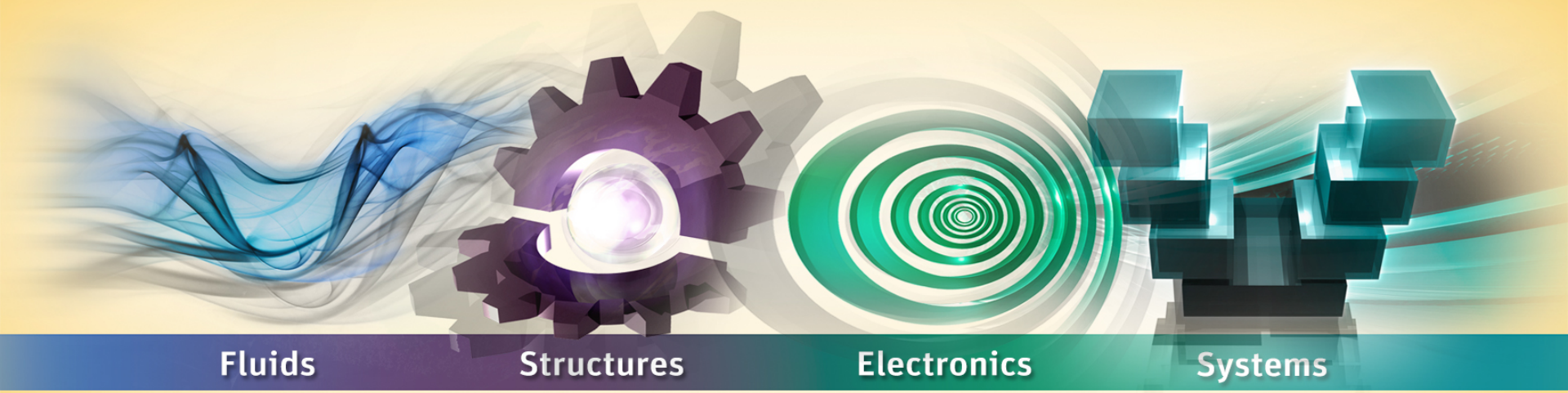
ANSYS

Realize Your Product Promise®

SCADE Suite Modeling with Import of Simulink and Stateflow Models



ESTEREL Technologies

Agenda

SCADE vs. Simulink

From Simulink to SCADE

From Stateflow to SCADE

From SCADE to Simulink

SCADE VS. SIMULINK

SCADE vs. Simulink: SCADE

SCADE SUITE®

- Safety critical software design
- Software simulation & verification
- Automatic embeddable code generation

SCADE vs. Simulink: Matlab/Simulink

Physical & regulation laws description (hydraulic, mechanic, electric…)

System Prototyping

Simulation at physical system level

The Simulink Gateway supports models built from Simulink/Stateflow 7.0 to 8.1 (Matlab R2007b to R2013a) file formats on input

SCADE vs. Simulink: Different Tools For Different Purposes

Simulink: Simulation environment

- Prototyping. Excellent at quickly representing graphically numerical equations/control laws, and simulating them
- Extremely flexible requiring no programming constraint
- But not designed by construction to generate safety critical code

SCADE: SW Design environment for critical control systems

- The SCADE methodology, technology and tools are designed to meet the strictest embedded software requirements (safety critical systems in avionics and automotive industry)
- SCADE offers a fully integrated and seamless design environment from specification to safe embedded production code qualifiable to strict industry standards

Components Of The Gateway

Simulink Translator

- Translates discrete controllers specified in Simulink models into SCADE

Stateflow Importer

- Graphically and syntactically translates Stateflow diagrams into SCADE State Machines

Simulink Wrapper

- Allows simulation of SCADE models by Simulink in:
  - black box (SCADE generated code running in a S-Function)
  - white box (co-simulation: SCADE as a slave simulator)

FROM SIMULINK TO SCADE

Principles

Translate a Project

Updating the Translation

Translated Blocks/Supported Blocks

Simulink Model Prerequisites

Type Inference

Note: Modular translation, customization using UCF files and exercises are addressed in the corresponding advanced training course

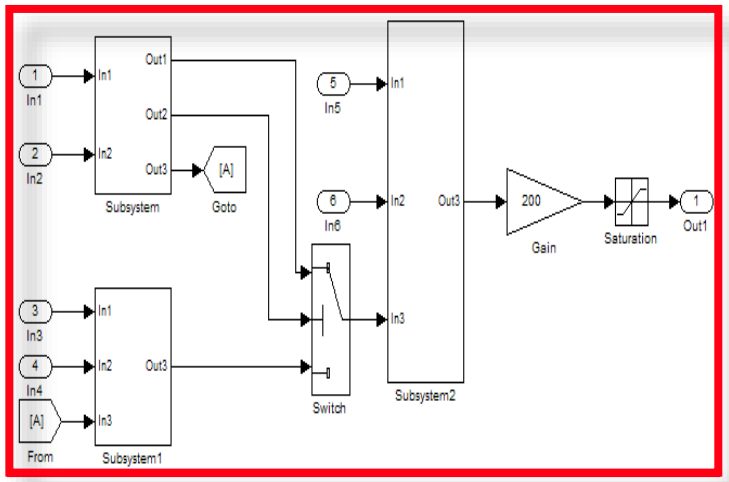
Principles

Structure preserved: same hierarchy

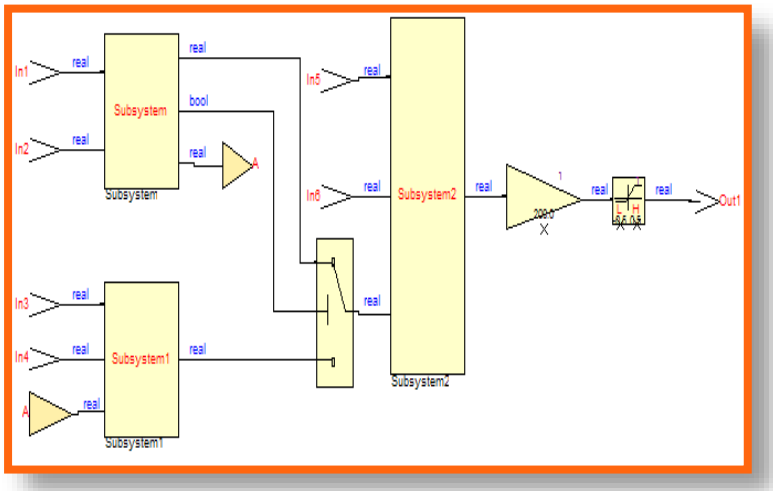
Graphical look kept: element position and size

Interface and names kept (size, variables, operators, …)

Matlab language eligible subset is translated



Simulink

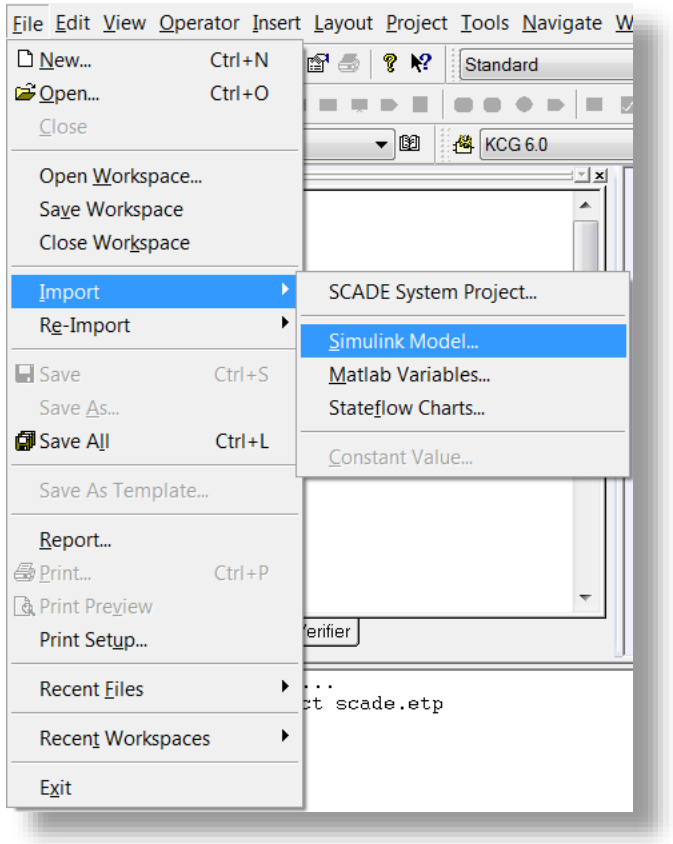


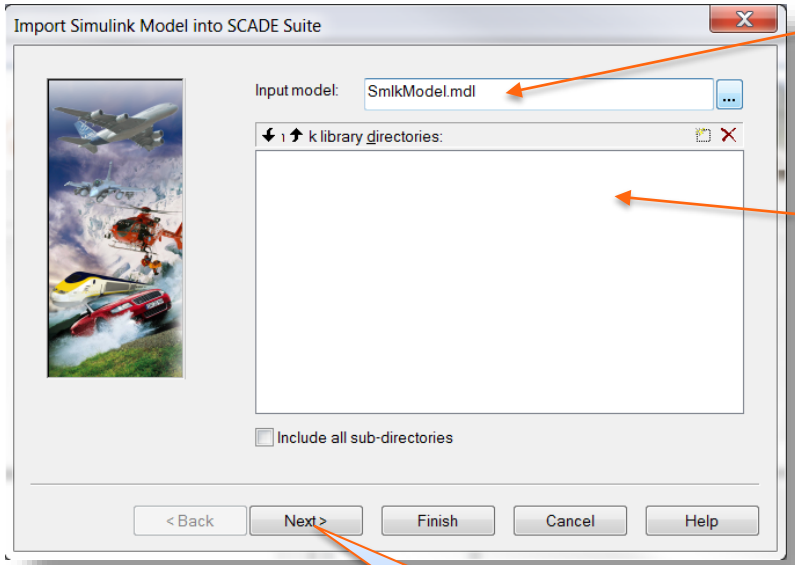
SCADE

Translate a Simulink Model

Create a new project in SCADE Suite

File -> Import -> Simulink Model



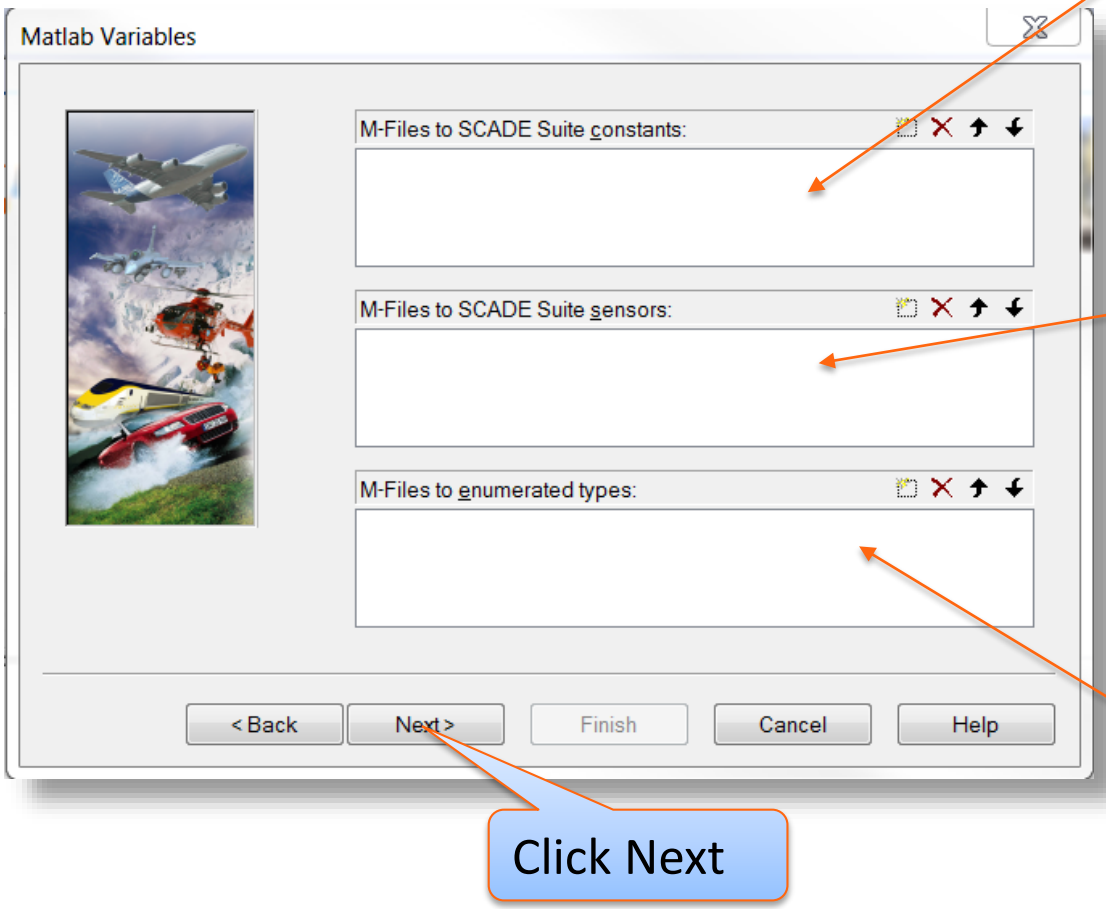


The Simulink model file (MDL or SLX files)

Include path for Simulink libraries

Click Next

The Simulink Model Properties



Specify the M-files that contain Matlab variables to translate into SCADE constants

Specify the M-files that contain Matlab variables to translate into SCADE sensors

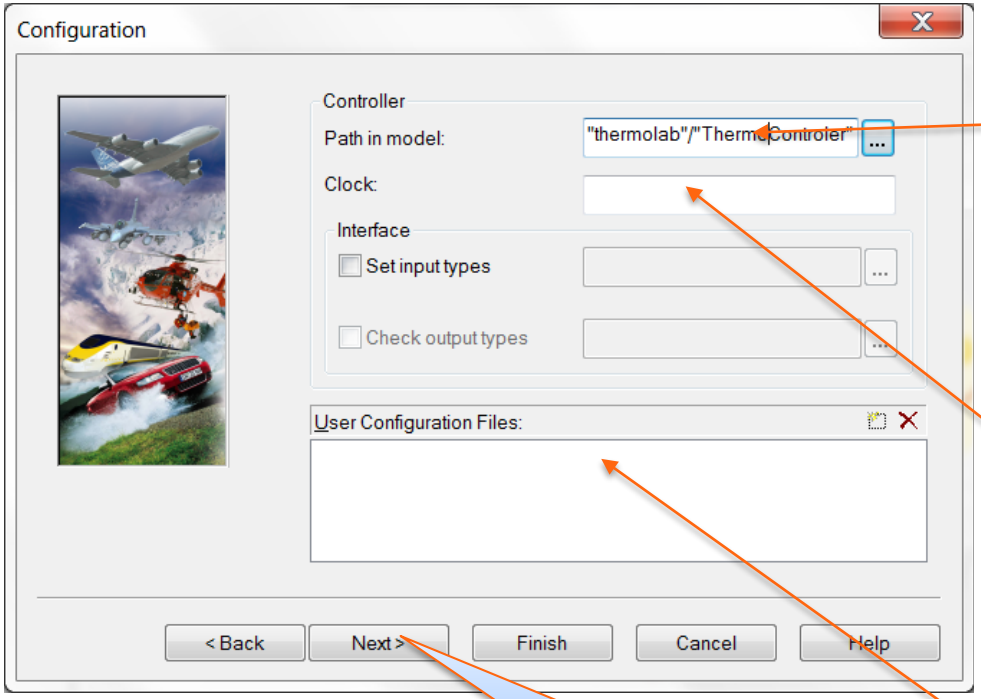
Specify the M-files that contain Matlab variables to translate into SCADE enumerated types

Click Next

The Controller Properties (1/2)

All fields are optional

All fields can be set in the UCF (for experts)



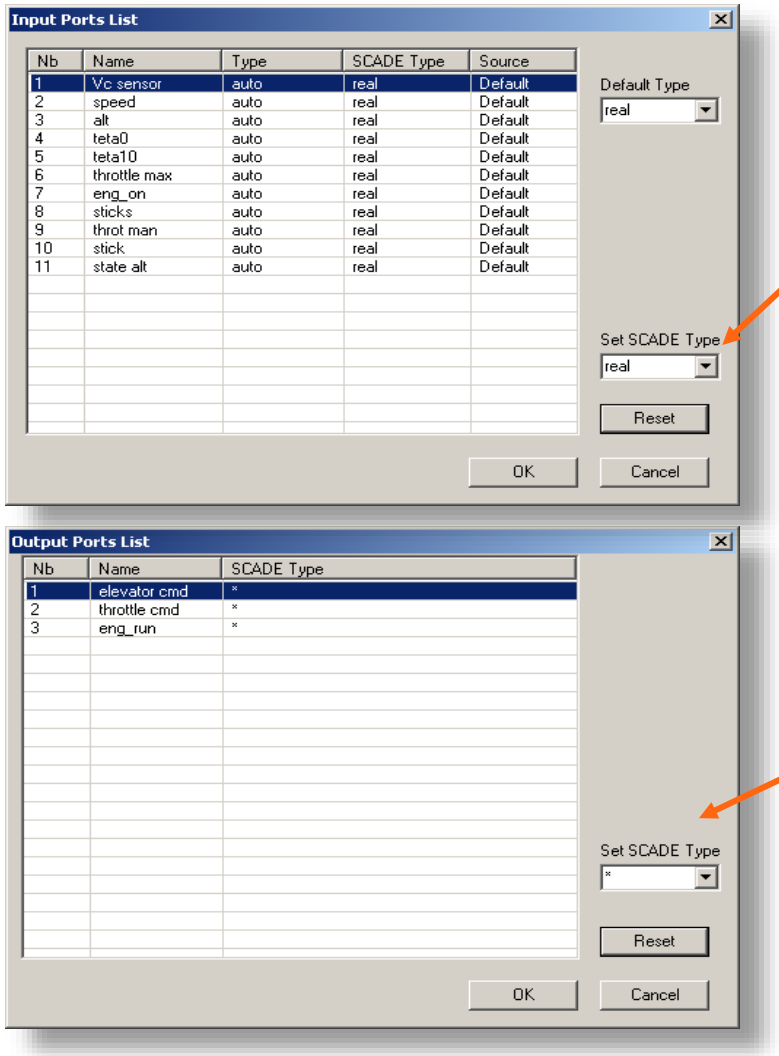
Path identifying the subsystem that is to be translated. If this field is empty the whole Simulink model is translated

Sets the clock (sample time value) of the Controller inputs

UCF files: to customize the translation in the advanced mode

Click Next

The Controller Properties (2/2)



- **Set the inputs types:**

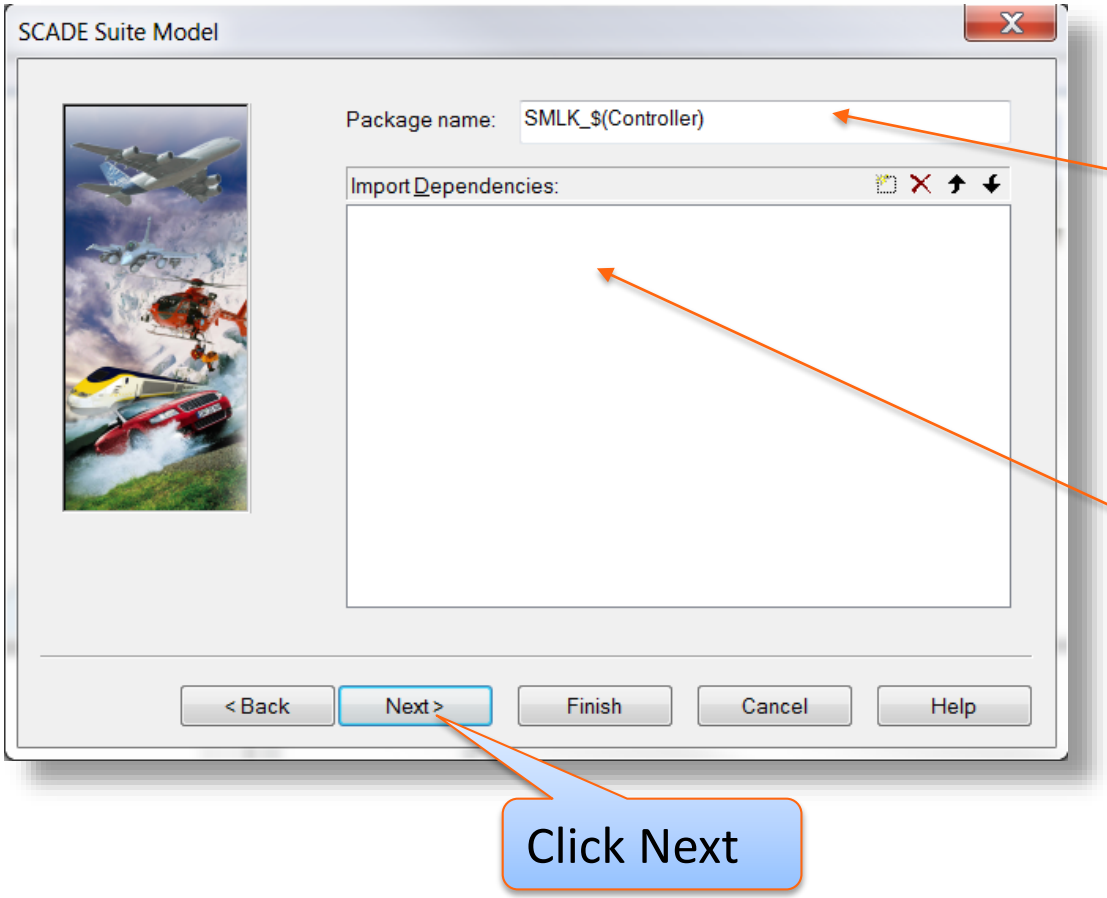
1. Select the inputs to modify
2. Chose a basic type in “set SCADE Type”
3. For array, in the “set SCADE Type” field, write int^3

- **Check the outputs types:**

1. Select the outputs to modify
2. Chose a basic type in “Set SCADE Type”
3. For array, in the “Set SCADE Type” field, write int^3

Tip : Use the "*" symbol as SCADE type for outputs to prevent the gateway from checking the type of one or several outputs specified in the output ports list

Translation Options (1/2)



Name of the package that will be created

Specify the list of packages previously translated (see modular translation in next part)

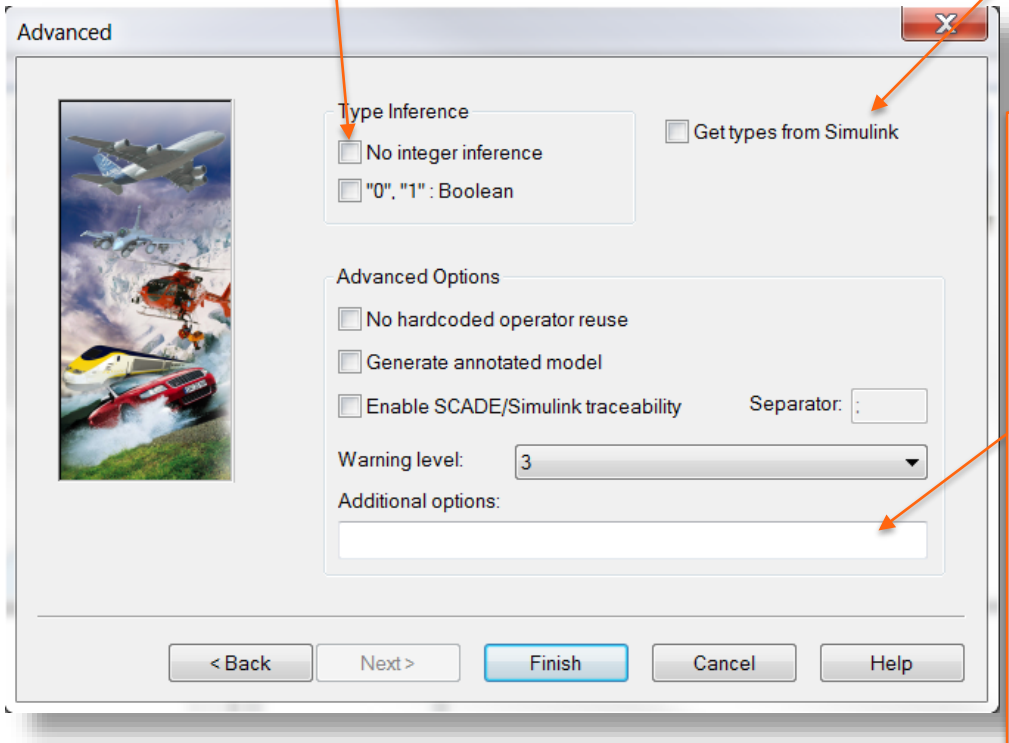
Click Next

Translation Options (2/2)

Type inference customization:

- Integers inferred as real
- 0/1 literals inferred as bool

Disable type inference : use types inferred by Simulink

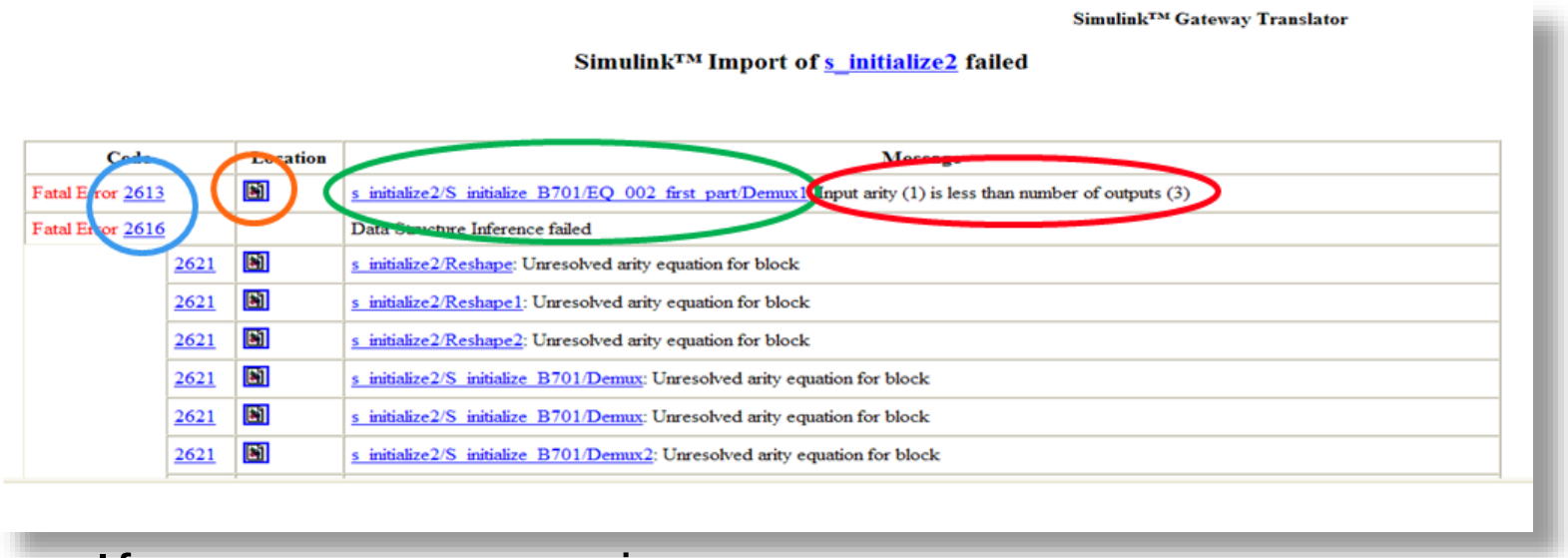


Additional options (expert use)

- -C<FileName>: replaces default main configuration file by the specified file.
- -q<FileName>: overrides the default Annotation Configuration files.
- -n<integer>: sets the maximum valid name length in a range from 5 to 255 (Default = 25).

Translation Results

- Possibly error and warning messages:
  - Problem type ID and link to details
  - Shortcut to Simulink or Stateflow model
  - Path to the Simulink block or model source of the problem
  - Nature of the problem



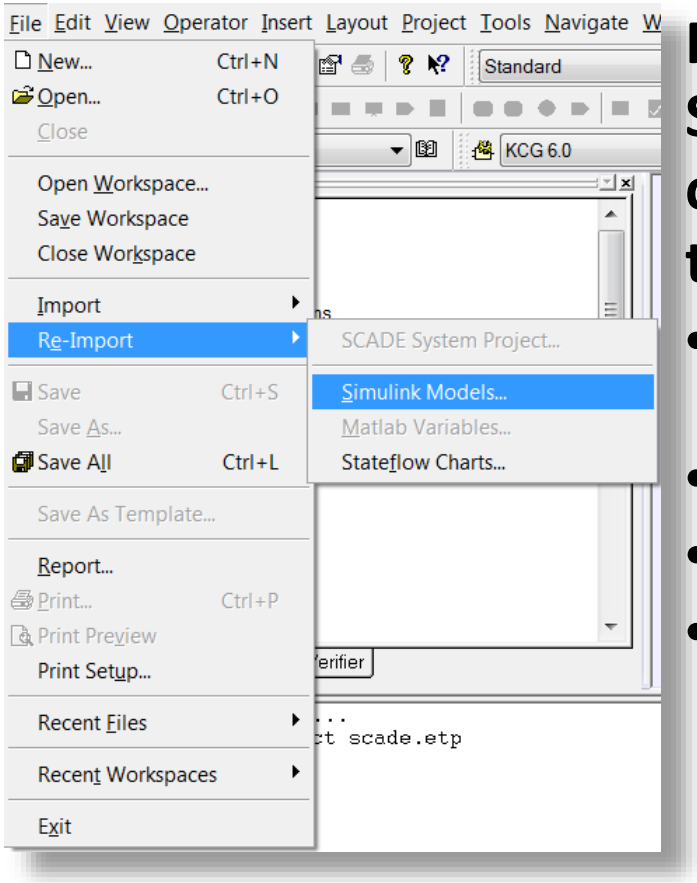
Simulink™ Gateway Translator

Simulink™ Import of s_initialize2 failed

Code	Location	Message
Fatal Error 2613		s_initialize2/S_initialize_B701/EQ_002_first_part/Demux1: Input arity (1) is less than number of outputs (3)
Fatal Error 2616		Data Structure Inference failed
	2621	s_initialize2/Reshape: Unresolved arity equation for block
	2621	s_initialize2/Reshape1: Unresolved arity equation for block
	2621	s_initialize2/Reshape2: Unresolved arity equation for block
	2621	s_initialize2/S_initialize_B701/Demux: Unresolved arity equation for block
	2621	s_initialize2/S_initialize_B701/Demux: Unresolved arity equation for block
	2621	s_initialize2/S_initialize_B701/Demux2: Unresolved arity equation for block

- If no errors or warnings
  - Imported model
  - Success message

Translation updates



Re-Import: The same Simulink (or Stateflow or Matlab variables) model can be re-translated after the initial translation.

- **SCADE generated package is overwritten by the new translation**
- **All previously generated files are discarded**
- **Files added by the user are kept**
- **If update translation fails, the previous generated package is kept. The import wizard remembers the last entered options.**

Translated Blocks/Supported Blocks

Supported Simulink blocks are translated using:

- **Predefined mapping: Main Configuration File (MCF) defines the mapping of Simulink blocks to SCADE operators.**
- **Built-in rules (complex blocks): the mapping is "hardcoded" in the Translator when MCF cannot define the mapping.**
- **User-provided mapping: User Configuration File (UCF) can be passed to the translator to support additional blocks**

Supported blocks and built-in rules are given in the guidelines

Most of the discrete blocks are supported

Typical unsupported blocks are blocks incompatible with design of safety critical embedded system (Ex: State Space block, etc.)

Simulink Model Prerequisites

Simulink model must be discrete

The import starts from a single hierarchical root (called hereafter Controller)

Algebraic loops are prohibited (introduce ambiguity and/or unbounded execution time)

The model must only use supported Simulink blocks

Type Inference

In SCADE, everything must be typed; in Simulink the typing is not mandatory. The type of all Simulink variables is inferred:

Automatically:

- **Types**

Simulink	SCADE
`boolean`	`bool`
`uint8, int8, uint16, int16, uint32, int32`	`int`
`float,double`	`real`

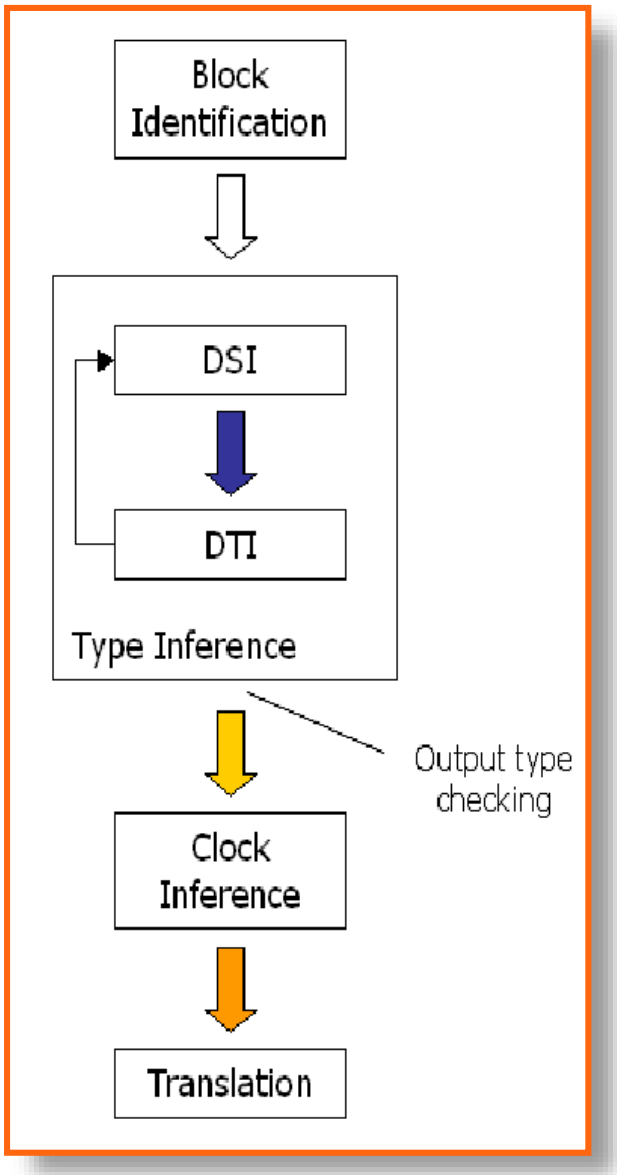
- **Constants**

Simulink	SCADE
`2.0`	`real`
`2`	`int (or real on option)`
`1`	`int (or bool or real on option)`
`True/False`	`bool`

Manually by the user:

- **UCF file: The user specifies the type inference of the block**
- **Input ports: The user specifies the type inference in the interface of the model**

Data Structure Inference (1/2)



Block identification: identifies Simulink blocks to determine the configuration to apply: Configured block, Hardcoded block, Unknown block

Type Inference: computes the structure (scalar, array, …) and types (bool, real, int) by analyzing the signal values and the translation rules (MCF / UCF files and hardcoded rules)

Clock inference: infers the clock of all signals

Translation: Identified Simulink blocks are translated into SCADE operators according to the rules

Data Structure Inference (2/2)

The Simulink gateway creates:

- Type variables: for each output of each block
- Type equations: between the inputs and outputs of the blocks

Type equations system is then solved:

- To find the value of each variable
- To determine the Data Structure of each signal

The structure can be:

- A scalar
- An array of known dimensions and sizes
- A bus whose field names are known and whose field types are scalar, array or bus.

Matlab Variables

Variable in Matlab ➔ constant or sensor in SCADE

.m files are evaluated:

.m file	SCADE
`A=2.5`	`real A=2.5`
`B=1+A`	`real B=3.5`
`V=[2,A]`	`Vector_2r V={2.0,2.5}`

Expression with Matlab variables ➔ SCADE expression

Sample Time vs SCADE Clock (1/3)

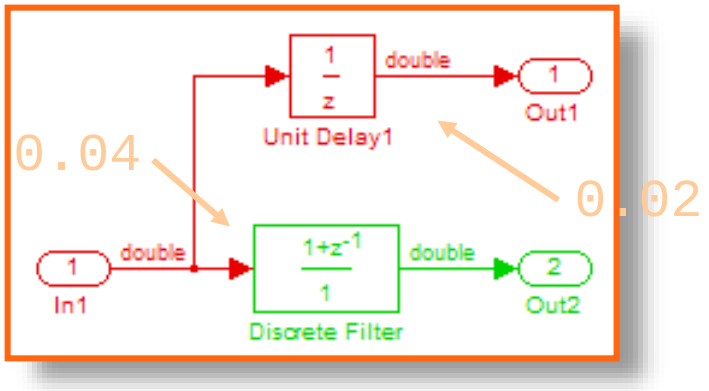
Simulink time	SCADE time
• Each Simulink block has a sample time (in physical units)	• SCADE root node is periodically called (basic clock)
• For discrete blocks, sample time is a period and an offset	• Using “activate” higher order operators, sub-nodes can be run at multiples of basic clock

Clock inference produces "logical" period and offsets (multiples of the basic clock), based on:

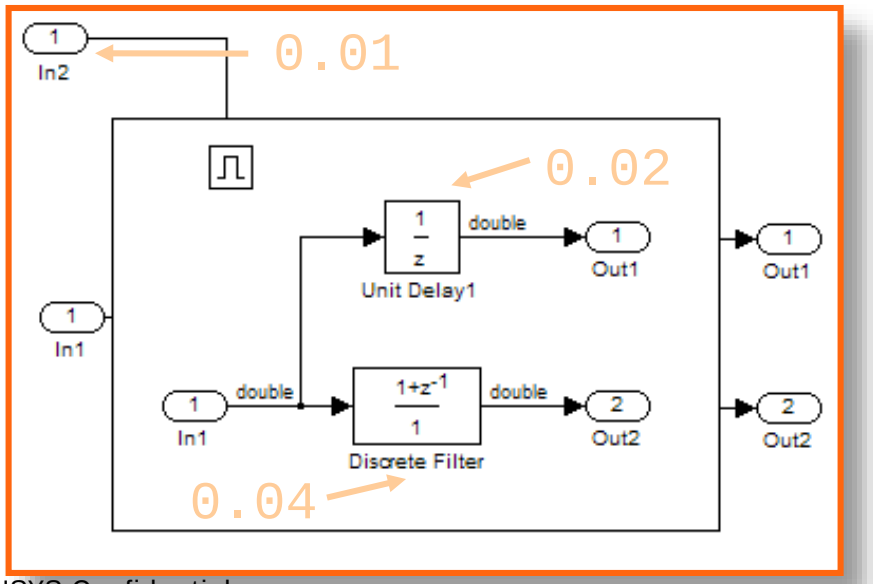
- Sample time of Simulink blocks
- Clock value from the wizard, giving the Sample Time of controller's inputs

Sample Time vs SCADE Clock (2/3)

- General clock or sample time is set to 0.01
- In Simulink: two different sample times:
  - 0.04
  - 0.02



Nearly equivalent



Sample Time vs SCADE Clock (3/3)

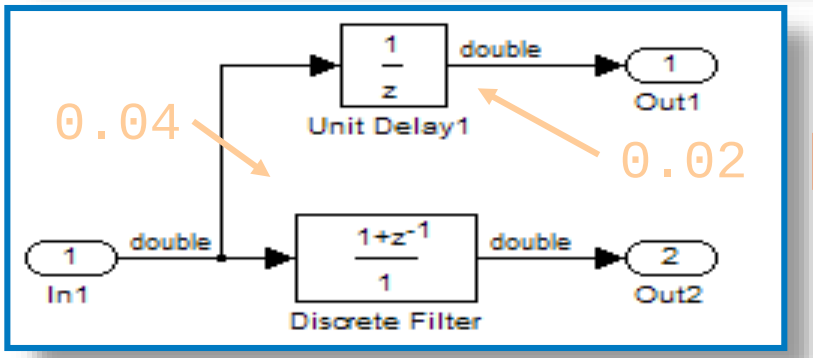
- **After translation:**

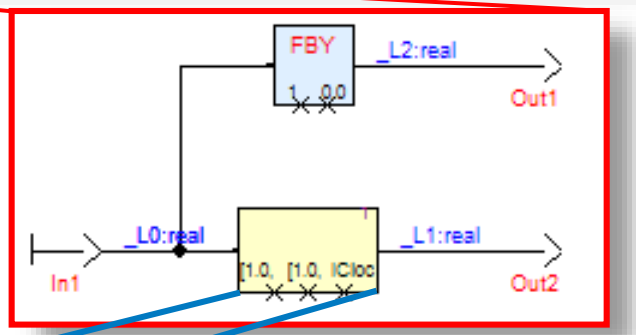
Subsystems or blocks whose clock is different from their parent subsystem are activated through an ”activate”

```
IClock_4 = smlk::ClockGen(4, 0);
IClock_2 = smlk::ClockGen(2, 0);
_L0 = 0.0;
_L1 = 0.0;
Output1, Output2 = (activate clockTest every IClock_2 initial default (_L0, _L1))(Input1, IClock_4);
```

True each 4 cycles, no offset

True each 2 cycles, no offset






```
_L0 = 0.0;
Output1 = (activate filters::Filter11 every IClock_4 initial default (_L0))(Input1, Input2, Input3);
```

Translation in a textual ”activate” operator

FROM STATEFLOW TO SCADE

State Machines

Translate a State Machines Project

Translation of Graphical Constructs

Translation of Transitions

Translation of Connective Junctions

Translation of Transition Actions

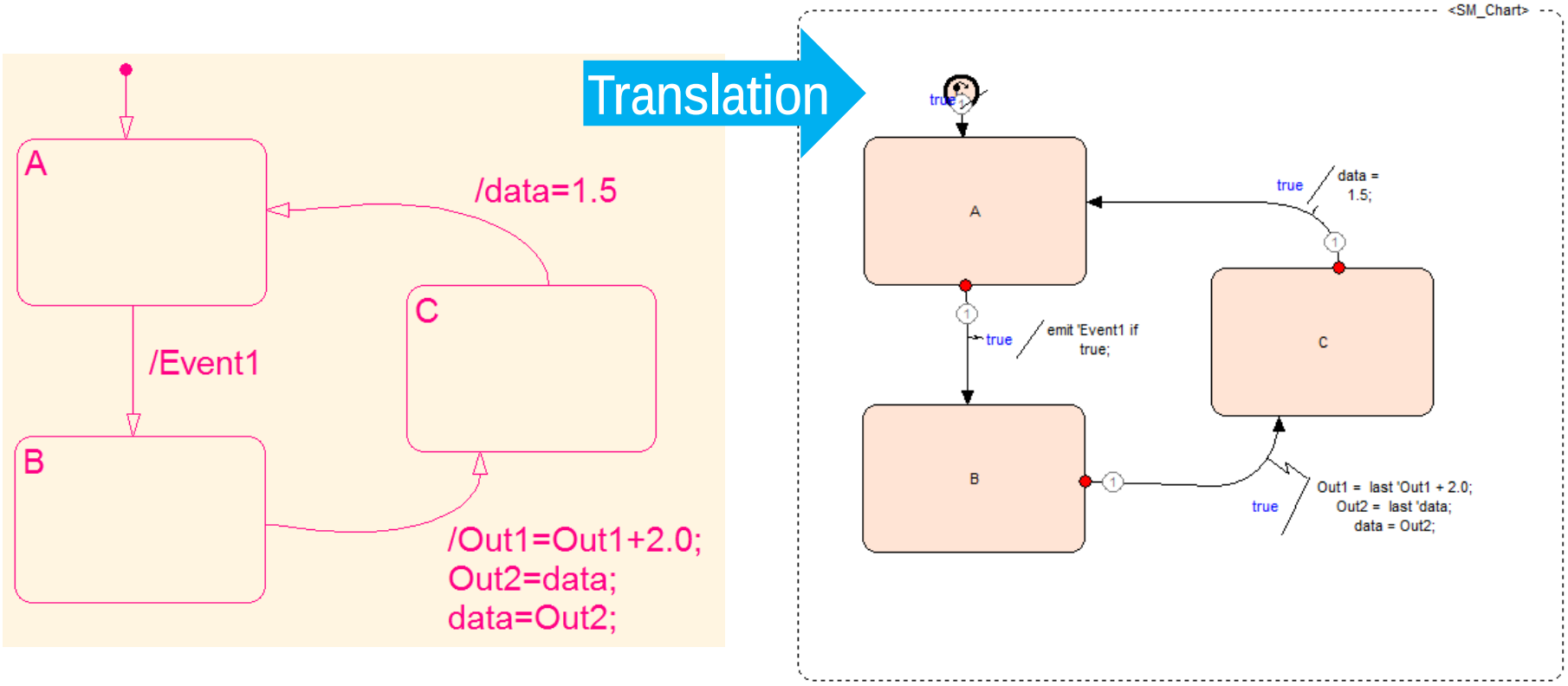
Translation Limitations

Note: exercises are addressed in the corresponding training course

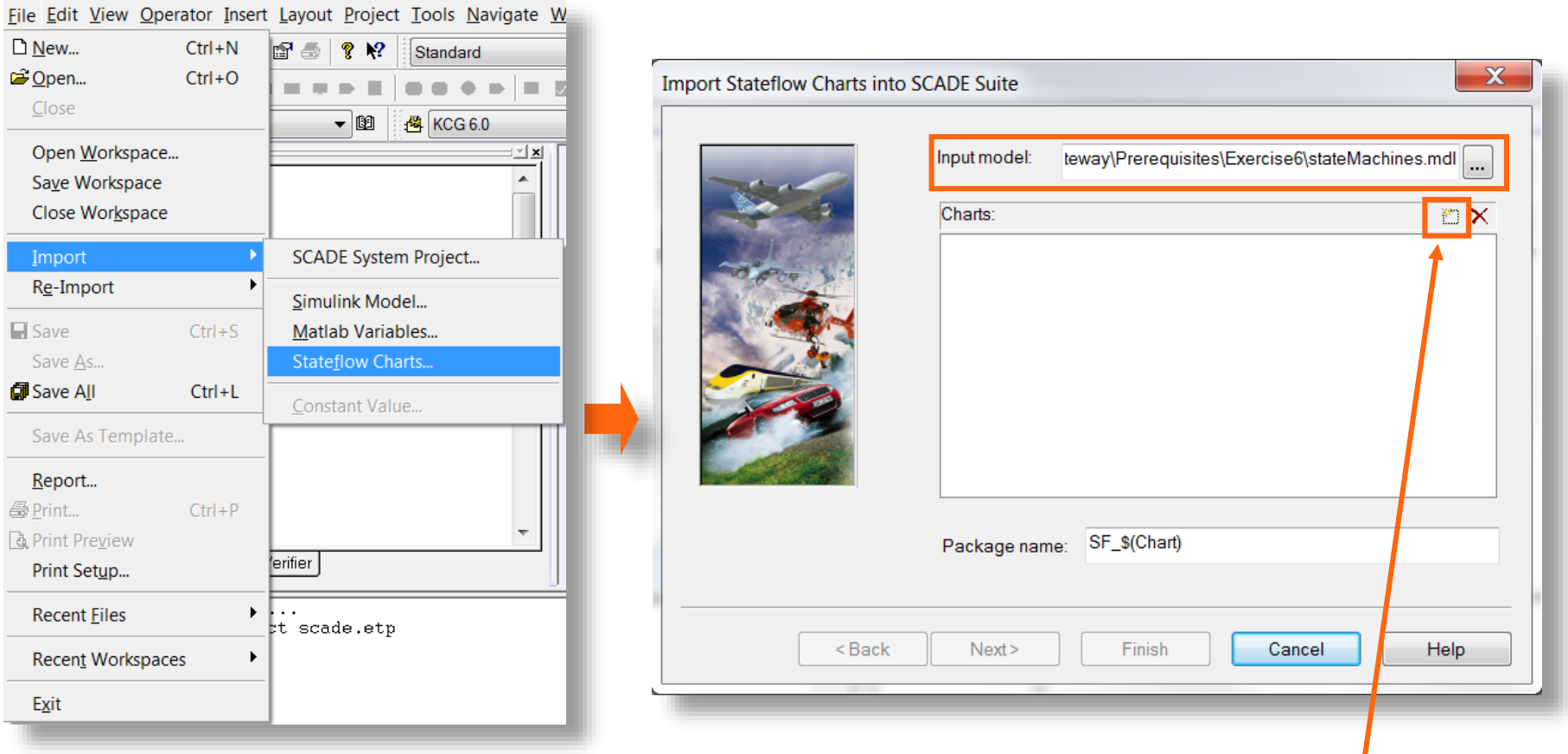
State Machines

All identified Stateflow elements are translated into SCADE control flow objects with respect to the definitions

Same graphical construction



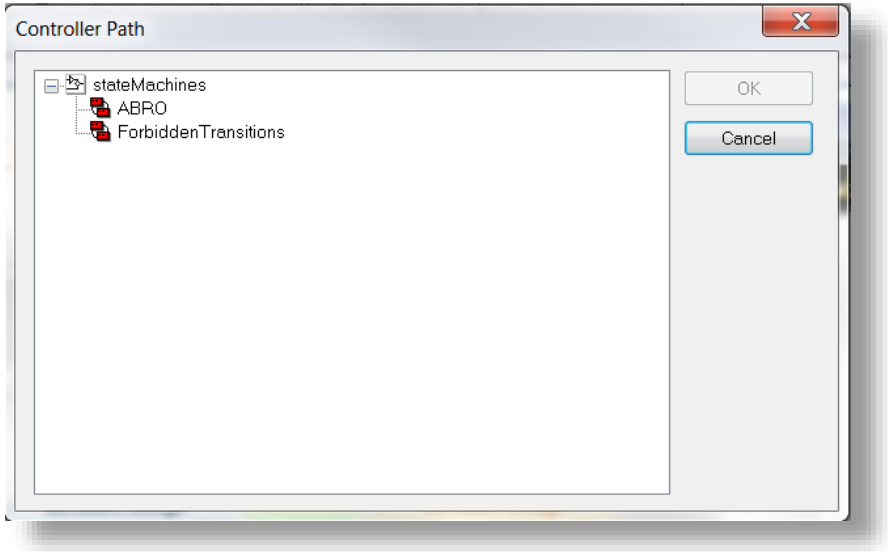
Translate a State Machines Project (1/2)



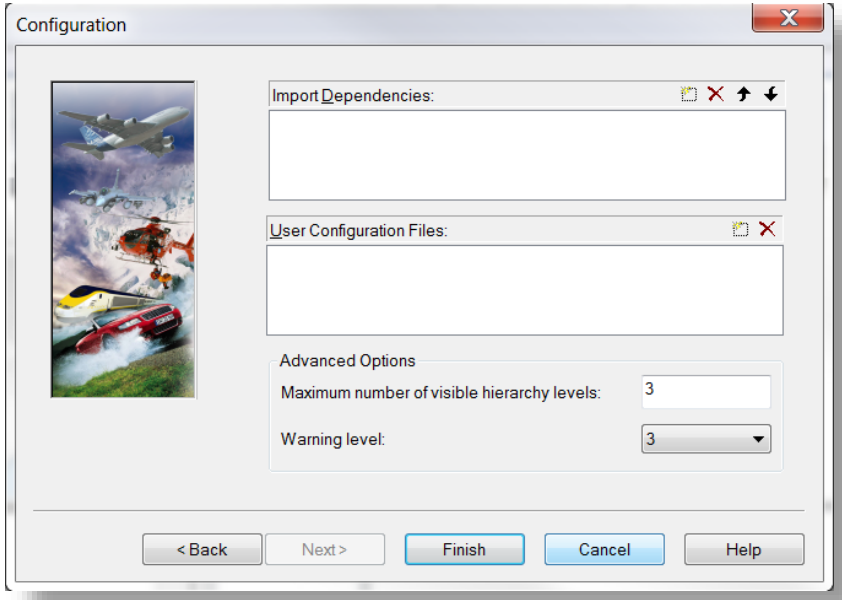
1. Create a new project or open an existing project
2. File -> Import -> Stateflow Charts
3. Set your Stateflow input model (MDL or SLX file)
4. Choose your chart

Translate a State Machines Project (2/2)

- Specify the list of Stateflow charts to be imported for translation.



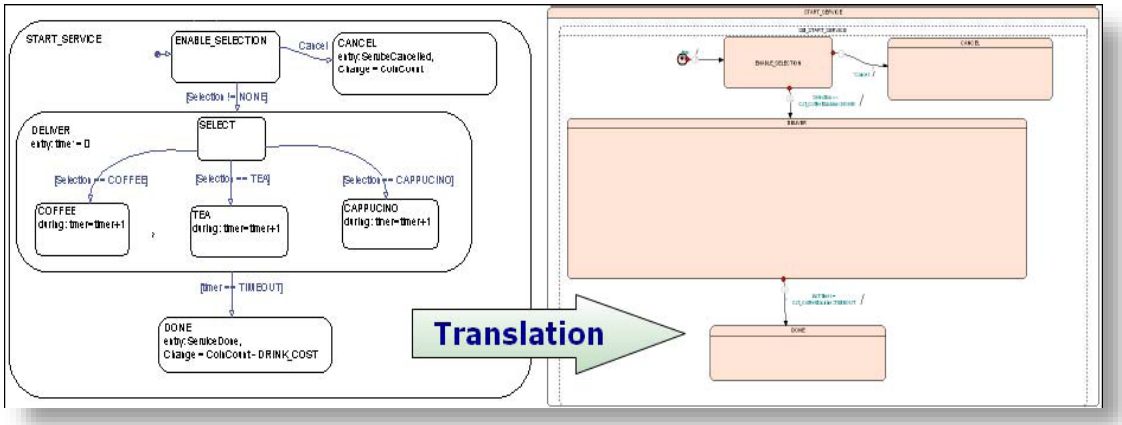
- To completely translate a Simulink model containing State Machines, it is necessary to use the modular translation.

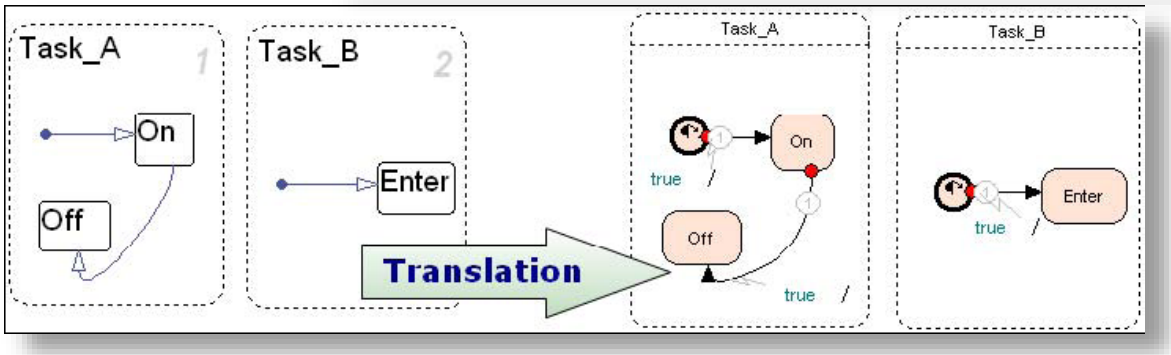


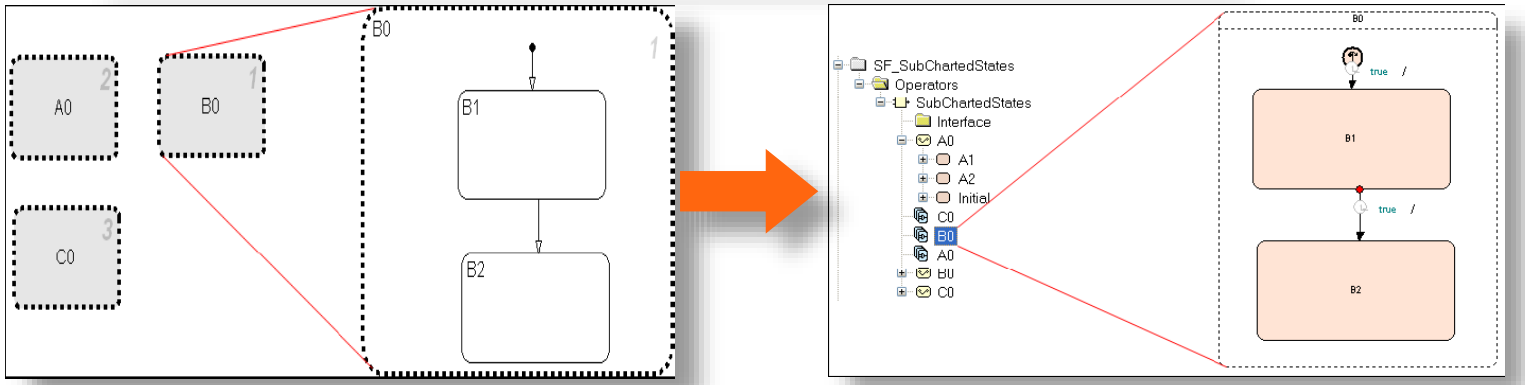
- Import Dependencies: specify the list of packages previously translated

Translation of Graphical Constructs

- Hierarchy
- Parallelism
- Subcharted Parallel States







Translation of Transitions (1/2)

In Stateflow transition priorities depend on the:

- **Definition on the transition**

or

- **Definition of transition labels**
- **Graphical position**

In SCADE transition priorities are explicitly defined

- **On each transition**
- **Do not depend on their graphical position**

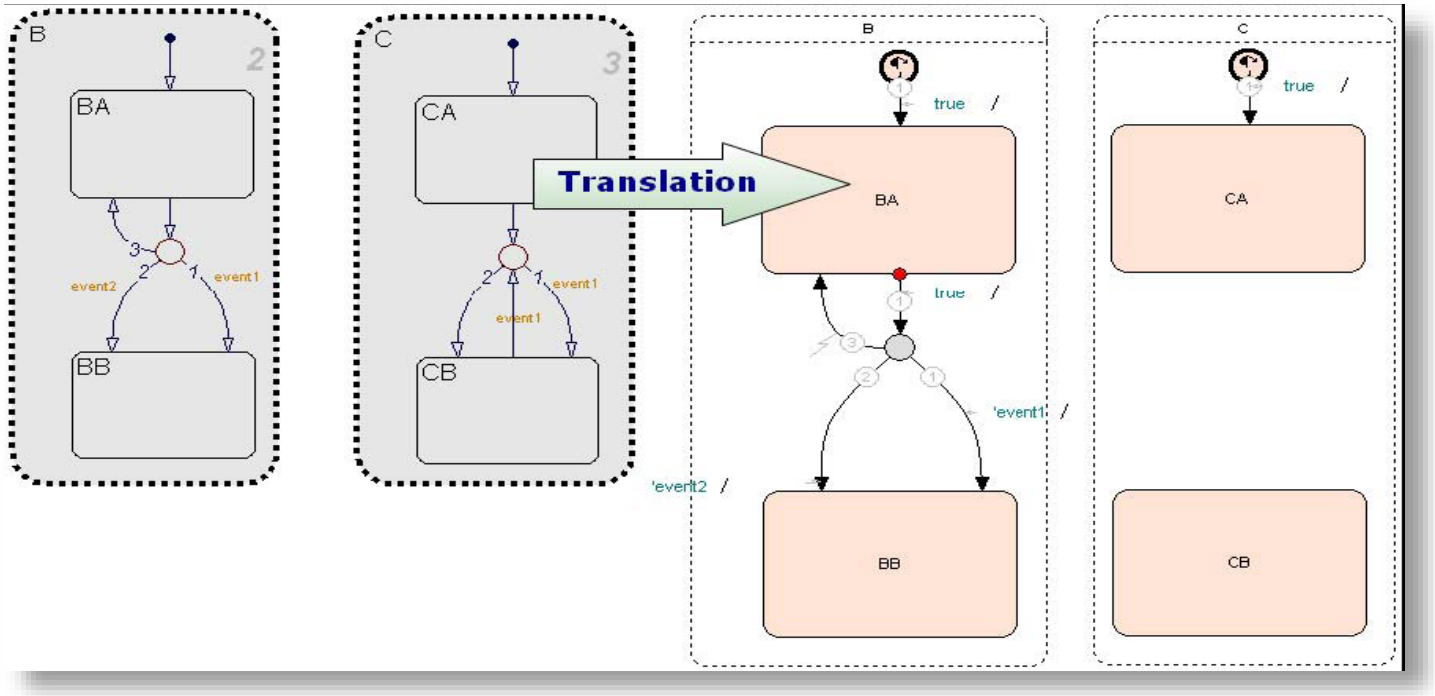
Specific transition constructs in Stateflow need an attention in translated models

Translation of Transitions (2/2)

Stateflow	SCADE
Transition	Strong transition
Default transition	Initial state with strong outgoing transition
Priorities according to: • explicit definition on each transition **or** • content of transition labels • graphical position of transitions	Priorities are explicitly defined on each transition

Translation of Connective Junctions

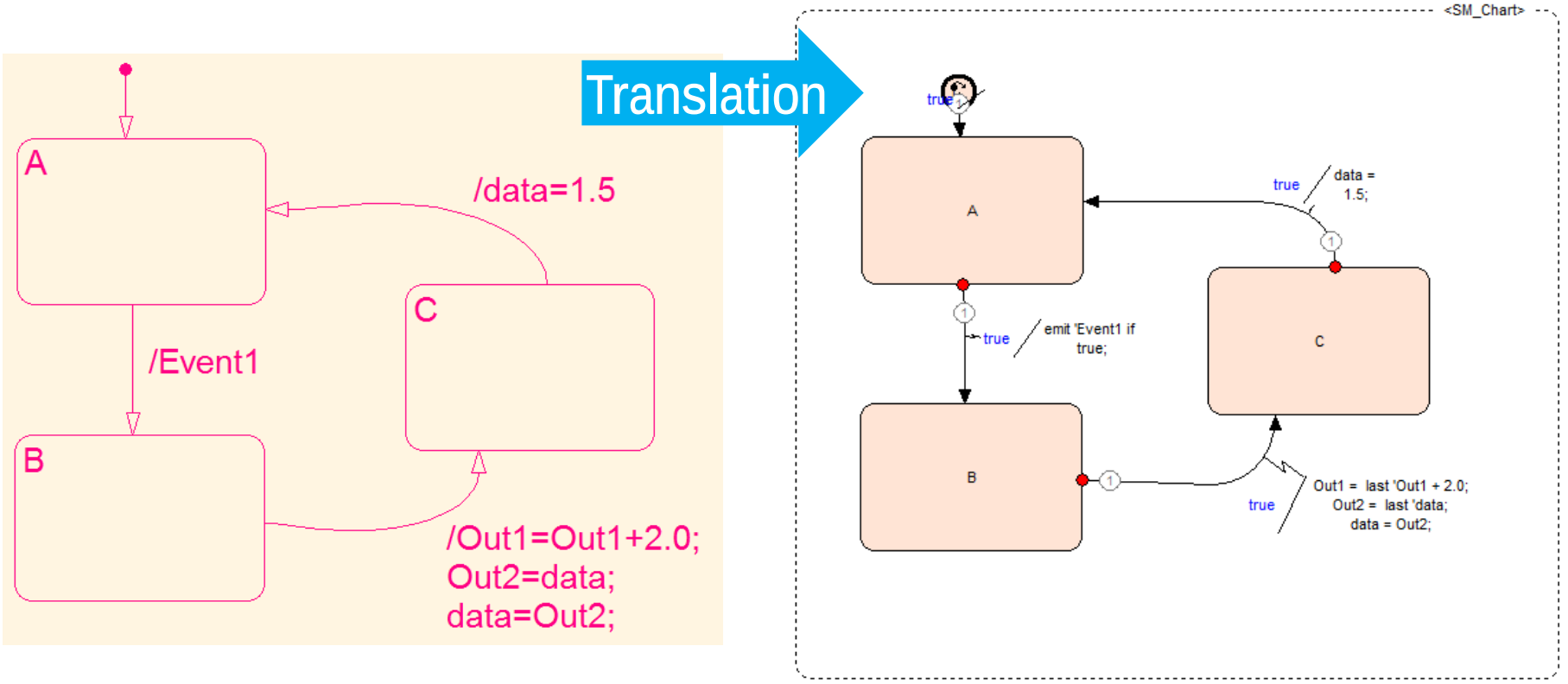
Stateflow connective junction…	…in SCADE is:
Single incoming transition	Translated in SCADE fork
Several incoming transitions	Not translated into SCADE



Translation of Transition Actions

Syntax of transition actions in SCADE :

`<event_trigger> { <condition> } / <transition_action>`



Translation Limitations

What is not translated:

Inner transitions

Crossing transition links

History junctions

Truth table functions

Embedded MATLAB functions

FROM SCADE TO SIMULINK

SCADE in Simulink

S-Function Generation for black-box simulation

Black Box Simulation

SCADE / Simulink Co-simulation

SCADE in Simulink

SCADE generated code is integrated into Simulink as a "S-Function"

The SCADE model can be a model translated from Simulink or made by hand

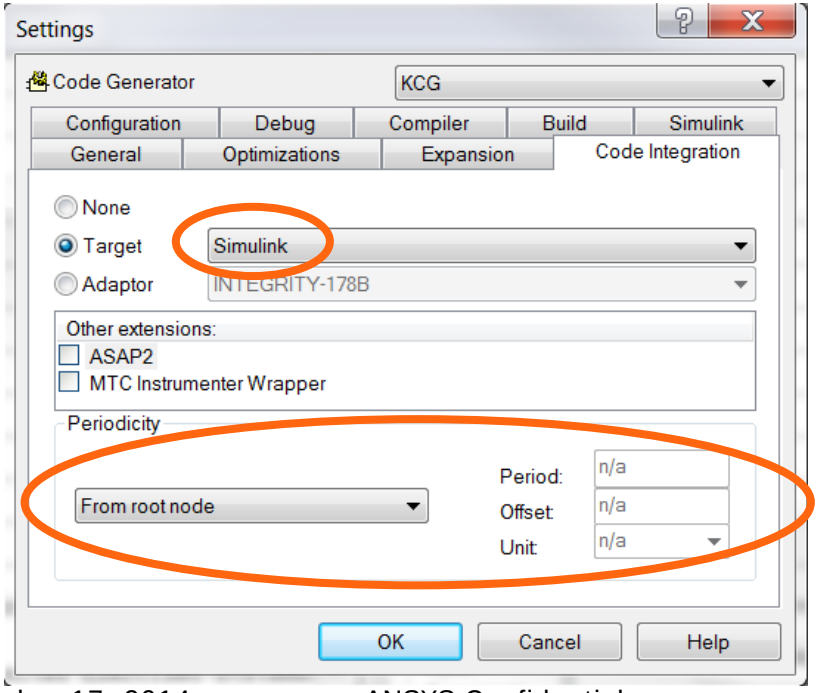
Simulation under Simulink:

- **The SCADE operator can be a black box**
- **The SCADE operator can be a white box**
  => co-simulation using the SCADE simulator within Simulink simulation
- **The generated code interacts with Simulink environment**

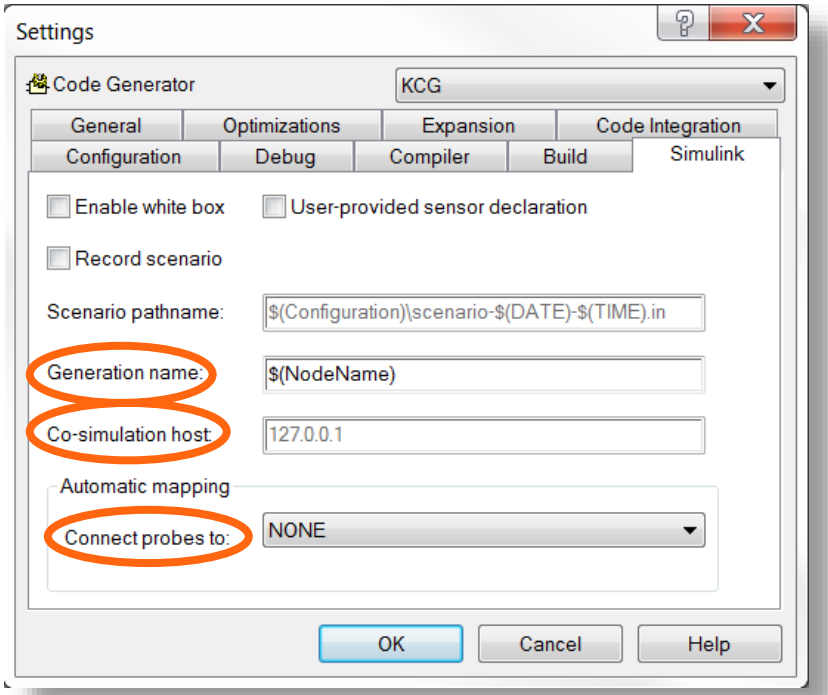
Simulink Wrapper (1/2)

The S-Function is generated by code generation with "Simulink" Target box selected in the General tab of the project settings

Sample time: period (in Simulink units) of the S-Function block (set in the Simulink tab of the project settings)



Simulink Wrapper (2/2)



- User-provided sensor declaration: prevent adding the sensors as inputs of the S-function block when the C definition of sensors is provided in an user C source file. When enabling this option, the wrapper discards the generation of the corresponding blocks in the S-Function wrapper
- Record scenario: enable the generated code of the S-function to produce a SCADE Suite scenario (.in format) at each simulation session running from Simulink environment
- Connect probes to: connect all SCADE probes through the S-Function. Probes can be connected to Display, Scope or Terminator block
- Generation name: Specify the S-Function file name and function name
- Co-simulation host (white box only): Specify the IP address of the host where you want to execute SCADE. You must create the M-file, the S-Function dynamic library, and the simulation executable on this same host

S-Function Dynamic Library and Hybrid Model

SCADE Suite generates the C code and a M-file that facilitates the creation of the S-Function dynamic library (used by Simulink for co-simulation with SCADE Suite)

Click Build to generate all necessary code

A Matlab script file is generated with the code:

`NodeName_SFCT_mk.m`

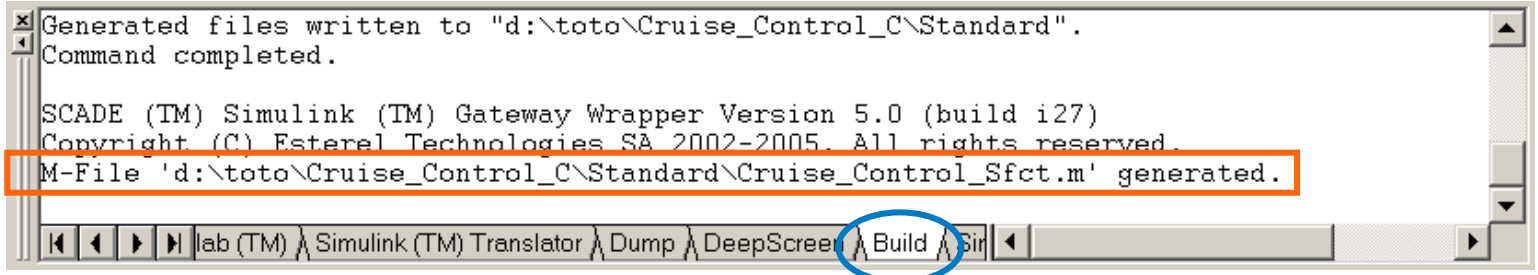
S-Function Dynamic Library and Hybrid Model

Execute the M-file to generate the hybrid model and create the S-Function dynamic library:

- **Open the File View, right-click the folder you want and select Insert Files, browse in the code generation folder of the project, right-click the <ControllerName>_Sfct.m file and select Execute Matlab M-File to start S-Function dynamic library and hybrid model creation with MATLAB**

Or

- **double-click on the corresponding line in the "Build" tab of the SCADE Output window, then click Yes when asked about M-file execution**



Black Box Simulation

In Simulink, click the Start simulation button

Execute simulation operations using Simulink commands

White Box Simulation

In Simulink, click the Start simulation button

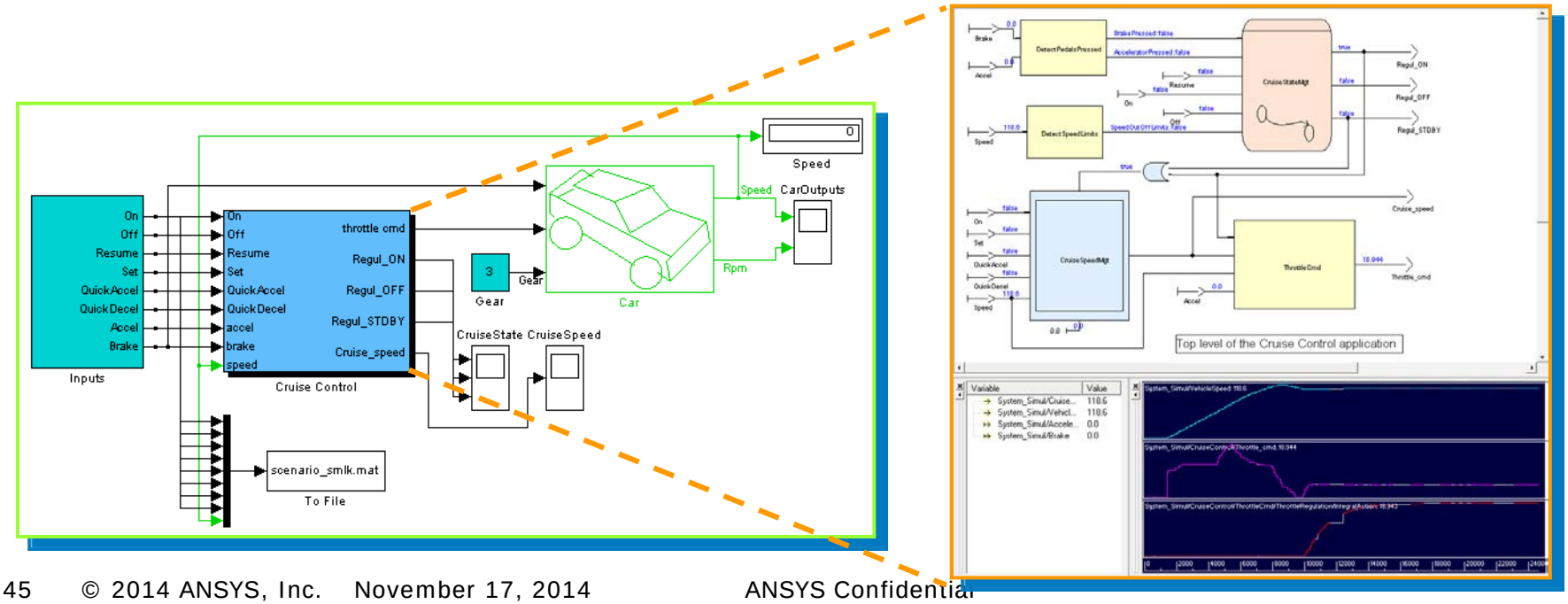
Execute simulation operations using either SCADE Suite or Simulink commands

White Box Simulation

The use of the full SCADE Simulator within Simulink enables high-quality verification activities of the software transformations with respect to the system requirements

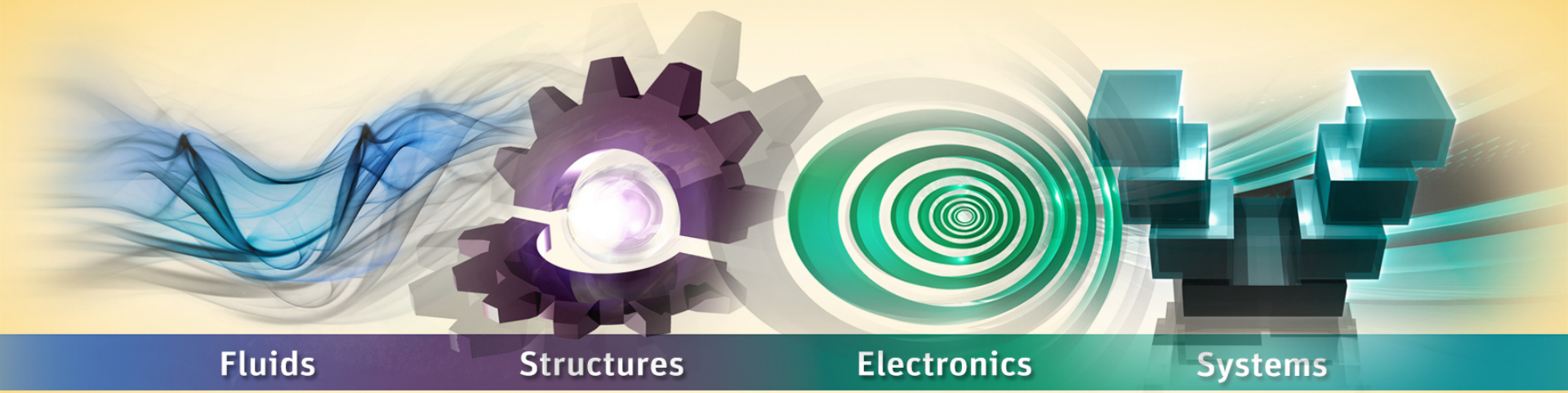
- **SCADE graphical simulation of the design, generic or implementation, in the Simulink environment**

All the following SCADE Simulator features are supported: value display, hierarchical navigation, breakpoints, wave forms.



ANSYS®

Realize Your Product Promise®



ESTEREL Technologies