Realize Your Product Promise®

Model-Based Design
with
SCADE Suite®

Training Resources

Day 1

ANSYS

SCADE Suite Basic Training Program

Day 2

1. Type declaration
2. Constant declaration
3. Operator and interface declaration
4. Control flow declaration

Day 3

1. Arrays
2. Iterators
3. Code Generation
4. Imported Code

© 2013 ANSYS, Inc.

© Esterel Technologies - An ISO 9001:2008 Certified Company - Confidential & Proprietary

SCADE Suite Basic Training Program

Day 4

1. Simulation
2. Generated Code
3. Integration

And the modules selected by trainees…

© 2013 ANSYS, Inc.

© Esterel Technologies - An ISO 9001:2008 Certified Company - Confidential & Proprietary

AGENDA

What is a Critical Real-Time Embedded Software

Formal Model Driven Engineering

SCADE Suite Basic Concepts

© 2016 ANSYS, Inc.

Brainstorming

Exercise

Objective:

Define Critical Real Time Embedded Software

Preparation time: 5 min

Requirements:

Create group of 2/3 people

Each group select one of the following terms to explain:

- Safety
- Critical
- Real-Time
- Embedded
- Determinism

Each group presents to others its ideas

© 2016 ANSYS, Inc.

What is an Embedded System?

Wikipedia

An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints

Examples

© 2016 ANSYS, Inc.

What is Real Time?

Transformational systems

- Inputs available on execution start
- Outputs delivered on execution end

e.g. Mathematical computation

Interactive systems

- Interact with the environment
- Have subjective speed requirements

e.g. Windows, Powerpoint

Reactive systems

- Interact with the environment
- Have **objective** speed requirements

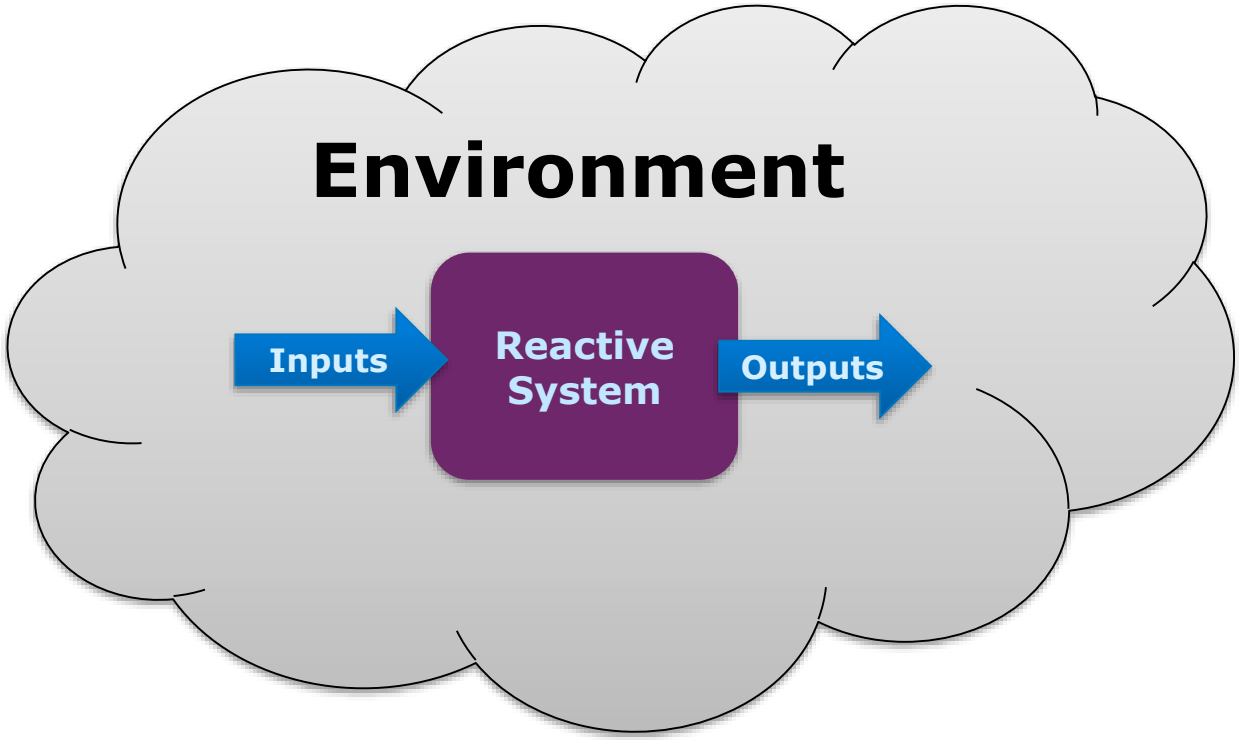
e.g. Control / Command of a spacecraft

© 2016 ANSYS, Inc.

Reactive Systems

… interact with the environment:

- Receive inputs
- Provide outputs



© 2016 ANSYS, Inc.

Determinism

The same cause implies the same effect

Key requirements for safety-embedded systems:

- The same sequence of inputs always produces the same sequence of outputs
- This implies a reproducible behavior

Not a strong requirement for non-critical systems:

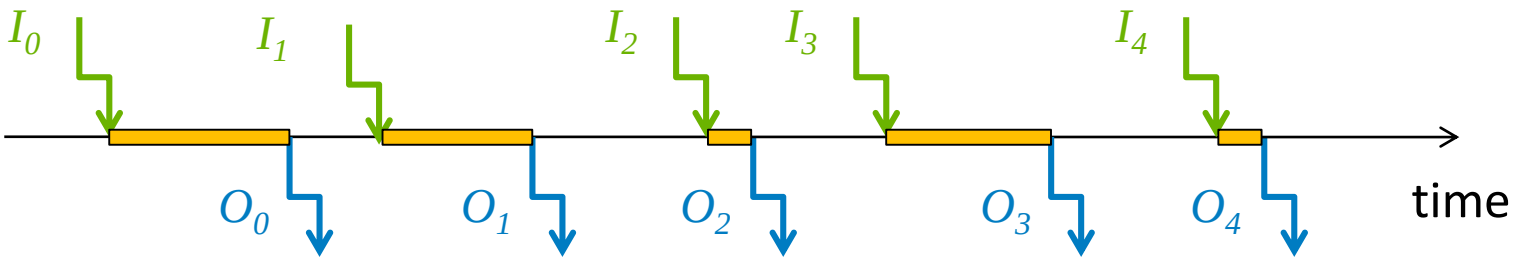
- Characteristic of interactive systems (e.g. OS, Internet)

© 2016 ANSYS, Inc.

Reactive Systems have Time Constraints

Receive inputs

The outputs shall be produced before the reception of the next input



Provide outputs

Time to compute the output

© 2016 ANSYS, Inc.

What is Critical?

Intuitively, a critical system is a system in which failure can have severe impacts:

- Nuclear
- Aeronautic
- Automotive
- Railway
- Space
- Military
- Medical
- …

© 2016 ANSYS, Inc.

Software Criticality Levels

Standards define precisely software criticality levels:

- DO-178C for airborne systems
- EN-50126/EN-50128, for railway applications
- IEC-61508, applied in the industry
- ISO-26262, for road vehicles
- etc.

© 2016 ANSYS, Inc.

DO-178C Criticality Levels

Severity	Consequence
Catastrophic	Failure conditions which would prevent continued safe flight and landing
Hazardous / Severe-Major	Failure conditions which would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions to the extent that there would be: • a large reduction in safety margins or functional capabilities, • physical distress or higher workload such that the flight crew could not be relied on to perform their tasks accurately or completely, or • adverse effects on occupants including serious or potentially fatal injuries to a small number of those occupants
Major	Failure conditions which would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions to the extent that there would be, for example, a significant reduction in safety margins or functional capabilities, a significant increase in crew workload or in conditions impairing crew efficiency, or discomfort to occupants, possibly including injuries
Minor	Failure conditions which would not significantly reduce aircraft safety, and which would involve crew actions that are well within their capabilities. Minor failure conditions may include, for example, a slight reduction in safety margins or functional capabilities, a slight
No effect	Failure conditions which do not affect the operational capability of the aircraft or increase crew workload

© 2016 ANSYS, Inc.

Safety and Security

Wikipedia

Safety

Safety is the state of being "safe", the condition of being protected against […] consequences of failure, damage, error, accidents, harm or any other event which could be considered non-desirable. It can include protection of people or possessions

Security

Security is the degree of resistance to, or protection from, harm. It applies to any vulnerable and valuable asset […]

© 2016 ANSYS, Inc.

Safety and Security
in Software Engineering

Safety

The software must not harm the world

Security

The world must not harm the software

AGENDA

What is a Critical Real-Time Embedded Software

Formal Model Driven Engineering

SCADE Suite Basic Concepts

© 2016 ANSYS, Inc.

Brainstorming

Which issue do you know related to the C language (and similar)?

Time: 10 min

Syntax

Communication

Type

…

Exercise

© 2016 ANSYS, Inc.

Have an Unambiguous Statement Syntax

Example

C, C++

```
if ( light == red );
{
    Cancel_lift_off()
    ;
}
```

Legal statement, No warning

The call to Cancel_lift_off is always executed

© 2016 ANSYS, Inc.

Have an Unambiguous Statement Syntax

Example

C, C++

```
if ( light == red );
{
    Cancel_lift_off()
    ;
}
```

Legal statement, No warning

The call to Cancel_lift_off is always executed

Need more verifications to avoid this error

Ada

```
if  light = red then;
    Cancel_lift_off;
end if;
```

Illegal statement, No compilation

© 2016 ANSYS, Inc.

Have a Clearly Named Notation (1/2)

Example

C, C++

```
struct date {
        int day, month, year;
};
```


© 2016 ANSYS, Inc.

Have a Clearly Named Notation (2/2)

Example

C, C++

What does it mean?

```
struct date today = {12, 1, 5};
```

We need to see the structure definition to understand the settings

Ada

```
type Date is record
      Day, Month, Year : Integer; end record;

Today: Date := ( Day => 12, Month => 1, Year => 5 );
```


© 2016 ANSYS, Inc.

Formal Methods Concept

Aim: Add mathematical reasoning in the system development flow

Goal: Specify better, reduce verification costs, increase system reliability

How: Provide concise formalism for specification and implementation

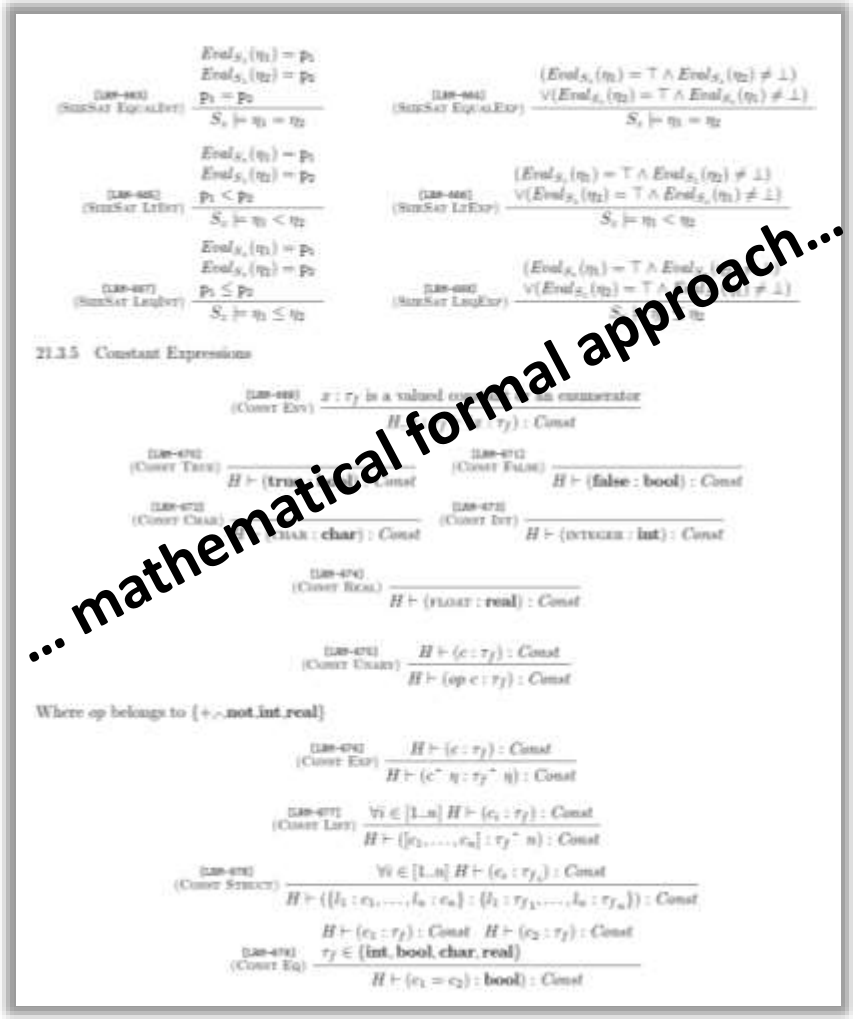
Challenge: Enable smooth integration in engineering framework

Better specification reduces verification!

© 2016 ANSYS, Inc.

The Challenge

From a…



21.3.5 Constant Expressions

$$\frac{x : \tau_f \text{ is a valued (...) in enumerator} \quad H \vdash \tau_f : Const}{H \vdash (x : \tau_f) : Const} \text{ (Const Env)}$$

$$\frac{}{H \vdash (\mathbf{true} : \mathbf{bool}) : Const} \text{ (Const True)} \qquad \frac{}{H \vdash (\mathbf{false} : \mathbf{bool}) : Const} \text{ (Const False)}$$

$$\frac{}{H \vdash (\text{CHAR} : \mathbf{char}) : Const} \text{ (Const Char)} \qquad \frac{}{H \vdash (\text{INTEGER} : \mathbf{int}) : Const} \text{ (Const Int)}$$

$$\frac{}{H \vdash (\text{FLOAT} : \mathbf{real}) : Const} \text{ (Const Real)}$$

$$\frac{H \vdash (c : \tau_f) : Const}{H \vdash (op\ c : \tau_f) : Const} \text{ (Const Unary)}$$

Where op belongs to {+, -, not, int, real}

$$\frac{H \vdash (c : \tau_f) : Const}{H \vdash (c^{\wedge} \eta : \tau_f^{\wedge} \eta) : Const} \text{ (Const Exp)}$$

$$\frac{\forall i \in [1..n]\ H \vdash (c_i : \tau_f) : Const}{H \vdash ([c_1, \ldots, c_n] : \tau_f^{\wedge} n) : Const} \text{ (Const List)}$$

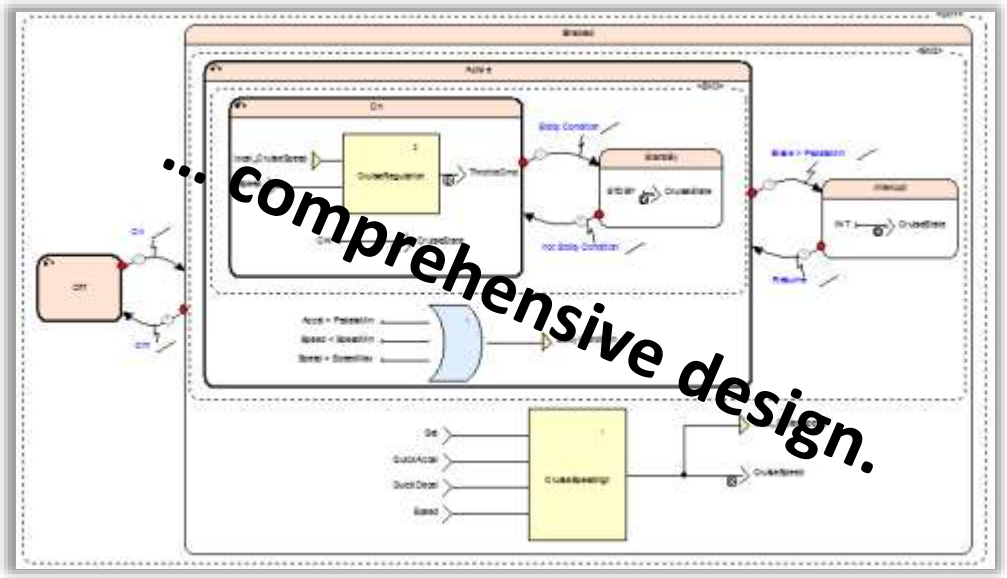
$$\frac{\forall i \in [1..n]\ H \vdash (c_i : \tau_{f_i}) : Const}{H \vdash (\{l_1 : c_1, \ldots, l_n : c_n\} : \{l_1 : \tau_{f_1}, \ldots, l_n : \tau_{f_n}\}) : Const} \text{ (Const Struct)}$$

$$\frac{H \vdash (c_1 : \tau_f) : Const \quad H \vdash (c_2 : \tau_f) : Const \quad \tau_f \in \{\mathbf{int}, \mathbf{bool}, \mathbf{char}, \mathbf{real}\}}{H \vdash (c_1 = c_2) : \mathbf{bool}) : Const} \text{ (Const Eq)}$$

Scade language formal definition (extract of SCADE Suite KCG SRS)

… mathematical formal approach…

To a…



… comprehensive design.

A SCADE Suite model

Formal Language Syntax

A good syntax shall be

Clear

Unambiguous

Intuitive

Another Simple Example

Example

Ada

```
case State is
  when State1 => Guard1 := X < 3; Guard2 := X > 3;
   if (EVENT1 and (Guard1 or Guard2)) then
        if (Guard1) then
            X := 5;
            State := State2;
        else
            if (Guard2) then
               X := 6;
            end if;
            State := State3;
        end if;
   end if;
  when State2 =>
     if (EVENT1) then
        X := 7;
        State := State1;
     end if;
 when State3 =>
     if (EVENT1 and EVENT1) then
        X := 8;
        State := State1;
     end if;
End case
```

Simple? Yes…

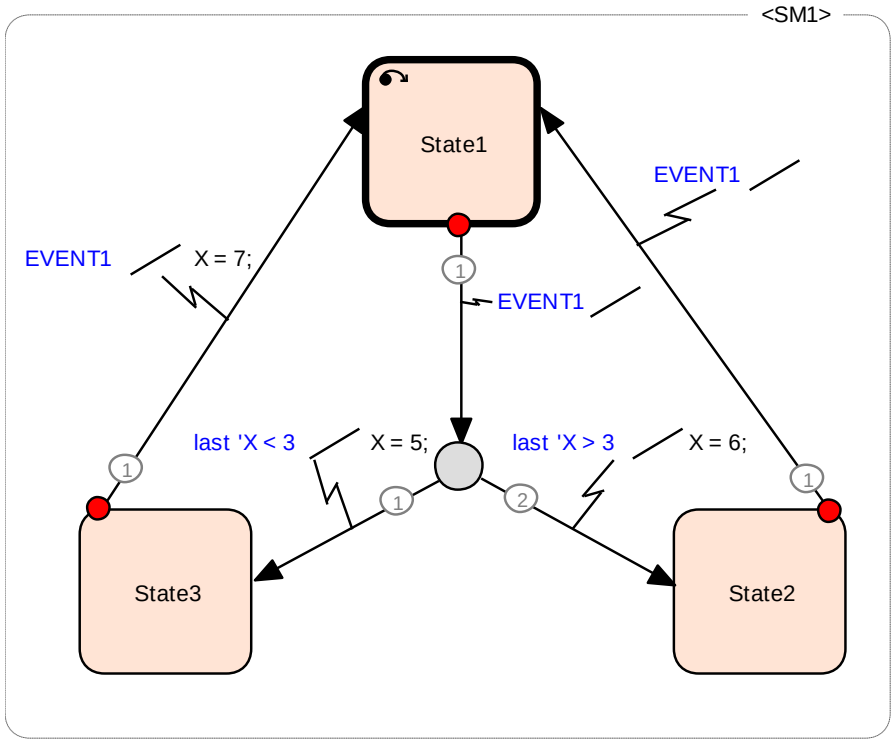
But what does this piece of code do?

Code alone is not sufficient for communication

© 2016 ANSYS, Inc.

Same Example Graphically

Example



A graphical language with a high level of abstraction facilitates the communication.

© 2016 ANSYS, Inc.

What is Model-Based Software Engineering?

SCADE® is a Model-based software engineering solution

Model-based software engineering means: **use diagrams instead of code**

More abstraction when defining the behavior and data of an application:

- Details are hidden or slipped to further refinements
- Independent from target language

© 2016 ANSYS, Inc.

What is Model-Based Software Engineering?

But it is more than just drawing models, **it is a complete methodology**, consisting of:

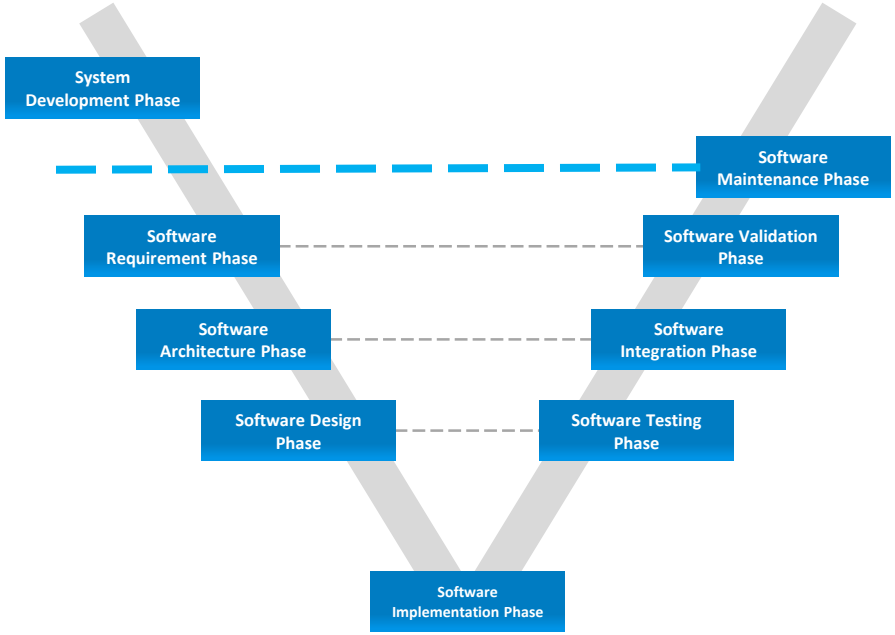
- Processes: Define the "What": a logical rationale that is performed to meet an objective
  - Examples: requirement analysis, model-based design, verification, configuration management, etc.
- Practices: Define the "How": a set of techniques for performing a process
  - Examples: prototyping, modeling, simulation, versioning, teamwork practices,…
- Tools: Means that facilitate the practices
  - Examples: SCADE®, UML, Synergy®, etc.

© 2016 ANSYS, Inc.

What is Software Lifecycle?

The Software life cycle brings rationale and scheduling among a Model-Based process. It is a division of software development work into distinct phases

Among various representations, the V-cycle representation comes with the advantage to show both a sequencing and a correspondence between design phases and V&V phases



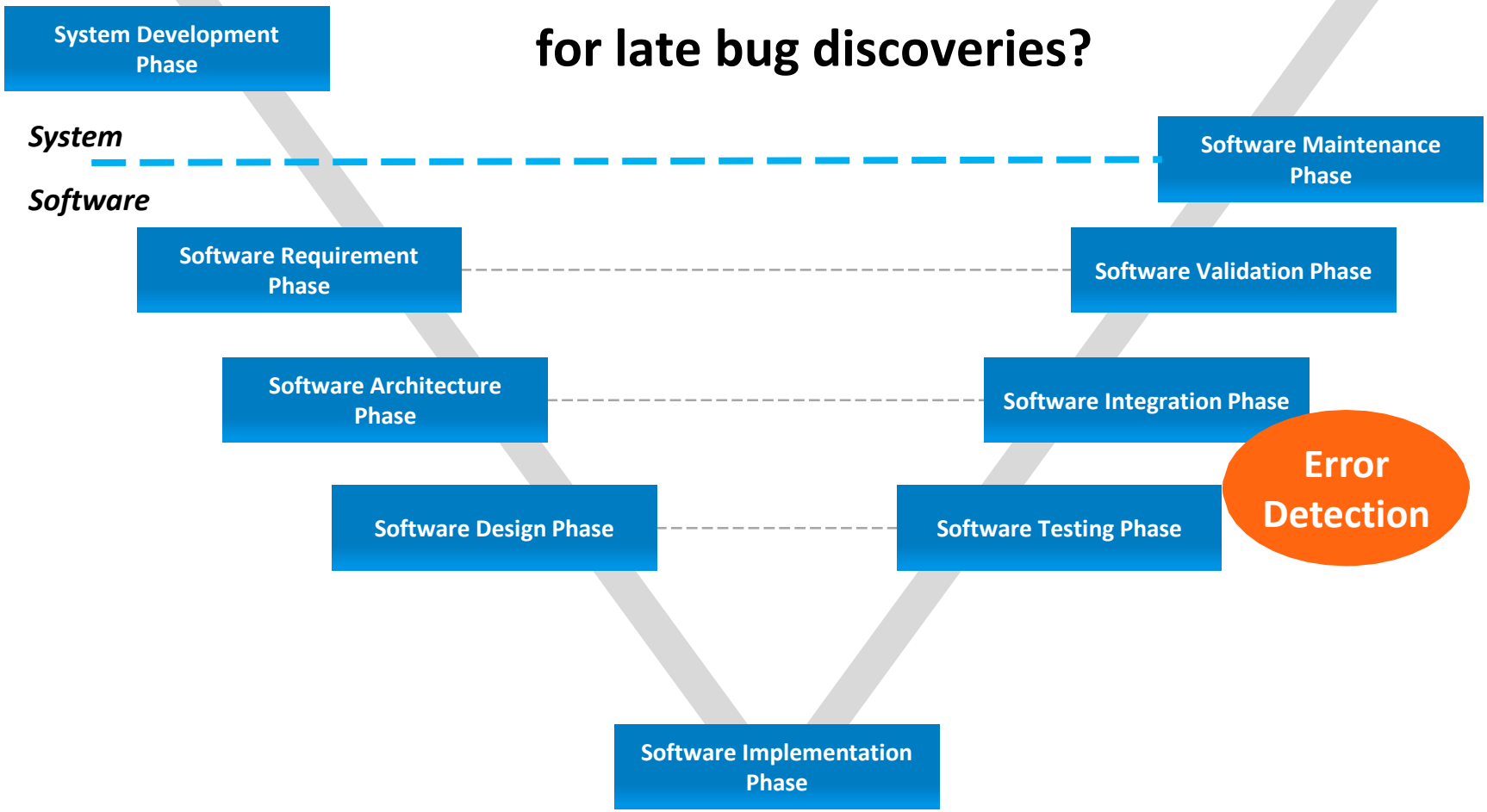
© 2016 ANSYS, Inc.

Brainstorming

Exercise

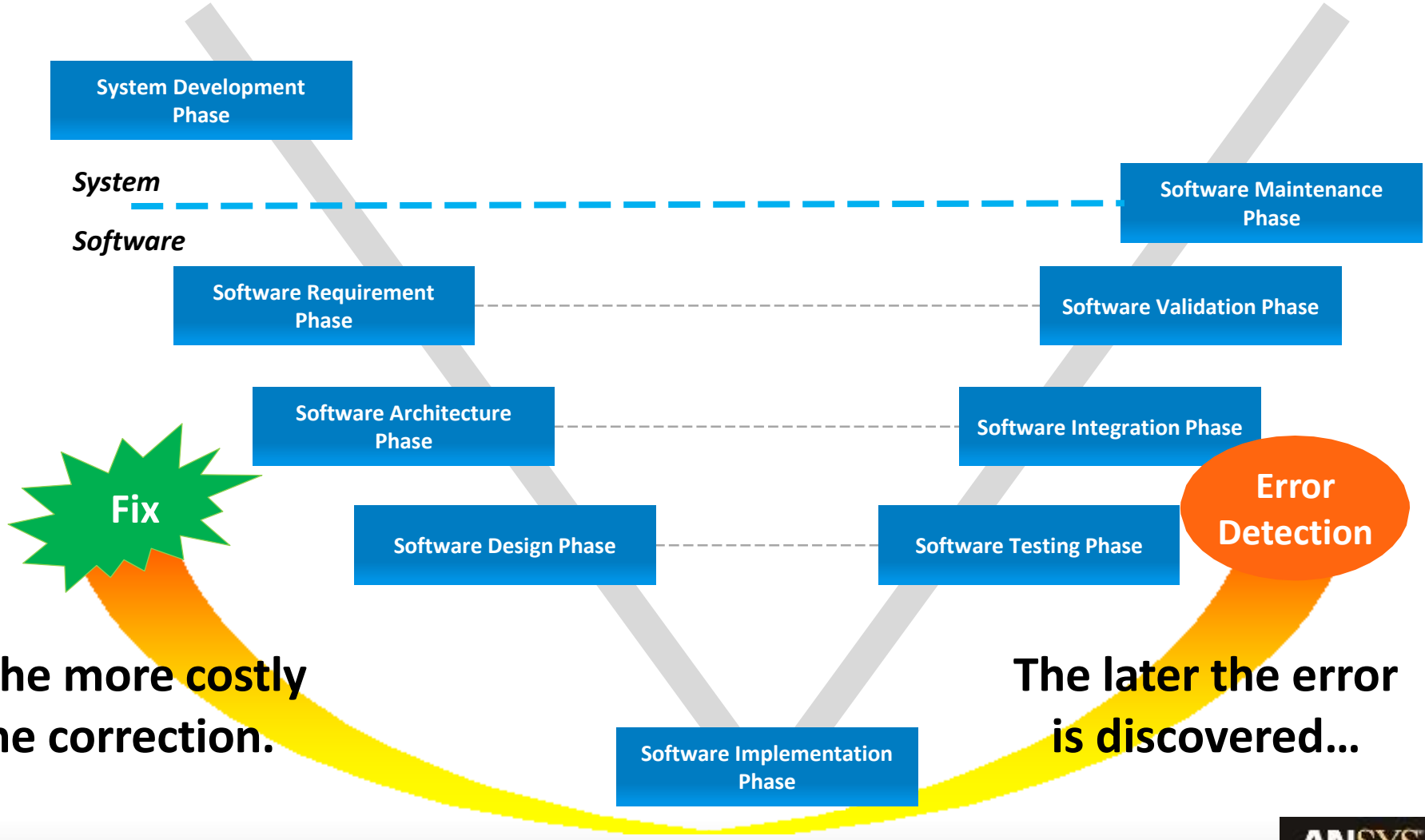
Time: 10 min

What are the issues
for late bug discoveries?



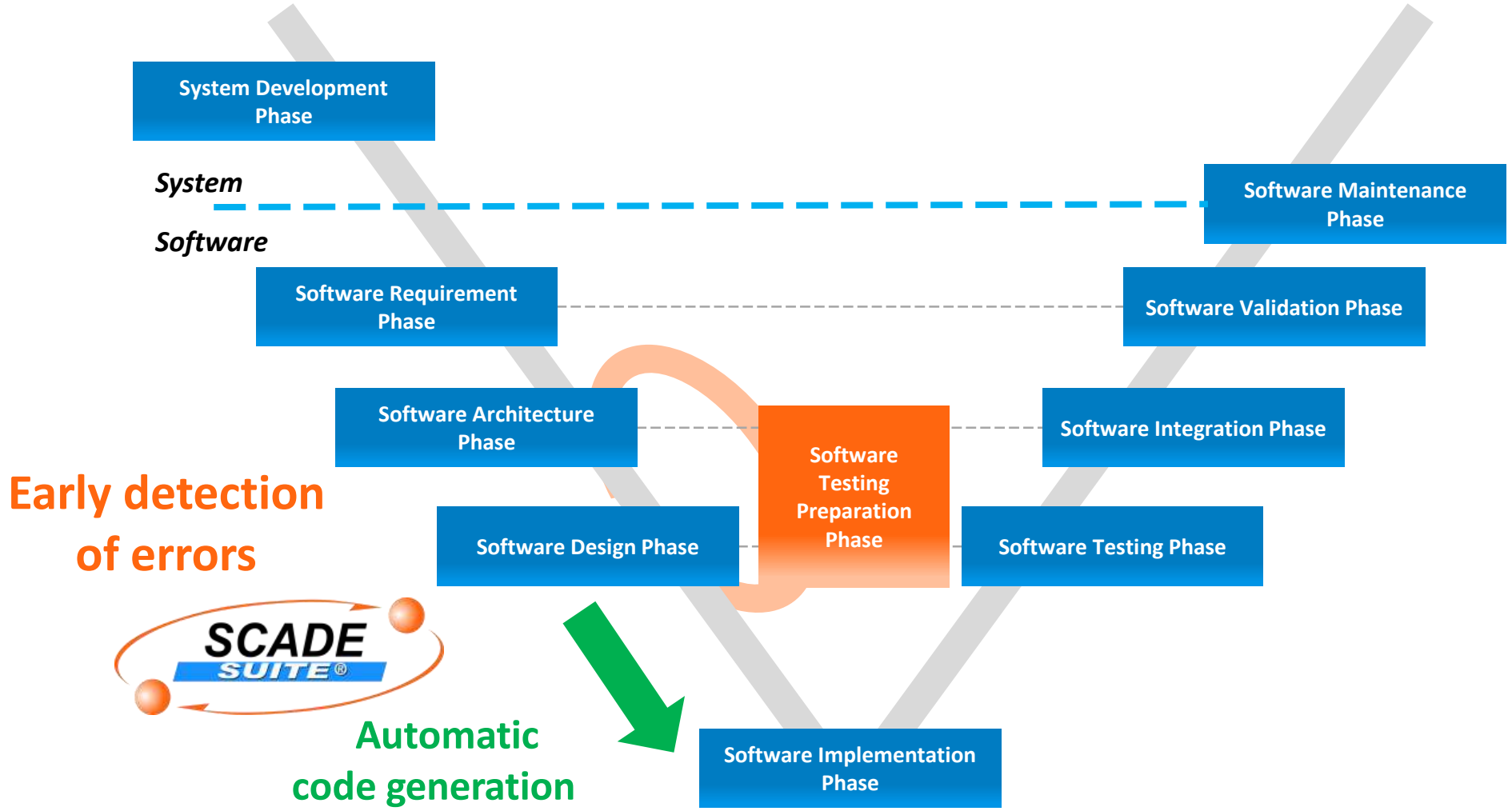
© 2016 ANSYS, Inc.

Cost of Error Detection



© 2016 ANSYS, Inc.

SCADE-Based Software Life Cycle



© 2016 ANSYS, Inc.

Other Software Process Issues and SCADE Model-Based Solution

Communication

- Model-based design as a common graphical language
- Documentation automatically produced and synchronized from formal models

Maintainability and Knowledge Capitalization

- Model maintenance versus code maintenance (faster and at a higher level of abstraction)
- Simulation and Executable specifications increase understanding
- Model-based development shortens the ramp-up of new designers
- Constant management of requirements
- Fully integrated tool chain

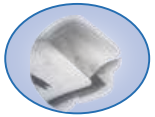
Fully Integrated Design Suite

Debugging & Simulation

Suite / Display Integration

Automatic Design Documentation

Integrated Configuration Management



Requirements Management Gateway

ANSYS

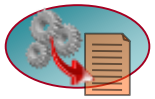
© 2016 ANSYS, Inc.

Other Software Process Issues and SCADE Model-Based Solution

Code Quality and Safety

- Formal language with qualified code generation

Formal Language

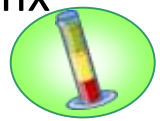


Qualified Code Generation

Discovery of bugs late in the process

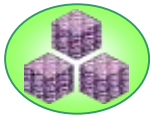
- Automatic verification and simulation enable the detection of design flaws early when they are cheaper to fix

Design Checking



Model Coverage Analysis

Formal Verification



Object Code Verification

Cost of V&V

- Low-level testing is suppressed due to code generation qualification credits
- Verification and simulation reduce costs

DO-178C

Certification Kit, Certificates & Handbooks

Certification Cost and Risk Mitigation

- Certification kits provided and SCADE process well known to certification authorities

© 2016 ANSYS, Inc.

AGENDA

What is a Critical Real-Time Embedded Software

Formal Model Driven Engineering

SCADE Suite Basic Concepts

© 2016 ANSYS, Inc.

History of Synchronous Languages

Created in the early 80's from a joint research effort by control engineering and computer sciences researchers

Lustre

P. Caspi – N. Halbwachs
(VERIMAG)

Signal

A. Benveniste – P. Le Guernic
(IRISA)

Esterel

JP Rigault, JP Marmorat – G. Berry
(INRIA/Ecole des Mines)

From Research to Industry

Mid-80's 90's

- SCHNEIDER Electric develops SAGA as the industrial version of Lustre for control command software for nuclear plants
- AIRBUS independently develops SAO, a formalism close to Lustre
- Verilog (then Telelogic) industrializes SCADE in collaboration with SCHNEIDER and AIRBUS
- SCADE 2.1 is released in 1996

Photo courtesy of AIRBUS

- In 1999, first Qualification for FCS (EC 135)
- In 1999, Esterel Technologies is created
- In 2001, Esterel Technologies acquires SCADE from Telelogic
- Feb 2002, second Qualification for FCS (A340/600)
- 2007: Esterel creates the Scade 6 language merging the best of Lustre and Esterel for safety critical embedded software development
- In 2009, SCADE Suite 6 and SCADE Display 6 integrated to form the Esterel SCADE

SCADE Suite was Created for Safety

Scade language (Lustre) is formally defined with key safety objectives:

- The language is very simple and stable, forbidding dangerous constructs (e.g. unbounded loops, wild goto's, dynamic memory allocation,…)
- Interpretation of a model does not depend on the readers or their environment

Very active research work for more than 20 years:

- Worldwide visibility in the safety critical embedded software field

Designed with close connections with certification authorities in the aeronautics & nuclear energy domains:

- SCADE Suite KCG is a C and Ada code generator developed with stringent qualification objectives (DO-178B/C DAL A, IEC 61508 SIL3, EN 50128 SIL3/4, ISO 26262 ASIL D)

© 2016 ANSYS, Inc.

Safety Critical

Errors lead to dramatic consequences involving human lives and huge costs.

The Synchronous approach helps the Safety-critical application development by:

Providing rigorous design methods

Providing formally defined languages and analyses

© 2016 ANSYS, Inc.

Semantics of Synchronous Hypothesis

Existence of a discrete clock:

- Software cyclically activated,
- Inputs read at the cycle beginning (no inputs changes during the cycle)
- No cycle overlap
- Outputs delivered at cycle end

$\Rightarrow$ The cycle execution duration is considered to be null

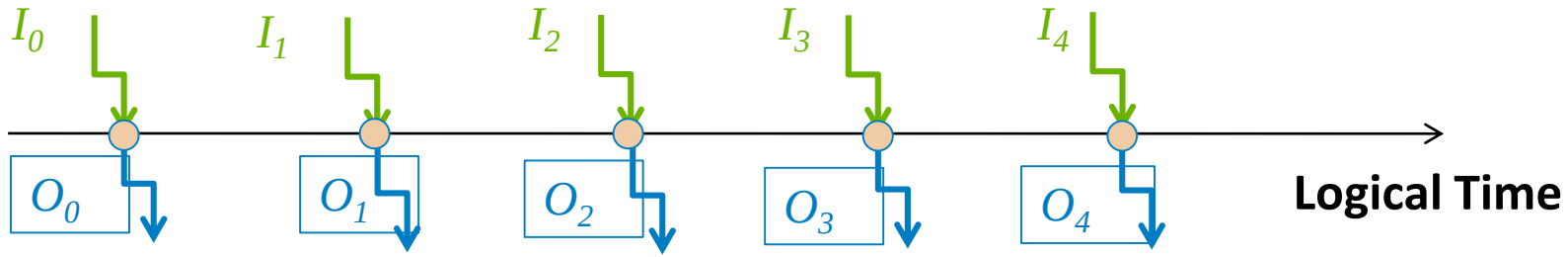
$\Rightarrow$ Reasoning is possible

© 2016 ANSYS, Inc.

Synchronous Time Model

Example

Time as a logical notion



Period of cycle

In theory, execution time is null…

… but how to assume this hypothesis?

© 2016 ANSYS, Inc.

Synchronous Language Implementation

No interaction between the program and the world during a computation cycle ➔ strong theoretical and practical properties:

- Inputs must not change during the execution cycle: using a double buffer or a write protection or any other architecture
- Internal variables and outputs are frozen as soon as they are computed during the execution cycle:
  - Once computed, they are never modified later during the cycle,
  - They only have one value at a given cycle

➔**Fully deterministic behavior, finite computation time**

➔**Much easier to verify than asynchronous programs**

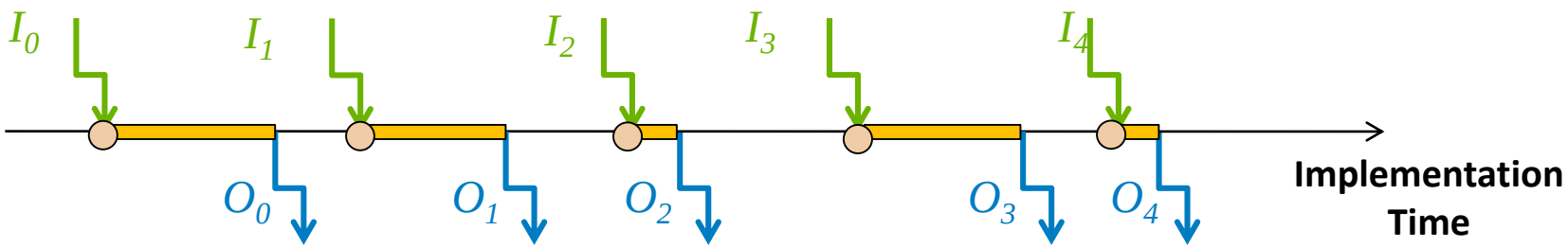
© 2016 ANSYS, Inc.

The Synchronous Time Implementation

Example

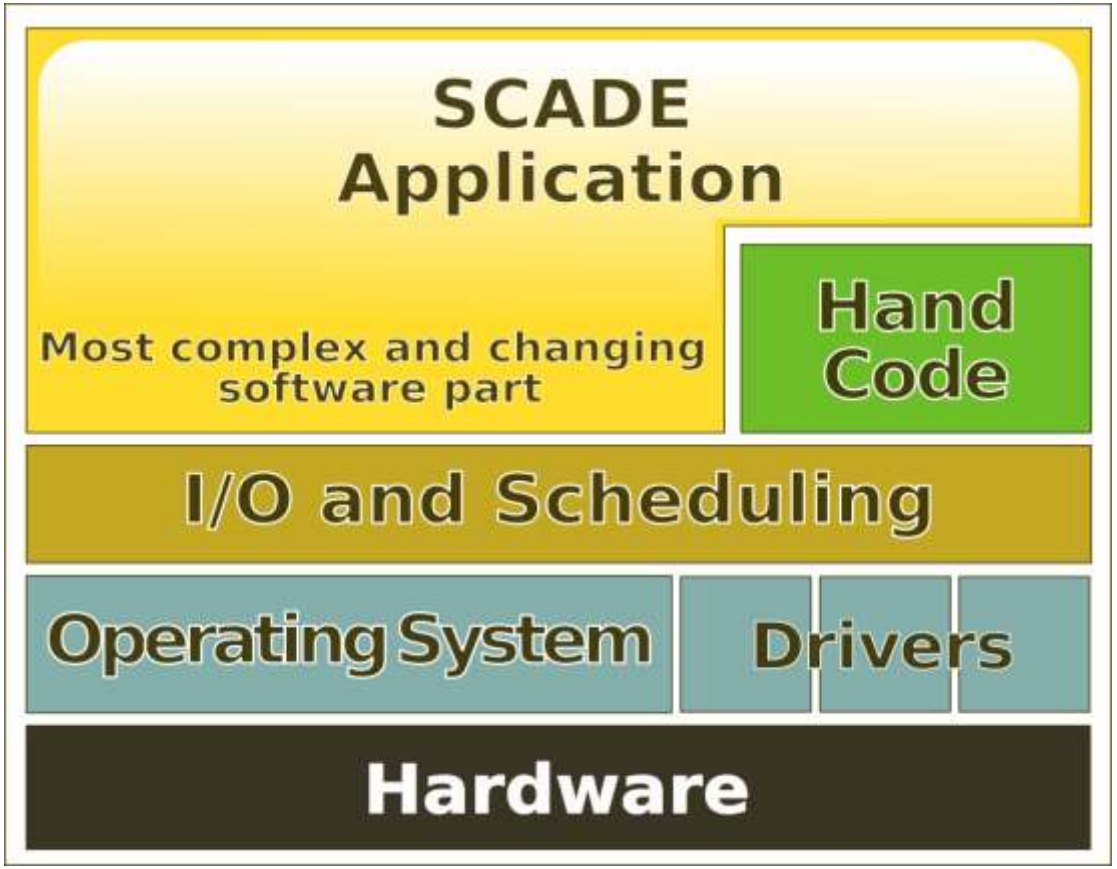
Finite computation time = WCET = guarantee of no-overlap

- *Period of cycle*



© 2016 ANSYS, Inc.

SCADE Suite addresses the Application Part



The application part is the largest and most frequently/lately changing part

© 2016 ANSYS, Inc.

Scade Fields of Application

Control algorithm:

- Performs data processing tasks using complex mathematics
- Updates actuator's outputs

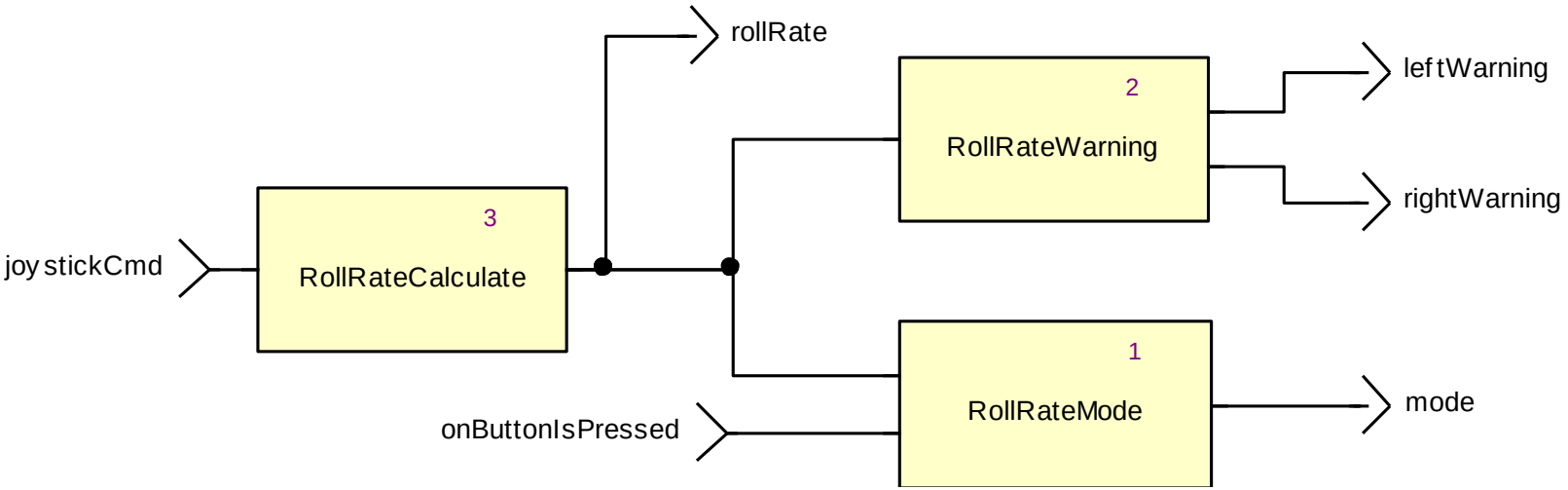
State or mode management:

- Reacts from external or internal events: sensors, alarms, threshold detection
- Performs combinational logic operations to compute the new system's state
- Updates actuator's outputs

© 2016 ANSYS, Inc.

Scade Language

A Simple Network of Operators

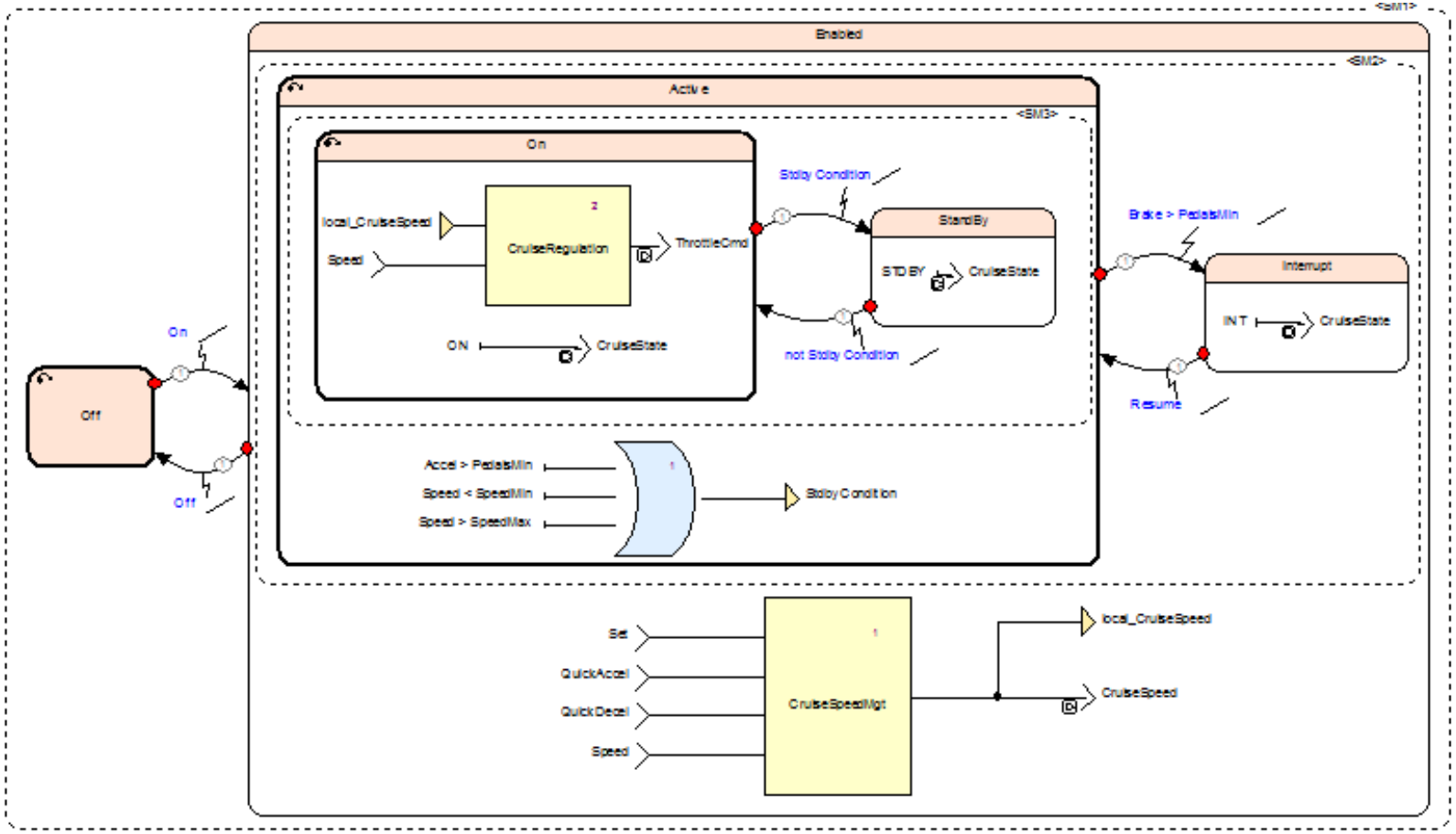


© 2016 ANSYS, Inc.

Scade Language

Control Flow and State Machines

Hierarchical state machines, that can be mixed with data-flow



© 2016 ANSYS, Inc.

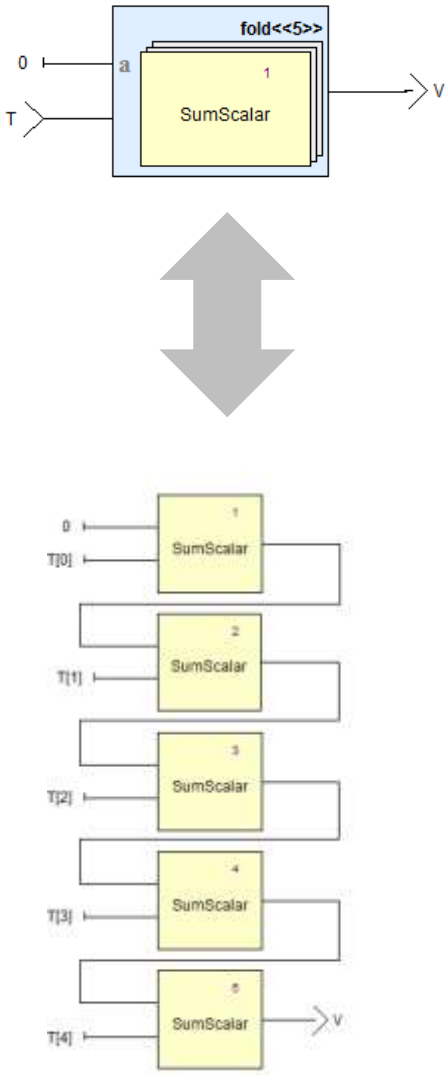
Scade Language

Arrays and Iterators

Scade provides array data types and iterators

An iterator applies an operator over arrays

This optimizes the design and the code while preserving the safety (bounded size)



© 2016 ANSYS, Inc.

Scade Language

Properties

Parallelism is naturally expressed

The computation order is based on dependencies (not on the diagrams' layout or on hidden properties):

- Each data element is computed before it is used, once and only once

The generated code is efficient and deterministic (no tasking overhead, no deadlock, no race condition)

Execution time is finite

© 2016 ANSYS, Inc.

SCADE Suite KCG

A Qualifiable / Certified Code Generator

Ensures that the input model complies with language syntax and semantics

Generates C or Ada code fitting safety-critical constraints

KCG is deterministic:

- A given input model and a given set of options always produce the same generated code

Qualified for DO-178B/DO-178C level A

Certified for:

- EN 50128 up to SIL 3/4
- IEC 61508 up to SIL 3
- with compliance for IEC 60880
- ISO 26262 up to ASIL D

ANSYS

© 2016 ANSYS, Inc.

Summary

Scade Language

Is dedicated to safety-critical embedded application development:

- Designed with academics, users and authorities

Mixes data-flow and control-flow constructs

Has a graphical notation understandable by any engineer

Is supported by the SCADE Suite model-based environment

Is used to produce embeddable code generated by the KCG qualified/certified code generator

© 2016 ANSYS, Inc.

SCADE Suite Basics

© 2016 ANSYS, Inc.

AGENDA

Integrated Development Environment

SCADE Suite Operator

Simulation introduction

Code generation and standalone executable

Reports

© 2016 ANSYS, Inc.

Editor Overview (1/2)

The SCADE Suite workspace includes various windows and views for:

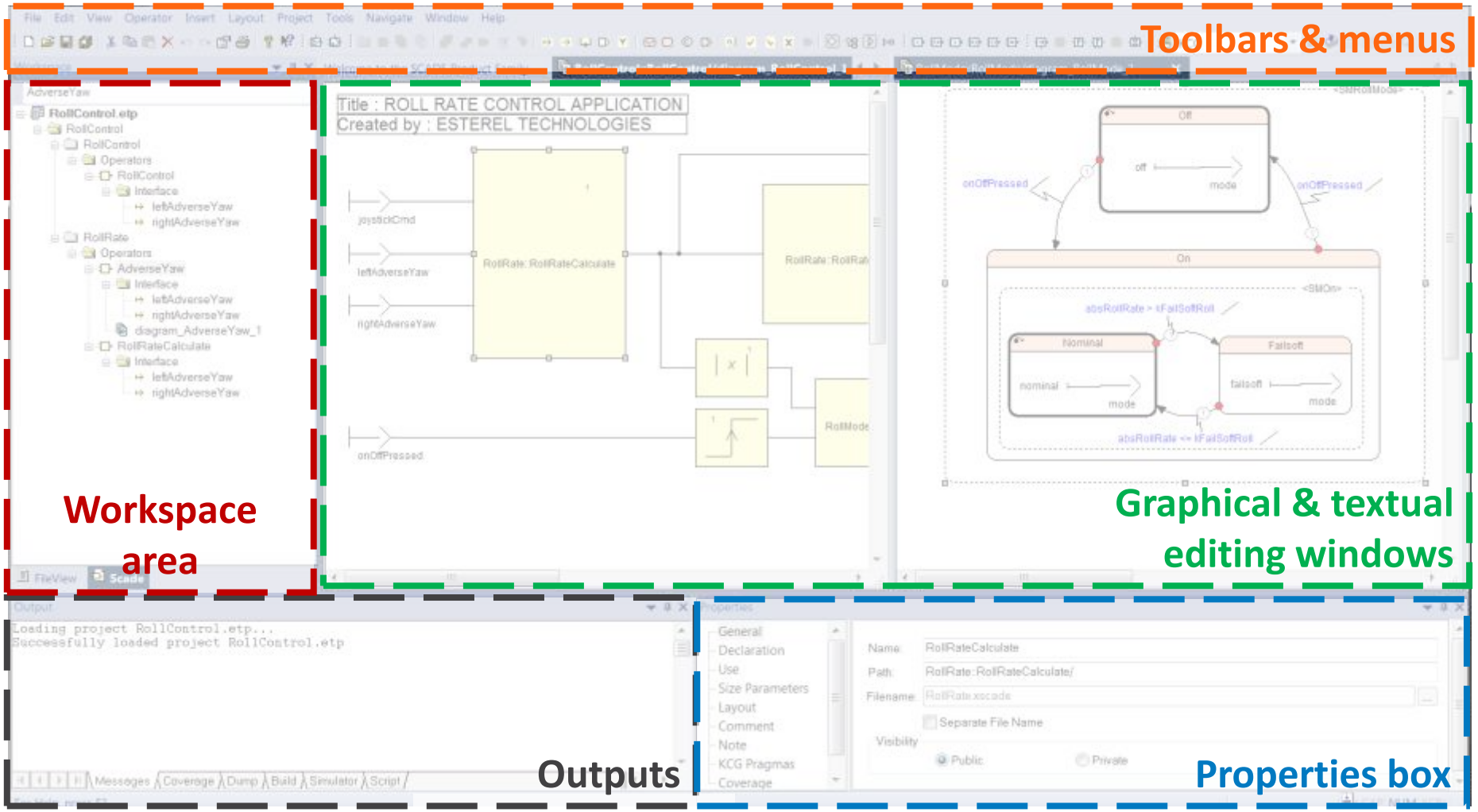
- Representing and editing functional models
- Setting integrated tools
- Displaying model information
- Checking status and results
- Adding traceability

The environment is customizable by the user with:

- Standard menus
- Dockable toolbars
- Different windows and tabs such as diagrams

© 2016 ANSYS, Inc.

Editor Overview (2/2)



© 2016 ANSYS, Inc.

ANSYS Confidential

SCADE Suite Workspace and Project

A workspace:

- Represents your work environment
- Contains the SCADE Suite projects
- Is necessary to be able to work in the SCADE Suite editor

A project:

- Contains your design information (constants, types, operators, …)
- Can be reused as a library in any other project

© 2016 ANSYS, Inc.

Project Files

A SCADE Suite project contains main following files:

.vsw: workspace file (with one or several projects)

.etp: project description file

.xscade: Scade graphical model file

.scade: Scade textual model file

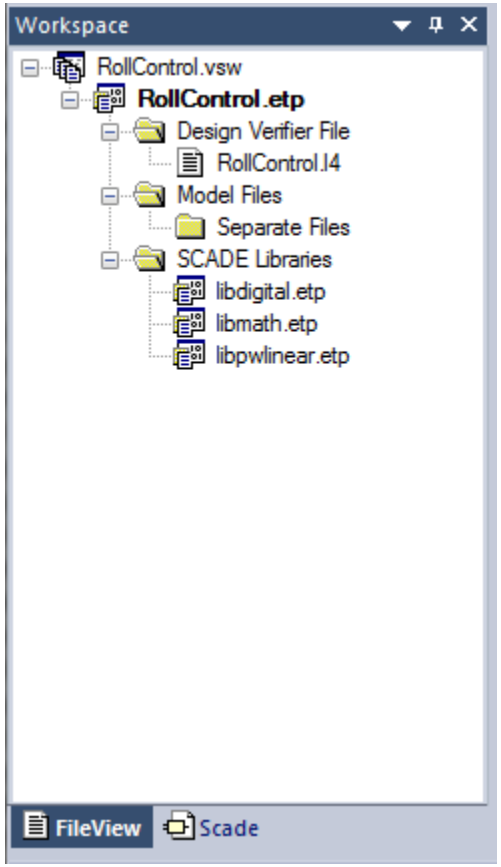
.ewo: user preferences associated with the project workspace

© 2016 ANSYS, Inc.

File View

The File View displays:

- SCADE Suite projects (RollControl.etp)
- Model files
- Project folders
- SCADE Suite project libraries
- Any user libraries imported into models



You can create virtual folders to classify your files:

- Right click on project > Insert Folder …

© 2016 ANSYS, Inc.

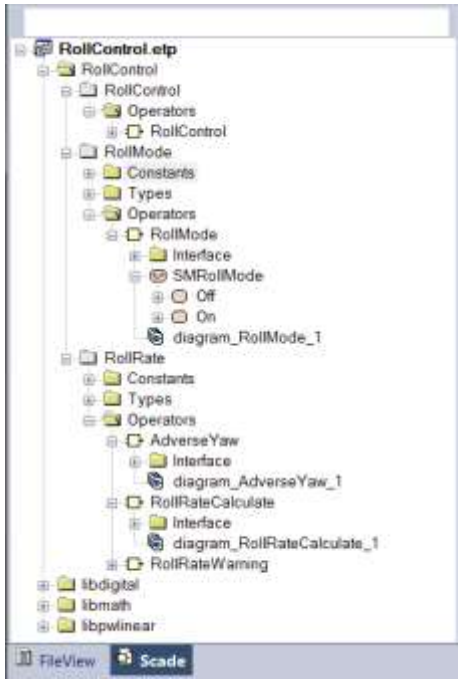
Scade View

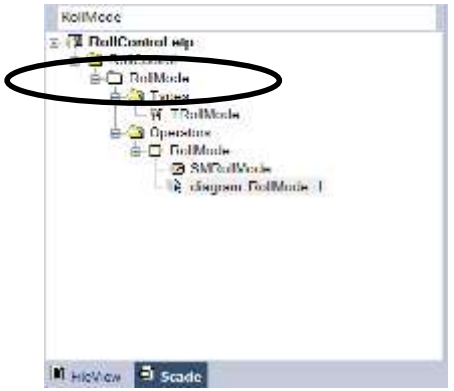
The Scade View shows the objects as declared in the project:

- Constants, Sensors and Types
- Operators (functions) with their:
  - Input / Output interface
  - Local variables
  - State machines
- Included libraries
- Possibility to group objects within a Package

Users can drag & drop objects to edit your design

Users can filter the tree display





© 2016 ANSYS, Inc.

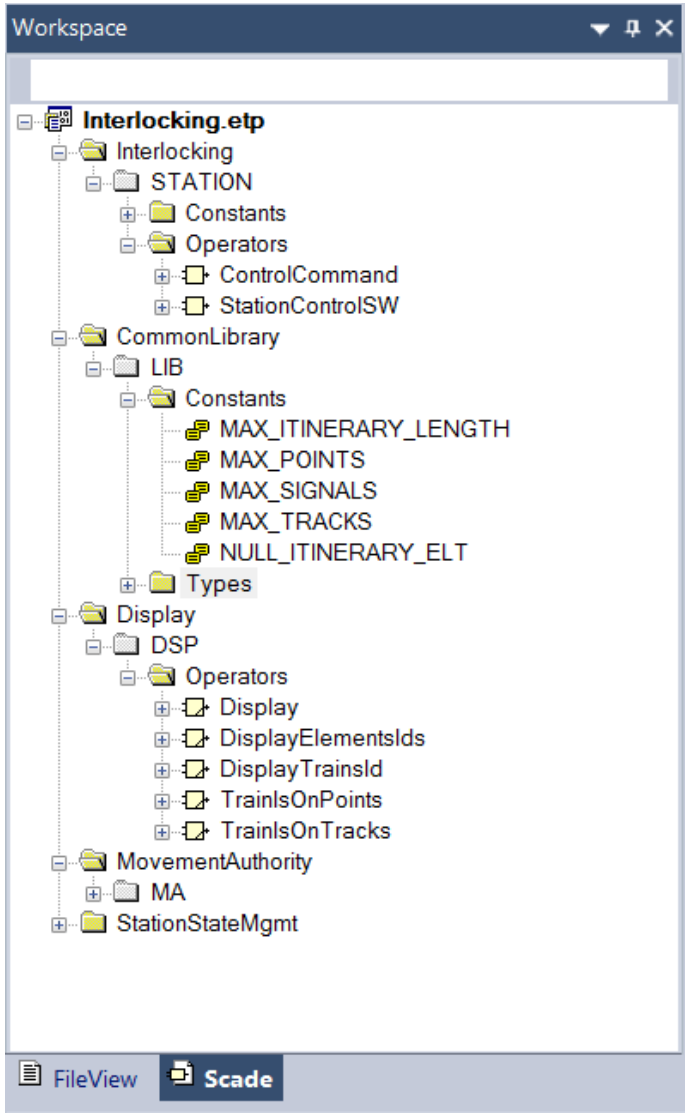
Libraries in SCADE Suite

Any SCADE Suite model can be a library

Any SCADE Suite model can use library components

Using libraries structures your SCADE Suite projects:

- Create your own library components
- Test them
- And reuse them in many projects



© 2016 ANSYS, Inc.

SCADE Suite Help

The user documentation is located in *<Installdir>\help\SCADE Suite Help Resources:*

- Getting Started
- Gateway Guidelines: VeriStand, RTOS, Simulink
- Manuals: SCADE Language Primer, tutorial, reference Manual, SCADE Suite libraries, User and Technical manuals…

Other documents are located in *<Installdir>\help\Common Help Resources*

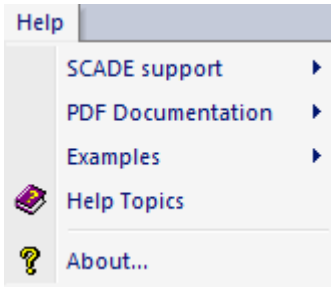
- Design Standard: SCADE Suite Software Development Standard
- Installation Guidelines
- Reference Cards (SCADE products glossary)
- Release Info (access to all SCADE Release Notes)
- Welcome (release content for all SCADE products)
- …

© 2016 ANSYS, Inc.

SCADE Suite Help

Access from Help menu:

- PDF Documentation
- SCADE Suite Model Examples
- Interactive Help Topics



PDF Documentation

Interactive Help Topics

© 2016 ANSYS, Inc.

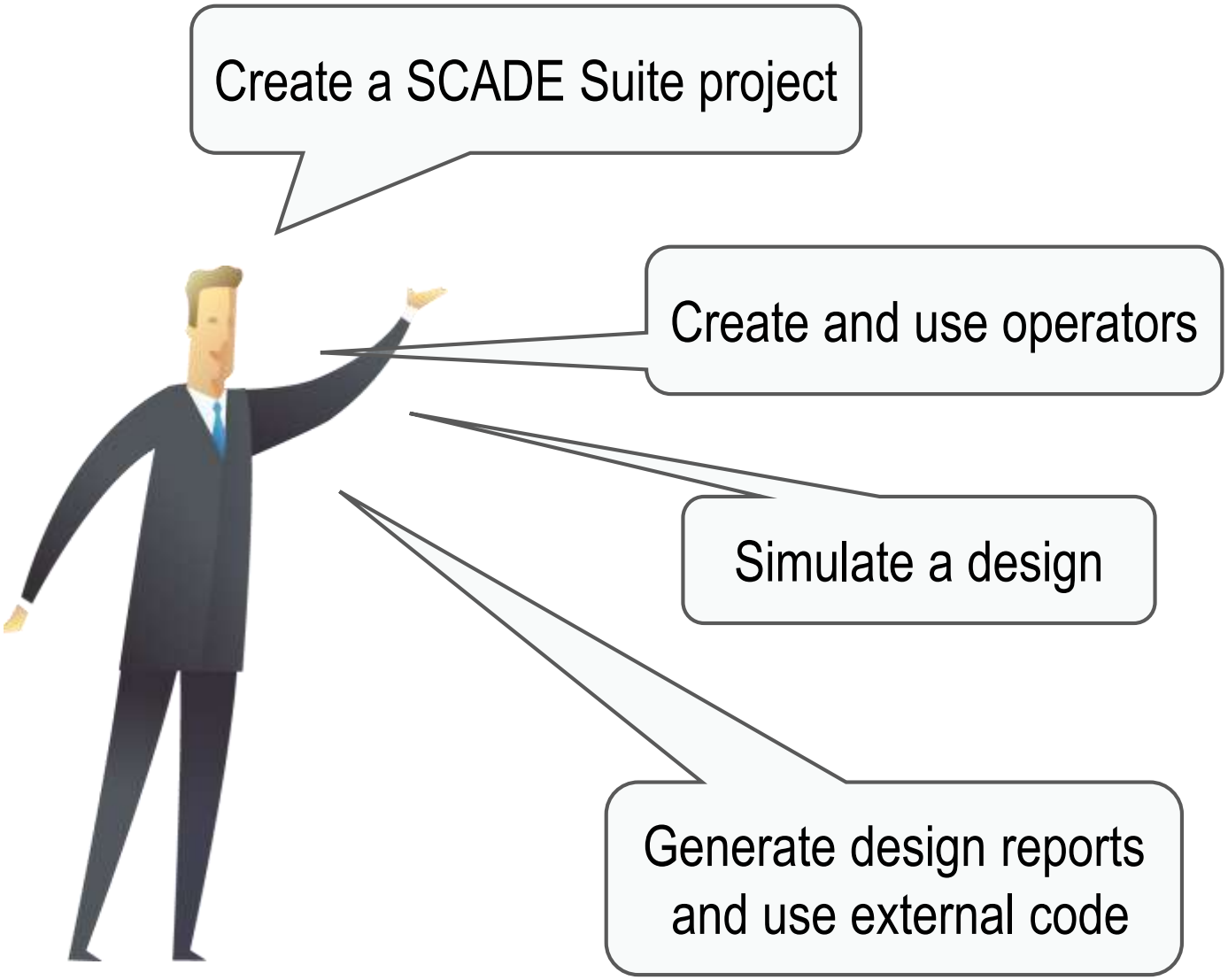
SCADE Suite Help

Access from Startup Page:

© 2016 ANSYS, Inc.

Lab Objectives

Lab

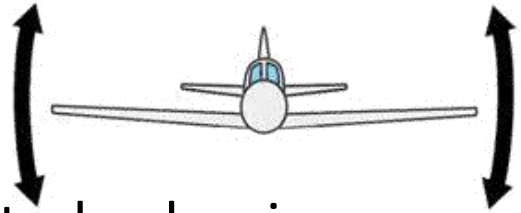


© 2016 ANSYS, Inc.

Lab Objectives

The system contains three subsystems:

- The control part to define the system mode
- The roll calculation
- A warning information when the roll is higher than a determined value



What is the roll?

The roll is the rotation around the front-to-back axis

The subsystems will be integrated together:

- We will create bricks and assemble them to obtain the whole application
- A graphical panel will be used to improve the simulation

Lab

© 2016 ANSYS, Inc.

Lab 1: How to Create a New Project (1/3)

Lab Prerequisite

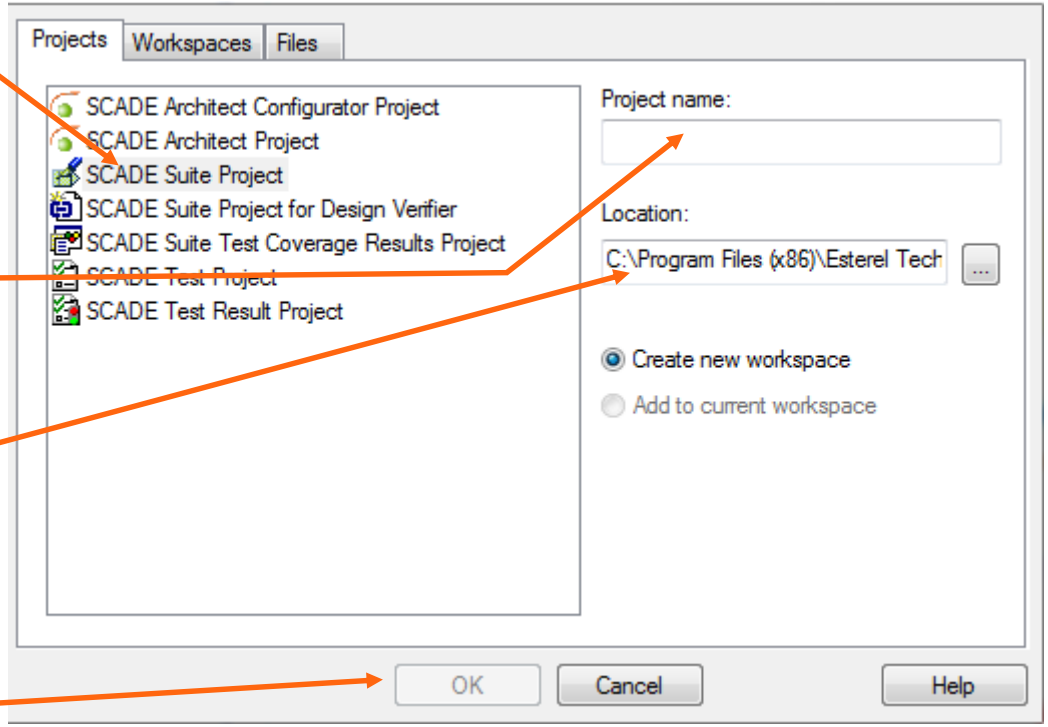
1. Launch SCADE Suite and select File - New

2. Select SCADE Suite Project

3. Enter project name

4. Enter a project location

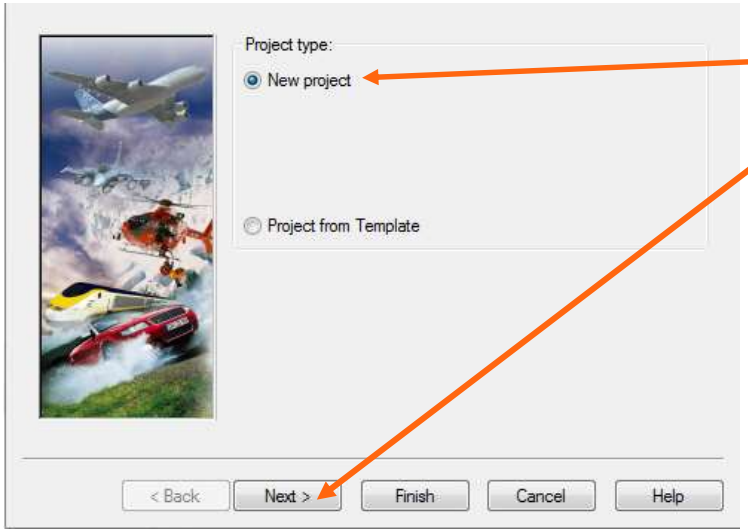
5. Click *OK*



© 2016 ANSYS, Inc.

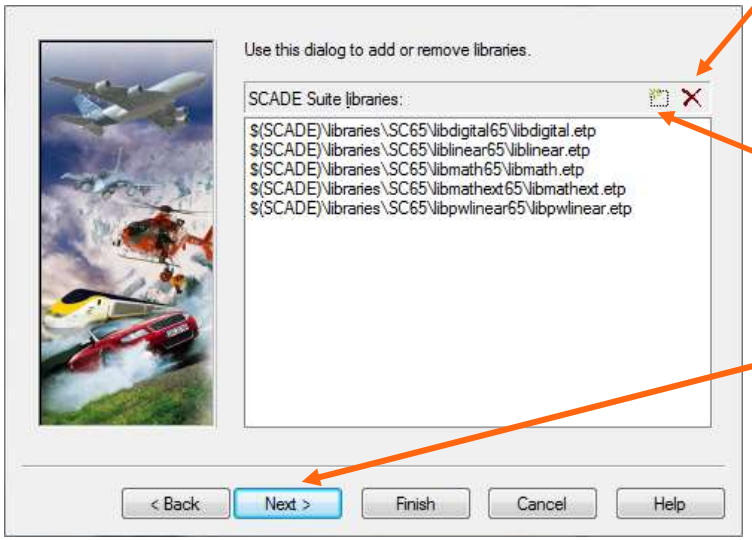
Lab 1: How to Create a New Project (2/3)

Lab Prerequisite



1. Select *New project*, and click on *Next*

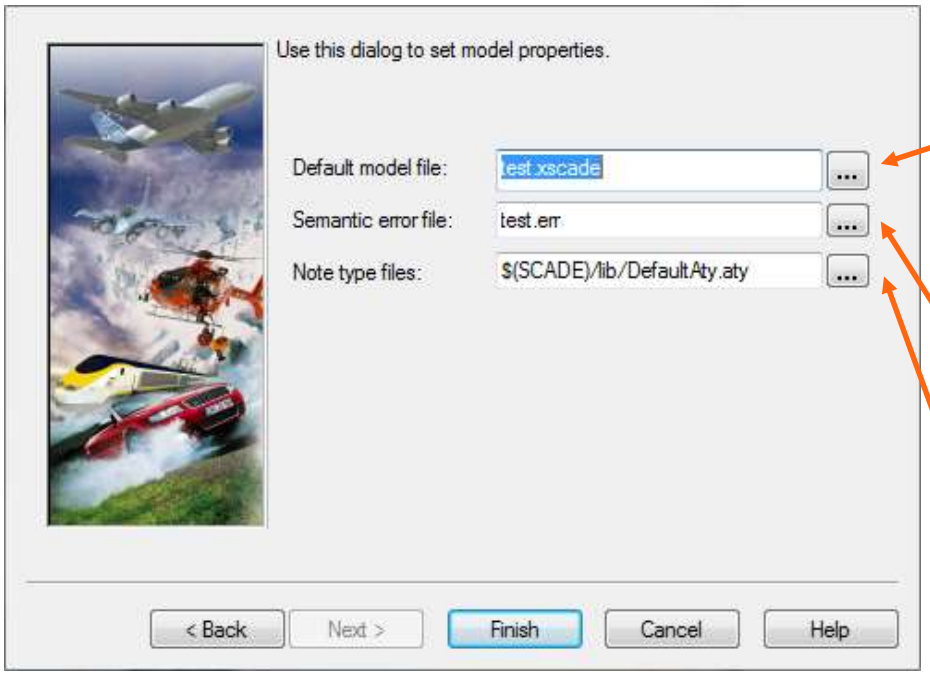
2. Select the default SCADE Suite libraries you do not need and remove them using the ✕ button



3. Add your own libraries by clicking on the ⊡ button and click on *Next*

© 2016 ANSYS, Inc.

Lab 1: How to Create a New Project (3/3)



Set the saving format (ensures editor compatibility with earlier versions)

Pathname of the error file

Pathname of the note type file

Click on Finish : the project with the workspace will be created automatically

Lab Prerequisite

© 2016 ANSYS, Inc. February 1, 2017 ANSYS Confidential

Lab 1: SCADE Suite Project

Lab

Objective:

Create your first SCADE Suite project

Lab Support p.5-9

Requirements:

Select a working folder

Time: 10 min

Open SCADE Suite and create your first project:

Project Name: `RollControl`

SCADE Suite libraries: `libpwlinear, libmath, libdigital`

© 2016 ANSYS, Inc.

AGENDA

Integrated Development Environment

SCADE Suite Operator

Simulation introduction

Code generation and standalone executable

Report

© 2016 ANSYS, Inc.

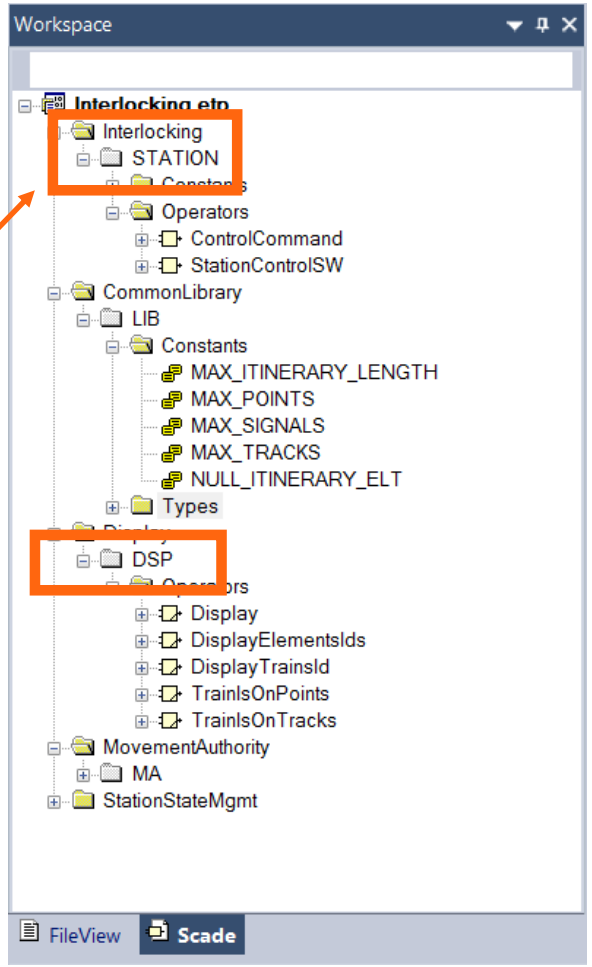
Model Packages

A package defines the namespace containing SCADE model elements

A package can be created only in a project folder

in Scade view, users can:

- Create as many packages as necessary
- Create a package inside another package

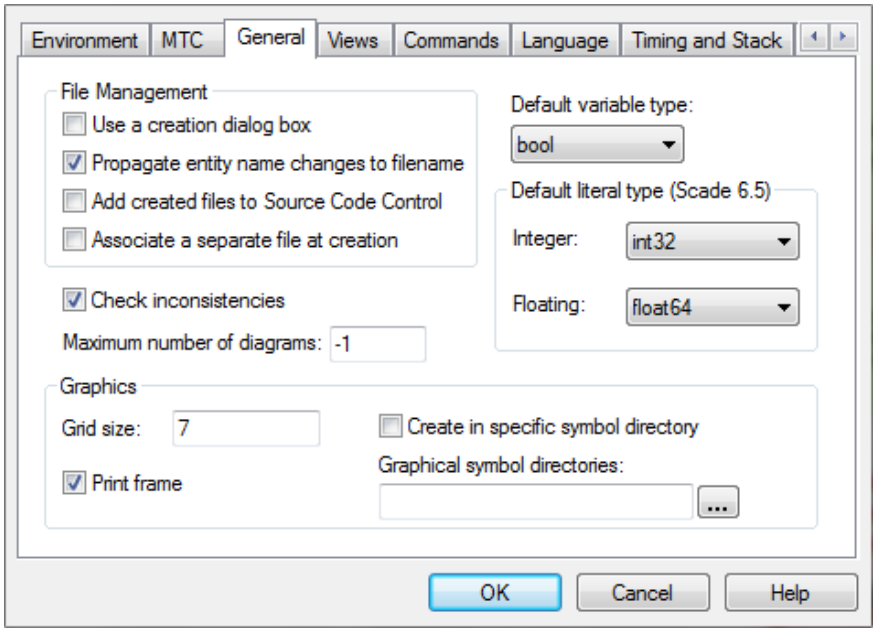


© 2016 ANSYS, Inc.

Package Option

Possible to set general options to propagate changes on the package name to the file name:

This option is set under *Tools > Options > General tab*

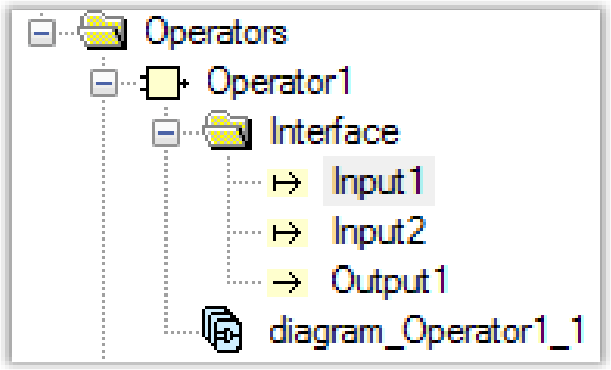


© 2016 ANSYS, Inc.

What is an Operator?

An operator is a basic building block in SCADE Suite.

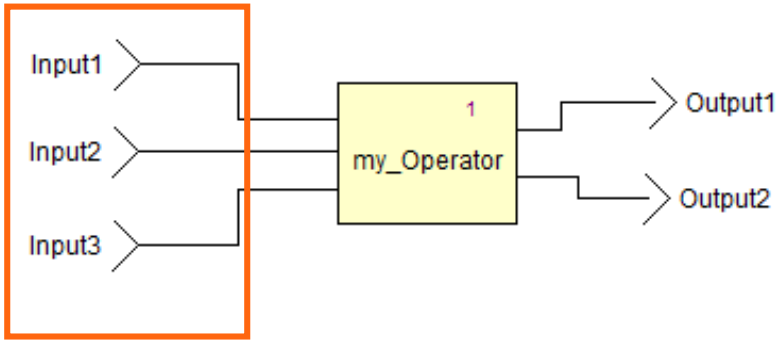
It is a computing unit with inputs and outputs.



© 2016 ANSYS, Inc.

Operator Interface (C Code)

Input values are passed to an operator through its input interface:

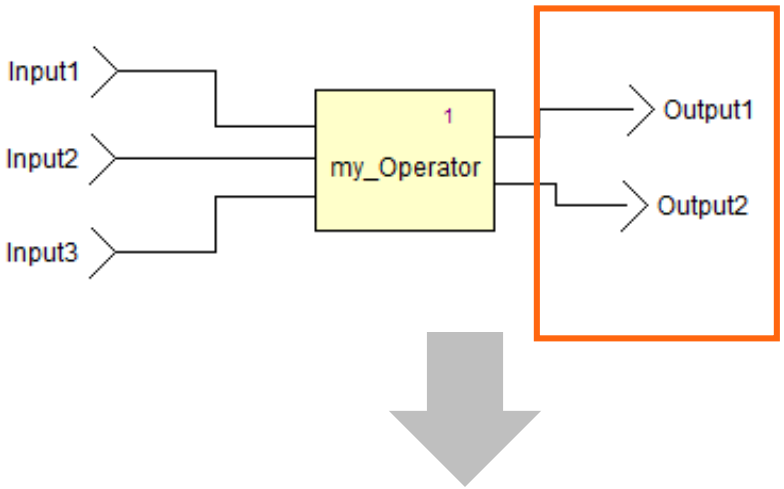



```
typedef struct {
  kcg_bool /* Input1/, _L1/ */ Input1;
  kcg_bool /* Input2/, _L2/ */ Input2;
  kcg_bool /* Input3/, _L3/ */ Input3;
} inC_my_Operator_Operators;
```


© 2016 ANSYS, Inc.

Operator Interface (C Code)

Computed output values are accessed through an operator's output interface:



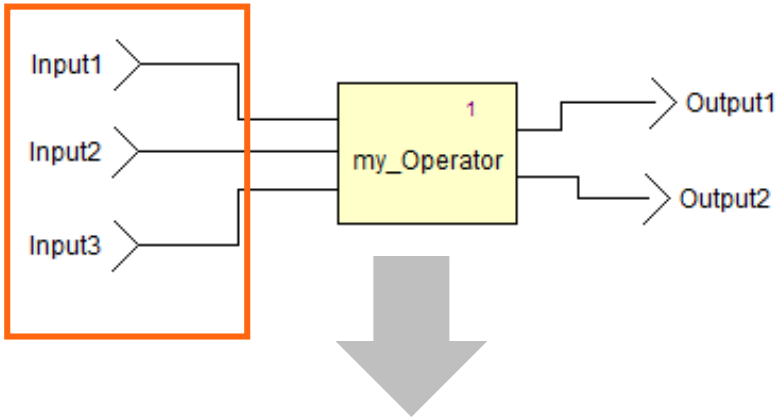

```
typedef struct {
  /* ---------------------------  outputs  --------------------------- */
  kcg_bool /* Output1/ */ Output1;
  kcg_bool /* Output2/, _L5/ */ Output2;
  /* ----------------------  no local probes  ---------------------- */
  /* ---------------------- local memories  ---------------------- */
  kcg_bool /* _L4/ */ _L4;
  /* -------------------- no sub nodes' contexts  -------------------- */
  /* ----------------- no clocks of observable data ----------------- */
} outC_my_Operator_Operators;
```


© 2016 ANSYS, Inc.

Operator Interface (Ada)

Ada

Input values are passed to an operator through its input interface:



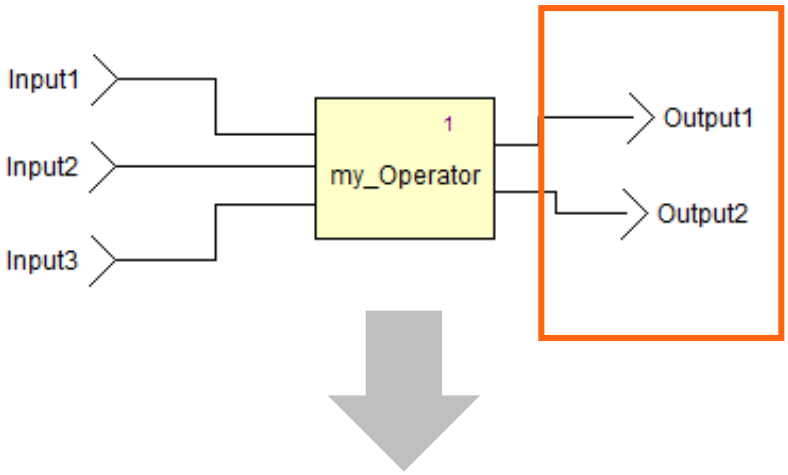

```
-- Operators::my_Operator/
procedure my_Operator(
   -- Input1/, _L1/
   Input1 : in Boolean;
   -- Input2/, _L2/
   Input2 : in Boolean;
   -- Input3/, _L3/
   Input3 : in Boolean;
   Ctx : in out Context_my_Operator);
```


© 2016 ANSYS, Inc.

Operator Interface (Ada)

Ada

Computed output values are accessed through an operator's output interface:




```
type Context_my_Operator is
  record
  -- --------------------------  outputs  --------------------
  -- Output1/
  Output1 : Boolean;
  -- Output2/, _L5/
  Output2 : Boolean;
  -- ----------------------  no local probes  ----------------
  -- ----------------- no clocks of observable data -----------
  -- ----------------------- local memories  ------------------
  -- _L4/
  L4 : Boolean;
  -- -------------------  no debug variables  ----------------
  -- ------------------  no assertion variables  --------------
  -- -------------------- no sub nodes' contexts  -------------
  end record;
```


© 2016 ANSYS, Inc.

Data Types

The Scade language is strongly typed:

- All the variables are explicitly typed
- Users can define as many types as necessary
- Conversion between values of different types are explicit

Scade predefined types:

- Boolean: `bool` and character: `char`
- Signed Integer: `int8, int16, int32, int64`
- Unsigned Integer: `uint8, uint16, uint32, uint64`
- Floating point: `float32, float64`

© 2016 ANSYS, Inc.

Lab 2: SCADE Suite Operator

Lab

Lab Support p.11-15

Objective:

Create your first SCADE Suite Operator

Requirements:

Continue on the current project (RollControl.etp)

Create a new package, a new operator and add I/Os

Package Name: `RollRate`

Operator Name: `RollRateWarning`

Time: 10 min

Name	Kind	Type
rollRate	input	float32
leftWarning	output	bool
rightWarning	output	bool

Double click on the operator to open the design view

ANSYS

© 2016 ANSYS, Inc. February 1, 2017 ANSYS Confidential

Customize User Environment

Different docking windows (output, properties, …), views (diagrams, constants, types, sensors) and toolbars to:

- Manage the tools integrated to SCADE Suite (Modeler, KCG, Simulation, SCADE Lifecycle Reporter, Traceability, …) related to the current activity
- Edit functional models

Right click on the toolbars area to select them

Use “To Shortcut” to put favorite toolbars in the shortcut window

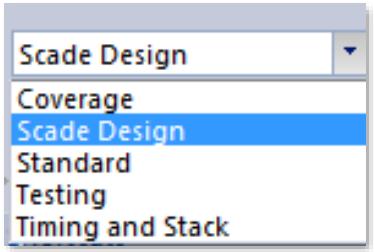
The user environment is stored into the active perspective (next slides)

© 2016 ANSYS, Inc.

Customize User Environment

Use **Perspectives** to store and retrieve environment configurations that fit the user context.

The SCADE Suite environment proposes default perspectives:



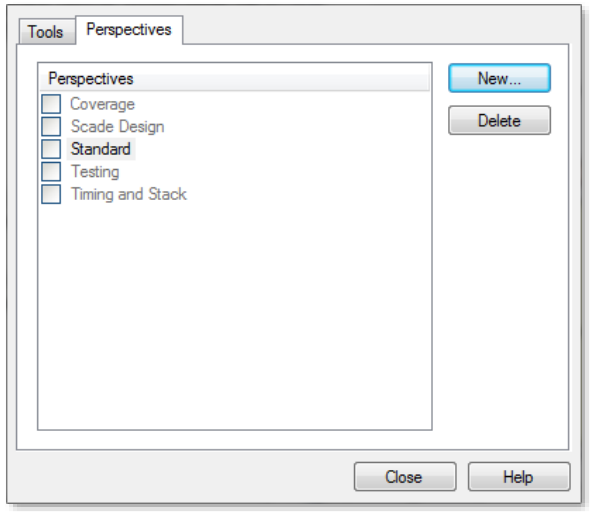
The active perspective can store any changes in the workspace organization.

Perspectives are attached to a kind of project (SCADE Suite, SCADE Test, …):

- Last used perspective is saved between sessions
- When opening a project, the last used configuration for this project is used

© 2016 ANSYS, Inc.

Customize User Environment



Create your own perspective:

View > Perspectives > Customize

Create a new perspective:

- Click New to open the Create Perspective window
- Enter the name of the perspective
- The new perspective is added and based on the active perspective

The default and active perspectives cannot be deleted

© 2016 ANSYS, Inc.

Customize User Environment

Users can position docking windows in the editor:

- Select, hold and drag the window to display helper buttons:
  - Grouped and individual buttons

- Move the window over an individual button (top / bottom / left / right side) to activate the position helper (turning to blue) and choose the position

- Release the dragged window at the selected position

© 2016 ANSYS, Inc.

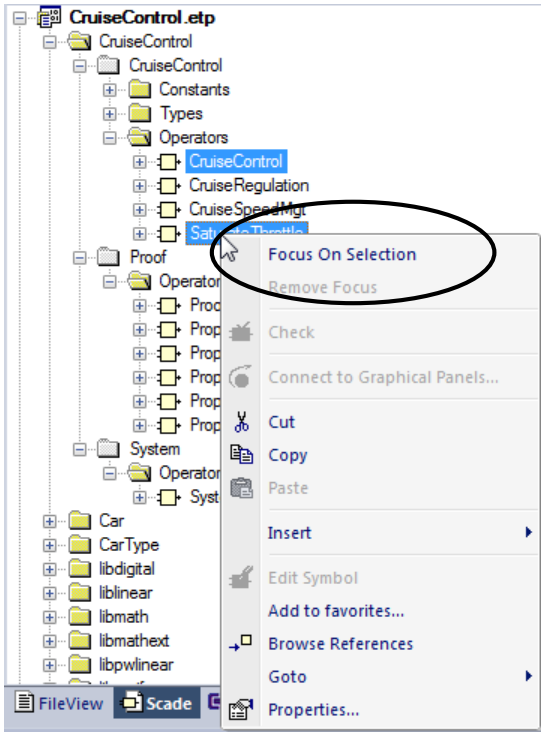
Customize User Environment

Multi-screen display:

- All docking windows and model editing views (diagrams, constants, types, sensors) can be moved for display on multi-screen environments:
  - For a view, first, set it as floating window, right-click the tab and select *Float*
  - Grab any docking window or view to move on another screen
- To set floating windows back to application window (editor)
  - For a view, right-click the tab and select *back to Main*
  - For a docking window, set the position with helper buttons (previous slide)

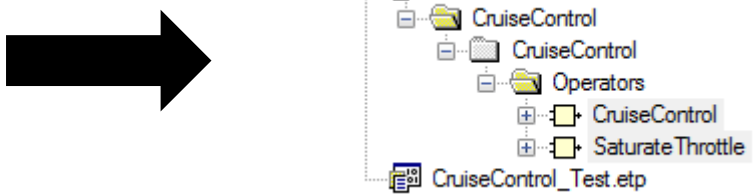
© 2016 ANSYS, Inc.

Customize User Environment



Tree selection display:

- From Scade view, select one or several objects and right-click *Focus on Selection* to display only selected object(s)



- From Scade view, select any object and right-click *Remove Focus* to display all objects

© 2016 ANSYS, Inc.

Lab 3: Workspace

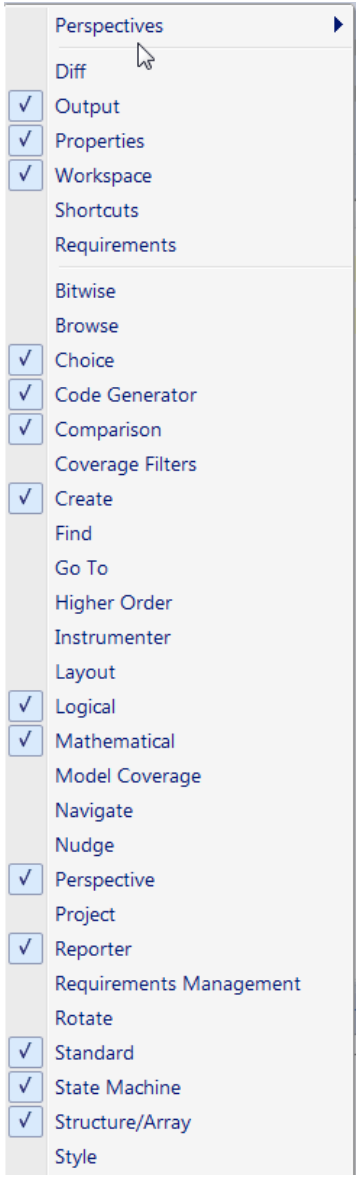
Time: 5 min

Objective:

Organize your workspace

Requirements:

Right click on the toolbars area to select the same options as on the menu on the right



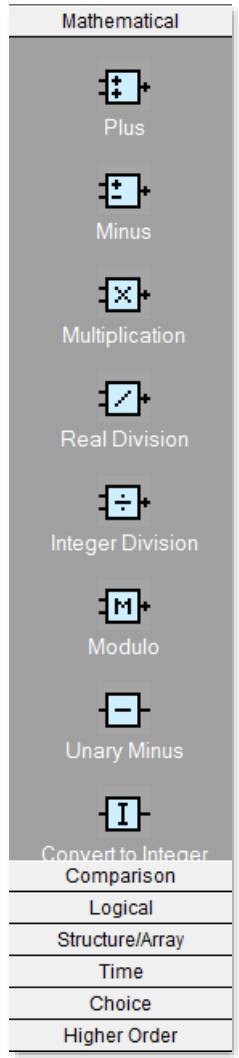
Lab

© 2016 ANSYS, Inc. February 1, 2017 ANSYS Confidential

Predefined Operators

SCADE Suite built in operators:

- Classified by functionalities (mathematical, logical, choice, …)
- Defined in the language (behaviour, type and interface)
- Used in user-defined operator but not editable
- Behaviour is precisely documented

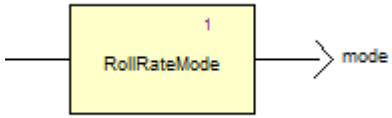


© 2016 ANSYS, Inc.

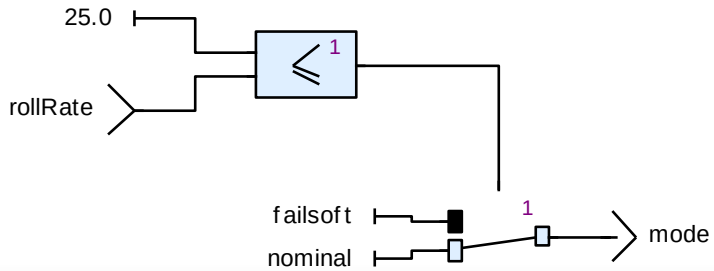
User Operators

Perform clearly defined functions to compute their outputs from their inputs and possible memories

Computation logic may be built using predefined operators and existing user-defined operators



Internal logic is captured in diagrams:

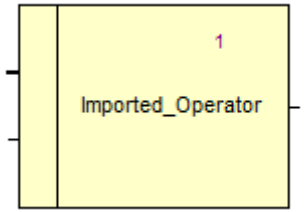


© 2016 ANSYS, Inc.

Imported Operators

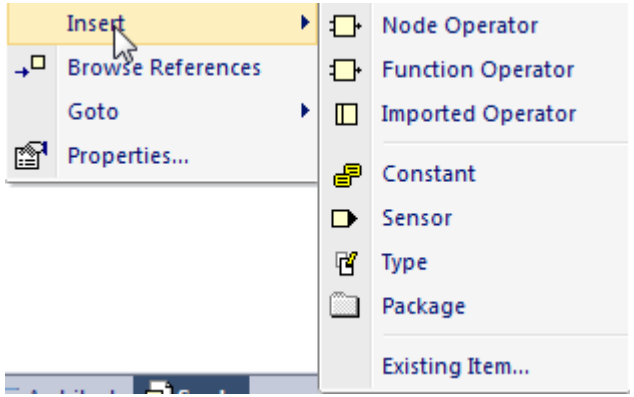
An operator defined in SCADE Suite with an externally defined implementation:

- Extends Scade language expressiveness (ex: trigonometric functions,…)
- Interface is defined in SCADE Suite
- Implementation has to be provided as target language source code



© 2016 ANSYS, Inc.

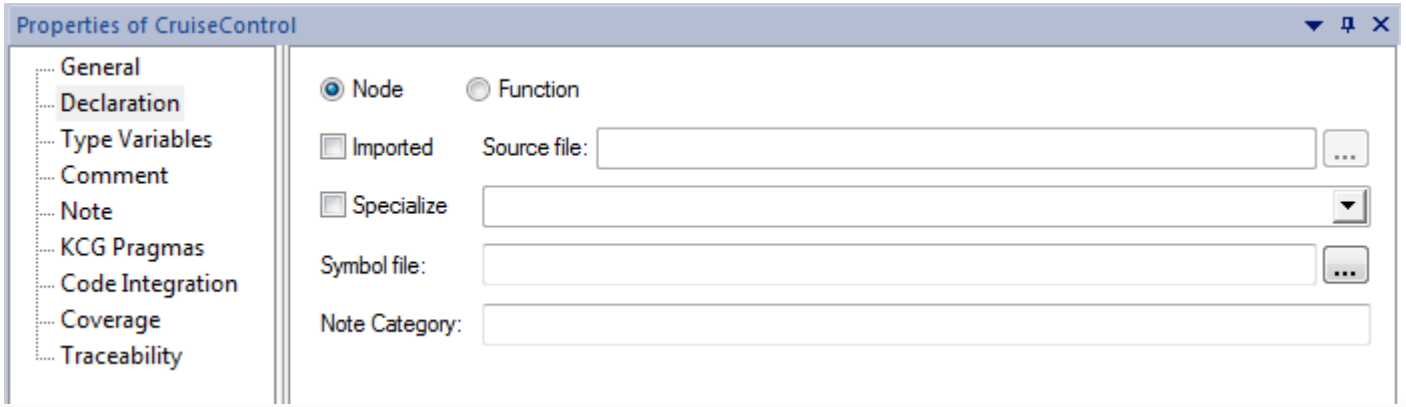
Nodes/Functions



Declare a node or function via contextual menu

- A node is an operator with internal memory
- A function is a stateless operator:
  - Contains no internal memory
  - Enables function specific code generator optimizations

Settings accessible via the properties dialog:



© 2016 ANSYS, Inc.

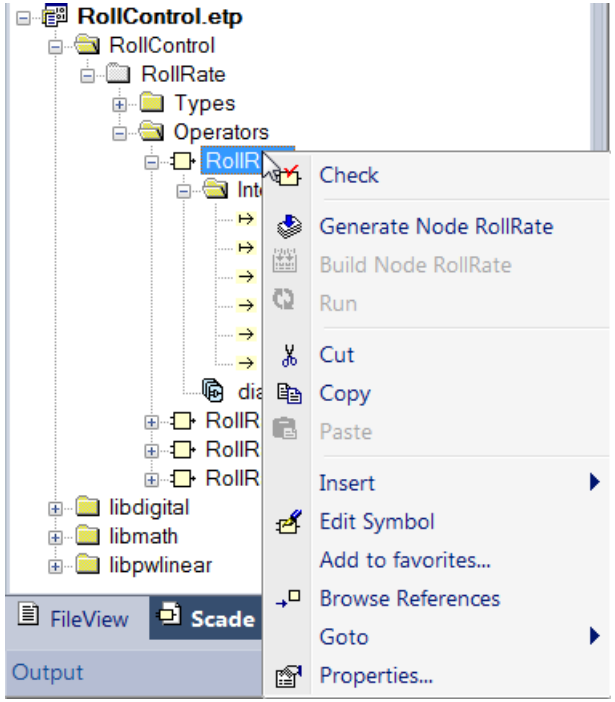
SCADE Suite Checker

SCADE Suite Checker:

- Is used to validate the model semantics
- Generates a report with hyperlinks to easily locate semantic errors in the model

Easy to use, a simple right click on the operator to check

The semantics check is performed on the operator and all SCADE Suite objects contained in its hierarchy



© 2016 ANSYS, Inc.

Lab 4: Create your First Design

Lab Support p.19-20

Objective:

Design the operator `RollRateWarning`, which computes the left warning and right warning alarms according to the airplane roll rate

Time: 10 min

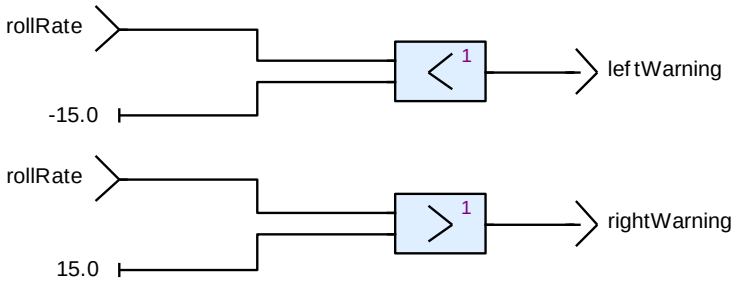
Requirement:

The operator `RollRateWarning` shall check the roll rate input and signaling if the value is out of bounds:

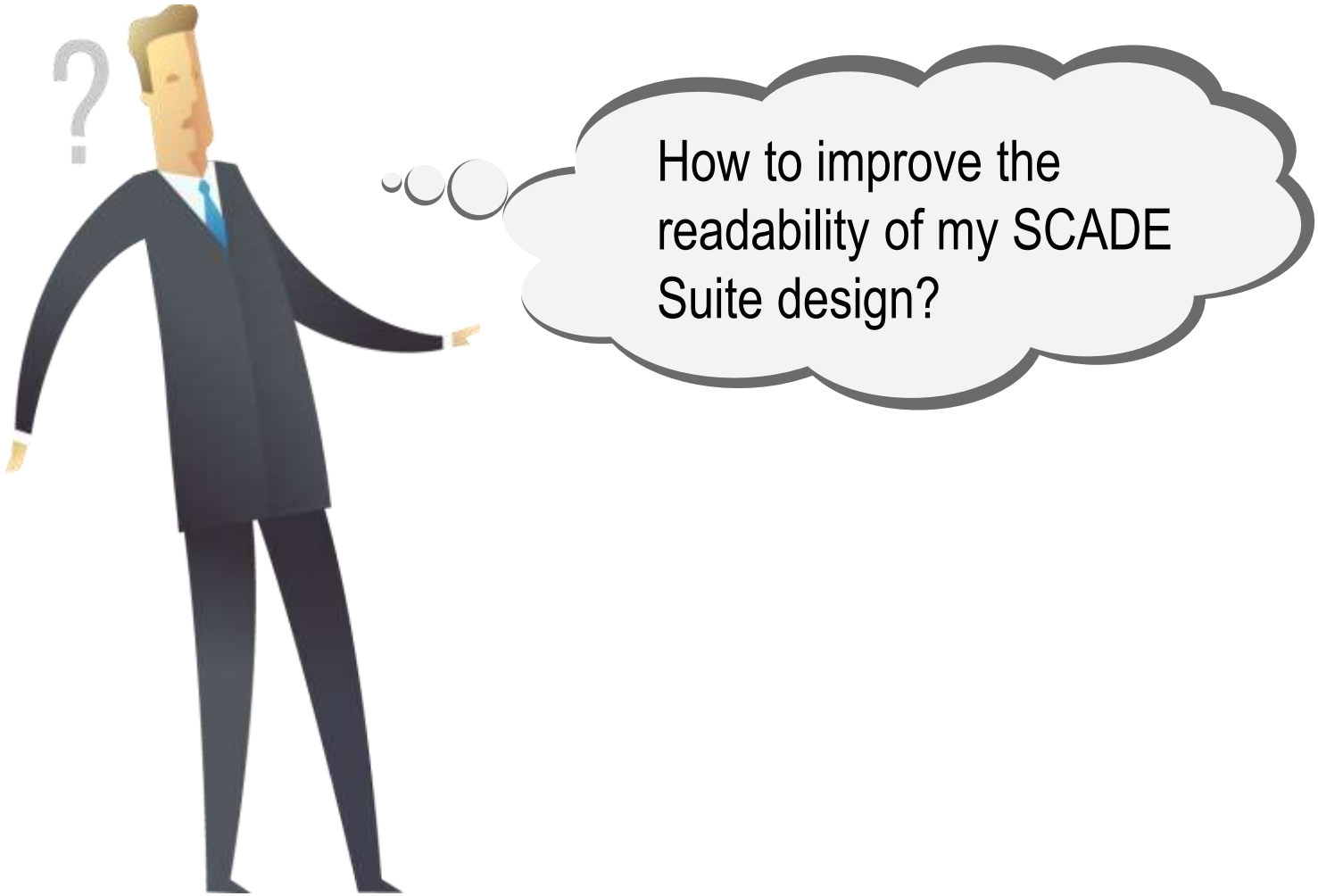
- when roll rate < -15.0, the operator indicates left warning
- when roll rate > 15.0, the operator indicates right warning

Tips: use  predefined operators

and  textual expression



© 2016 ANSYS, Inc.



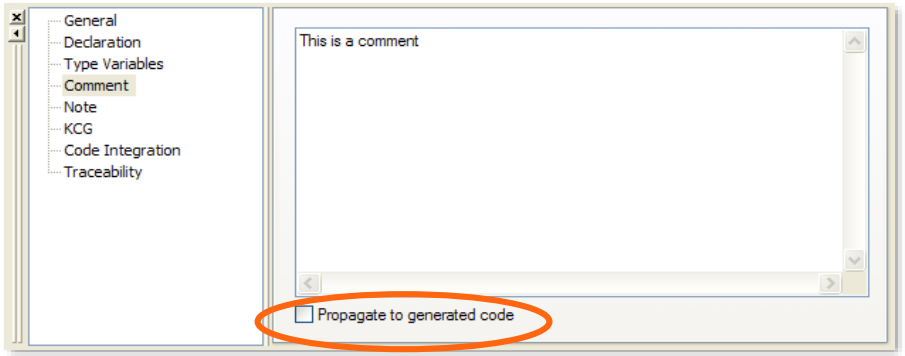
© 2016 ANSYS, Inc.

Comment Editing

A comment can be added in the properties of any of these objects:

- Packages, Constants, Sensors, Types, Diagram Blocks, Operators, Imported Operators, Inputs, Hidden Inputs, Outputs, Locals, etc..

Open the object properties window and enter comments in the “Comment” item

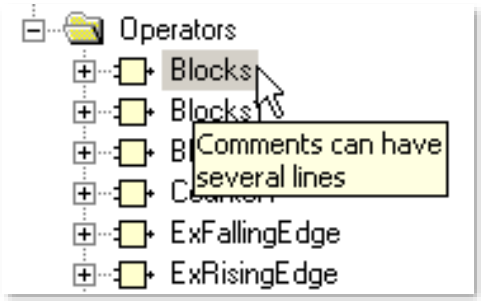


Comments can be propagated in the generated code by checking "Propagate to generated code"

© 2016 ANSYS, Inc.

Displaying Comments

In the Scade View, comments are displayed as mouse motion tips



For Constants, Sensors, Type, comments are displayed in the grid views

Constant	Type	Value	Comments
IInit	bool	false	Comments can have several lines
Y1	bool	false	
Y2	bool	false	

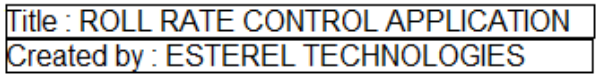
© 2016 ANSYS, Inc.

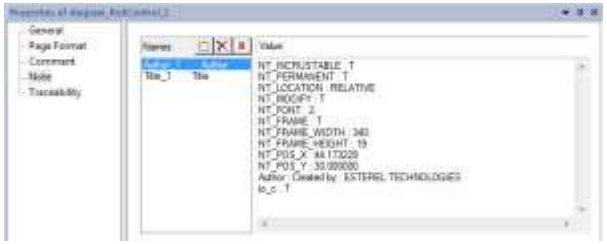
Annotations and Notes (1/4)

Annotations (notes) can be associated with any SCADE Suite entity: package, types, constants, nodes, inputs, outputs, ...

The notes types may be customized to implement any methodological guide or additional information:

- graphical notes
- mandatory notes
- cardinality rules
- document purpose
- C code propagation
- ...





Note: graphical notes on nodes are defined at the diagram block level

© 2016 ANSYS, Inc.

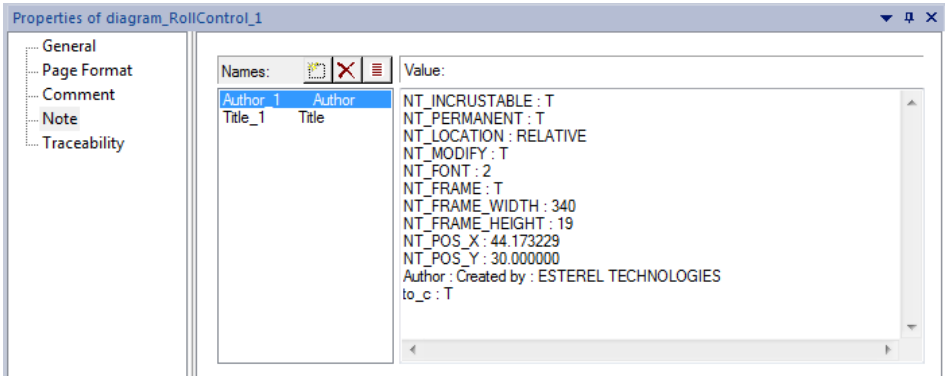
Annotations and Notes (2/4)

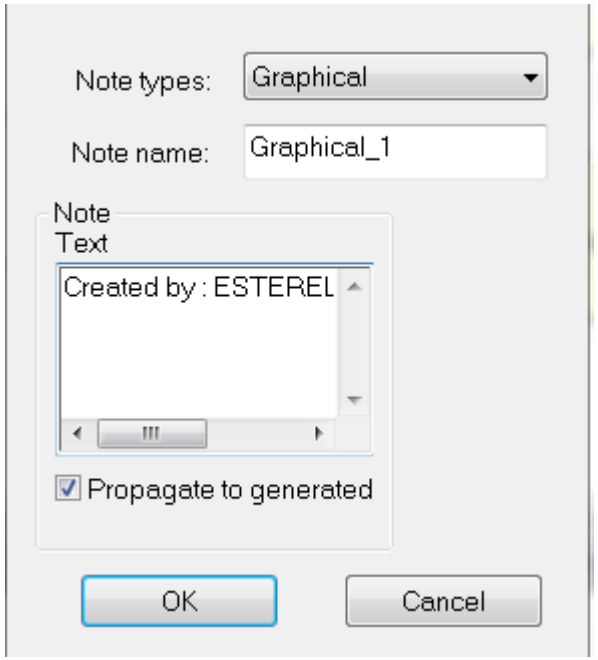
Select the object you want to comment, open its Properties box, and select the Note tab:

- the left column displays the notes name and type
- the right column displays the notes content.

Click on  to create a new note, on  to delete or on  to edit the selected note.

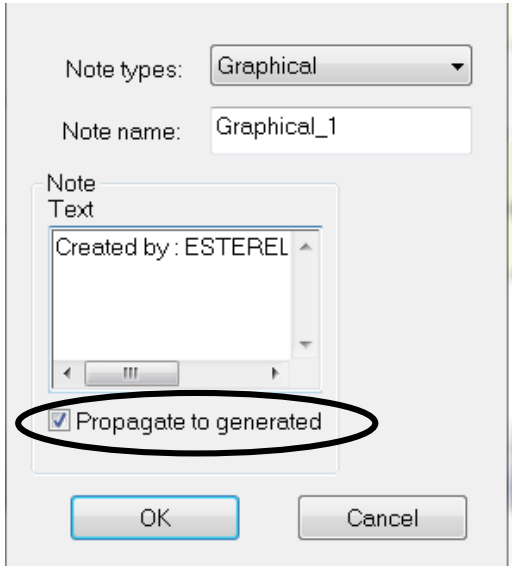
When creating/editing a note, a custom editing window opens





© 2016 ANSYS, Inc.

Annotations and Notes (3/4)

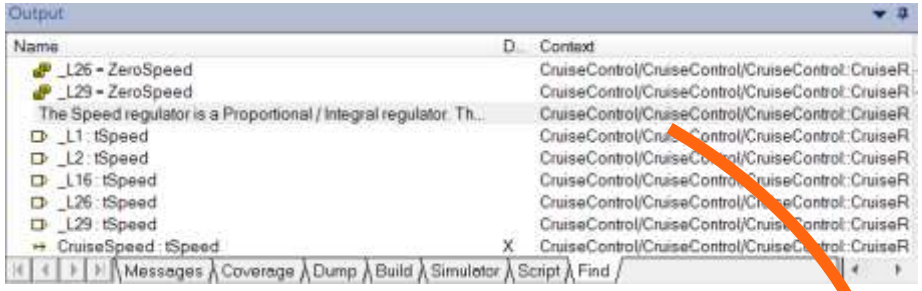


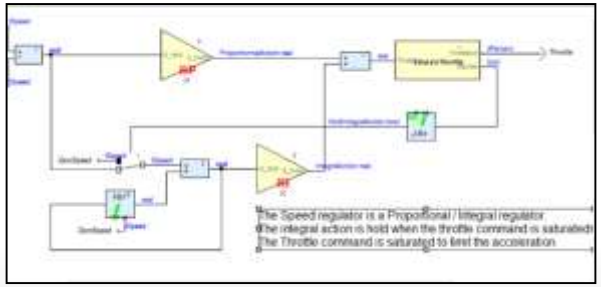
As for comments, annotations can be propagated in the generated code:

- "Propagate to generated code" in the annotation has to be checked if present

Annotations appear in the Find tab:

- It is possible to locate and go directly to any annotated object





© 2016 ANSYS, Inc.

Annotations and Notes (4/4)

Users can create their own type of annotations:

- Copy the <SCADE_dir>/lib/DefaultAty.aty under your project directory and name it <project name>.aty
- Create a new type of annotations (A)
- Specify to which item this new annotation can be associated (B)

```
MyAnnotation ::=
    SEQUENCE OF {
    SEQUENCE {
       annot_object OID,
       name STRING,
       information {
         Purpose TEXT {
            NT_DEFAULT_VALUE "Purpose : ",
            NT_FIELD_HEIGHT 1,
            NT_FIELD_WIDTH  60},
         Requirements TEXT {
            NT_DEFAULT_VALUE "Requirements :
",
            NT_FIELD_HEIGHT 1,
            NT_FIELD_WIDTH  20},
to_c BOOLEAN { NT_DEFAULT_VALUE T }
} } }
```

(A)

```
constant ::= {
                    {Remark F 0 99},
                    {GdC F 0 1}
                   {MyAnnotation F 0 1 }
}
```

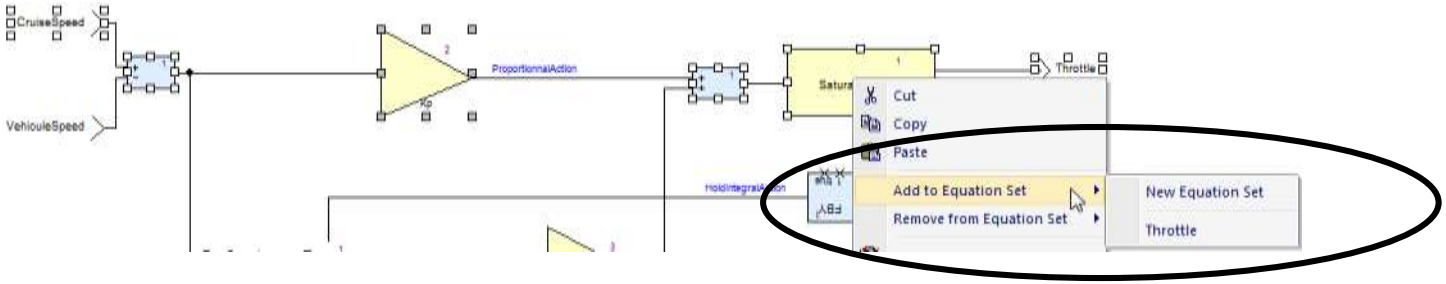
(B)

© 2016 ANSYS, Inc.

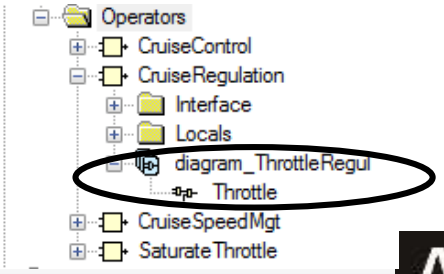
Equation Set

Users can create Equation Sets to refine the traceability or set specific information such as comments, and notes to a set of Scade elements

- Right-click several elements in a diagram and from the menu
- Select “New Equation Set” or an existing Equation Set to respectively add them in a new Equation set or in an existing one



- A new Equation Set is created (rename it) or the selected one is updated in Scade View

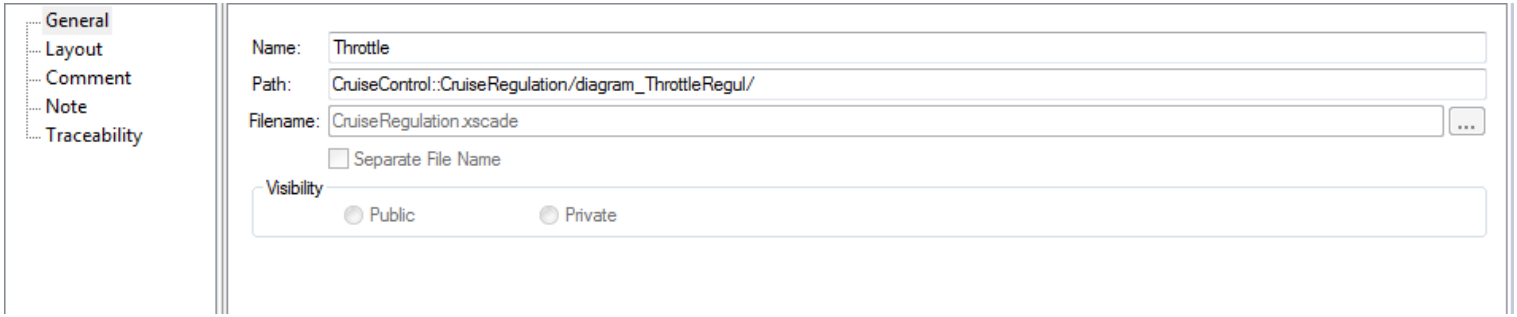


© 2016 ANSYS, Inc.

Equation Set

Select an existing Equation Set and from Properties, set

- A new name from General tab
- A new style in the Layout tab
- Comments in the Comment tab
- Notes in the Note tab
- Show or delete traceability requirements in the Traceability tab

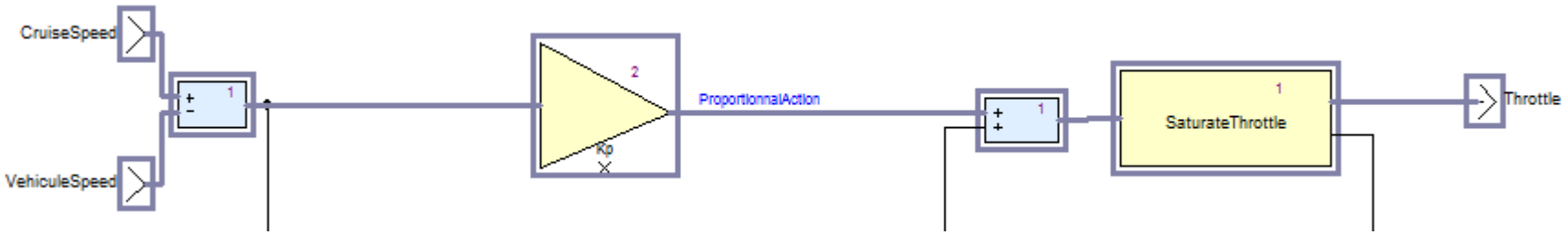


© 2016 ANSYS, Inc.

Equation Set

In Scade View, double-click on the Equation Set to see its content in the diagram (Scade elements):

- A style is used by default to highlight and display its content



Delete several elements from the Equation Set:

- Right-click several elements in a diagram and from the menu
- Select “Remove from Equation Set” to delete them from the Equation Set

© 2016 ANSYS, Inc.

Lab 5: Comment your Design

Use Lab Support p.22-23

Objective:

Improve the readability of `RollRateWarning` design

Requirements:

Time: 10 min

In your `RollRateWarning` operator design, add:

- a comment on the `RollRateWarning` **operator** to describe the behaviour
- Notes on `RollRateWarning` **diagram** to complete information: “Title”, “Author”, and “Text In Frame”

Lab

© 2016 ANSYS, Inc.

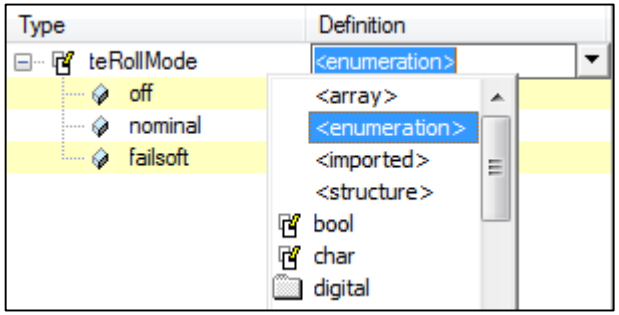
User Data Types

Users can create own types:

- Array, enumeration, structure,…
- Use the predefined and user types

Right Click on project > package to create a new type:

- Select the type definition of this new type from the Definition column
- Click and click (not double click)

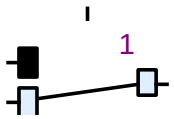


© 2016 ANSYS, Inc.

“Choice” Operator: if-then-else

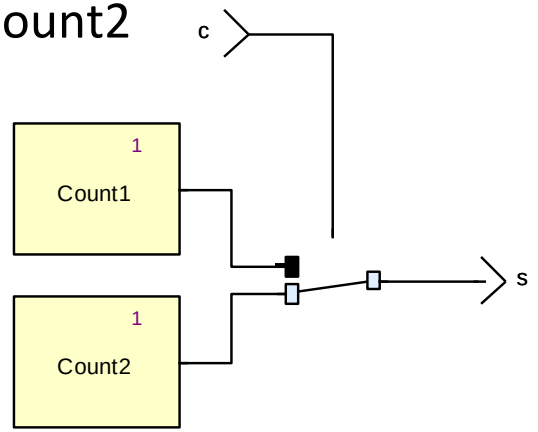
Receives two flows and outputs one of them depending on a Boolean condition

Both “then” and “else” expressions are always evaluated independently of the condition value.



Example:

- Consider 2 counter operators count1 and count2
- s = if c then Count1() else Count2();
- Each counter is incremented at each cycle regardless of c
- This behaviour complies with the intuition behind the reading of the graphical representation.



 © 2016 ANSYS, Inc.

Lab 6: Use Type and if-then-else (1/2)

Lab Support p.25-28

Objective:

Design the operator `RollRateMode`, which manages the mode

Requirements:

Time: 10 min

The RollRate system has 3 modes: off, nominal, failsoft

- off mode is when ON button is not pressed
- nominal roll rate is [-25.0;25.0]
- failsoft mode is out of these boundaries

Operator Name: `RollRateMode`

Name	Kind	Type
rollRate	input	float32
onButtonIsPressed	input	bool
mode	output	teRollMode

Lab

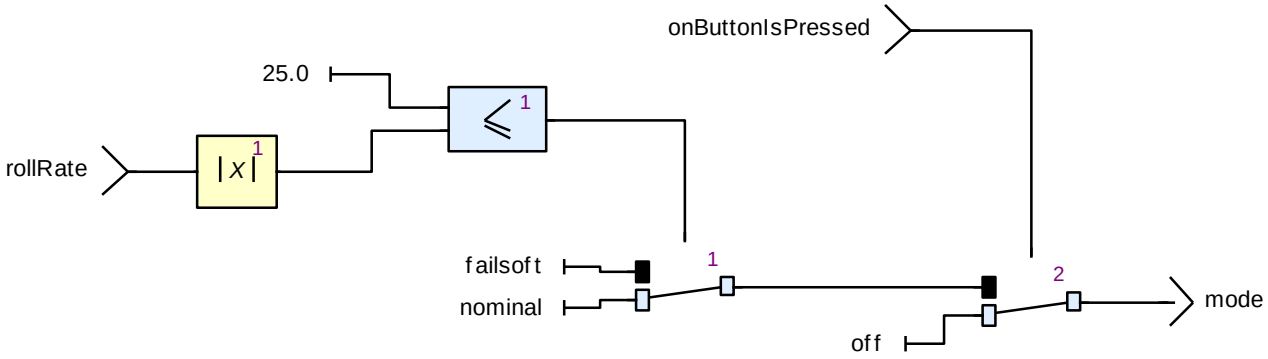
© 2016 ANSYS, Inc.

Lab 6: Use Type and if-then-else (2/2)

Create a new type `teRollMode` in the `RollRate` package:

Type	Definition
teRollMode	<enumeration>
off	
nominal	
failsoft	

Design the `RollRateMode` operator



Tips: Use "`math : Abs`" library operator to design the `RollRateMode` operator

© 2016 ANSYS, Inc.

SCADE Suite Standard Libraries (1/2)

SCADE Suite is delivered with examples of libraries of user operators:

- They are given as examples of common operations often used in control applications
- These operators have not been certified
- To use them in a certified project, you must:
  - Copy them into the project repository and manage them under configuration control
  - Verify and test them against the user requirements in the project context

© 2016 ANSYS, Inc.

SCADE Suite Standard Libraries (2/2)

Located in $(SCADE)\libraries

- **libdigital:** analysis operators for digital signals such as edges detection, flip-flop, truth tables…
- **liblinear:** operators for linear signal such as gain, derivative, memory, filters…
- **libmath:** math operators such as mean, range, min, max, round, vector and matrixes basic operations...
- **libpwlinear:** conversion operators from linear to digital such as counters, dead band, hysteresis, Look-up tables…
- **libmathext:** operators for trigonometric linked from the standard C libmath.
- **libverif:** operators for safety property modelling for formal verification

© 2016 ANSYS, Inc.

Lab 7: Use SCADE Suite Libraries (1/2)

Lab Support p.30-32

Objective:

Design the operator `RollRateCalculate`, which computes the airplane roll rate depending on the joystick command's input

Time: 10 min

Requirement:

The operator `RollRateCalculate` shall limit the joystick command between -25.0 and 25.0 and indicate if the output is saturated

Operator Name: `RollRateCalculate`

Name	Kind	Type
joystickCmd	input	float32
rollRate	output	float32

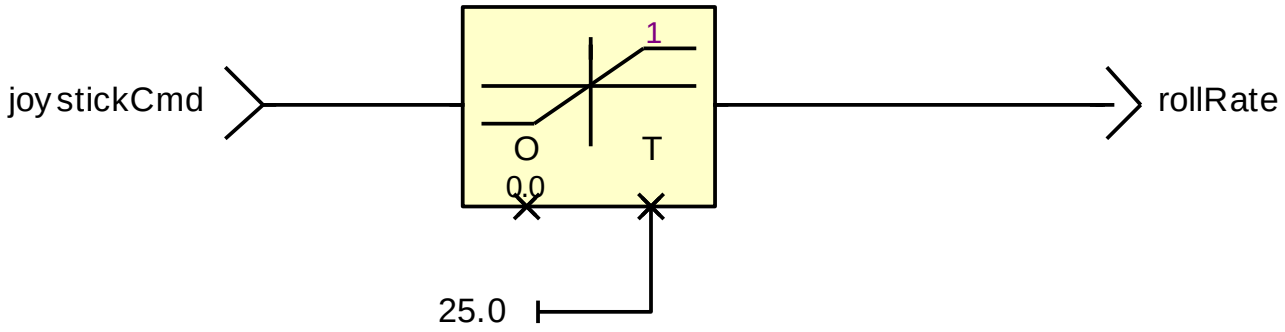
Lab

© 2016 ANSYS, Inc.

Lab 7: Use SCADE Suite Libraries (2/2)

Lab

Design the `RollRateCalculate` operator



© 2016 ANSYS, Inc.

Lab 8: Connect Operators (1/2)

Lab Support p.34-36

Objective:

Design the architecture operator `RollRate`

Time: 10 min

Requirements:

Create the `RollRate` operator and connects the previous designed operators to manage the system

Operator Name: `RollRate`

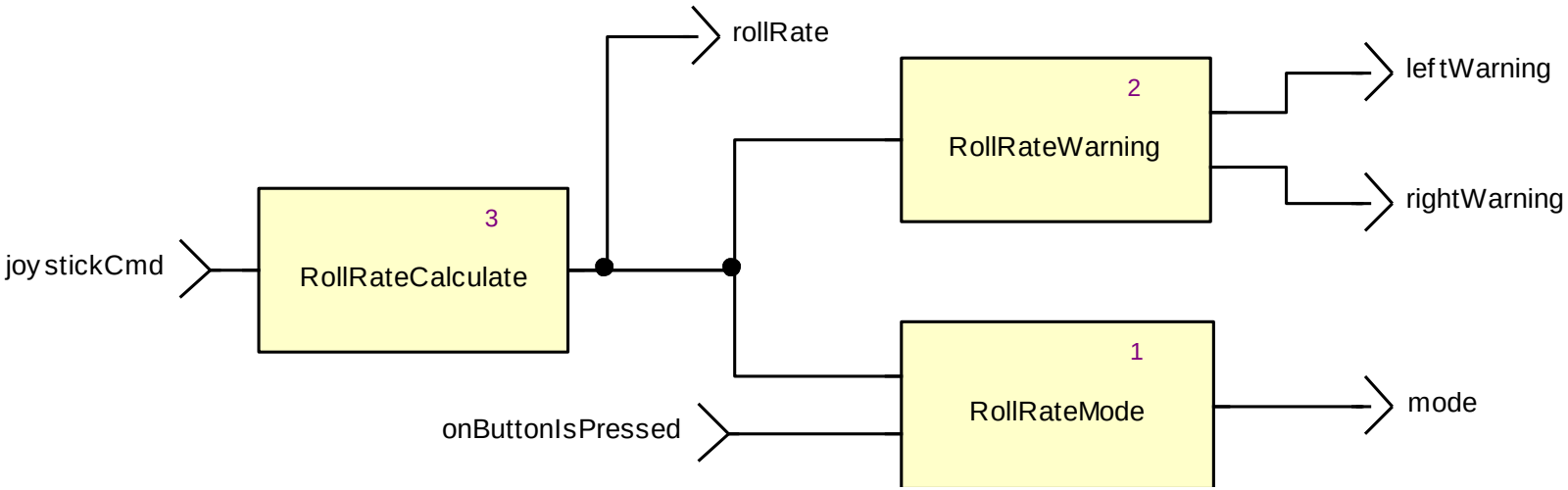
Name	Kind	Type
joystickCmd	input	float32
onButtonIsPressed	input	bool
rollRate	output	float32
mode	output	teRollMode
rightWarning	output	Bool
leftWarning	output	bool

Lab

© 2016 ANSYS, Inc.

Lab 8: Connect Operators (2/2)

Design the `RollRate` operator



Lab

© 2016 ANSYS, Inc.

AGENDA

Integrated Development Environment

SCADE Suite Operator

Simulator Introduction

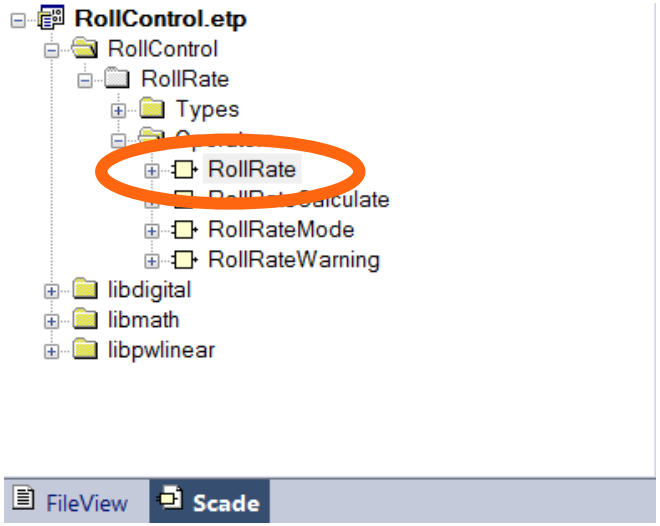
Code generation and standalone executable

Report

© 2016 ANSYS, Inc.

Simulator's Interface (1/3)

Select the operator you want to simulate from the Scade view



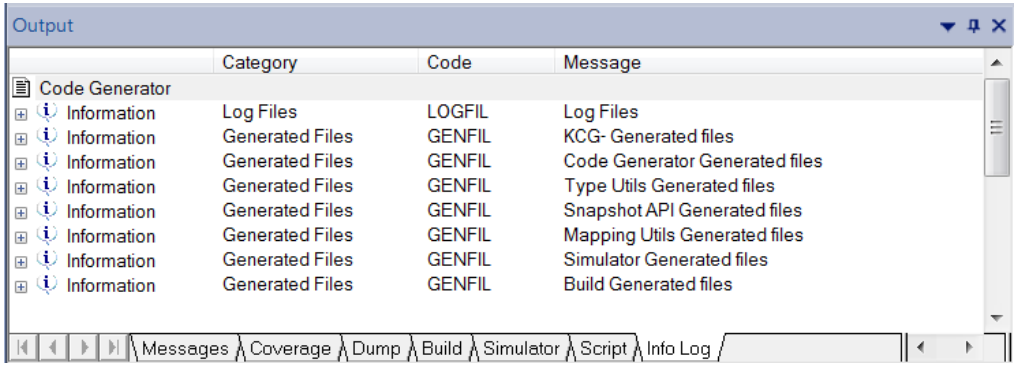
From the Code Generator toolbar

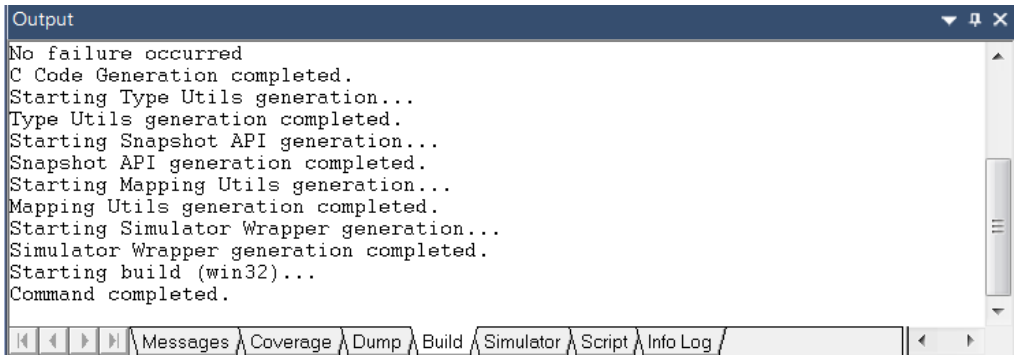
- Select the Simulation configuration
- Build th

© 2016 ANSYS, Inc.

Simulator's Interface (2/3)

Look under the "Info Log" and "Build" tabs if the simulation build is completed:





Launch the simulation:

© 2016 ANSYS, Inc.

Simulator Interface (3/3)

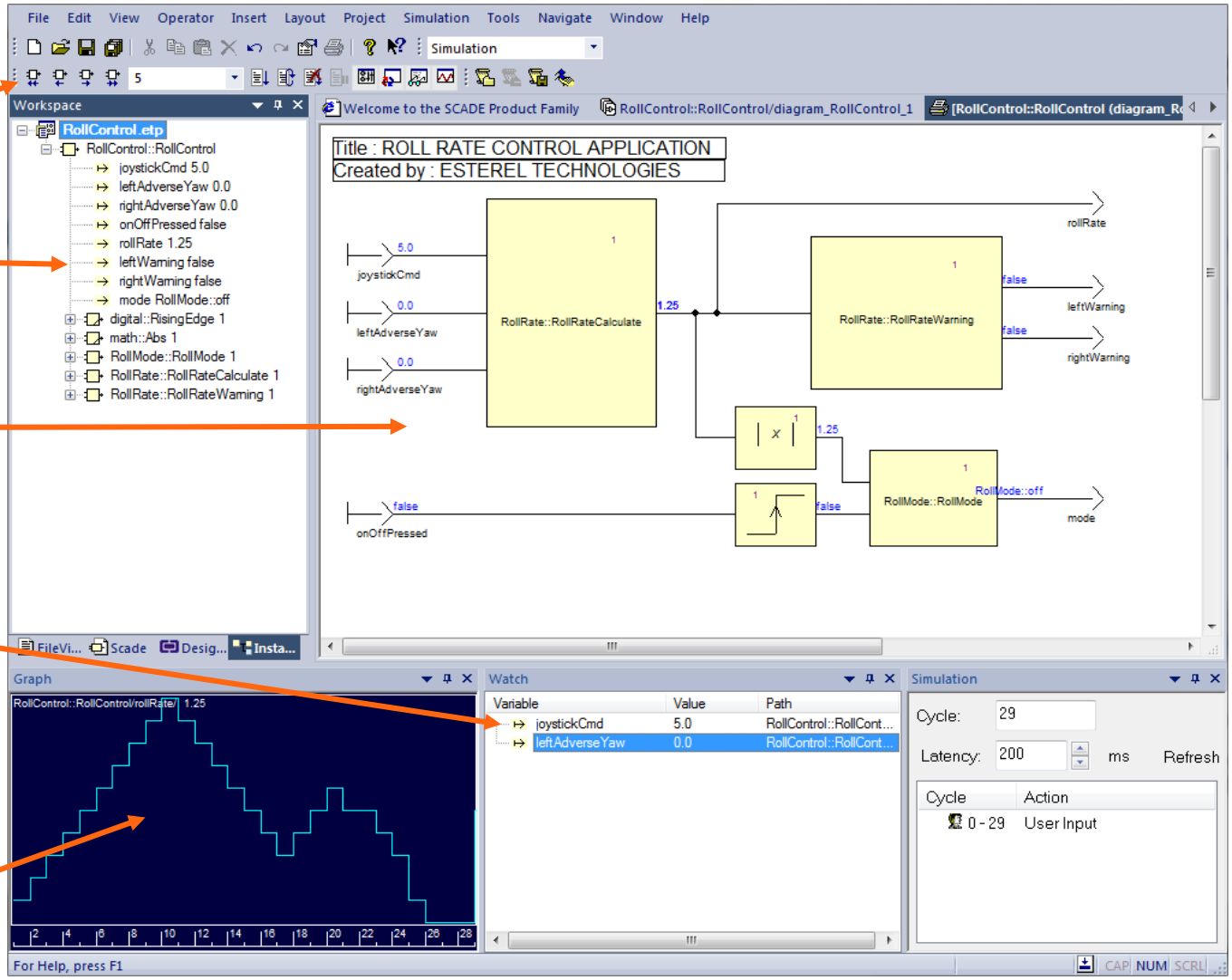
Play buttons

User input

Updated variables

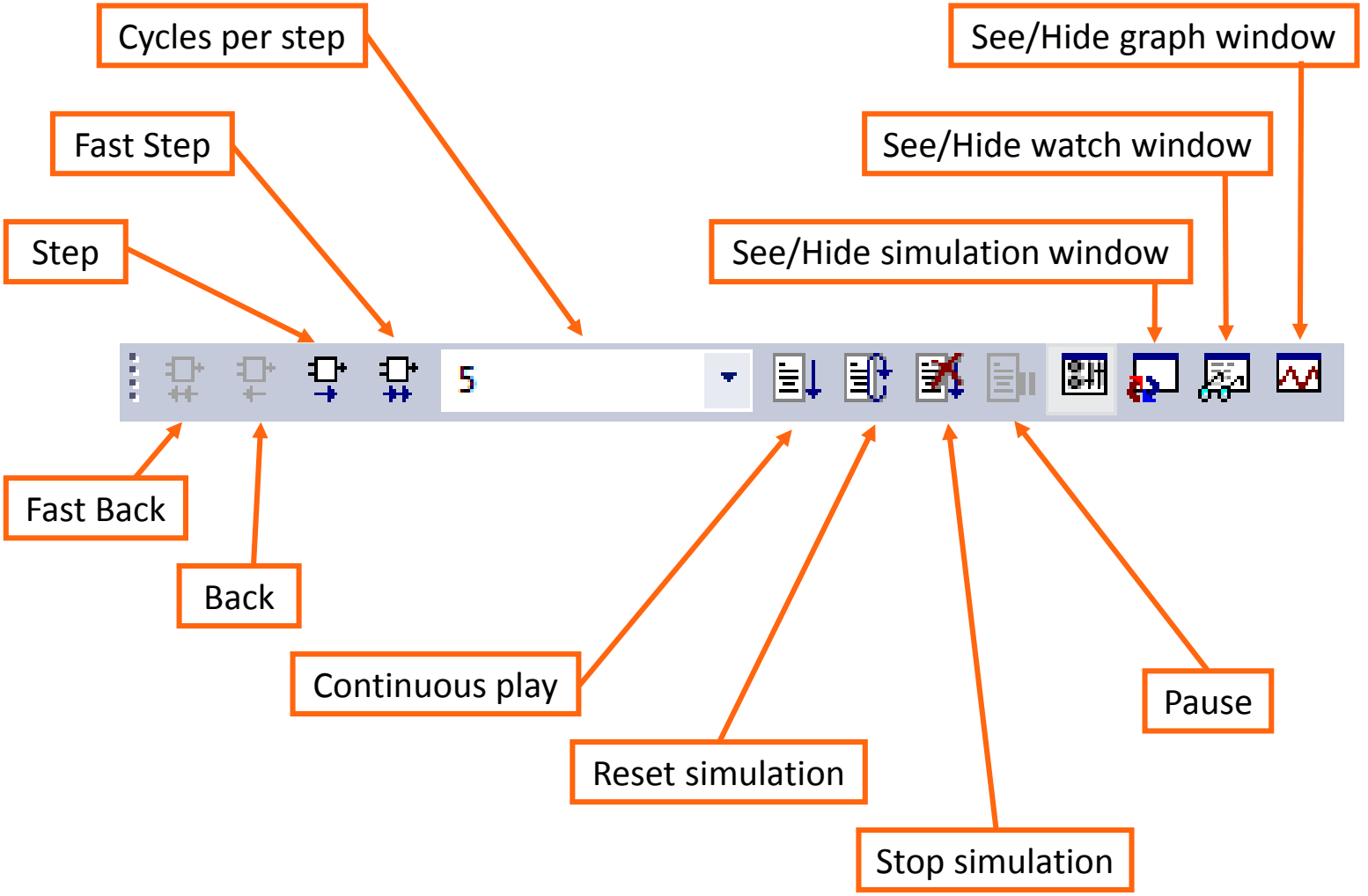
Watched variables

Graphical view of variables



© 2016 ANSYS, Inc.

Main Simulator Toolbars

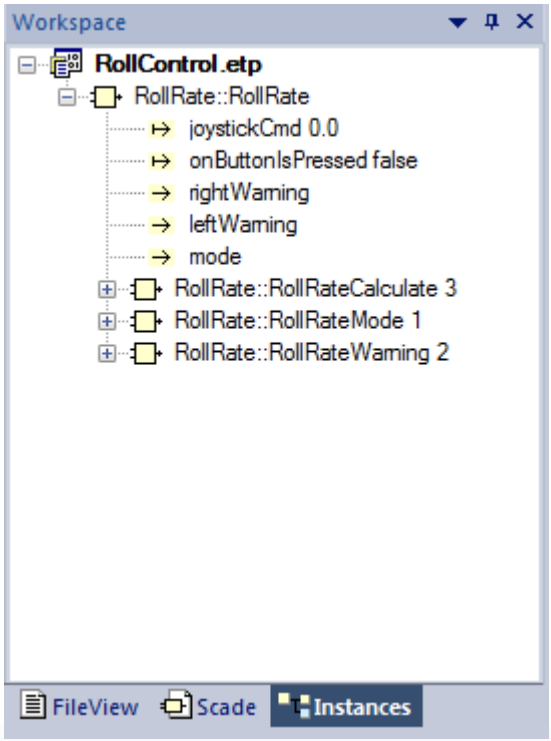


© 2016 ANSYS, Inc.

Set Input Values

To assign values to the model's inputs for each cycle:

- Directly set the value for an input in the Instances tab (use F2 to enter in edition mode)
- Tips: use t, for true, and f, for false for boolean values



© 2016 ANSYS, Inc.

Lab 9: SCADE Suite Simulator

Lab

Objective:

Simulate RollRate system to check the design

Requirements:

Generate the code and run the simulation

Time: 10 min

Play with the features of the SCADE Suite Simulator:

- Change the inputs values
- Run step by step or in continuous mode
- Open the watch window and select a data to observe (right click and select add to watch or drag and drop it to the watch window)
- Play the simulation in the graph window…

© 2016 ANSYS, Inc.

AGENDA

Integrated Development Environment

SCADE Suite Operator

Simulation introduction

Code generation and standalone executable

Report

© 2016 ANSYS, Inc.

SCADE Suite KCG

Qualifiable/certified code generator

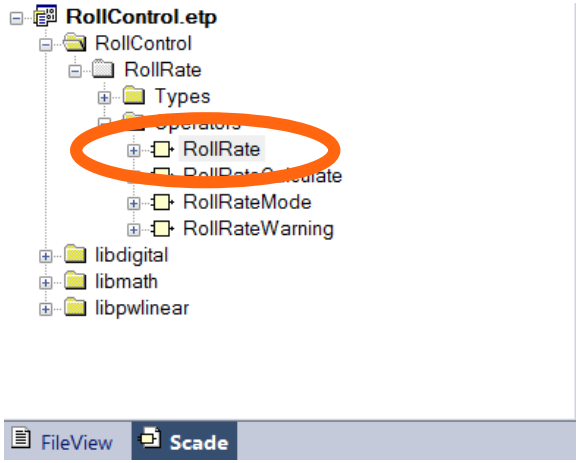
Generated code properties:

- Readable, traceable (names and annotations propagation)
- Portable (independent from the target and the system scheduler)
- Modular
- Static memory allocation
- Finite execution duration
- Size optimizations

© 2016 ANSYS, Inc.

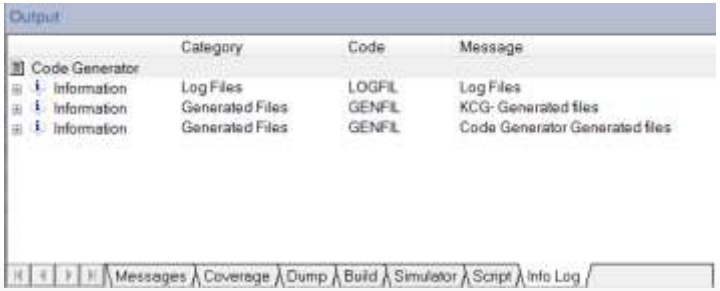
How to Generate Code from IDE

Select the operator you want to generate from the Scade view:



Generate the code:

Check in the “Info Log” tab if the code generation is successfully completed



© 2016 ANSYS, Inc.

Lab 10: Generate KCG C Code

Lab

Objective:

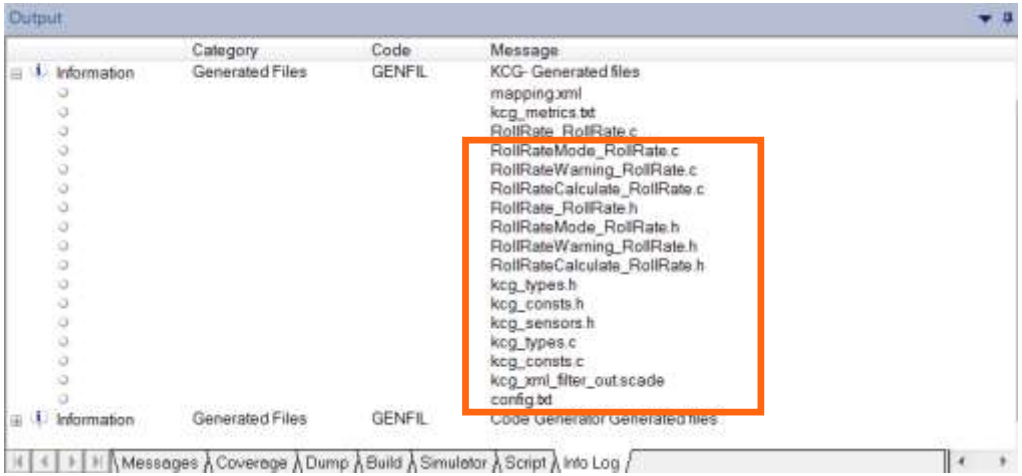
Generate KCG code and observe generated files

Requirements:

Generate the code with KCG configuration

Observe the generated files:

Time: 10 min



© 2016 ANSYS, Inc.

Lab 10: Generate KCG Ada Code

Ada

Lab

Objective:

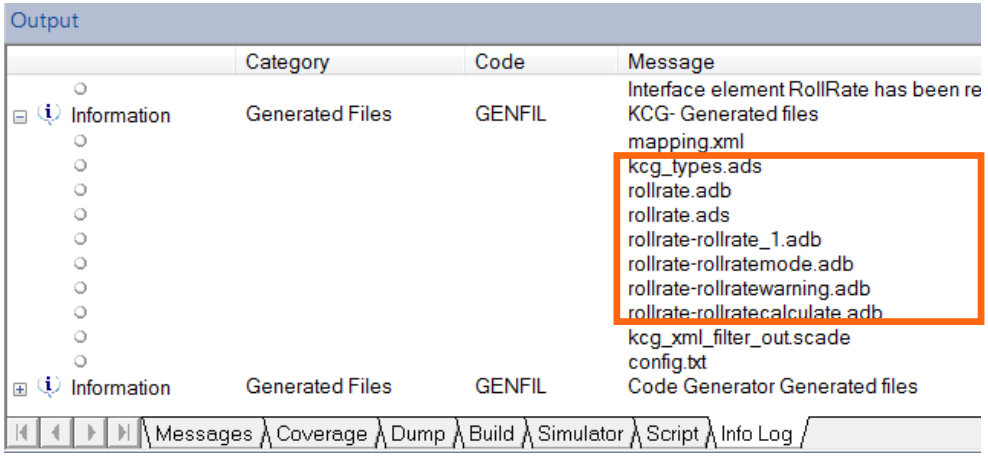
Generate KCG Ada code and observe generated files

Requirements:

Time: 10 min

Generate the code with KCGAda configuration

Observe the generated files:



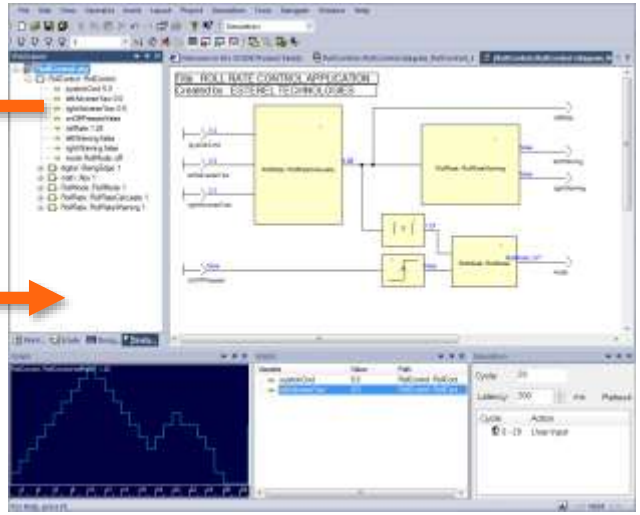
© 2016 ANSYS, Inc. February 1, 2017 ANSYS Confidential

Rapid Interactive Simulation (C Code Only)

Rapid Interactive Simulation of SCADE Suite models

- Quick & comfortable graphical SCADE Suite models validation
- Library of predefined widgets for rapid prototyping

SCADE Suite Rapid Prototyper
Control Panel



SCADE Suite Simulator
Embedded Controller Application

Target applications:

- Any SCADE Suite application in any industry domain (engine control, flight control, nuclear reactor power control, train control, etc.)

© 2016 ANSYS, Inc.

Standalone Executable (C Code Only)

Automatic generation of Standalone Executables:

- One-click
- Portable (runs on any Windows, Android or iOS mobile platform)
- No run-time fee
- Code integration easily customizable

© 2016 ANSYS, Inc.

Lab 11: Create a Standalone Executable (C Code Only)

Lab Support p.44-49

Objective:

Generate a Standalone Executable with a Graphical Panel

Requirements:

Time: 10 min

Add a SCADE Rapid Prototyper in your SCADE Suite project

Connect SCADE Suite I/Os with a SCADE Rapid Prototyper Panel I/Os

Create a new “Standalone Executable” configuration

Generate and execute the executable

Lab

© 2016 ANSYS, Inc.

AGENDA

Integrated Development Environment

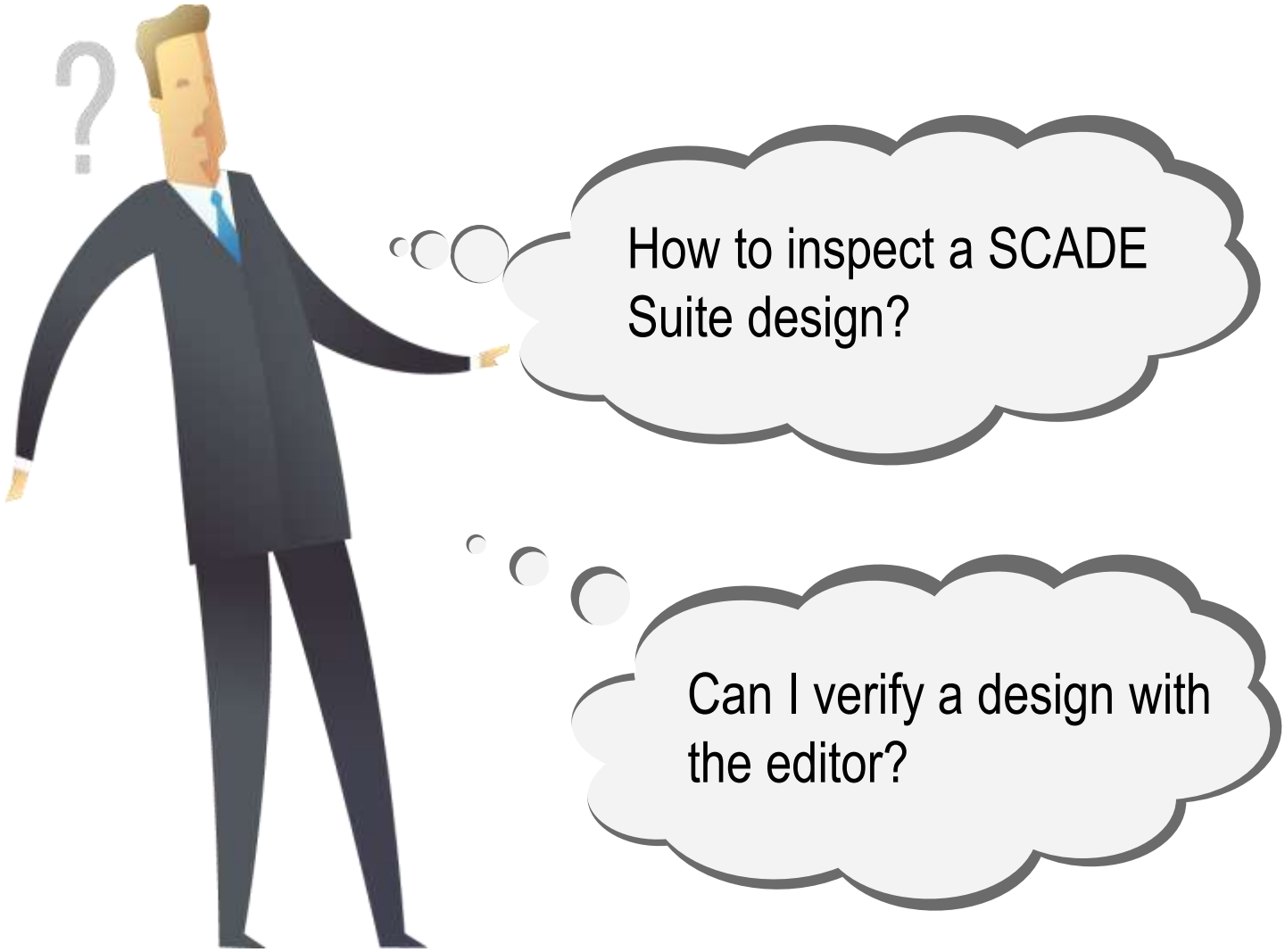
SCADE Suite Operator

Simulation introduction

Code generation and standalone executable

Report

© 2016 ANSYS, Inc.



 © 2016 ANSYS, Inc.

SCADE LifeCycle Reporter

Automatic project documentation generation enables:

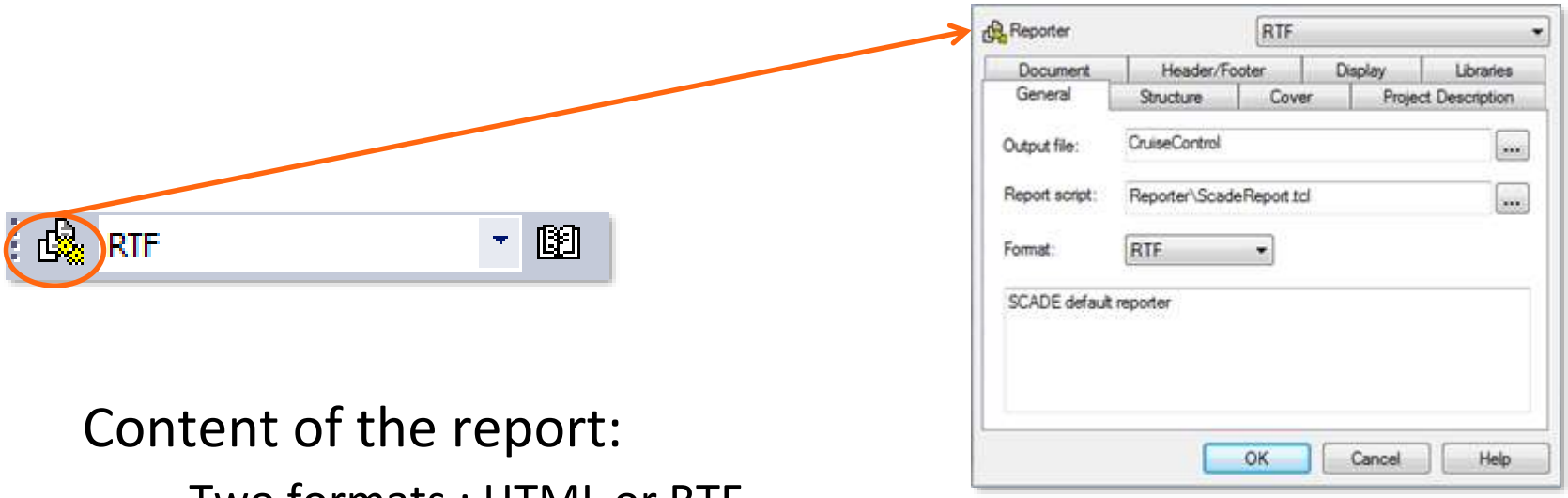
- Review of a SCADE Suite model
- Verification whether a SCADE Suite design complies with the current standards, and with Software Requirements

© 2016 ANSYS, Inc.

Reporter: General Settings

Generate project reports using predefined configurations (HTML or RTF) or user configurations

Use the toolbar to open the Settings dialog:



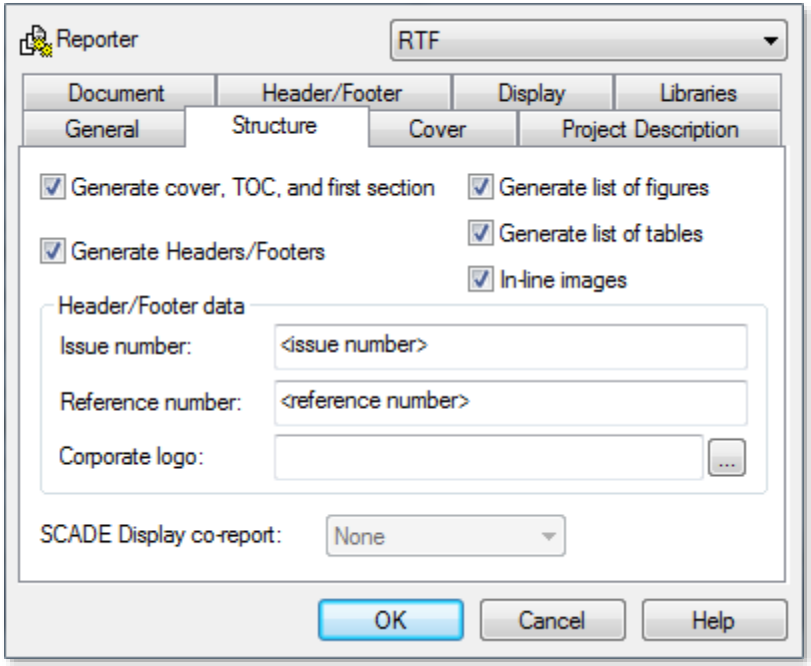
Content of the report:

- Two formats : HTML or RTF
- Choose a TCL script to generate a content fully customizable
  - A default one is available: ScadeReport.tcl

© 2016 ANSYS, Inc.

Reporter: Structure Settings

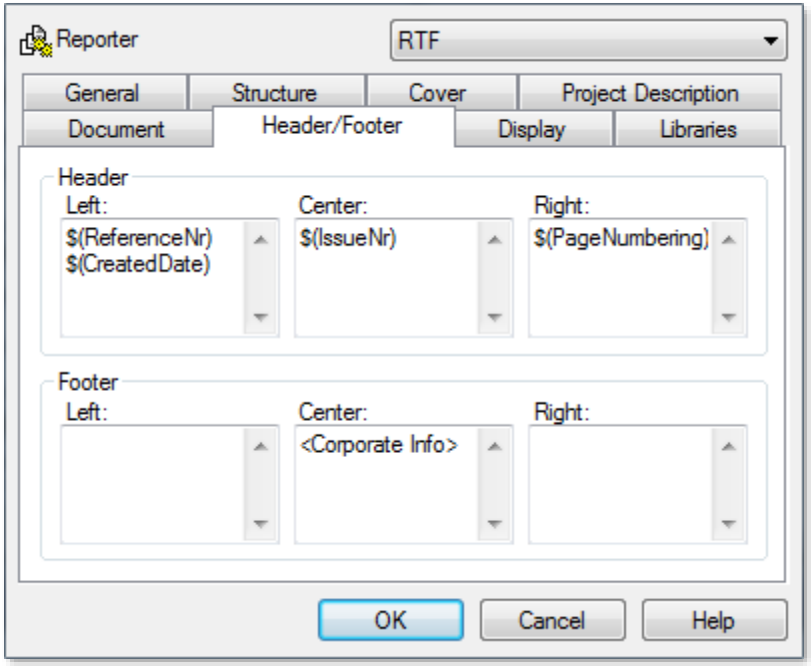
Customization of the structure (Call graph, Table of contents, etc..)



 © 2016 ANSYS, Inc.

Reporter: Structure Settings

Easy management of the headers/footers



 © 2016 ANSYS, Inc.

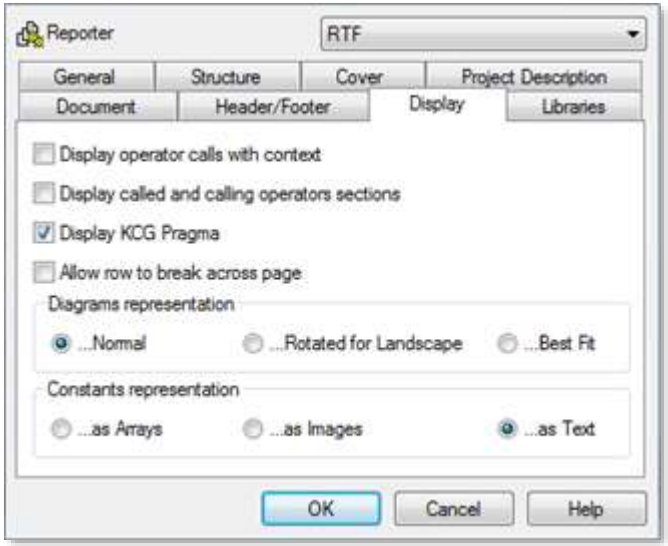
Display Options

Constants:

- Three possible representations :

Name	Type	Value	Comments/Annotations
Iengine	real	0.025	
KBRAKE	real	200.0	
KTRASM	Vec5	1: 4.75; 2: 2.68; 3: 1.87; 4: 1.42; 5: 1.17	
MASSE	real	1450.0	
TCYCLE	real	0.1	
Tengine	real	-0.04	
TorqMax	real	400.0	
VehicleDynamic	real	2.5	

As Arrays



Constant	Type	Value	Comments
Iengine	real	0.025	
KBRAKE	real	200.0	
KTRASM	Vec5		
1	real	4.75	
2	real	2.68	
3	real	1.87	
4	real	1.42	
5	real	1.17	
MASSE	real	1450.0	
TCYCLE	real	0.1	
Tengine	real	-0.04	
TorqMax	real	400.0	
VehicleDynamic	real	2.5	

As Images

Name	Type	Value	Comments/Annotations
Iengine	real	0.025	
KBRAKE	real	200.0	
KTRASM	Vec5	[4.75 , 2.68 , 1.87 , 1.42 , 1.17]	
MASSE	real	1450.0	
TCYCLE	real	0.1	
Tengine	real	-0.04	
TorqMax	real	400.0	
VehicleDynamic	real	2.5	

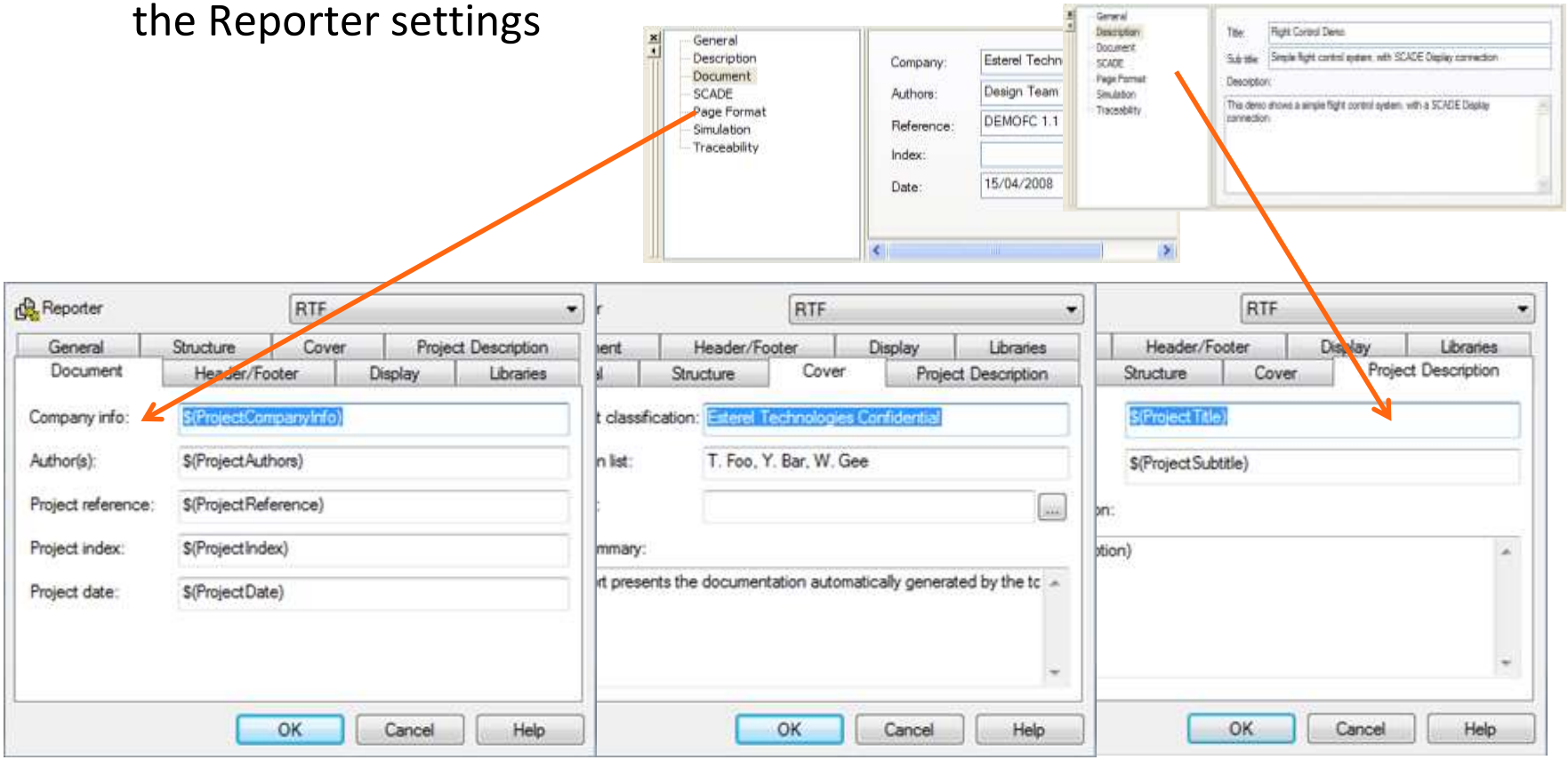
As Text

© 2016 ANSYS, Inc.

Custom Attributes

Manage document attributes (classification, distribution list, …)

Customize cover, structure, description from project properties or from the Reporter settings



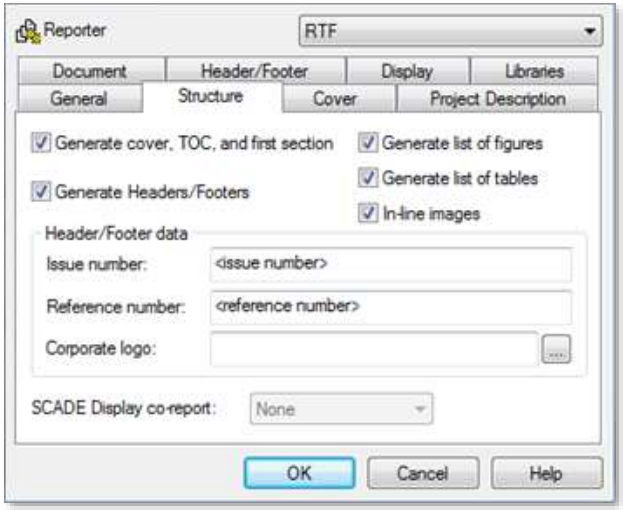
 © 2016 ANSYS, Inc.

SCADE Display Integration

When there is at least one SCADE Display specification in the project or one operator connected to a specification.

SCADE display co-report:

- Complete: generate the "SCADE Display Integration" section in the "Software Architecture" chapter
- Connection Tables: same as "Complete", without the generation of the SCADE Display reports
- None: no report of SCADE Display elements



 © 2016 ANSYS, Inc.

Report Generation

After selecting a configuration, to produce a report, click on:

A HTLM or RTF document according to the selected format is displayed.

 © 2016 ANSYS, Inc.

Report Document

Example

<document classification>

Pilot Example

Auto Pilot

ESTEREL Technologies

Summary:
<summary>

Company: Esterel Technologies
Authors: S.SABATHIER
Reference: 1.4
Index: 2
Date: 19/10/2007

Distribution List: <distribution list>

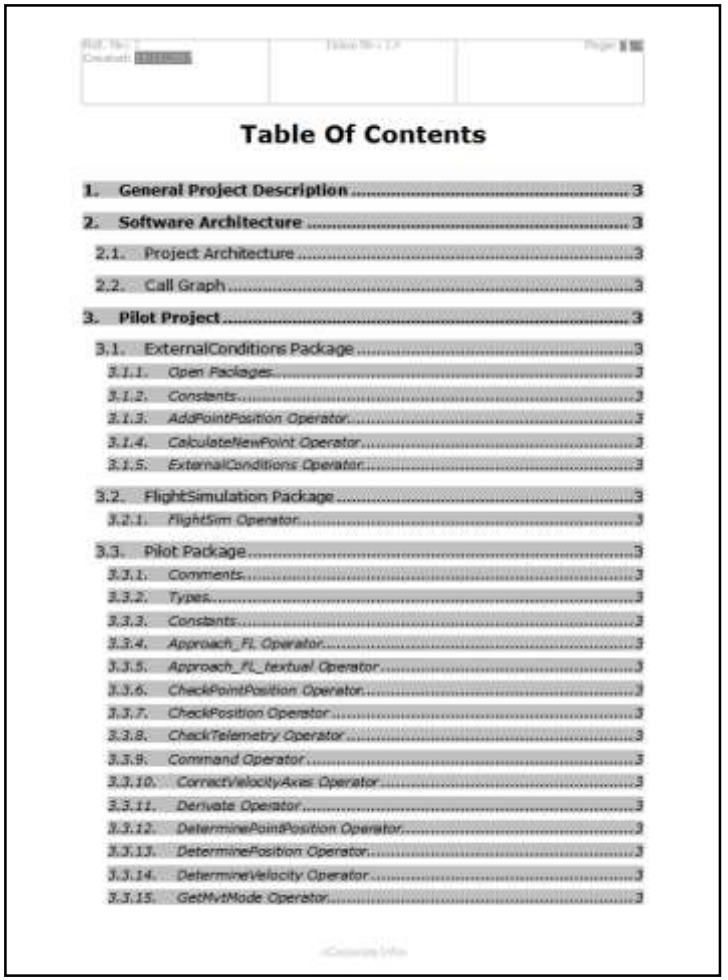


Table Of Contents

1. General Project Description 3
2. Software Architecture 3
2.1. Project Architecture 3
2.2. Call Graph 3
3. Pilot Project 3
3.1. ExternalConditions Package 3
3.1.1. Open Packages 3
3.1.2. Constants 3
3.1.3. AddPointPosition Operator 3
3.1.4. CalculateNewPoint Operator 3
3.1.5. ExternalConditions Operator 3
3.2. FlightSimulation Package 3
3.2.1. FlightSim Operator 3
3.3. Pilot Package 3
3.3.1. Comments 3
3.3.2. Types 3
3.3.3. Constants 3
3.3.4. Approach_FL Operator 3
3.3.5. Approach_FL_textual Operator 3
3.3.6. CheckPointPosition Operator 3
3.3.7. CheckPosition Operator 3
3.3.8. CheckTelemetry Operator 3
3.3.9. Command Operator 3
3.3.10. CorrectVelocityAxis Operator 3
3.3.11. Derivate Operator 3
3.3.12. DetermineP0intPosition Operator 3
3.3.13. DeterminePosition Operator 3
3.3.14. DetermineVelocity Operator 3
3.3.15. GetMvtMode Operator 3

© 2016 ANSYS, Inc.

Report Document

Example

2. Software Architecture

2.1. Project Architecture

This section displays the package hierarchy of projects.

Project Pilot
- ExternalConditions
- FlightSimulation
- Pilot

2.2. Call Graph

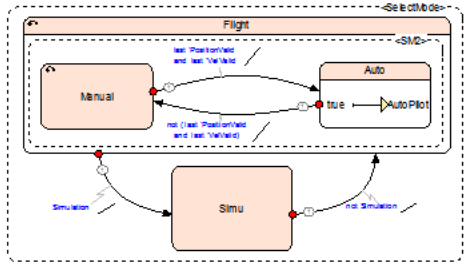
This Call Graph displays the dependency tree of model operators.

1. FlightSimulation::FlightSim
   - 1.1. ExternalConditions::ExternalConditions (L59=)
     - 1.1.1. ExternalConditions::CalculateNewPoint (L69=)
       - 1.1.1.1. ExternalConditions::AddPointPosition (L9=)
   - 1.2. Pilot::Pilot (L56=)
     - 1.2.1. Pilot::CheckPosition (L37=)
       - 1.2.1.1. Pilot::CheckTelemetry (L13=)
         - 1.2.1.1.1. Pilot::CheckPointPosition (L25=)
         - 1.2.1.1.2. pwlinear::ClockCounter (L31=)
       - 1.2.1.2. Pilot::DeterminePosition (L47=)
         - 1.2.1.2.1. Pilot::DeterminePointPosition (L3=)
       - 1.2.1.3. Pilot::DetermineVelocity (L19=)
         - 1.2.1.3.1. Pilot::Derivate (L52=)
         - 1.2.1.3.2. Pilot::Module (L55=)
           - 1.2.1.3.2.1. mathext::SqrtR (L10=)
           - 1.2.1.3.2.2. Pilot::SquareSum (L12=)
     - 1.2.2. Pilot::Command (L24=)
       - 1.2.2.1. Pilot::MvtCalculus (TheoricVel=)
         - 1.2.2.1.1. Pilot::Approach_FL_textual (L48=)
         - 1.2.2.1.2. Pilot::Rising_FL_imported (L46=)
         - 1.2.2.1.3. Pilot::Waiting_FL (L45=)
       - 1.2.2.2. Pilot::VelocityRegulation (L21=)
         - 1.2.2.2.1. Pilot::CorrectVelocityAxes (L144=)
     - 1.2.3. Pilot::GetMvtMode (L30=)
2. Pilot::Approach_FL
3. Pilot::Rising_FL

3.2.1.1. Node Hierarchy

state-machine : SelectMode
- state : Simu
- state : Flight
  - state-machine : SM2
    - state : Manual
    - state : Auto

3.2.1.2. Graphical Views



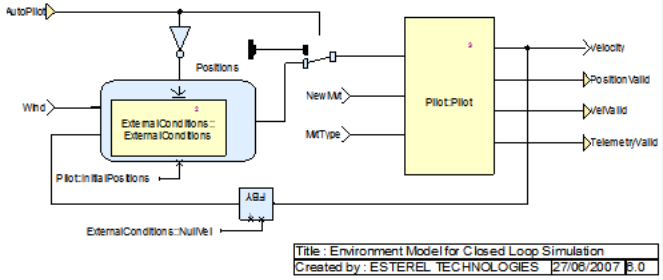


Figure 4: FlightSim

Table 13: Transitions of FlightSim

Source/Target	#	Conditions/Actions	Comments/Annotations
Source: SelectMode:Flight:SM2:Manual **Target:** SelectMode:Flight:SM2:Auto	1	**Condition:** last 'PositionValid and last 'VelValid	

<Corporate Info>

© 2016 ANSYS, Inc. February 1, 2017 ANSYS Confidential

Qualified Verification Tool

SCADE LifeCycle Reporter qualified as verification tool:

- Certification Kit DO-178C Level A (Criteria 3 / TQL-5)
- Qualified for SCADE Suite and SCADE Display
- Qualified report generation only in the batch mode
- Qualified tool for the RTF format (forced option) and a defined environment:
  - Use the required “Reporter\ScadeQualifiedReport.tcl” script
  - A dedicated reporter configuration named “Qualified Reporter” is available

 © 2016 ANSYS, Inc.

Qualified Verification Tool

In the scope of the qualification, several “standard” reporter properties values are forced as follows:

Reporter property	Forced value	Reporter GUI tab	Reporter GUI field name
ImagesInLineWithText	true (default)	Structure	In-line images
ReportHeaderAndFooter	true (default)	Structure	Generate Headers/Footers
ReportStructure	true (default)	Structure	Generate cover, TOC, and first section
AllowRowToBreak	true	Display	Allow row to break across page
cstDisplayType	Flat	Display	Constants representation/ …as Text
DisplayCalledAndCalling	true	Display	Display called and calling operators sections
DisplayKCGPragma	true	Display	Display KCG Pragma

© 2016 ANSYS, Inc.

Batch Generation

```
SCADE -report <project> -configuration <configuration>
```

Notes:

- <project> "project name".etp
- To call the qualified reporter, the "reporter script" property must be set to "Reporter\ScadeQualifiedReport.tcl" in the configuration

© 2016 ANSYS, Inc.

Advanced TCL Customization

The customization of SCADE LifeCycle Reporter can be performed via TCL scripts:

- Use Reporter-related TCL commands to set the content and display of model documentation and reports (show details in the User Manual document)

The customization script files must be registered in SCADE Suite environment for proper evaluation.

Use this tool with user customized TCL procedures (or options outside the qualification context), need additional verification activities.

© 2016 ANSYS, Inc.

Lab 12: Generate a Design Report

Lab

Objective:

Generate a RTF report of your design

Requirements:

Time: 10 min

Select RTF format and generate a design report

Open the generated report

Change settings and regenerate. Observe the differences

© 2016 ANSYS, Inc.

Contacts

Legal Contact
Esterel Technologies SAS
14/15, Place Georges Pompidou
78180 Montigny Le Bretonneux
FRANCE
Phone: +33 1 30 68 61 60
Fax: +33 1 30 68 61 61

Technical Support
Esterel Technologies SAS
Parc Avenue - 9 rue Michel Labrousse
31100 Toulouse FRANCE
Phone: +33 5 34 60 90 50
Fax: +33 5 34 60 90 41

Submit questions to Technical Support: scade-support@ansys.com
Contact one of our Sales representatives at: scade-sales@ansys.com
Direct general questions about Esterel Technologies to: scade-info@ansys.com
Discover the latest news on our products and technology at: http://www.ansys.com/products/embedded-software

Legal Information

Copyrights ©2017 ANSYS, Inc. All rights reserved. ANSYS®, SCADE®, SCADE Suite®, SCADE Display®, SCADE Architect®, SCADE LifeCycle® are trademark or registered trademarks of ANSYS, Inc or its subsidiaries in the U.S. or other countries. All other trademarks and trade names contained herein are the property of their respective owners.