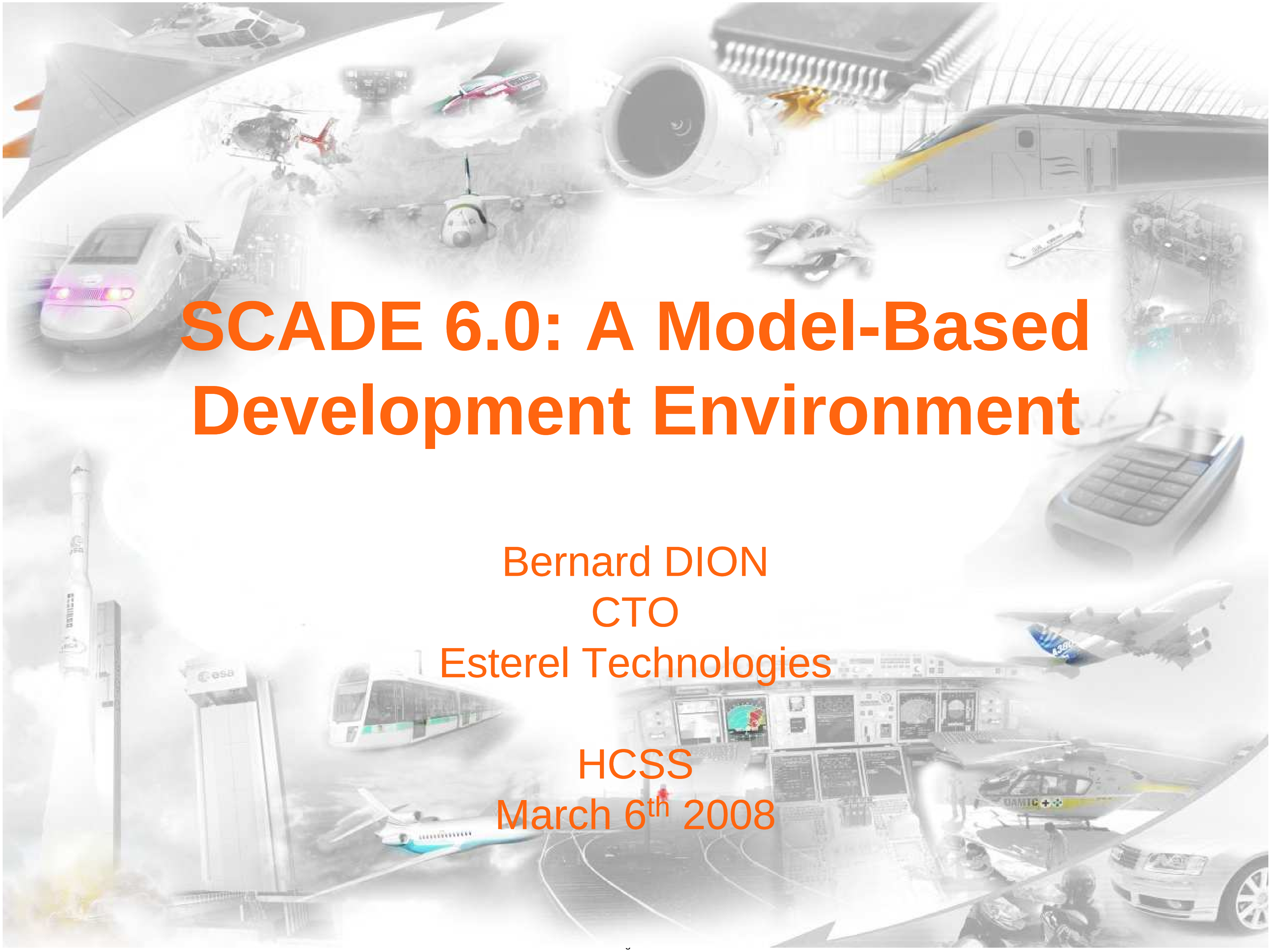


SCADE 6.0: A Model-Based Development Environment

Bernard DION
CTO
Esterel Technologies

HCSS
March 6th 2008

Agenda

- Esterel Technologies' Introduction
- Objectives of this Presentation
- The SCADE 6 Language
- The SCADE 6 Toolset
- Summary of the Benefits of Using SCADE 6

Agenda

- **Esterel Technologies' Introduction**
- **Objectives of the Presentation**
- The SCADE 6 Language
- The SCADE 6 Toolset
- Summary of the benefits of SCADE 6

Esterel Technologies

- We are committed to Mission and Safety-Critical Application Development
- We provide the SCADE Model-based Development Environment used extensively by systems and software engineers
- SCADE contains FAA / EASA / TUV Certified Code Generation for applications such as Airbus A380 Flight Controls, Boeing 787 Landing Gear …

Objectives of the Presentation

- Improve specification quality & accuracy
  - Show how we can provide formal and deterministic specifications
- Eliminate the cost of finding bugs late
  - Show how we can eliminate specification and design flaws early
- Reduce the cost of software implementation
  - Demonstrate that automatic and certified code generation ensures application integrity and cost effectiveness

Development and Verification objectives are imposed by standards such as DO-178B, IEC 61508, EN 50128, IEC 60880

Agenda

- Esterel Technologies' Introduction
- **The SCADE 6 Language**
- The SCADE 6 Toolset
- Summary of the Benefits of Using SCADE 6

SCADE was created for Safety

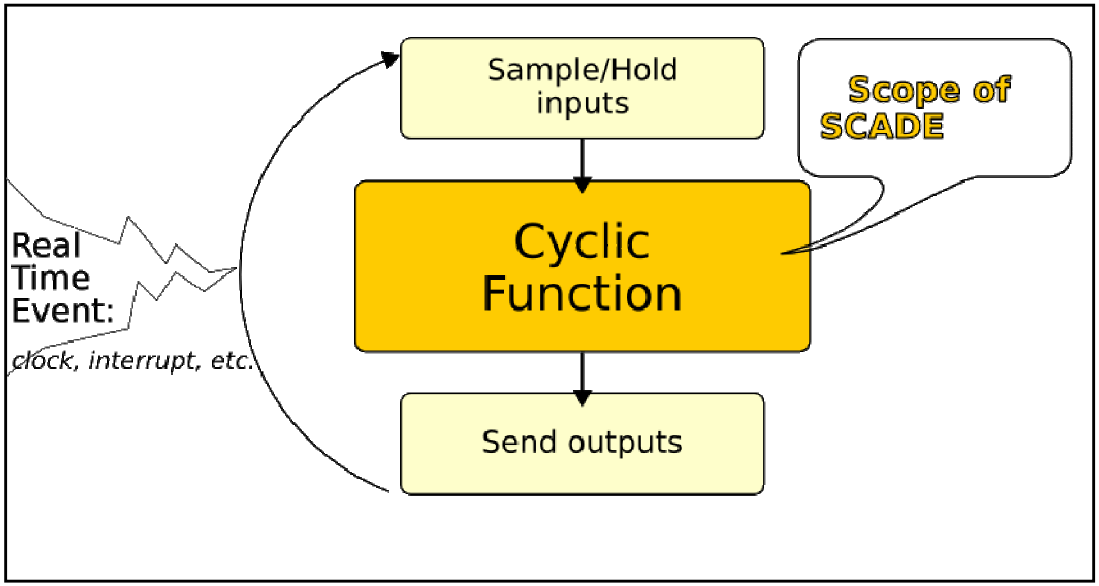
- The SCADE language is **formally defined** with **key safety objectives**:
  - Deterministic models
  - Simple and stable language, forbidding dangerous constructs (e.g. wild goto's, dynamic memory allocation,…)
  - Modular, strongly typed, explicit specification
  - Interpretation of a SCADE model does not depend on the reader nor its environment
  - Very active research work for more than 10 years (Lustre & Esterel synchronous languages)

- Designed in close connection with certification authorities in the aeronautics, transportation & nuclear energy domains

Cycle-Based Execution Model of SCADE

- For each cycle, the program generated from SCADE computes its output vector from a stable input vector

- No interaction between the program and its environment during a computation cycle, and therefore strong practical properties **(fully deterministic behavior)**

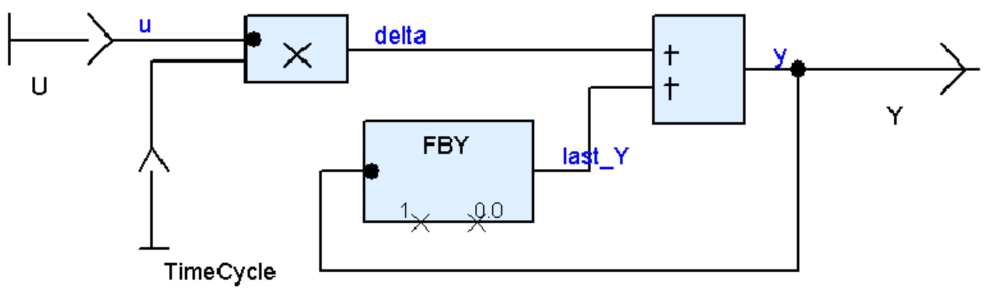


- A node is a functional module, defined by
  - a formal interface:

```
node IntegrFwd( U: real ; hidden TimeCycle: real)
returns ( Y: real);
```

  - a set of local variables declarations:

```
var
  delta : real;
  last_Y : real;
```



  - a set of equations:

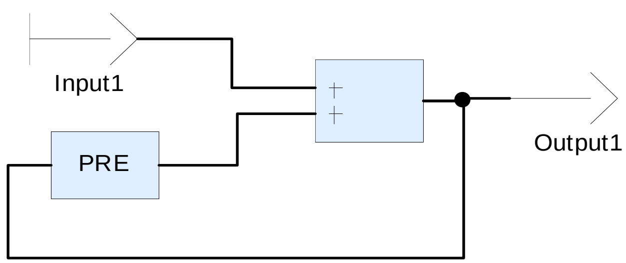
```
delta = u * TimeCycle;
y = delta + last_Y;
last_Y = fby(y , 1 , 0.0);
```

SCADE Time Operators

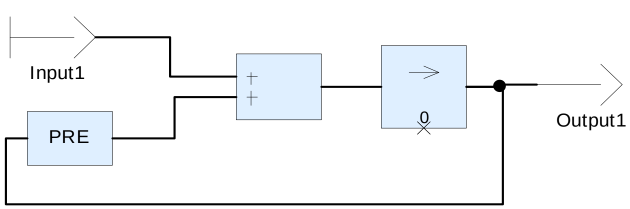
- **pre**: delay one cycle

$$y = az^{-1}x \longrightarrow y = a \times pre\ (x)$$

- **->**: data flow initial value



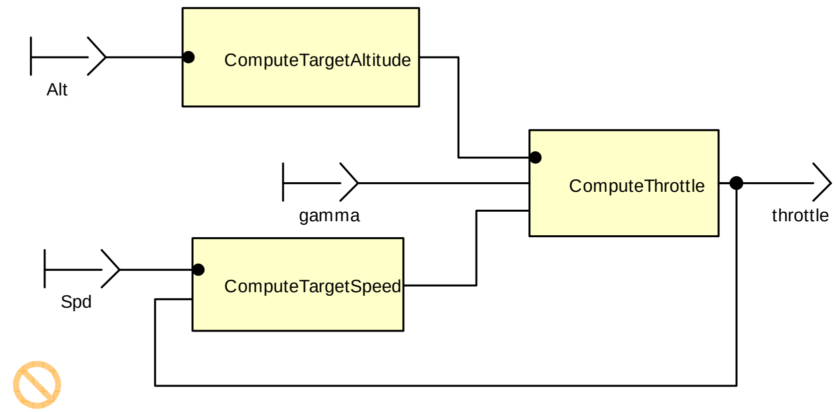

```
WARNING: NODE Count, VAR Output1,
please verify the initialization
of the variable(s)
```



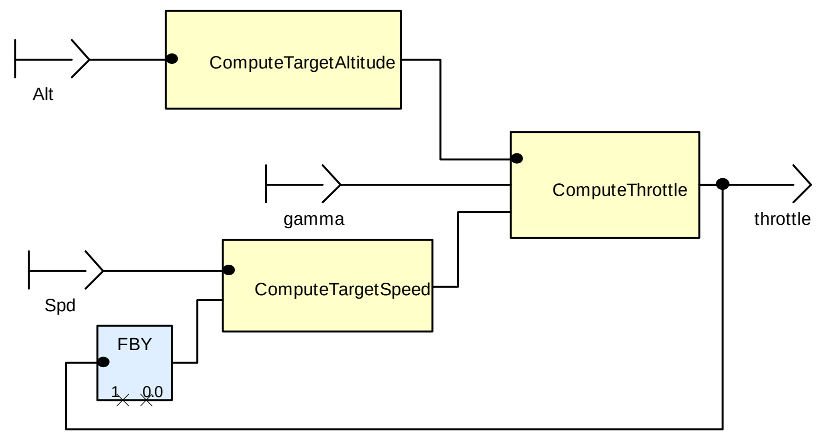

```
Translation completed with
0 semantic error(s),
0 semantic warning(s).
```

Causality Checks

- Causality implies that data have to be produced before being consumed
  (in a given cycle)




```
Error: recursion detected in
    variables definition
```



OK

Static Checks

- **Semantic errors detected at compile time** to prevent runtime errors:
  - Type check
  - Clock check
  - Initialization check
  - Causality check

From SCADE 5 to SCADE 6

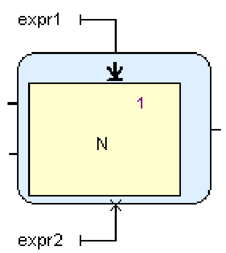
- SCADE 6 has **same safe foundations** as SCADE 5

- SCADE 6 is an **extension** of SCADE 5
  - In addition to data flow intensive applications, SCADE 6 addresses
    - Control intensive applications
    - Array intensive applications

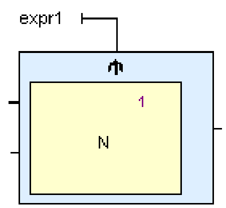
SCADE 6 for control intensive applications

- SCADE 5 and SCADE 6 semantics rely on clocked data flows
  - Precise and simple semantics
  - Low level constructs available to sample, merge, ... clocked data flows

- SCADE 6 extends SCADE 5 with higher level constructs
  - Conditional activation of an operator: “activate <N>”



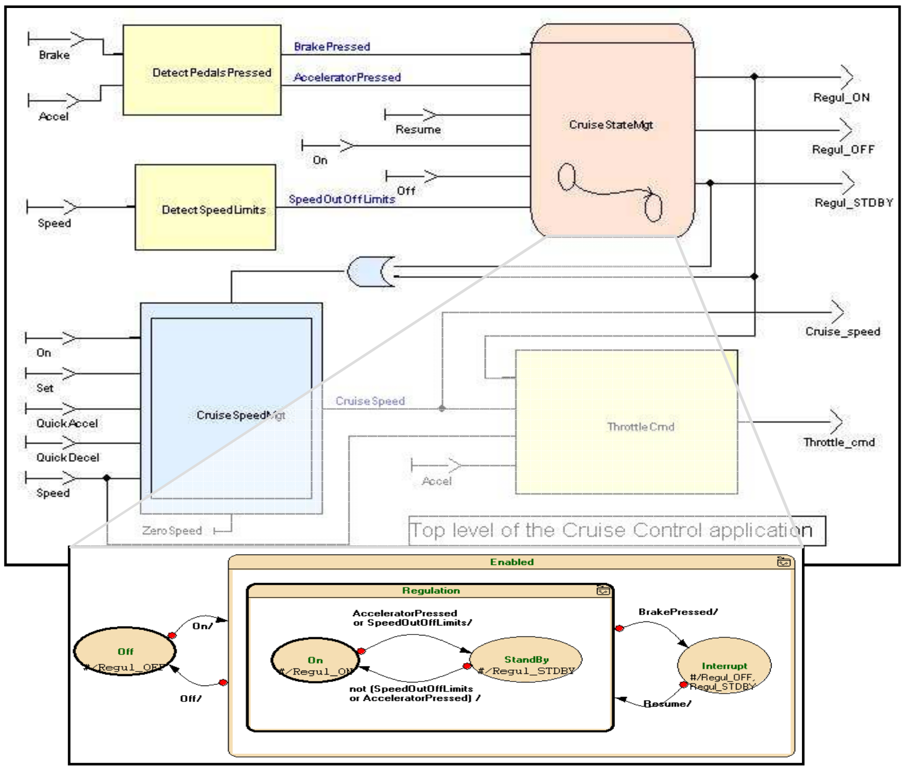
  - New construct: “restart <N>”
    - Perform a simple reset of complex operators



SCADE 6 for control intensive applications
Unified Modeling Style

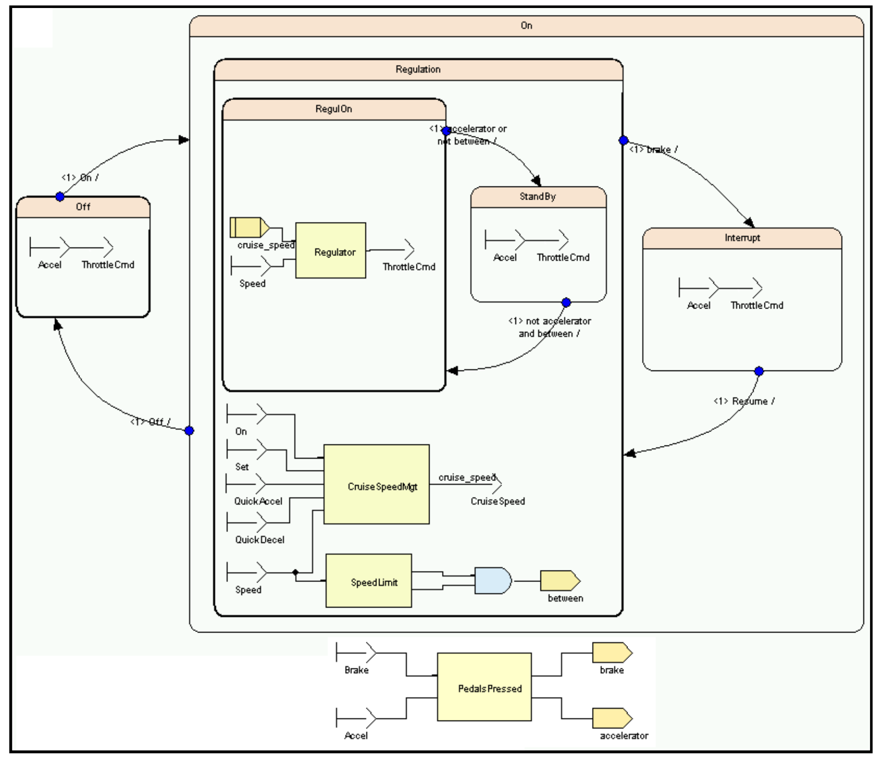
- SCADE 6: **merging dataflow and control flow designs**

- Traditional separation of Control Flow and Data Flow



Pure Control Flow into
Pure Data Flow

- Power of Nested Data Flow & Control Flow



*Data Flow & Control Flow become "**One**"*

ESTEREL Technologies

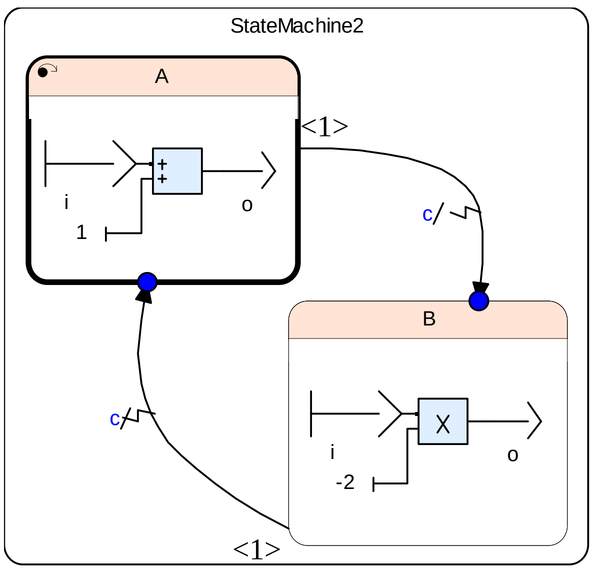
SCADE 6 for control intensive applications

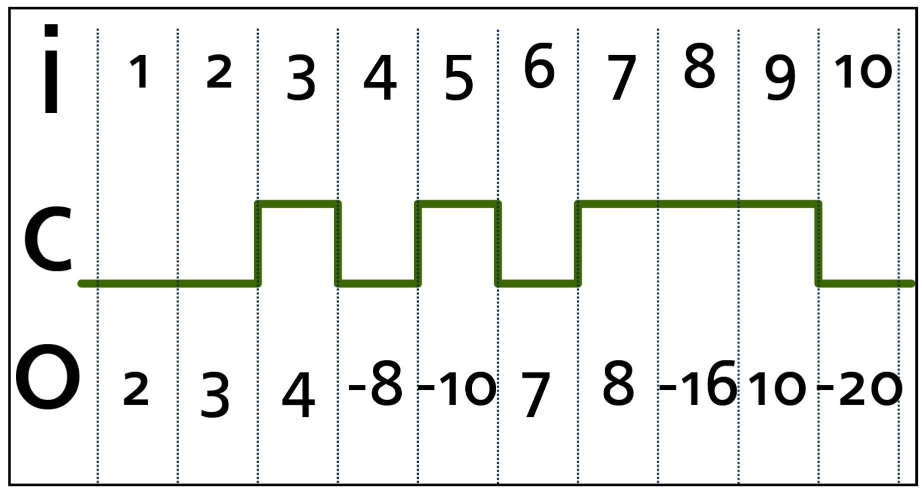
Safe & powerful state machines

- Principles
  - Only one value set per variable and cycle
  - One active state, and one transition fired per state-machine and cycle

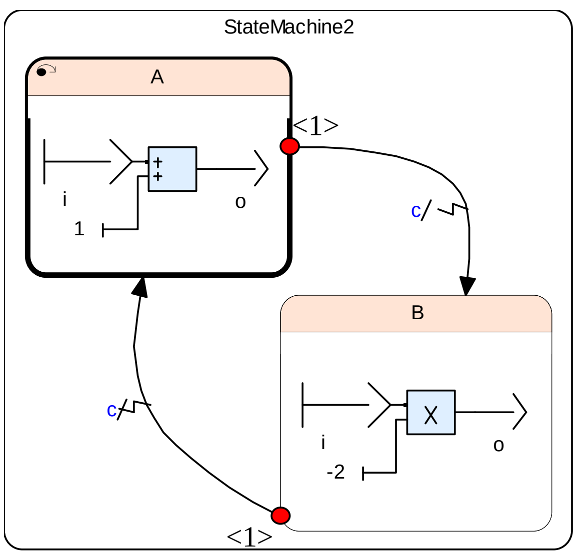
- State machine transitions
  - Weak or strong: target state executed in next or in current cycle
  - Restart or resume: target state reset or not

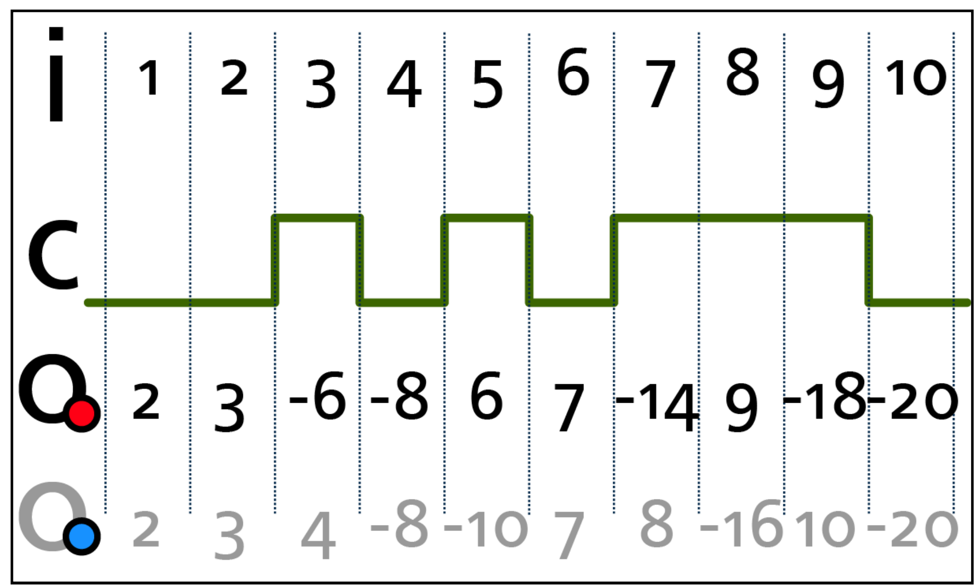
- Weak transition
  - Target state activated at the next cycle only





- Strong transition
  - Specifies the state where the computation in the current cycle will take place





SCADE 6 for array intensive applications

- Arrays are required for:
  - Computation: matrices, vectors, and operations
    - Simple vector computations, filters, polynomials
  - Code multi-instantiation
    - Several instantiation of the same operator

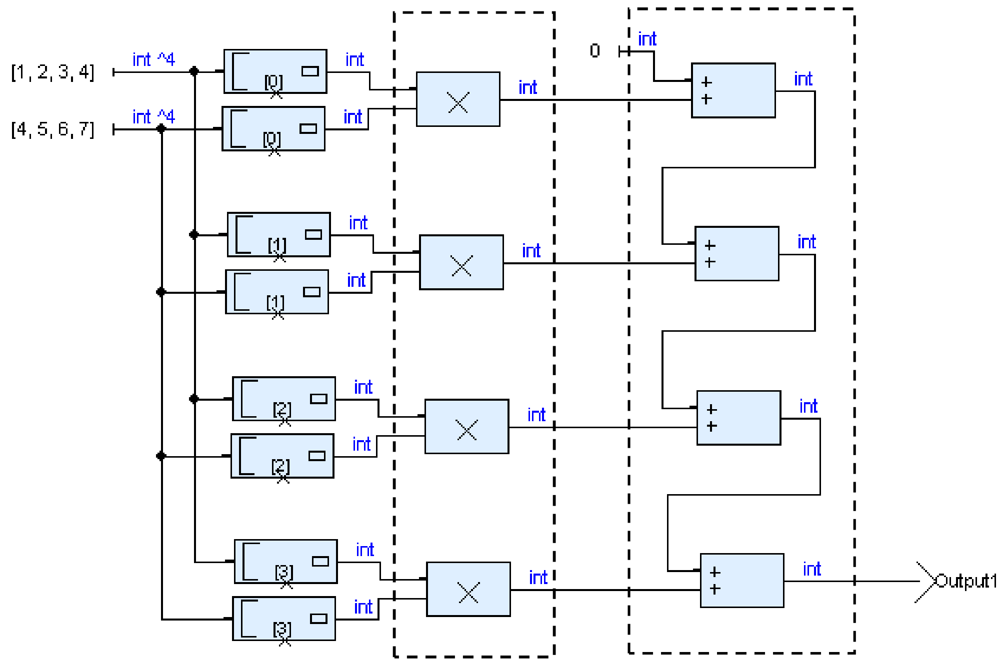
- SCADE 6 provides
  - Native array operators (transpose, mirror, concatenation, slice, …)
  - Regular computation mechanisms on arrays thanks to safe schemes: array iterators

SCADE 6 for array intensive applications

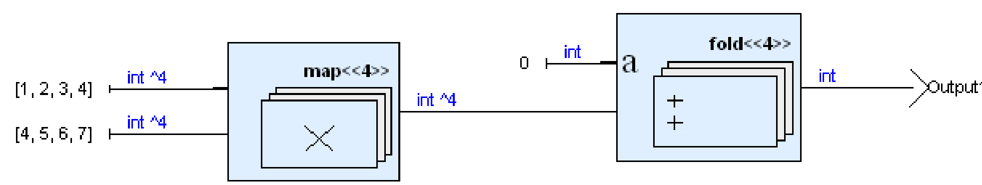
Safe loop (array iterators)

- Optimize the design, while preserving safety
  - Example Scalar Product

Without iterators

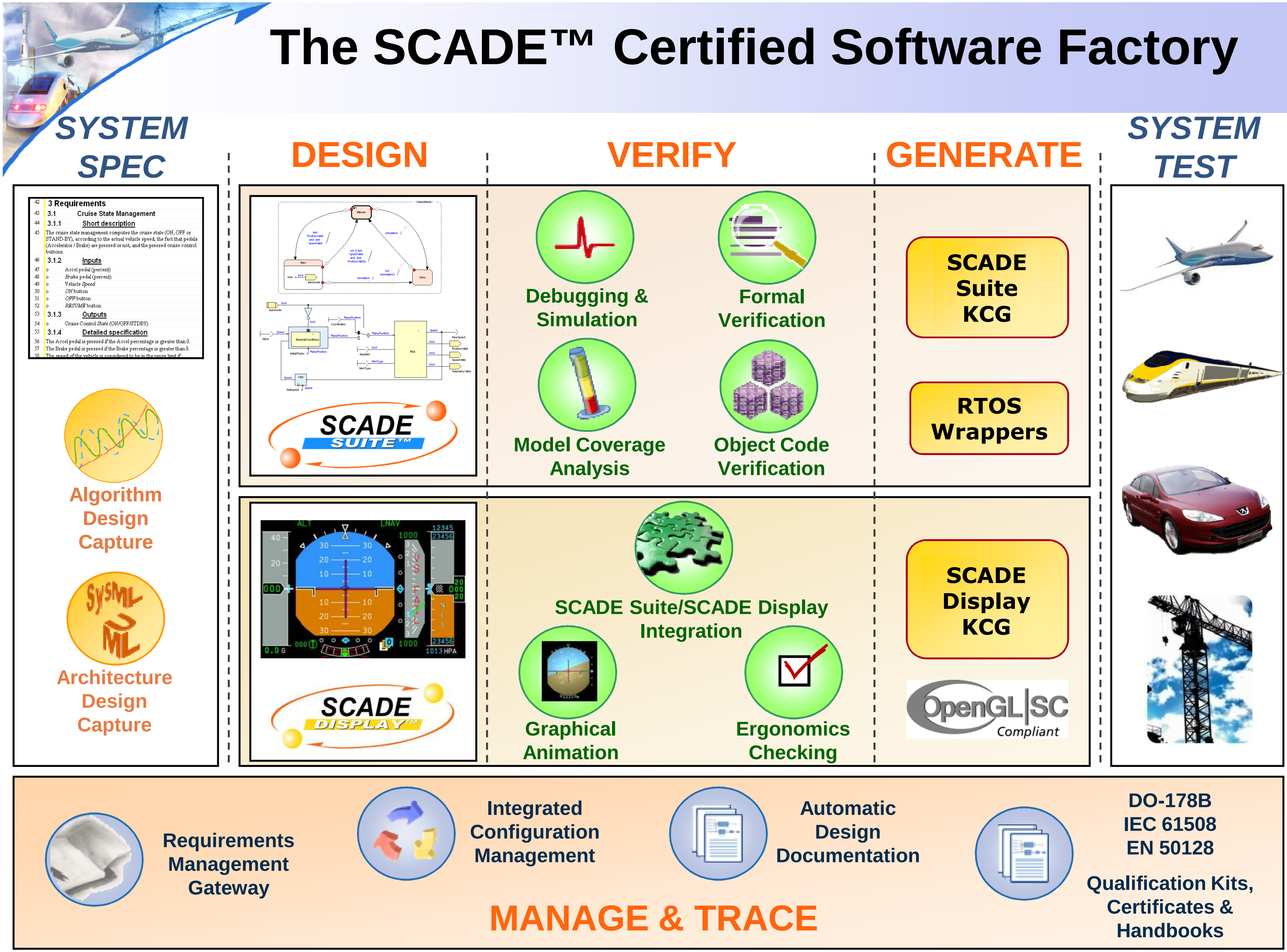


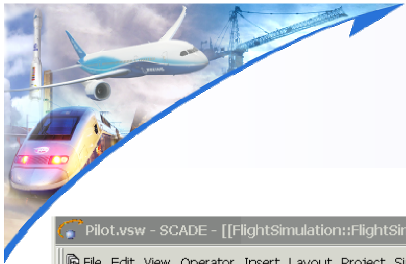
With SCADE 6 iterators



Agenda

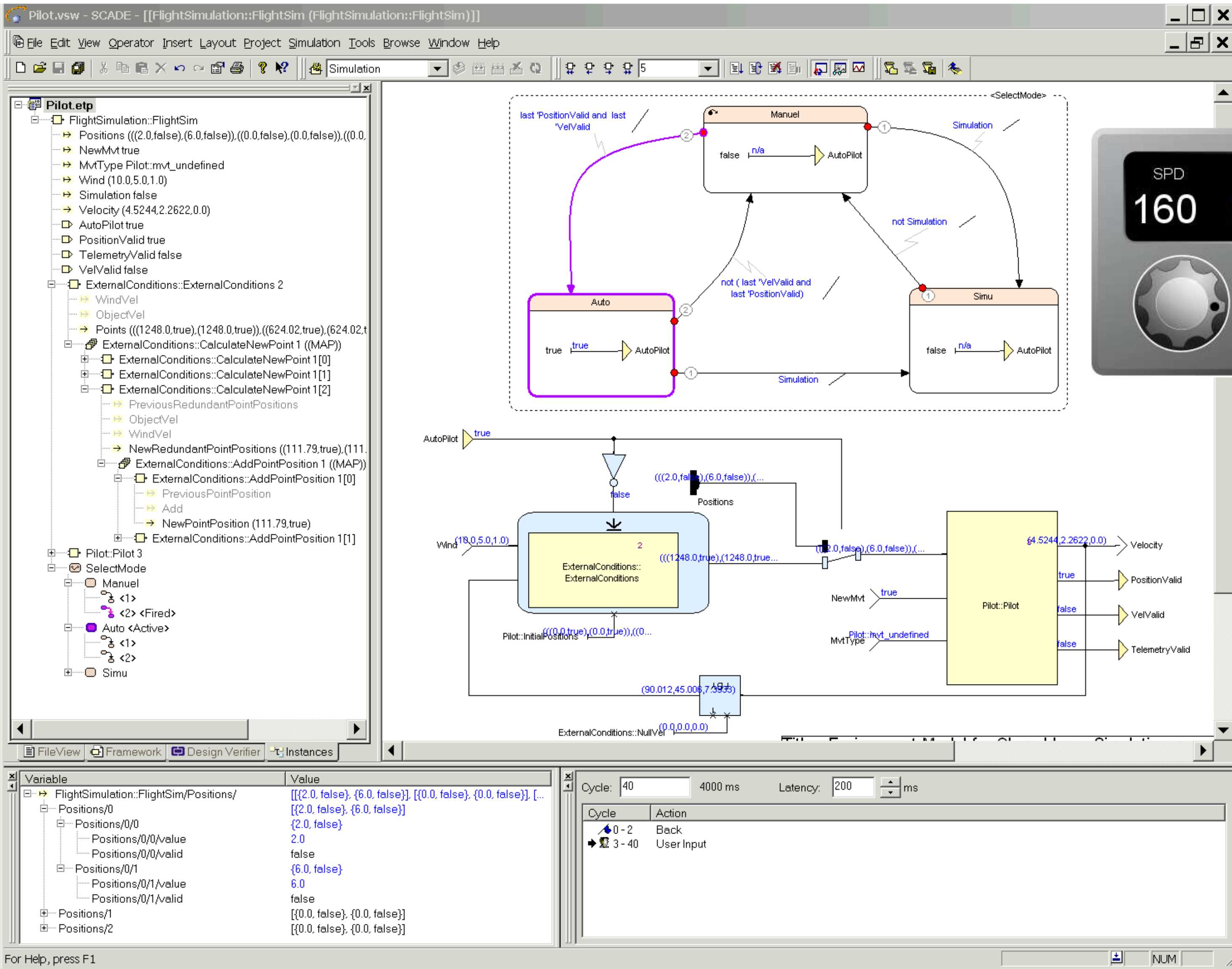
- Esterel Technologies' Introduction
- Objectives of the Presentation
- The SCADE 6 Language
- **The SCADE 6 Toolset**
- Summary of the Benefits of Using SCADE 6


The SCADE™ Certified Software Factory
SYSTEM SPEC
DESIGN
VERIFY
GENERATE
SYSTEM TEST
Algorithm Design Capture
Architecture Design Capture
SCADE SUITE™
SCADE DISPLAY™
Debugging & Simulation
Formal Verification
Model Coverage Analysis
Object Code Verification
SCADE Suite KCG
RTOS Wrappers
SCADE Suite/SCADE Display Integration
Graphical Animation
Ergonomics Checking
SCADE Display KCG
OpenGL SC Compliant
Requirements Management Gateway
Integrated Configuration Management
Automatic Design Documentation
DO-178B
IEC 61508
EN 50128
Qualification Kits, Certificates & Handbooks
MANAGE & TRACE



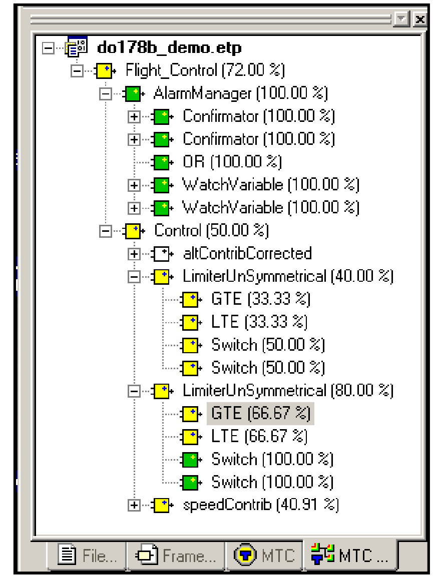
Simulator

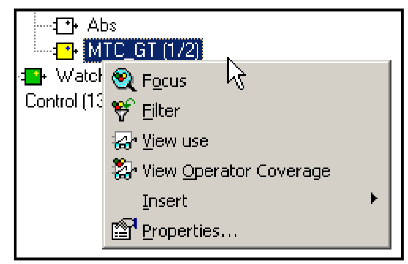
Debugging & Simulation at Model Level



Model Test Coverage (MTC)
Model Coverage Analysis

- Coverage ***Analysis*** at Model level:
  - Enables **requirements-based** tests
  - Shows how **thoroughly** the SCADE model has been **tested**
  - Shows the **role of each test case** in covering operator instances of the SCADE model

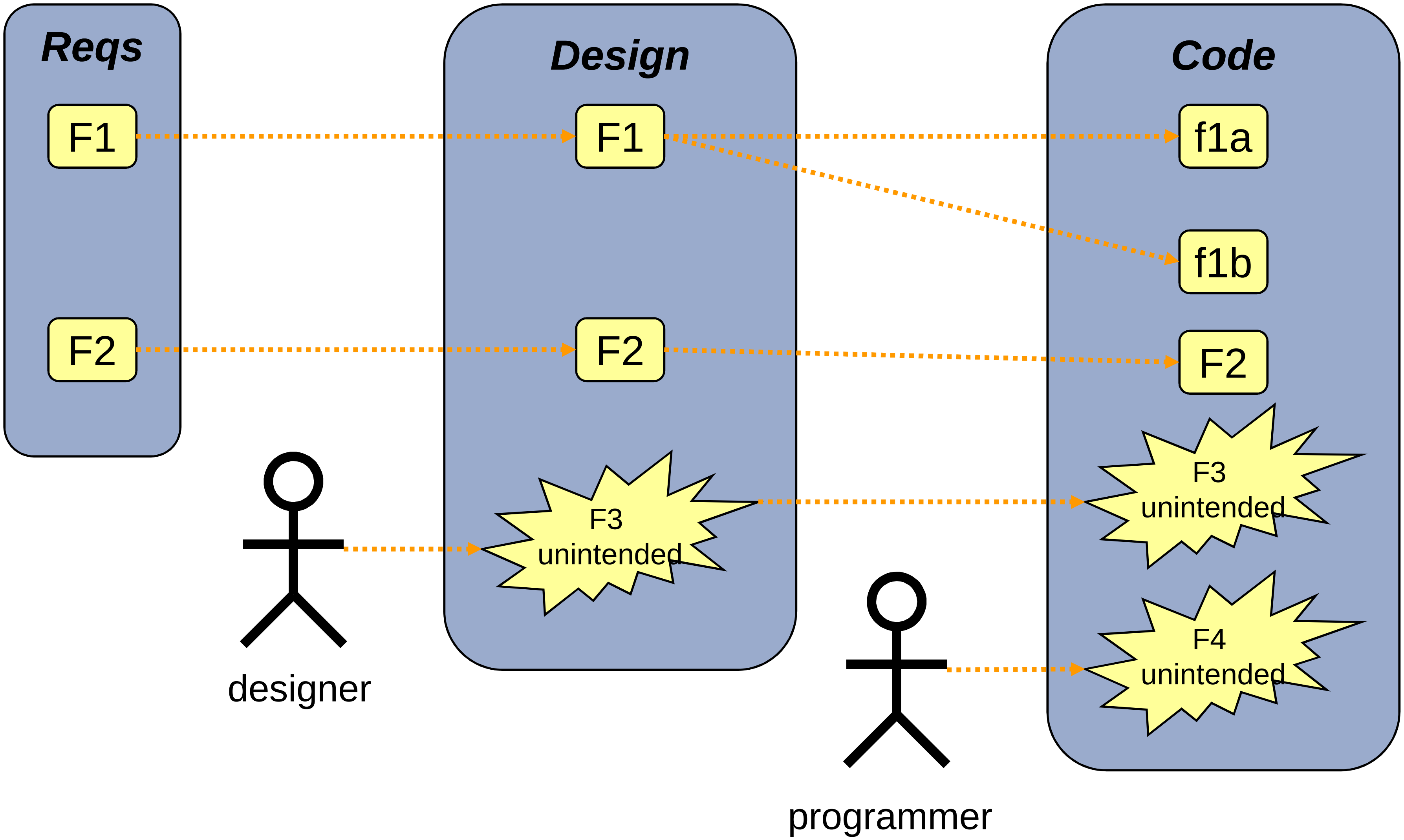




Acquisition Records	Nominal	Reseted	Over Saturated	Under Saturated
PI	X			X
PI_corrected	X		X	X
Pi_justification		X		

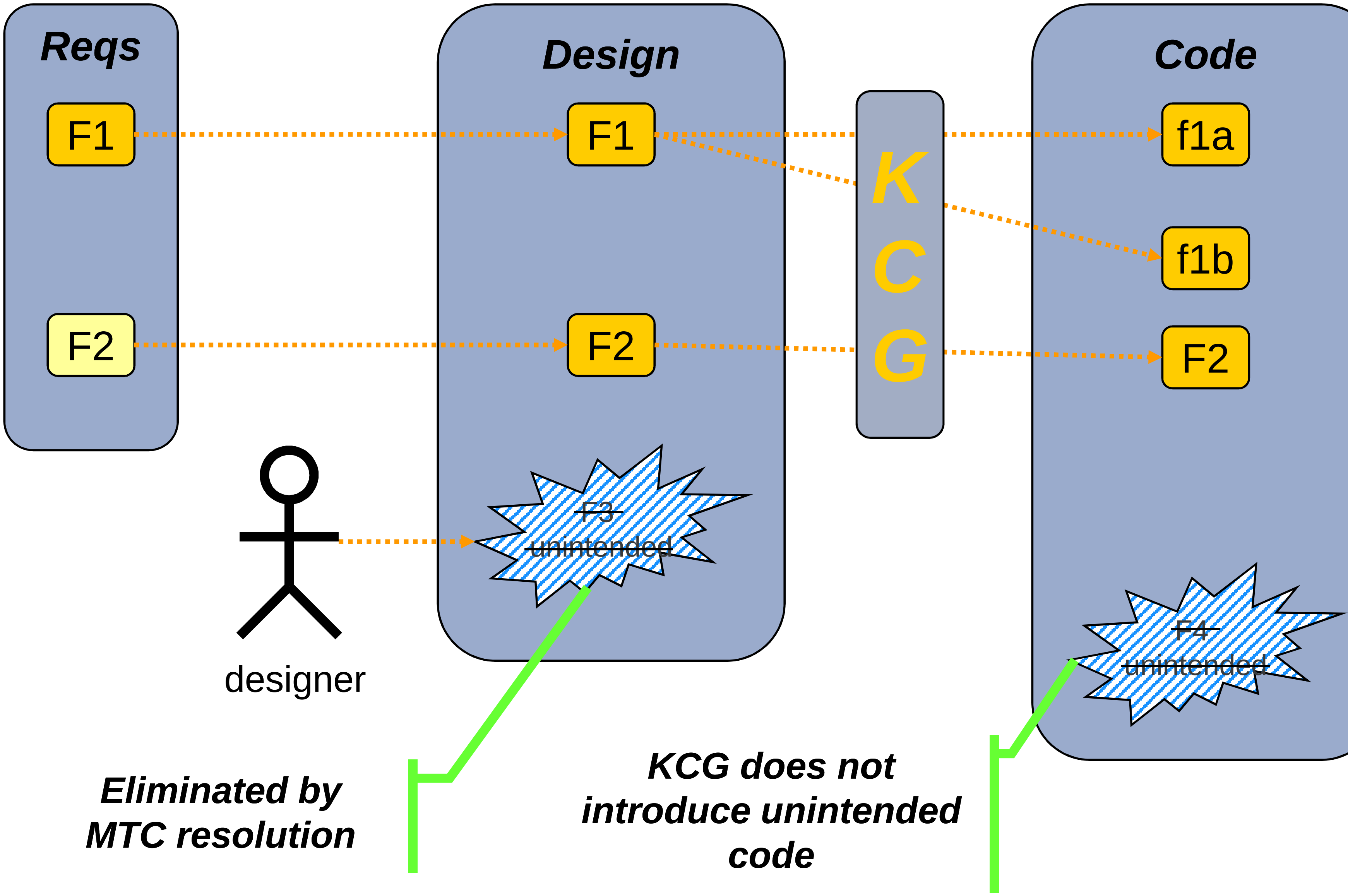
Model Test Coverage (MTC)

Unintended Functions In Traditional Process



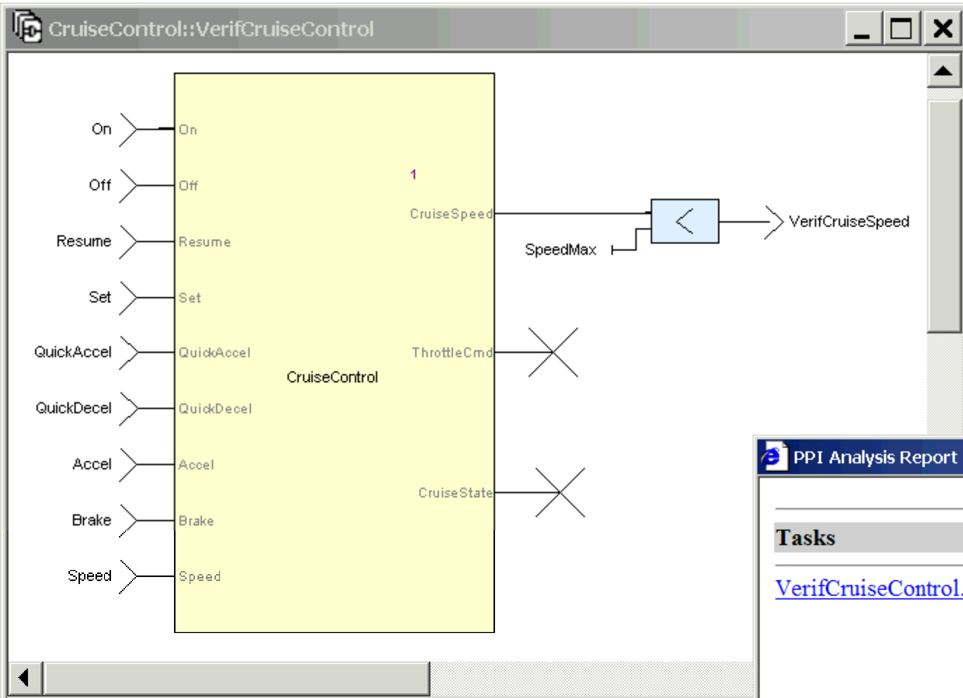
Model Test Coverage (MTC)

Eliminate Unintended Functions with MTC



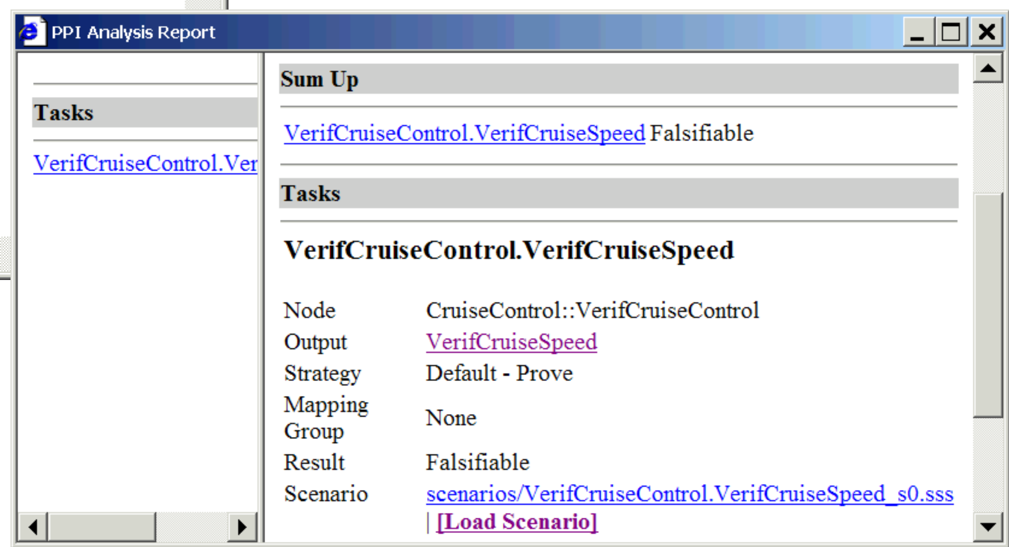
Design Verifier

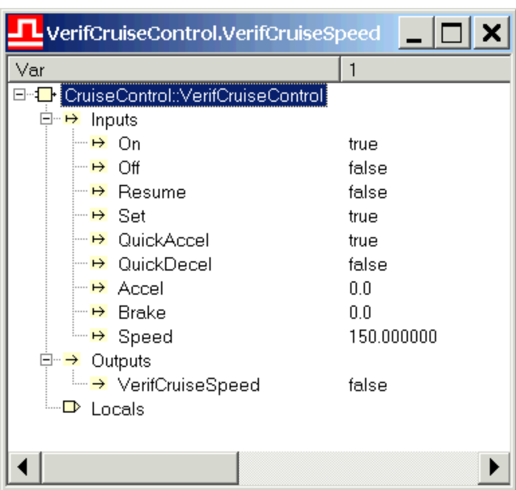
Formal Verification Assistant



Property definition

Design Verifier report





Generated counter example

SCADE KCG

Qualified/Certified Code Generation

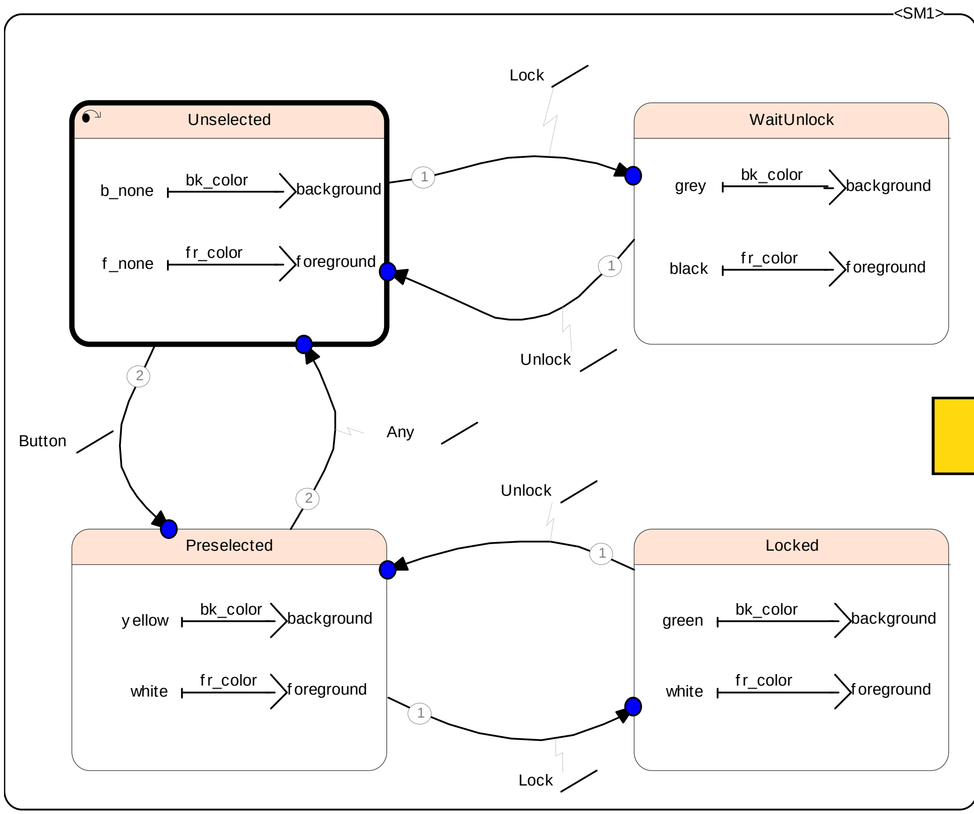
- SCADE KCG was developed according to ***DO-178B***, ***EN 50128 & IEC 61508*** guidelines
  - Qualified as a Development Tool under **DO-178B, up to level A by FAA, Transport Canada & EASA**
  - Certified as a Product under **IEC 61508, up to SIL 3 by TÜV**
  - Certified as a Product under **EN 50128, up to SIL 4 by TÜV**

  - Unambiguous 1-to-1 relationship between SCADE model and the generated C code
  - Complete traceability from model to code

SCADE Suite KCG
Generated Code Properties

- SCADE Suite KCG produces simple C code that fits the constraints of safety-critical embedded software
  - **Portable** (ANSI C, compiler, target and OS independent)
  - MISRA™ compliant
  - **Readable** and **traceable** with respect to the design (name / annotation propagation)
  - Size or speed **optimization** for all constructs (3 levels of optimization)
  - Structured (by functions or by blocks)
  - Static memory allocation
  - No pointer arithmetic
  - No recursion, bounded loops only
  - Bounded execution time

SCADE Suite KCG
Example of SSM code

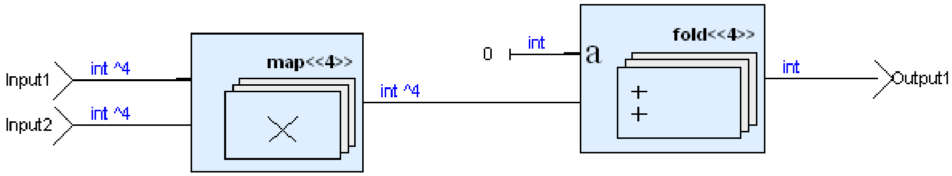



```
[…]
void Button_ABC_N(inC_Button_ABC_N *inC,
   outC_Button_ABC_N *outC)
{
  /* ABC_N::Button::SM1::SSM_SM1_dispatch_sel */
  SSM_Button_SM1_ST SSM_SM1_dispatch_sel;

  if (outC->init)
   {
    outC->init = kcg_false;
    SSM_SM1_dispatch_sel = SSM_SM1_Unselected__ABC_N;
   }
  else
   {
    SSM_SM1_dispatch_sel = outC->M_pre_;
   }
  switch (SSM_SM1_dispatch_sel) {
   case SSM_SM1_Locked__ABC_N :
    outC->foreground = white_ABC_N;
    outC->background = green_ABC_N;
    if (inC->Unlock)
     {
      outC->M_pre_ = SSM_SM1_Preselected__ABC_N;
     }
    else
     {
      outC->M_pre_ = SSM_SM1_Locked__ABC_N;
     }
    break;
   case SSM_SM1_WaitUnlock__ABC_N :
    outC->foreground = black_ABC_N;
    outC->background = grey_ABC_N;
    if (inC->Unlock)
     {
      outC->M_pre_ = SSM_SM1_Unselected__ABC_N;
     }
    else
     {
      outC->M_pre_ = SSM_SM1_WaitUnlock__ABC_N;
     }
    break;
[…]
```

Example of arrayed data flow code

- Generate the code, with safe loops
  - Example Scalar Product



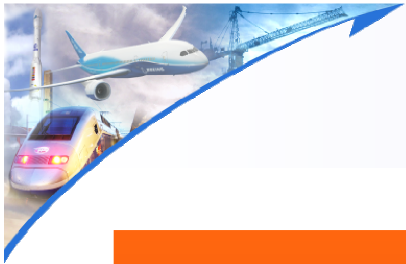

```
kcg_int ScalarProduct6(
  array_1 *Input1 /* ScalarProduct6::Input1 */,
  array_1 *Input2 /* ScalarProduct6::Input2 */)
{
  /* ScalarProduct6::Output1 */
  kcg_int Output1;

  kcg_int i;

  Output1 = 0;
  for (i = 0; i < 4; i++)
    {
      Output1 = Output1 + (*Input1)[i] * (*Input2)[i];
    }
  return Output1;
}
```

Agenda

- Esterel Technologies' Introduction
- Objectives of the Presentation
- The SCADE 6 Language
- The SCADE 6 Toolset
- **Summary of the Benefits of Using SCADE 6**



What we achieved with SCADE 6

Communication

SCADE provides a common representation between systems and software teams sharing models

Certification

SCADE automates critical SW production
FAA & EASA Qualified & TÜV Certified Tools

Portability

SCADE generates portable C or Ada Code which is RTOS, hardware & bus platform independent

Results

SCADE users have experienced a 2X speed-up improvement in time-to-certification and a 37% reduction in project development costs


ESTEREL
Technologies