MathWorks
AUTOMOTIVE CONFERENCE 2018

Simulink를 이용한 효율적인 레거시 코드 검증 방안

류성연

Agenda

- Overview to V&V in Model-Based Design
- Legacy code integration using Simulink
- Workflow for legacy code verification

Model-Based Design With Legacy C/C++ Code?

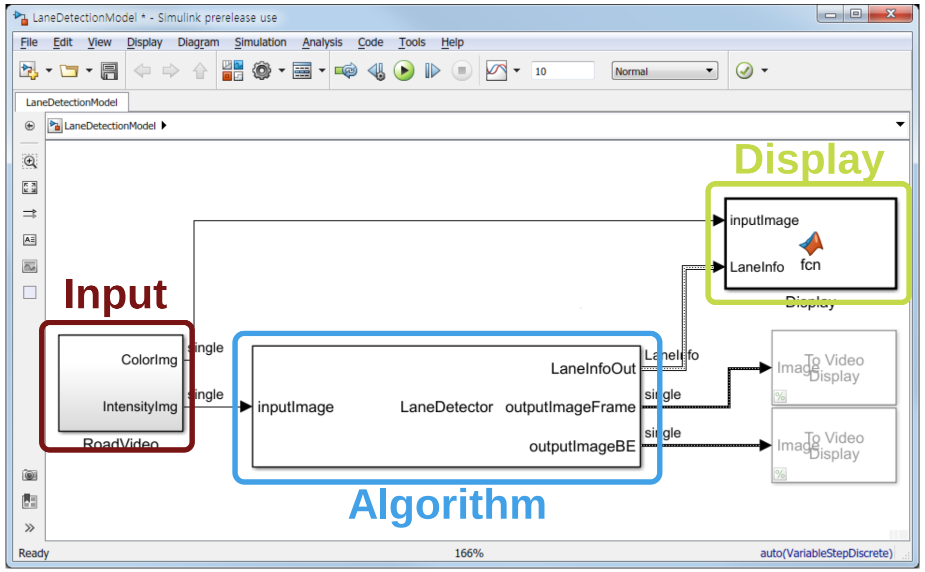
Hand Coding ⟷ Full MBD

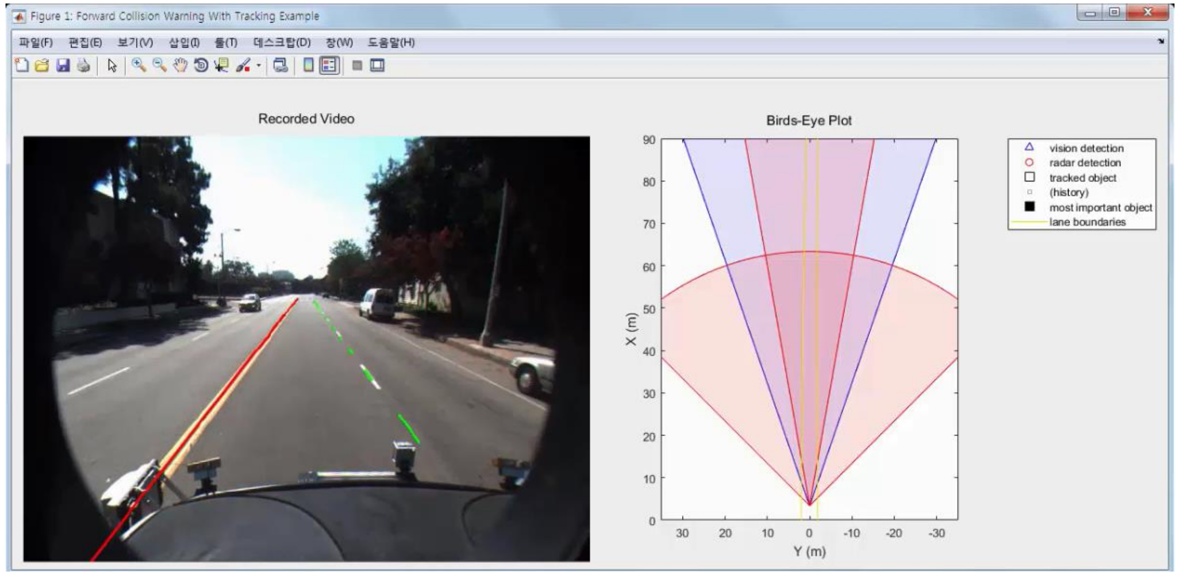
Legacy code verification using Simulink ?

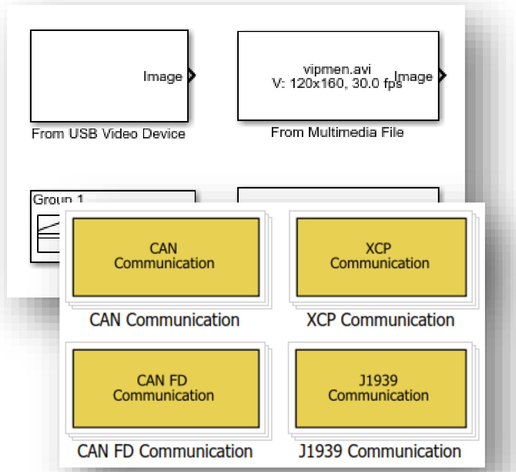
MBD with C/C++ code?

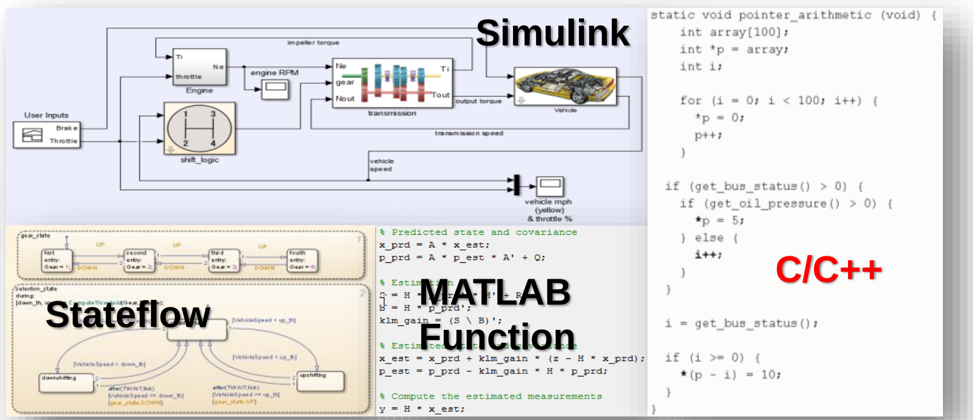
Model-Based Design

Why Using Simulink for Legacy Code Testing?



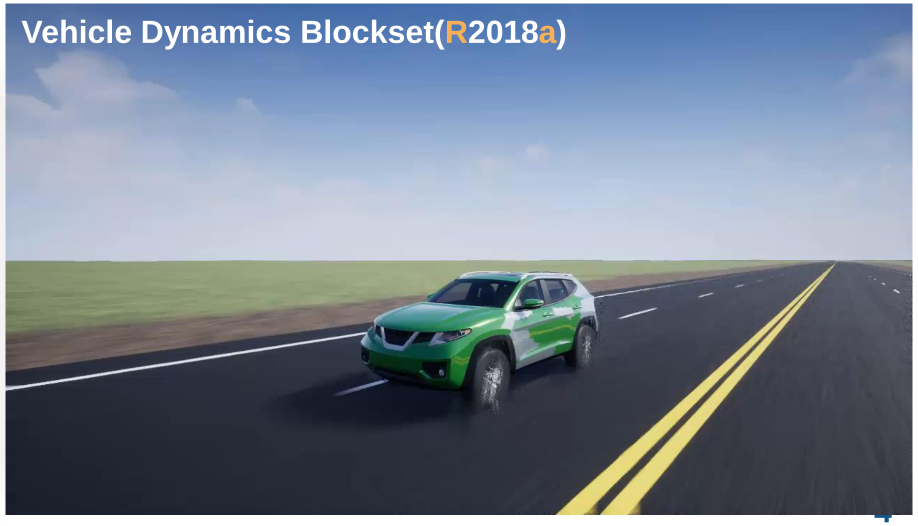






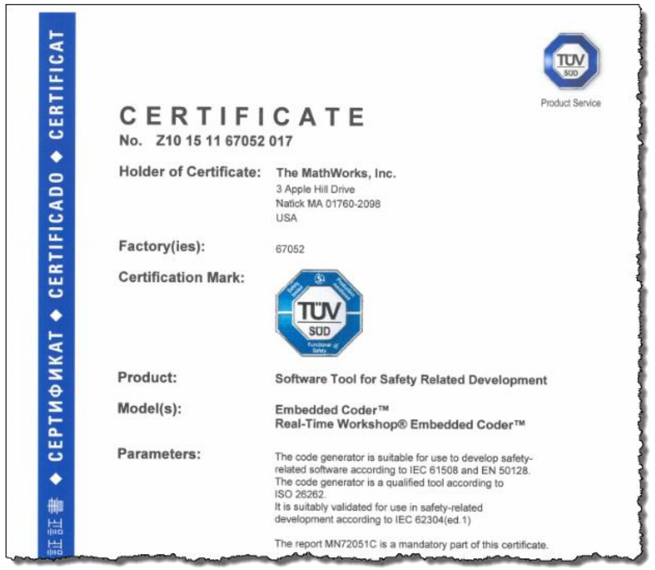
```
% Predicted state and covariance
x_prd = A * x_est;
p_prd = A * p_est * A' + Q;

% Estimation
S = H * p_prd' * H' + R;
B = H * p_prd';
klm_gain = (S \ B)';

% Estimated state and covariance
x_est = x_prd + klm_gain * (z - H * x_prd);
p_est = p_prd - klm_gain * H * p_prd;

% Compute the estimated measurements
y = H * x_est;
```

```
static void pointer_arithmetic (void) {
    int array[100];
    int *p = array;
    int i;

    for (i = 0; i < 100; i++) {
      *p = 0;
      p++;
    }

  if (get_bus_status() > 0) {
    if (get_oil_pressure() > 0) {
      *p = 5;
    } else {
      i++;
    }
  }

  i = get_bus_status();

  if (i >= 0) {
    *(p - i) = 10;
  }
}
```



ISO26262 “Road Vehicles - Functional Safety”

- **Functional safety standard** for passenger cars
  - Concerned with avoidance of unreasonable risks due to hazards caused by E/E systems
  - Recommends tool certification, but offers little guidance
- Serves as an umbrella standard for industry specific adaptions including:
  - **ISO 26262 - Automotive**
  - EN 50128 - Rail
  - IEC 62304 - Medical
  - IEC 61511 - Process Control



СЕРТИФИКАТ ◆ CERTIFICAT ◆ CERTIFICADO ◆ CERTIFICAT

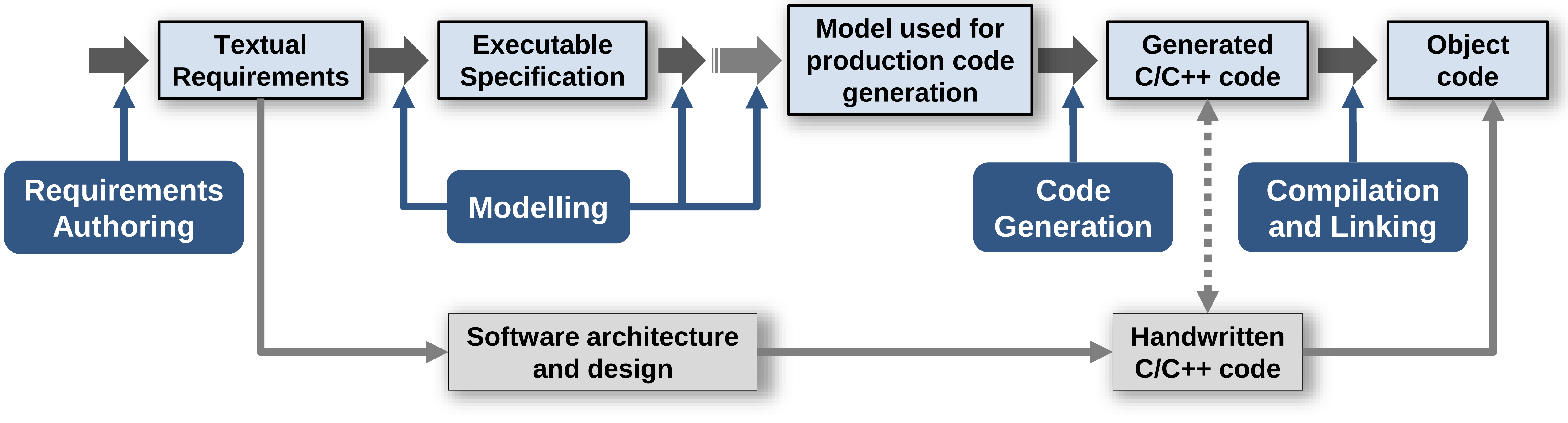
Product Service

CERTIFICATE

No. Z10 15 11 67052 017

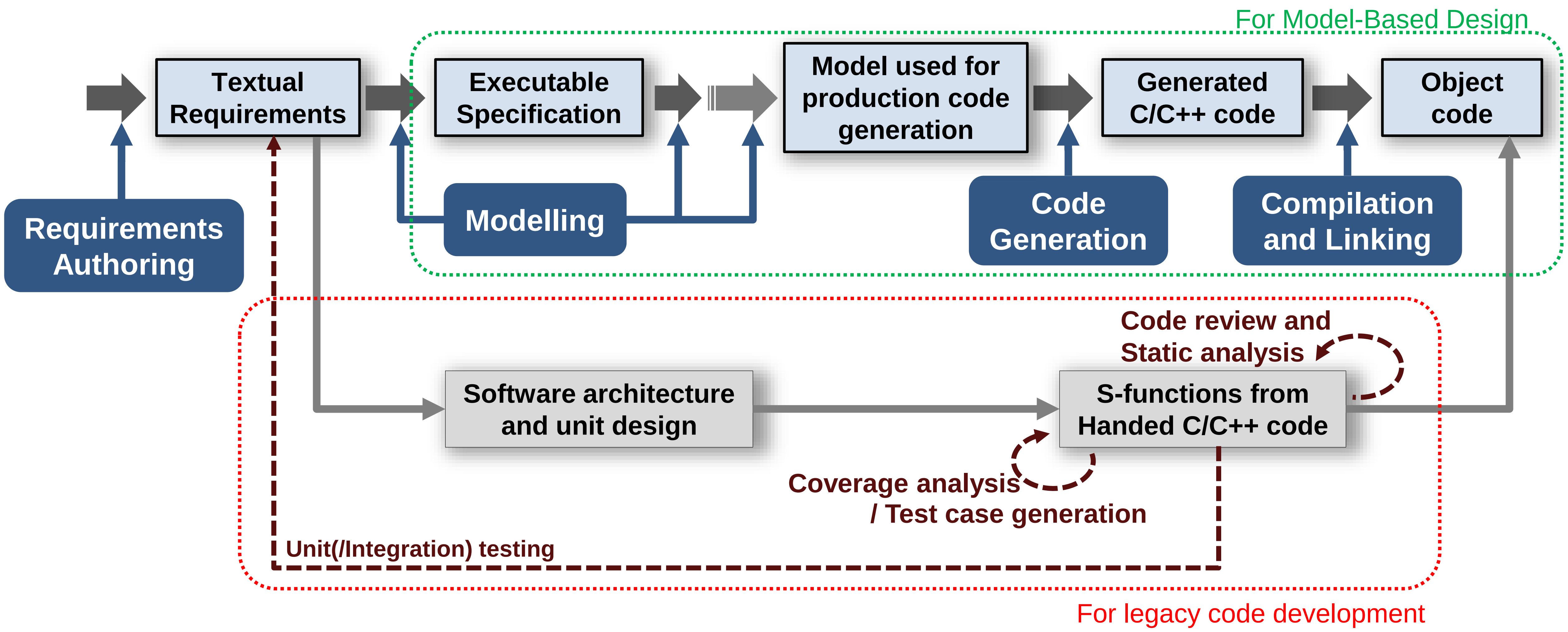
Holder of Certificate:	The MathWorks, Inc. 3 Apple Hill Drive Natick MA 01760-2098 USA
Factory(ies):	67052
Certification Mark:	
Product:	Software Tool for Safety Related Development
Model(s):	Embedded Coder™ Real-Time Workshop® Embedded Coder™
Parameters:	The code generator is suitable for use to develop safety-related software according to IEC 61508 and EN 50128. The code generator is a qualified tool according to ISO 26262. It is suitably validated for use in safety-related development according to IEC 62304(ed.1) The report MN72051C is a mandatory part of this certificate.

Software Development Workflow for Embedded Applications



Requirements Trace
Documentation
Version Control
Tool Qualification

Legacy Code Verification Overview

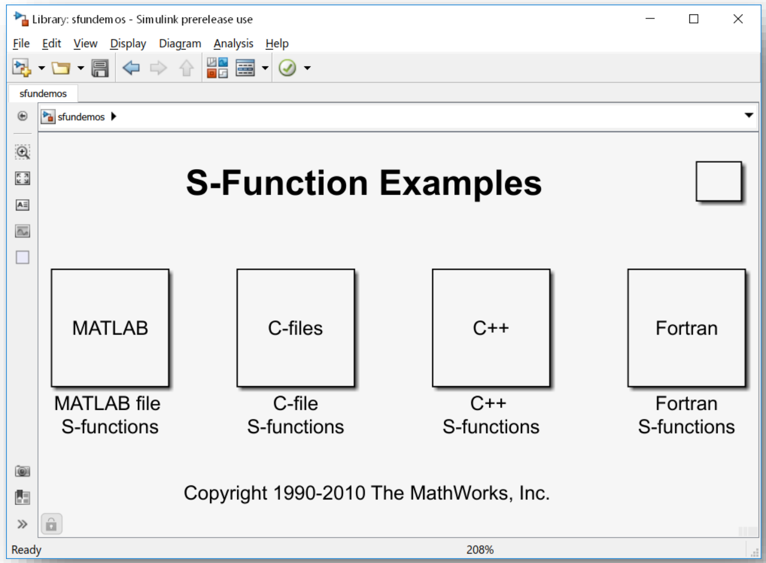

For Model-Based Design
Textual Requirements
Executable Specification
Model used for production code generation
Generated C/C++ code
Object code
Requirements Authoring
Modelling
Code Generation
Compilation and Linking
Code review and Static analysis
Software architecture and unit design
S-functions from Handed C/C++ code
Coverage analysis / Test case generation
Unit(/Integration) testing
For legacy code development

Agenda

- Overview to V&V in Model-Based Design
- Legacy code integration using Simulink
- Workflow for legacy code verification

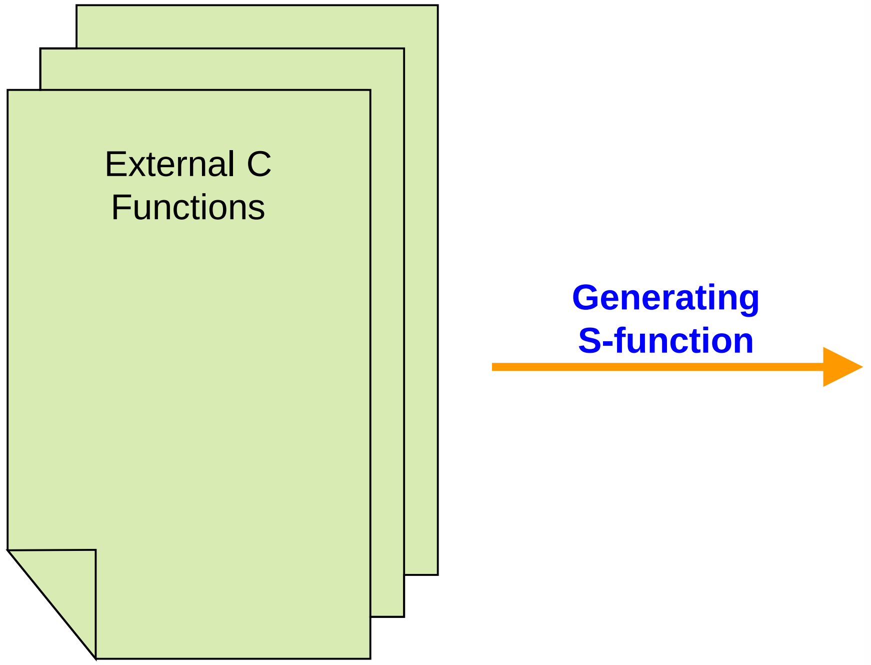
How to Import Legacy Code

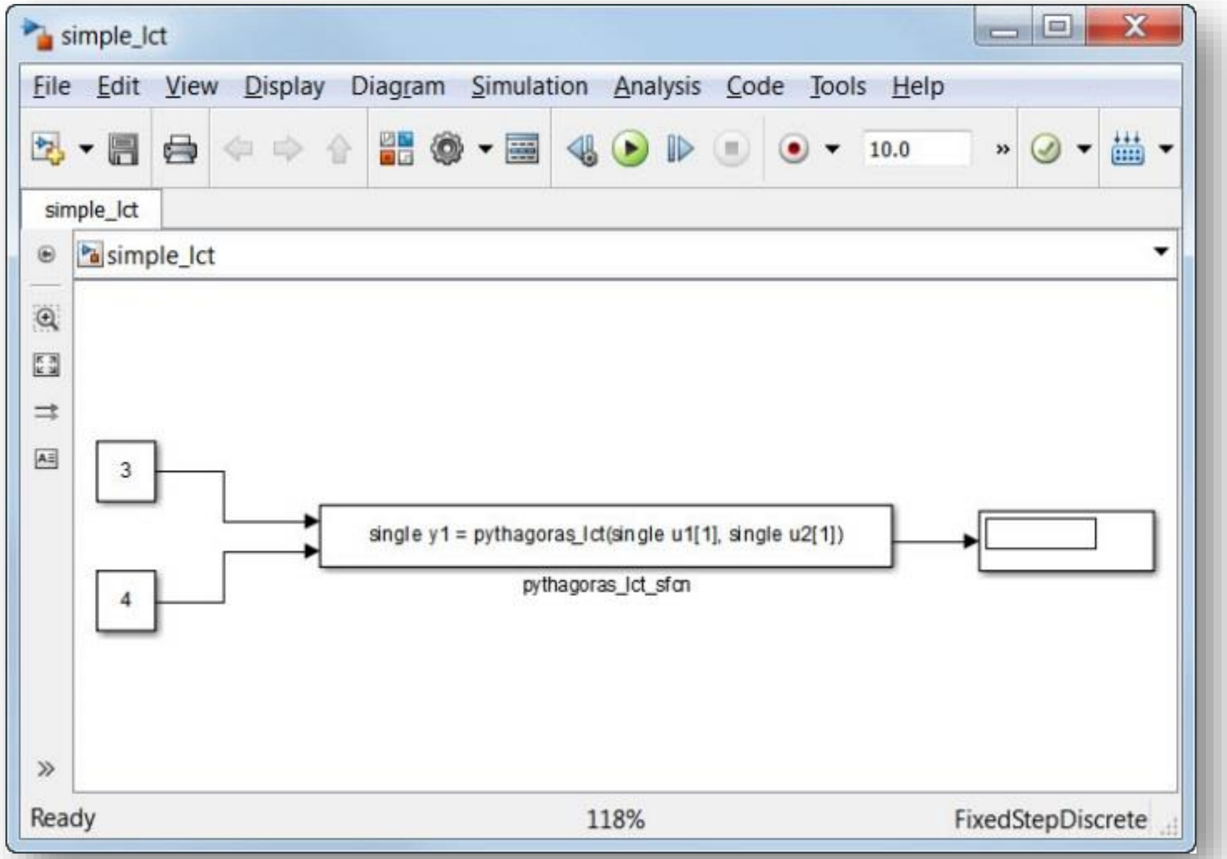
- Legacy Code Tool
- C Caller Block
- Legacy code integration in Stateflow



What legacy C code integration in Simulink means?

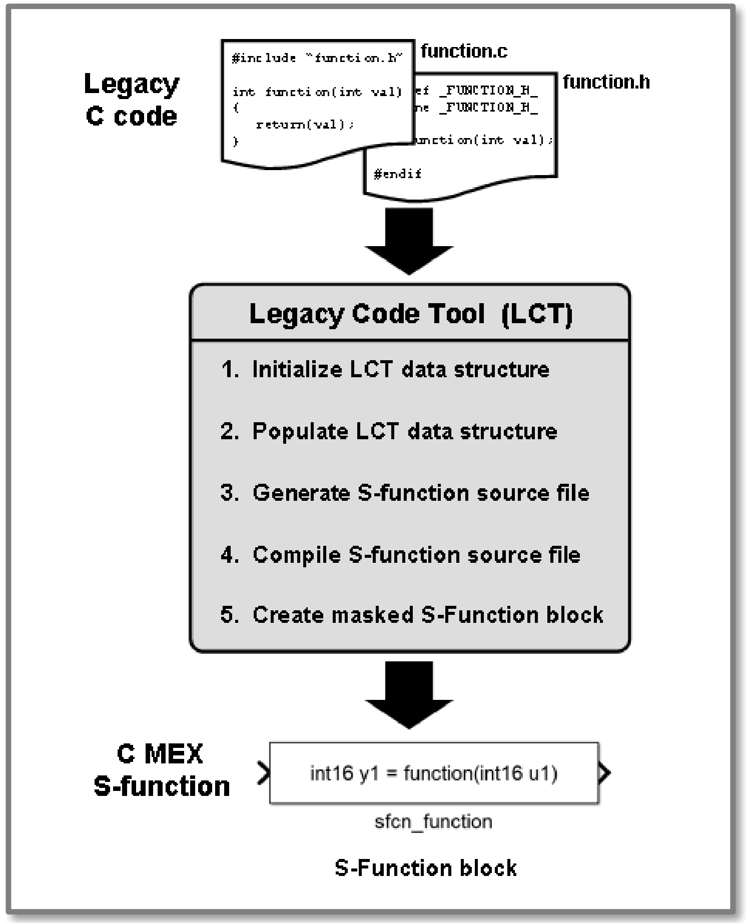
- Legacy Code Tool enables existing C code to be used in Simulink models




simple_lct
File Edit View Display Diagram Simulation Analysis Code Tools Help
10.0
simple_lct
simple_lct
3
4
single y1 = pythagoras_lct(single u1[1], single u2[1])
pythagoras_lct_sfcn
Ready
118%
FixedStepDiscrete

How to use Legacy Code Tool?

- General procedure for using Legacy Code Tool

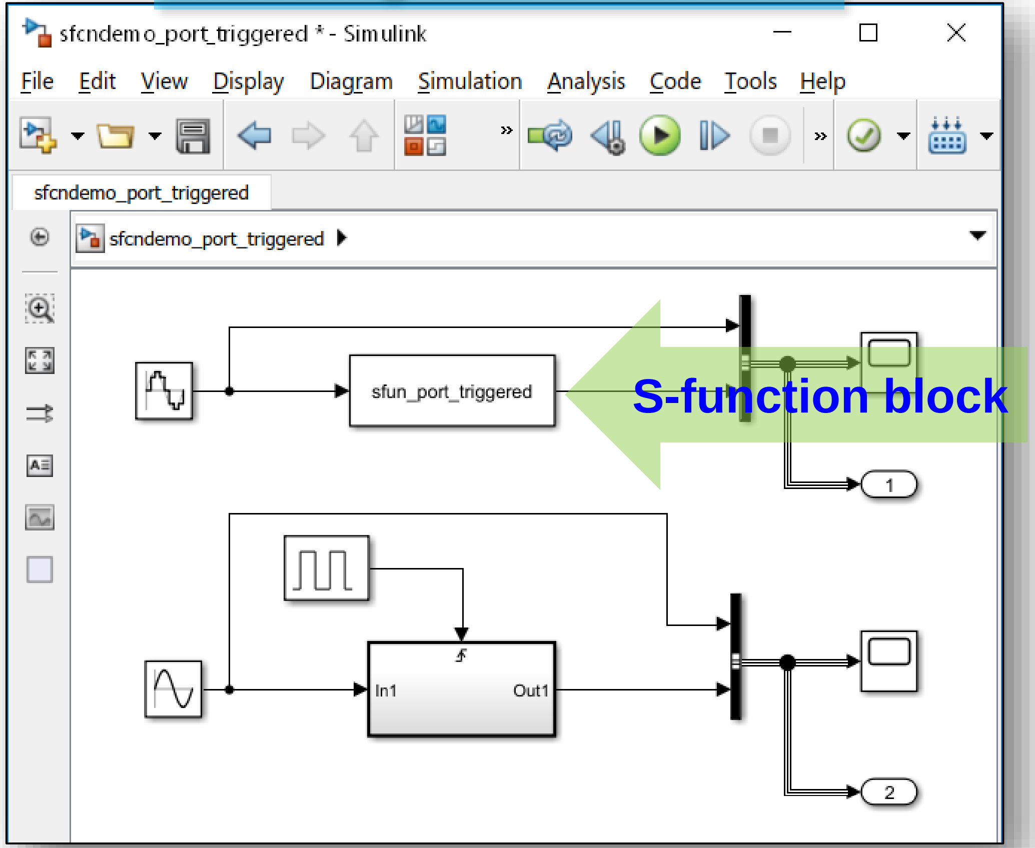


Legacy C code
function.c
function.h
Legacy Code Tool (LCT)
1. Initialize LCT data structure
2. Populate LCT data structure
3. Generate S-function source file
4. Compile S-function source file
5. Create masked S-Function block
C MEX S-function
int16 y1 = function(int16 u1)
sfcn_function
S-Function block

```
#include "function.h"
int function(int val)
{
   return(val);
}
```

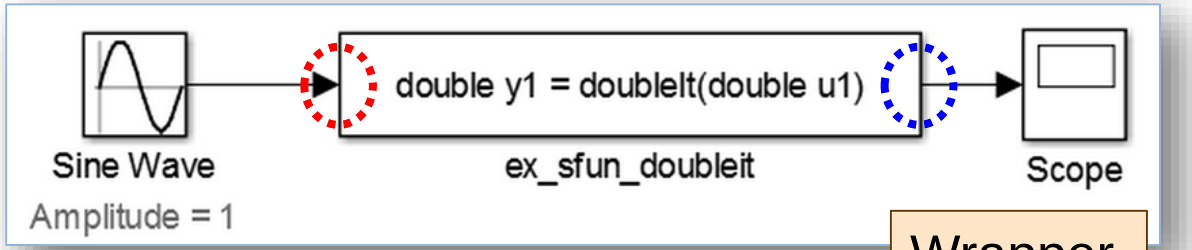
```
ef _FUNCTION_H_
ne _FUNCTION_H_
unction(int val);
#endif
```

Modeling with S-Function



Prerequisite to use Legacy Code Tool

- What is wrapper code?
  - Root-level C function having in/output variables for S-Function block's in/out ports



Wrapper code

```
double doubleit(double u1)
{
    MainInp = u1;

    double_main();

    y1 = MainOutp;
    return y1;
}
```

or

```
void doubleit(double u1, double *y1)
{
    MainInp = u1;

    double_main();

    *y1 = MainOutp;
}
```

Main function of legacy code

MATLAB Script to Build and Generate S-Function Block

- m-script file: compiling C files and generate a S-Function Block

```
Simulink.importExternalCTypes('ex_myTypes_LCT.h');

def = legacy_code('initialize');

def.SFunctionName = 'sfun_ex_mySrc_LCT';
def.SourceFiles = {'ex_mySrc_LCT.c'};
def.HeaderFiles = {'ex_myTypes_LCT.h'};

def.OutputFcnSpec = 'void myFcn(sigStructType u1[1], paramStructType p1[1], sigStructType y1[1])';

def.IncPaths = {'rtwdemo_lct_src'};
def.SrcPaths = {'rtwdemo_lct_src'};

legacy_code('sfcn_cmex_generate', def);

legacy_code('compile', def);

legacy_code(‘slblock_generate', def);
```

① C files to integrate in Simulink

② S-Function block specification

③ Include folders

④ Compile and s-function generation

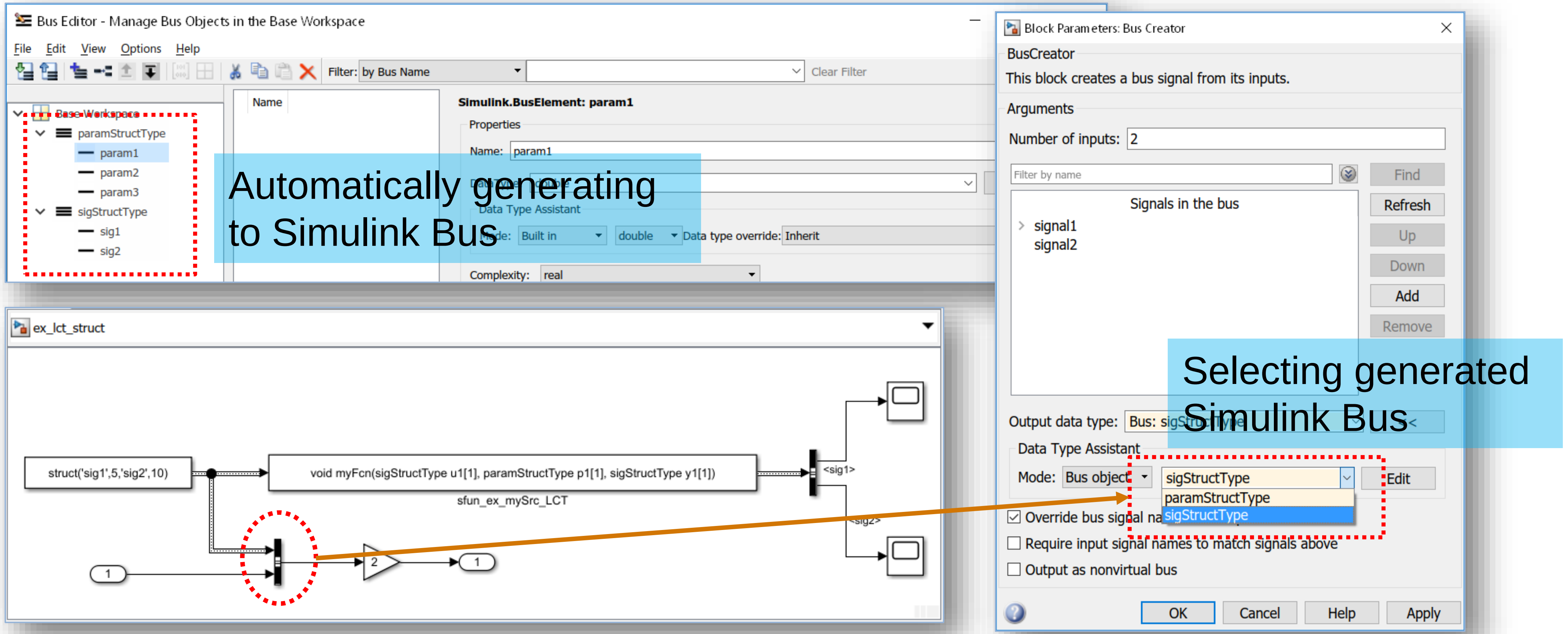
https://www.mathworks.com/help/releases/R2018a/simulink/sfg/integrating-existing-c-functions-into-simulink-models-with-the-legacy-code-tool.html?s_tid=doc_srchtitle

MathWorks R2017a

Generate Simulink Representations from C or C++ Code

- Import external C header file and generate available Simulink data types

```
Simulink.importExternalCTypes('ex_myTypes_LCT.h’);
```



Automatically generating to Simulink Bus

Selecting generated Simulink Bus

Bus Editor - Manage Bus Objects in the Base Workspace

Filter: by Bus Name

Base Workspace
- paramStructType
  - param1
  - param2
  - param3
- sigStructType
  - sig1
  - sig2

Simulink.BusElement: param1

Properties

Name: param1

Mode: Built in | double | Data type override: Inherit

Complexity: real

ex_lct_struct

struct('sig1',5,'sig2',10)

void myFcn(sigStructType u1[1], paramStructType p1[1], sigStructType y1[1])

sfun_ex_mySrc_LCT

Block Parameters: Bus Creator

BusCreator

This block creates a bus signal from its inputs.

Arguments

Number of inputs: 2

Signals in the bus
- signal1
- signal2

Output data type: Bus: sigStructType

Data Type Assistant

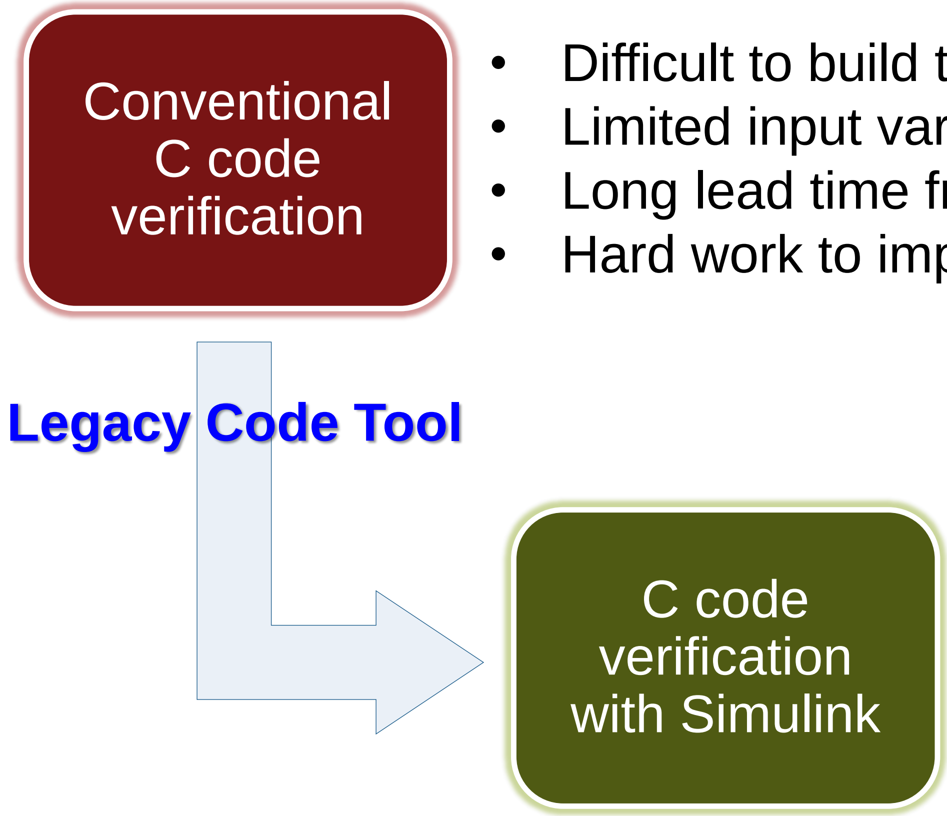
Mode: Bus object | sigStructType

paramStructType
sigStructType

☑ Override bus signal names from inputs
☐ Require input signal names to match signals above
☐ Output as nonvirtual bus

Issues for Legacy Code Tool

- There are still technical challenges to make S-Function Block

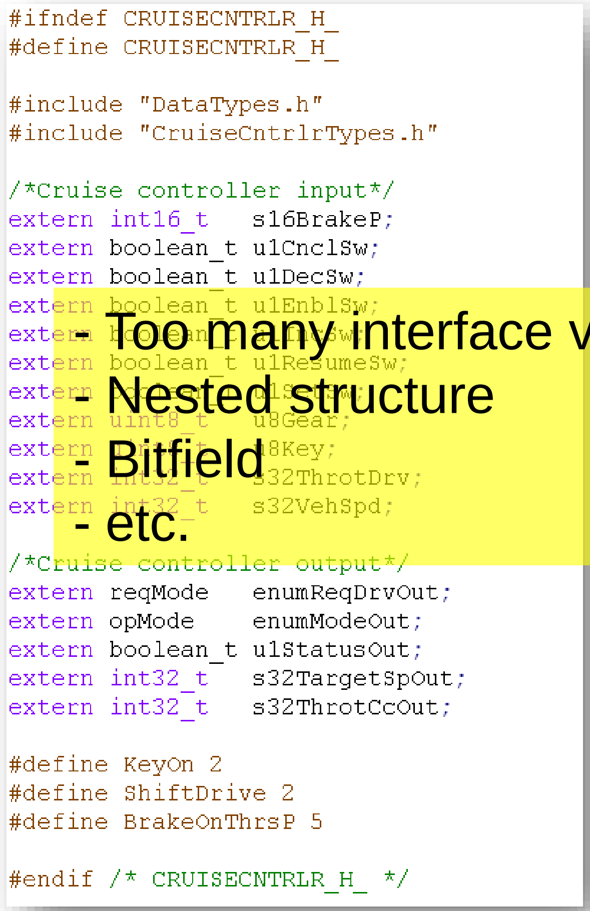

Conventional C code verification
Legacy Code Tool
C code verification with Simulink

- Difficult to build test cases
- Limited input variation
- Long lead time from development to test
- Hard work to improve test results

- No unified interfaces to interact with legacy code
- Hard to build S-Function Block
- No auto sync with custom C code change
- Still maintenance problem

Example Issue: Too Many Function Arguments

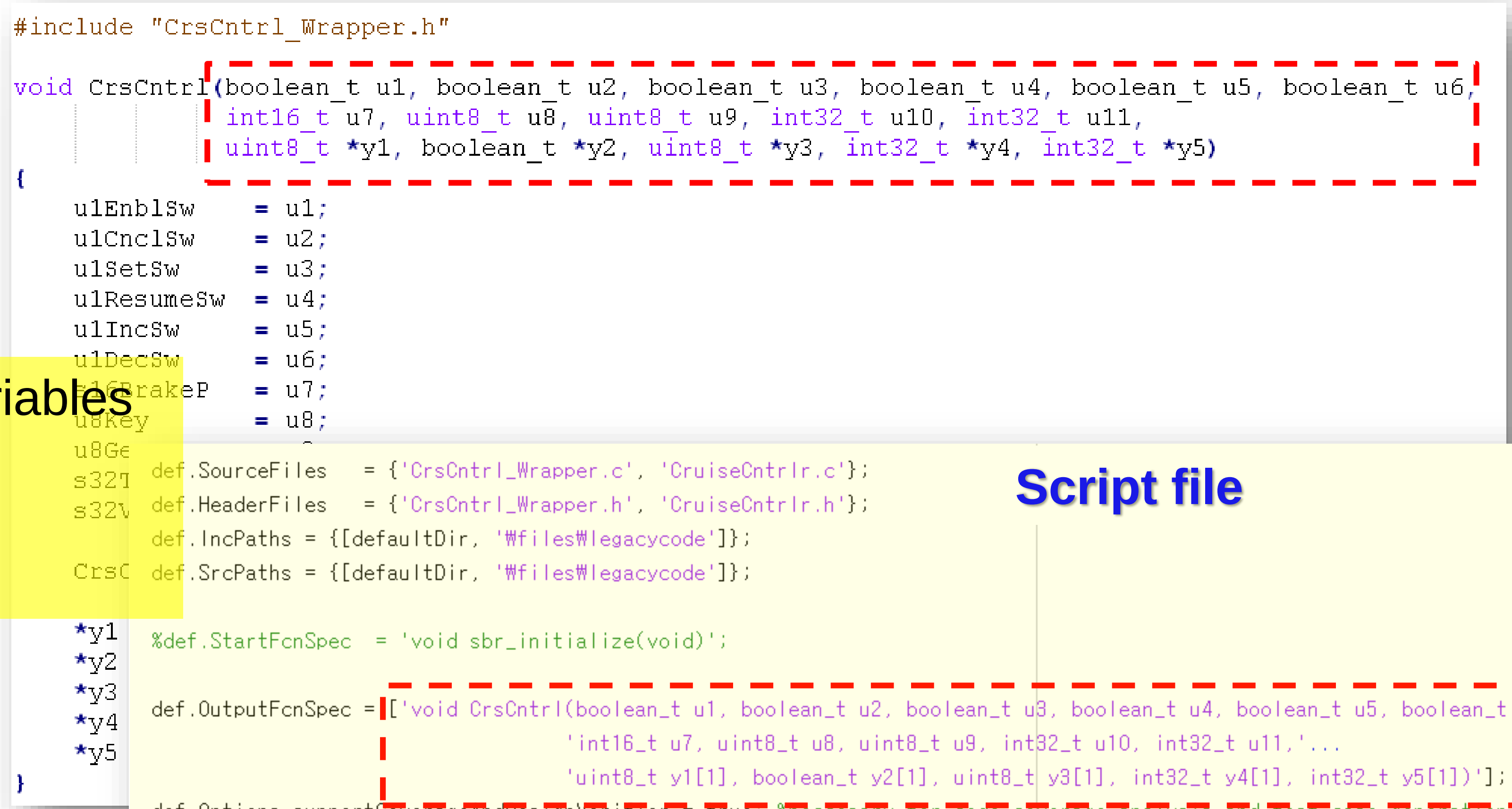
Legacy code



```
#ifndef CRUISECNTRLR_H_
#define CRUISECNTRLR_H_

#include "DataTypes.h"
#include "CruiseCntrlrTypes.h"

/*Cruise controller input*/
extern int16_t   s16BrakeP;
extern boolean_t u1CnclSw;
extern boolean_t u1DecSw;
extern boolean_t u1EnblSw;
extern boolean_t u1IncSw;
extern boolean_t u1ResumeSw;
extern boolean_t u1SetSw;
extern uint8_t   u8Gear;
extern uint8_t   u8Key;
extern int32_t   s32ThrotDrv;
extern int32_t   s32VehSpd;

/*Cruise controller output*/
extern reqMode   enumReqDrvOut;
extern opMode    enumModeOut;
extern boolean_t u1StatusOut;
extern int32_t   s32TargetSpOut;
extern int32_t   s32ThrotCcOut;

#define KeyOn 2
#define ShiftDrive 2
#define BrakeOnThrsP 5

#endif /* CRUISECNTRLR_H_ */
```

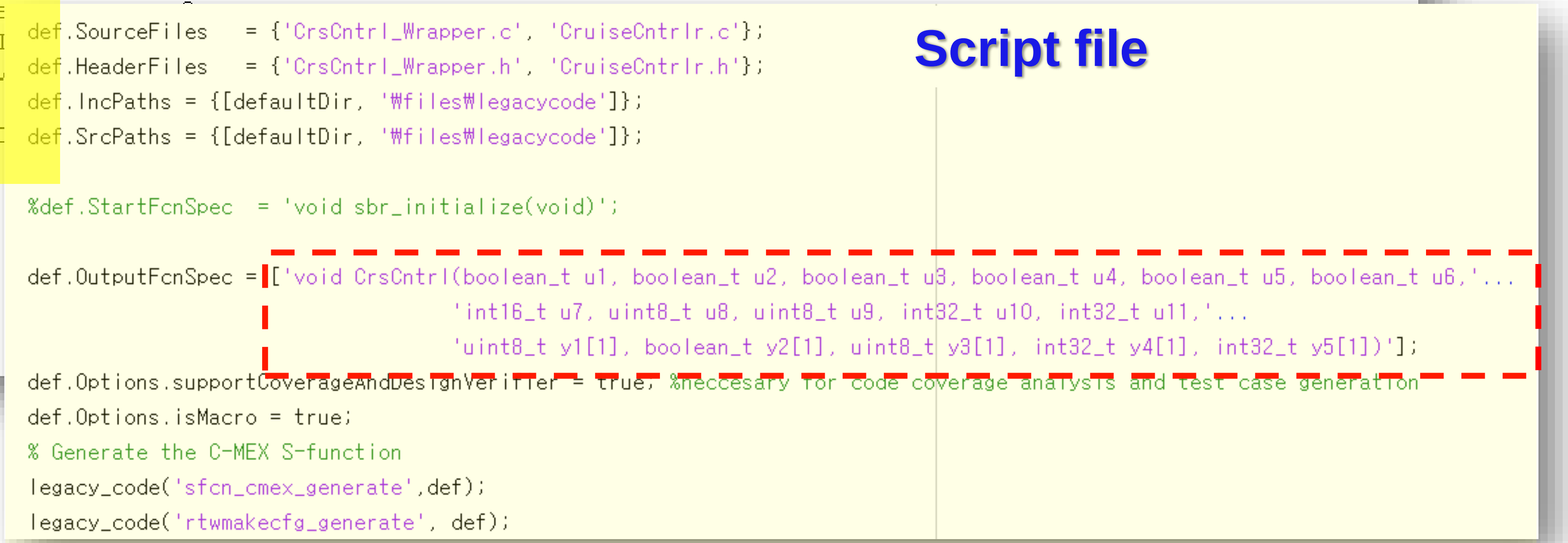
- Too many interface variables
- Nested structure
- Bitfield
- etc.

Wrapper code



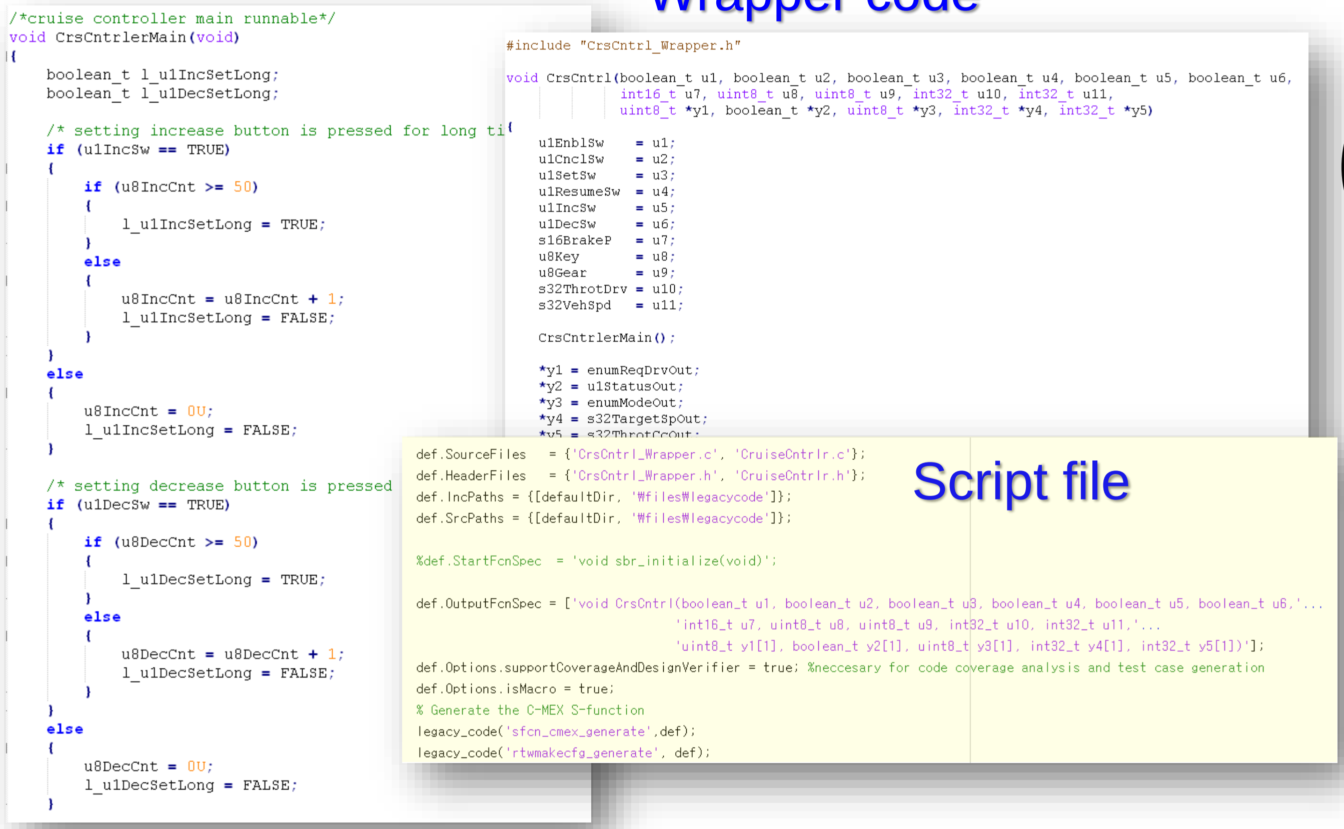
```
#include "CrsCntrl_Wrapper.h"

void CrsCntrl(boolean_t u1, boolean_t u2, boolean_t u3, boolean_t u4, boolean_t u5, boolean_t u6,
              int16_t u7, uint8_t u8, uint8_t u9, int32_t u10, int32_t u11,
              uint8_t *y1, boolean_t *y2, uint8_t *y3, int32_t *y4, int32_t *y5)
{
    u1EnblSw     = u1;
    u1CnclSw     = u2;
    u1SetSw      = u3;
    u1ResumeSw   = u4;
    u1IncSw      = u5;
    u1DecSw      = u6;
    s16BrakeP    = u7;
    u8Key        = u8;
    u8Ge
    s32T
    s32V

    CrsC

    *y1
    *y2
    *y3
    *y4
    *y5
}
```

Script file



```
def.SourceFiles   = {'CrsCntrl_Wrapper.c', 'CruiseCntrlr.c'};
def.HeaderFiles   = {'CrsCntrl_Wrapper.h', 'CruiseCntrlr.h'};
def.IncPaths = {[defaultDir, '₩files₩legacycode']};
def.SrcPaths = {[defaultDir, '₩files₩legacycode']};

%def.StartFcnSpec  = 'void sbr_initialize(void)';

def.OutputFcnSpec = ['void CrsCntrl(boolean_t u1, boolean_t u2, boolean_t u3, boolean_t u4, boolean_t u5, boolean_t u6,'...
                          'int16_t u7, uint8_t u8, uint8_t u9, int32_t u10, int32_t u11,'...
                          'uint8_t y1[1], boolean_t y2[1], uint8_t y3[1], int32_t y4[1], int32_t y5[1])'];
def.Options.supportCoverageAndDesignVerifier = true; %neccesary for code coverage analysis and test case generation
def.Options.isMacro = true;
% Generate the C-MEX S-function
legacy_code('sfcn_cmex_generate',def);
legacy_code('rtwmakecfg_generate', def);
```

Maintenance Problem…



Legacy code

```c
/*cruise controller main runnable*/
void CrsCntrlerMain(void)
{
    boolean_t l_u1IncSetLong;
    boolean_t l_u1DecSetLong;

    /* setting increase button is pressed for long ti
    if (u1IncSw == TRUE)
    {
        if (u8IncCnt >= 50)
        {
            l_u1IncSetLong = TRUE;
        }
        else
        {
            u8IncCnt = u8IncCnt + 1;
            l_u1IncSetLong = FALSE;
        }
    }
    else
    {
        u8IncCnt = 0U;
        l_u1IncSetLong = FALSE;
    }

    /* setting decrease button is pressed
    if (u1DecSw == TRUE)
    {
        if (u8DecCnt >= 50)
        {
            l_u1DecSetLong = TRUE;
        }
        else
        {
            u8DecCnt = u8DecCnt + 1;
            l_u1DecSetLong = FALSE;
        }
    }
    else
    {
        u8DecCnt = 0U;
        l_u1DecSetLong = FALSE;
    }
```

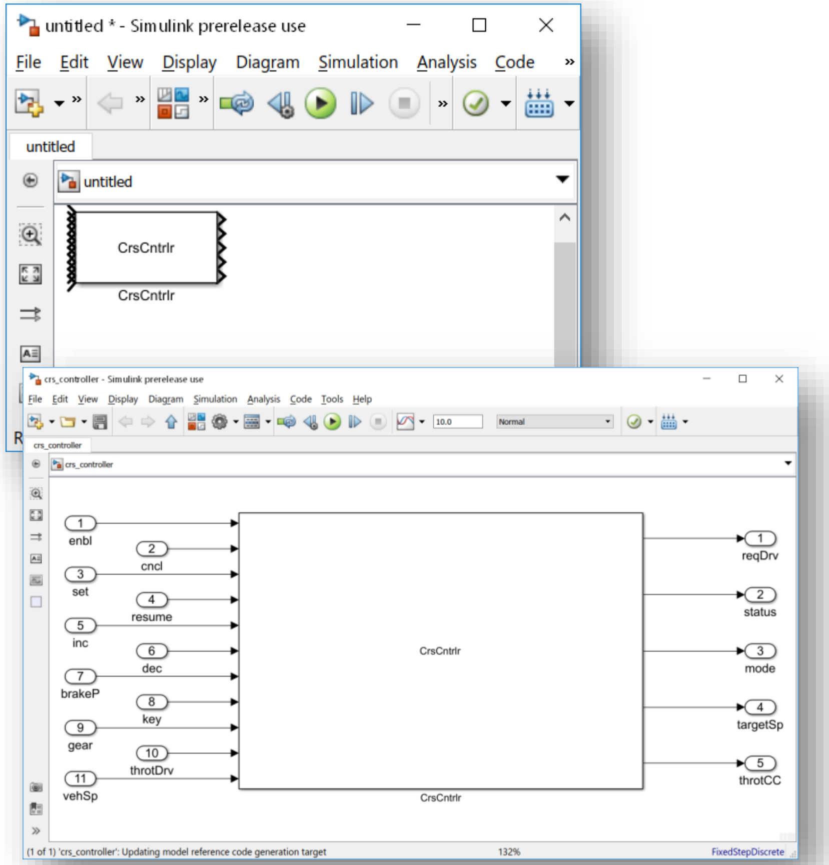
Wrapper code

```c
#include "CrsCntrl_Wrapper.h"

void CrsCntrl(boolean_t u1, boolean_t u2, boolean_t u3, boolean_t u4, boolean_t u5, boolean_t u6,
              int16_t u7, uint8_t u8, uint8_t u9, int32_t u10, int32_t u11,
              uint8_t *y1, boolean_t *y2, uint8_t *y3, int32_t *y4, int32_t *y5)
{
    u1EnblSw    = u1;
    u1CnclSw    = u2;
    u1SetSw     = u3;
    u1ResumeSw  = u4;
    u1IncSw     = u5;
    u1DecSw     = u6;
    s16BrakeP   = u7;
    u8Key       = u8;
    u8Gear      = u9;
    s32ThrotDrv = u10;
    s32VehSpd   = u11;

    CrsCntrlerMain();

    *y1 = enumReqDrvOut;
    *y2 = u1StatusOut;
    *y3 = enumModeOut;
    *y4 = s32TargetSpOut;
    *y5 = s32ThrotCcOut;
```

Script file

```matlab
def.SourceFiles   = {'CrsCntrl_Wrapper.c', 'CruiseCntrlr.c'};
def.HeaderFiles   = {'CrsCntrl_Wrapper.h', 'CruiseCntrlr.h'};
def.IncPaths = {[defaultDir, '₩files₩legacycode']};
def.SrcPaths = {[defaultDir, '₩files₩legacycode']};

%def.StartFcnSpec  = 'void sbr_initialize(void)';

def.OutputFcnSpec = ['void CrsCntrl(boolean_t u1, boolean_t u2, boolean_t u3, boolean_t u4, boolean_t u5, boolean_t u6,'...
                     'int16_t u7, uint8_t u8, uint8_t u9, int32_t u10, int32_t u11,'...
                     'uint8_t y1[1], boolean_t y2[1], uint8_t y3[1], int32_t y4[1], int32_t y5[1])'];
def.Options.supportCoverageAndDesignVerifier = true; %neccesary for code coverage analysis and test case generation
def.Options.isMacro = true;
% Generate the C-MEX S-function
legacy_code('sfcn_cmex_generate',def);
legacy_code('rtwmakecfg_generate', def);
```

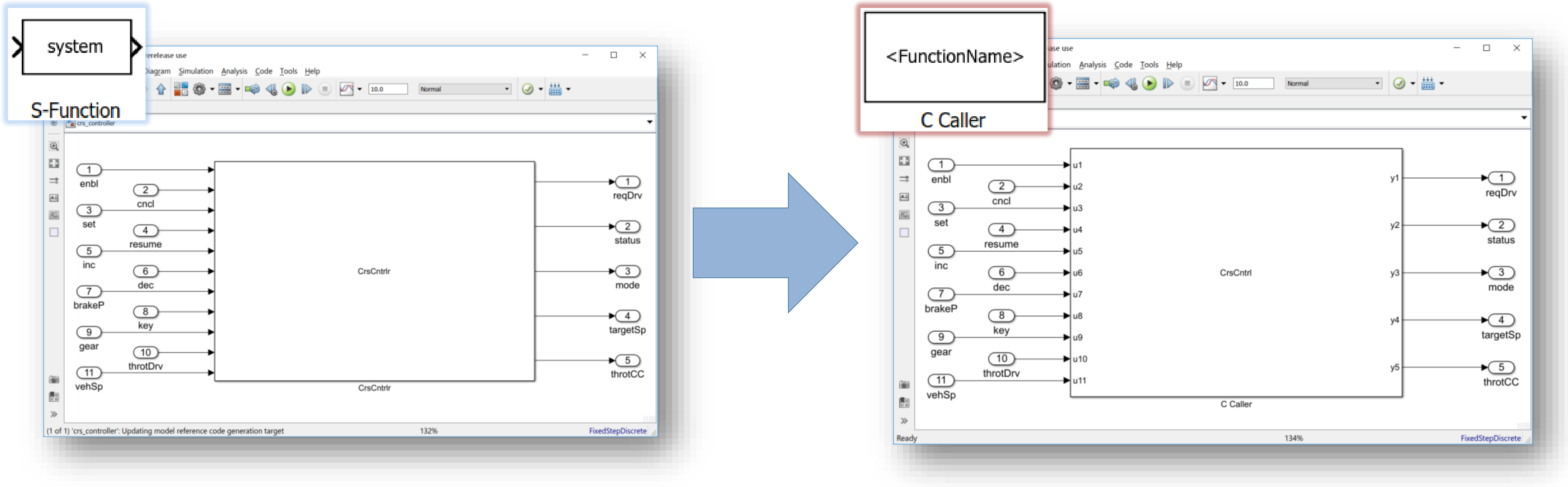
Modeling



(1 of 1) 'crs_controller': Updating model reference code generation target

Introducing C Caller Block

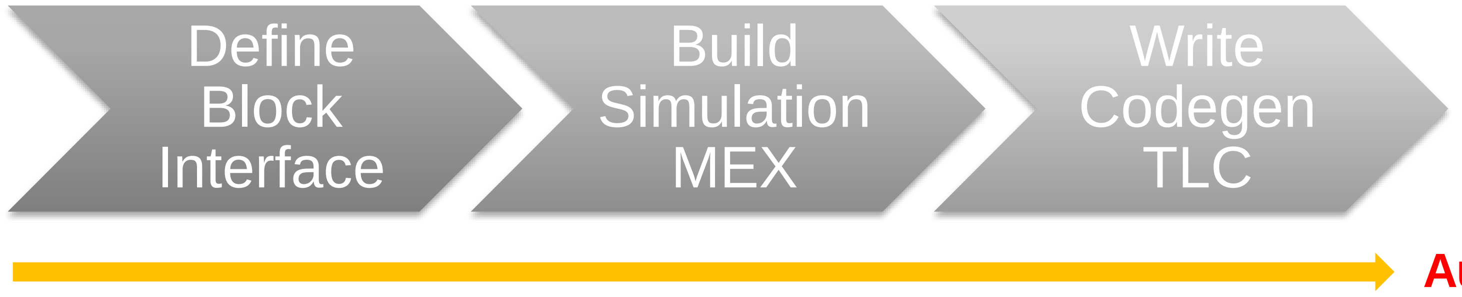
C Caller Block makes it easier to call C Functions in Simulink
→ It works for simulation and Code Generation



(1 of 1) 'crs_controller': Updating model reference code generation target

Key Features

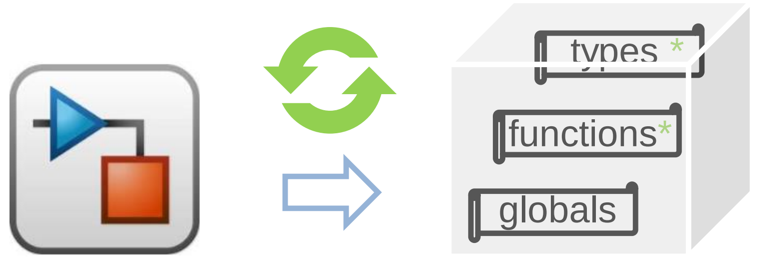
- Automate the process



- Tedious
- Error prone
- Hard to maintain

Automate

- Synchronize with custom code changes

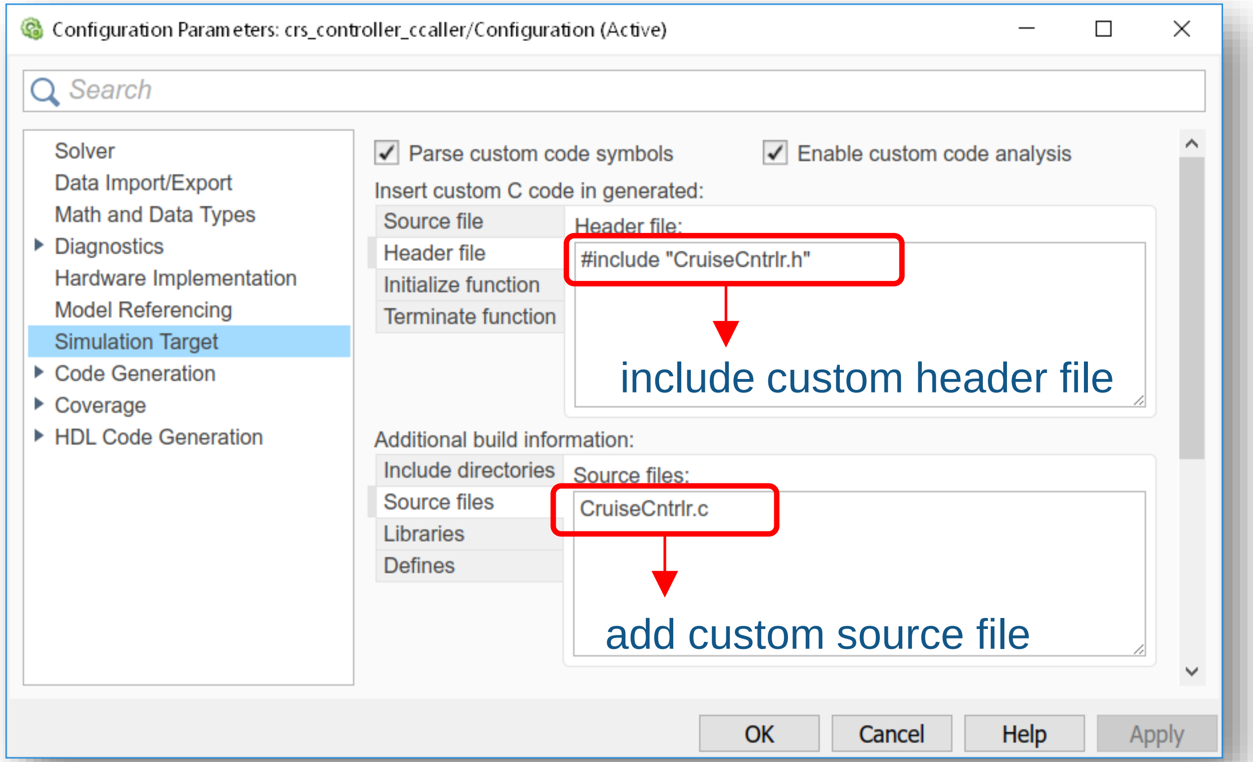


C/C++ Code

Using C Caller Block

1. Specify Custom Code in the Configuration Parameters

- Custom code is specified on the Configuration Parameters.
  - **The Header file section**: Any code that needs to be inserted into the header file
  - **The Source files section:** List of source files that needs to be compiled


Configuration Parameters: crs_controller_ccaller/Configuration (Active)

Simulation Target

✓ Parse custom code symbols
✓ Enable custom code analysis

Insert custom C code in generated:

Header file:

```
#include "CruiseCntrlr.h"
```

include custom header file

Additional build information:

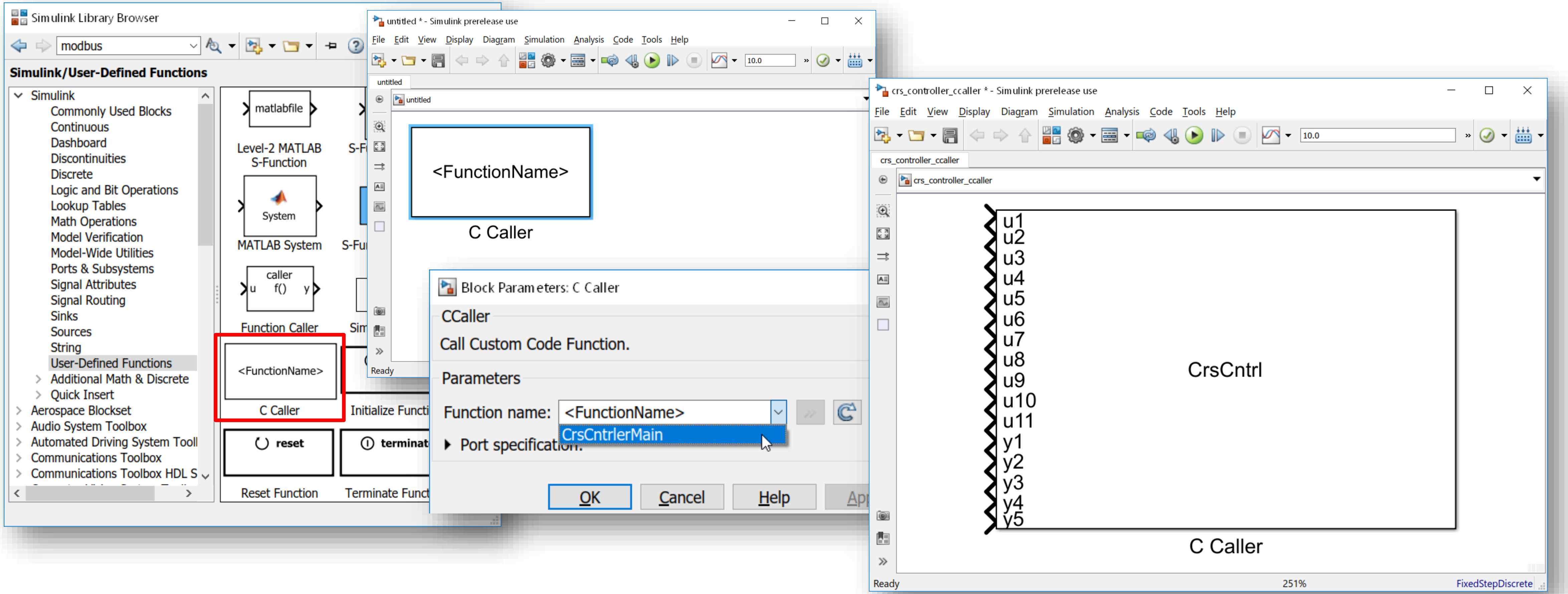
Source files:

```
CruiseCntrlr.c
```

add custom source file

Using C Caller Block

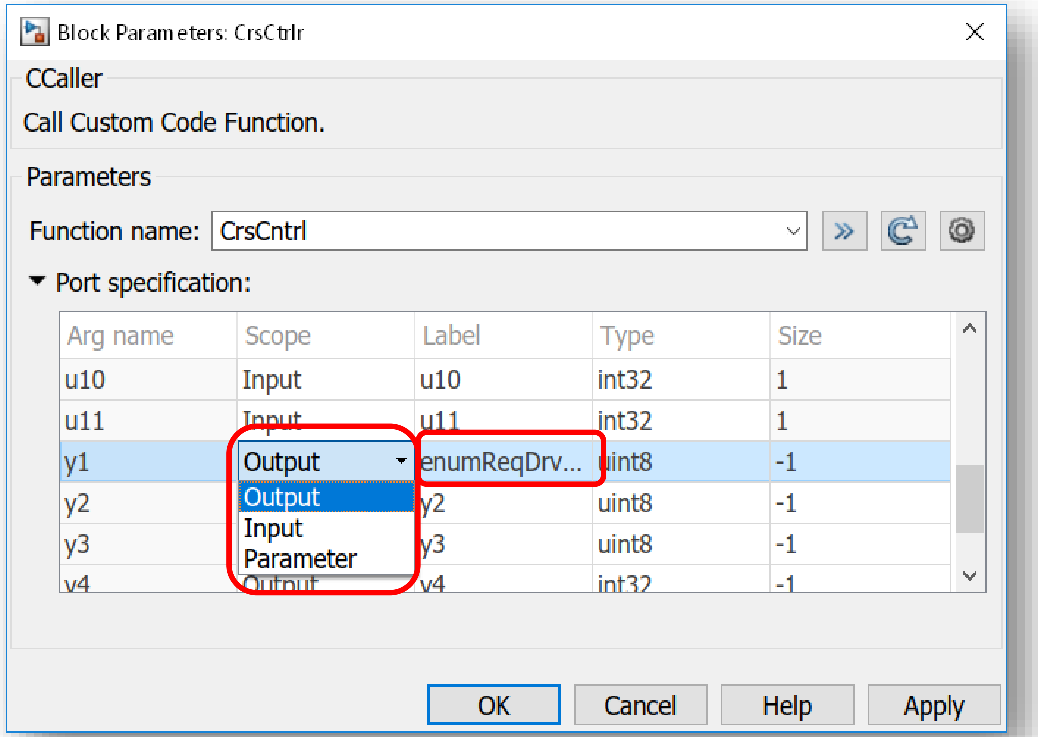
2. Select the function that you want to call



Using C Caller Block

3. Customize the function that you want to call

- Mapping inputs, outputs or parameters to C Caller Block



Block Parameters: CrsCtrlr

CCaller

Call Custom Code Function.

Parameters

Function name: CrsCntrl

Port specification:

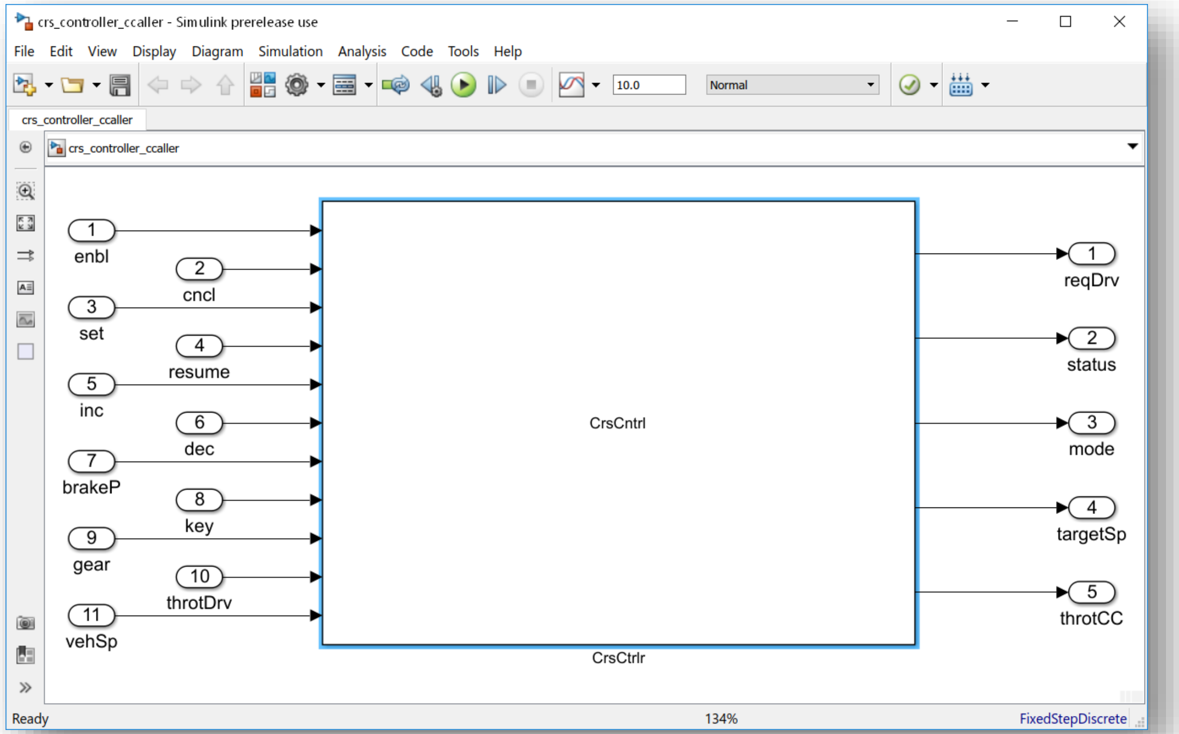
Arg name	Scope	Label	Type	Size
u10	Input	u10	int32	1
u11	Input	u11	int32	1
y1	Output	enumReqDrv...	uint8	-1
y2		y2	uint8	-1
y3		y3	uint8	-1
y4	Output	y4	int32	-1

Output

Input

Parameter

OK Cancel Help Apply

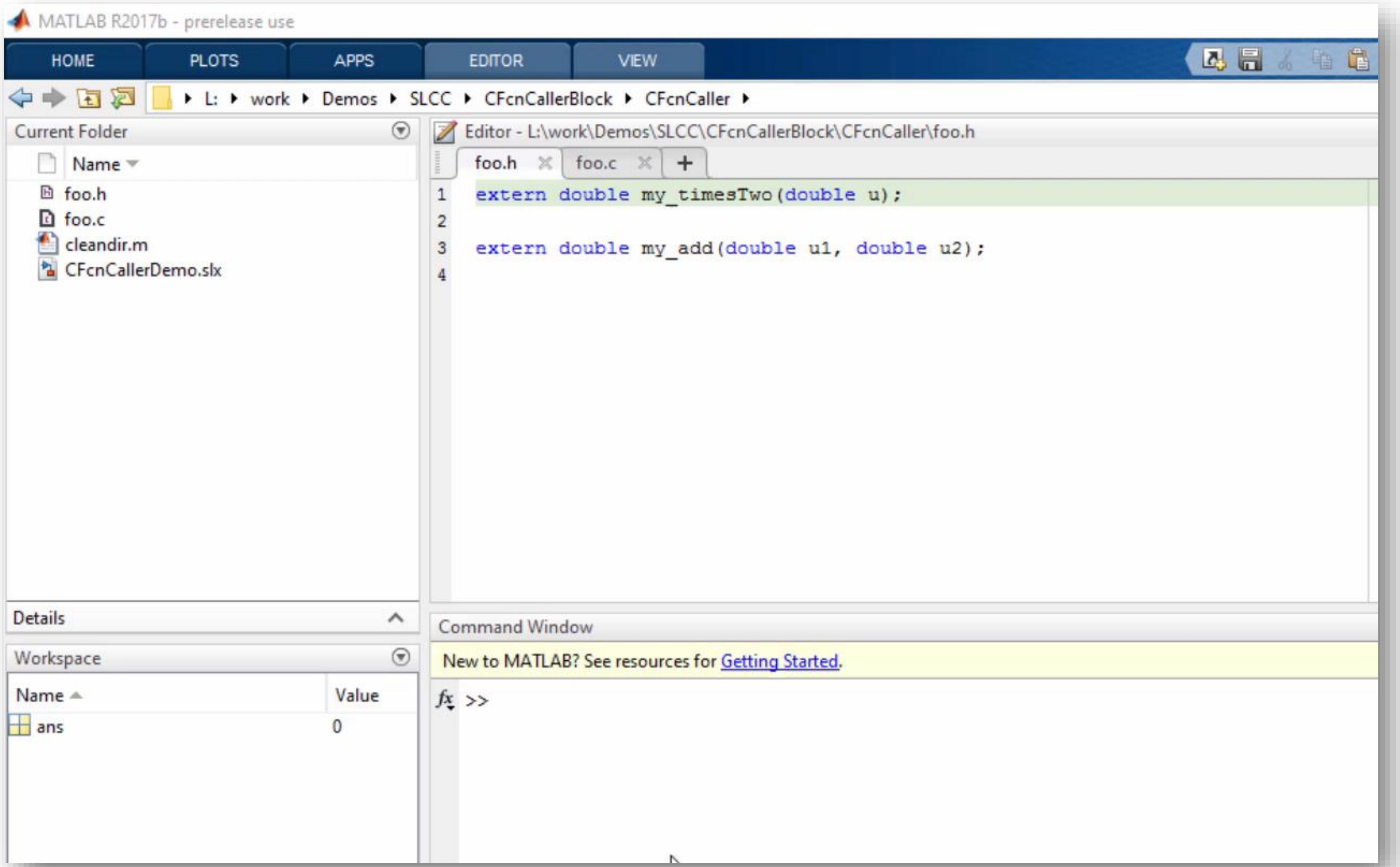


1) Change argument scope to "Output"

2) (Optional) Override with a better output name

3) Complete the test model with connecting signal ports

Demo: Simple C Caller



MATLAB R2017b - prerelease use

foo.h
foo.c
cleandir.m
CFcnCallerDemo.slx

Name	Value
ans	0

Editor - L:\work\Demos\SLCC\CFcnCallerBlock\CFcnCaller\foo.h

```
extern double my_timesTwo(double u);

extern double my_add(double u1, double u2);

```

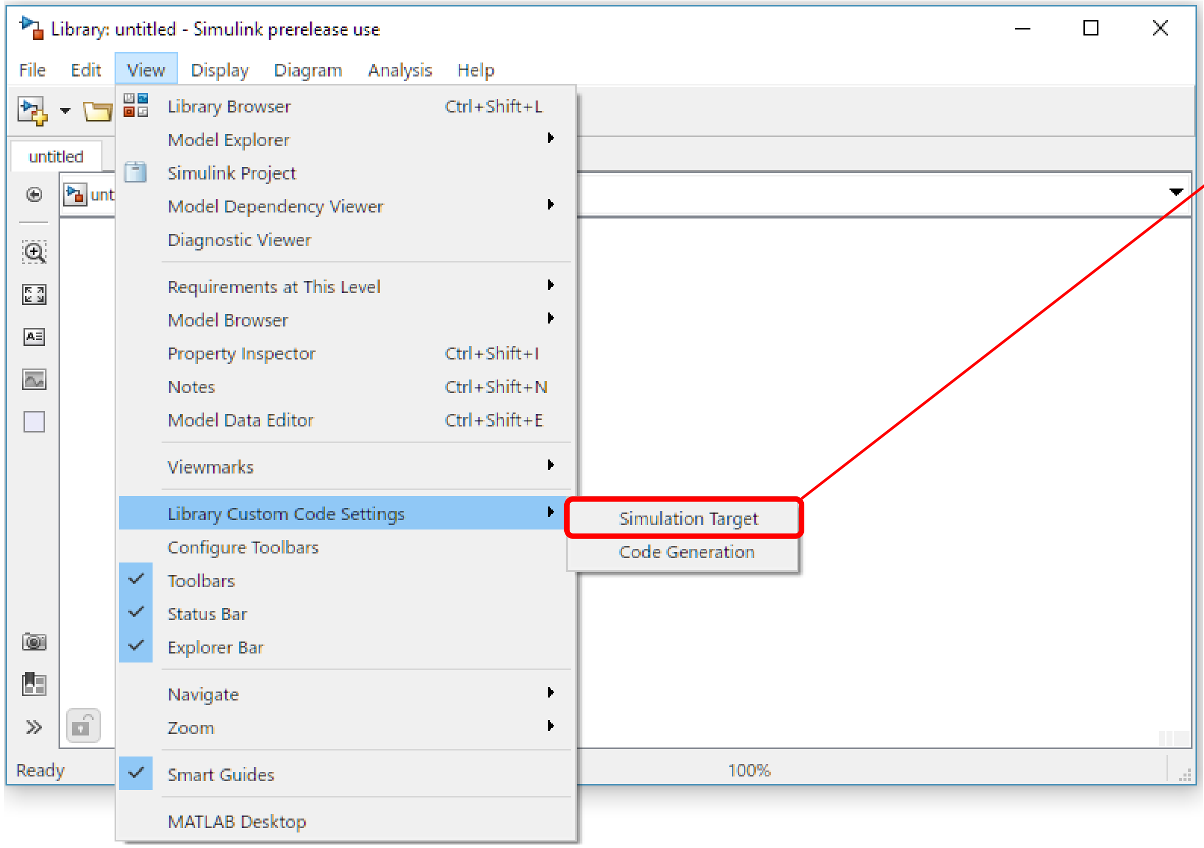
Command Window

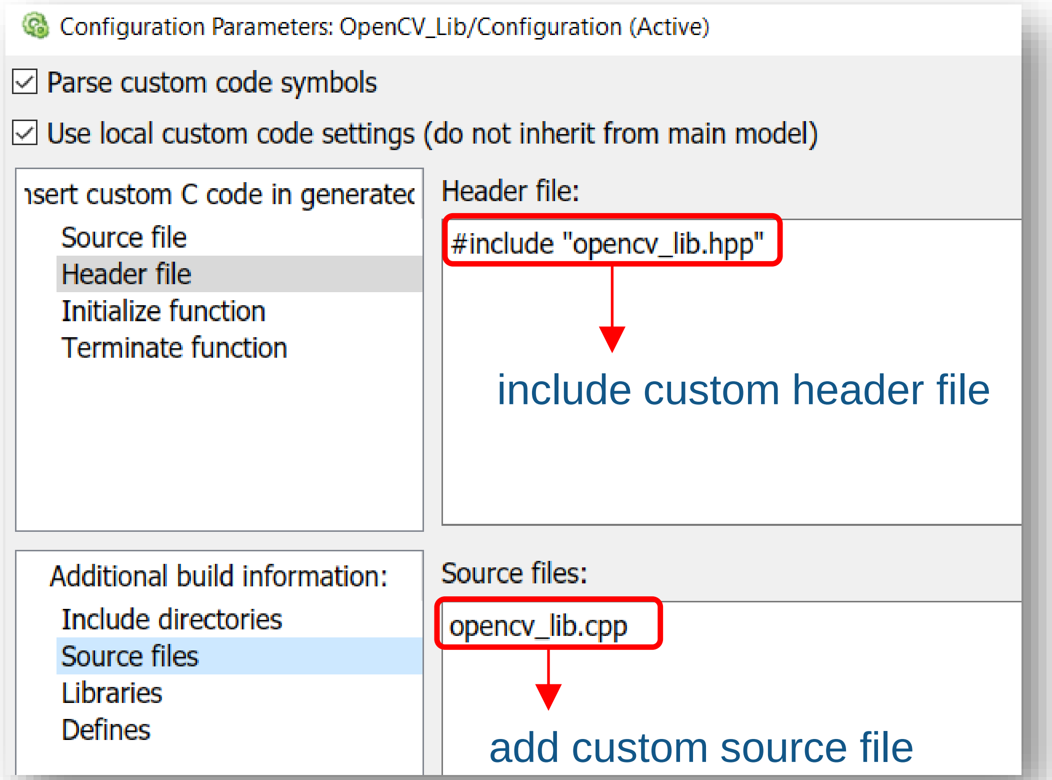
New to MATLAB? See resources for Getting Started.

```
>>
```

Library Workflow

- C Caller block can be configured as a library model
  - Custom Code Settings can be accessed from View Menu → Library Custom Code Settings




Configuration Parameters: OpenCV_Lib/Configuration (Active)

☑ Parse custom code symbols

☑ Use local custom code settings (do not inherit from main model)

Header file:

#include "opencv_lib.hpp"

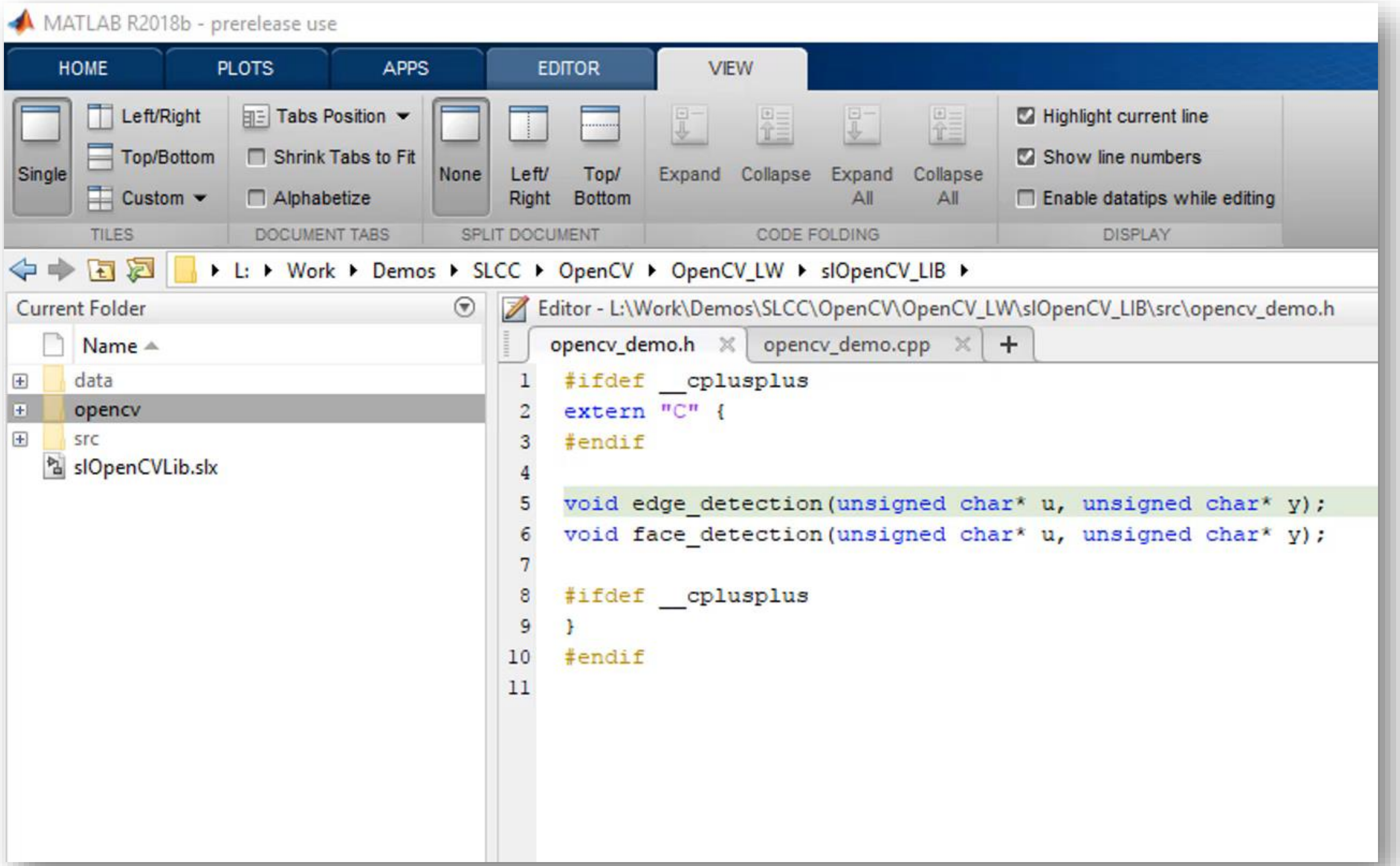
include custom header file

Source files:

opencv_lib.cpp

add custom source file

Demo: Reusable Library Workflow with OpenCV



[✓] Highlight current line
[✓] Show line numbers
[ ] Enable datatips while editing

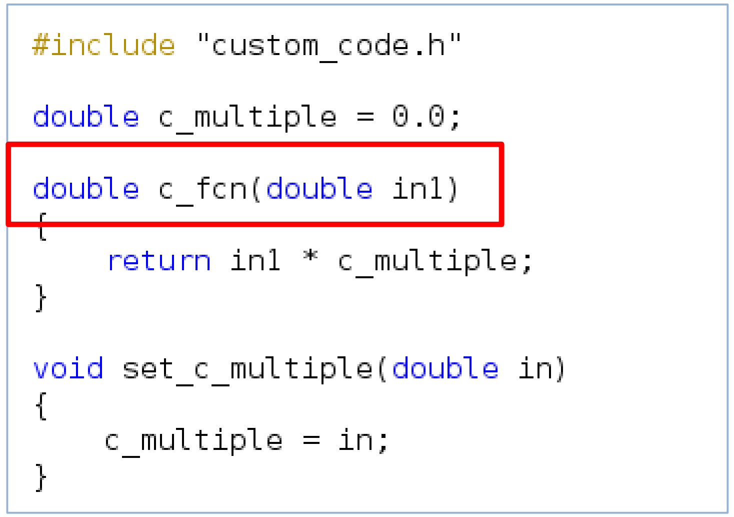
Current Folder:
- data
- opencv
- src
- slOpenCVLib.slx

Editor - L:\Work\Demos\SLCC\OpenCV\OpenCV_LW\slOpenCV_LIB\src\opencv_demo.h

```
#ifdef __cplusplus
extern "C" {
#endif

void edge_detection(unsigned char* u, unsigned char* y);
void face_detection(unsigned char* u, unsigned char* y);

#ifdef __cplusplus
}
#endif

```

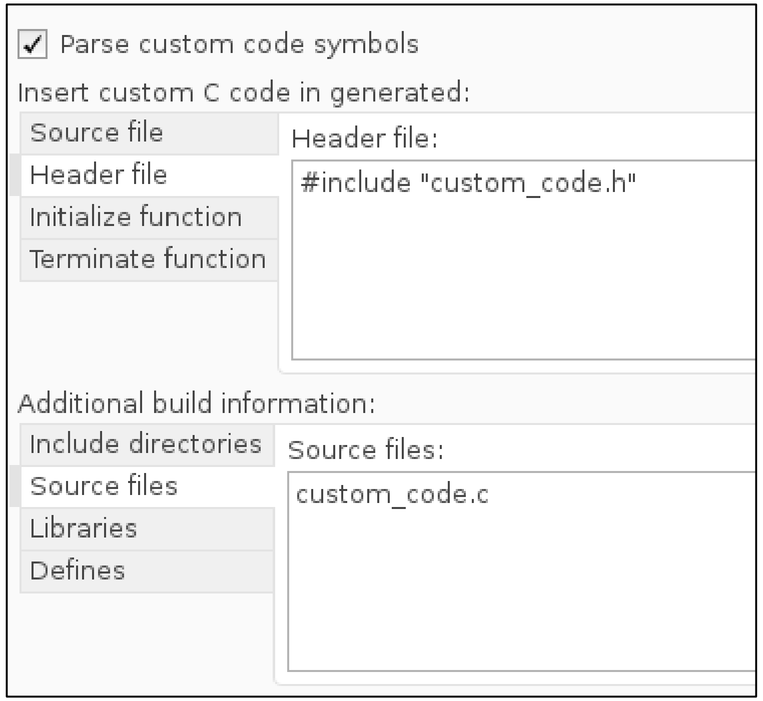
Legacy Code Evaluation in Stateflow

- Using legacy code in Stateflow chart



```
#include "custom_code.h"

double c_multiple = 0.0;

double c_fcn(double in1)
{
    return in1 * c_multiple;
}

void set_c_multiple(double in)
{
    c_multiple = in;
}
```

Step 1: Have C code



☑ Parse custom code symbols

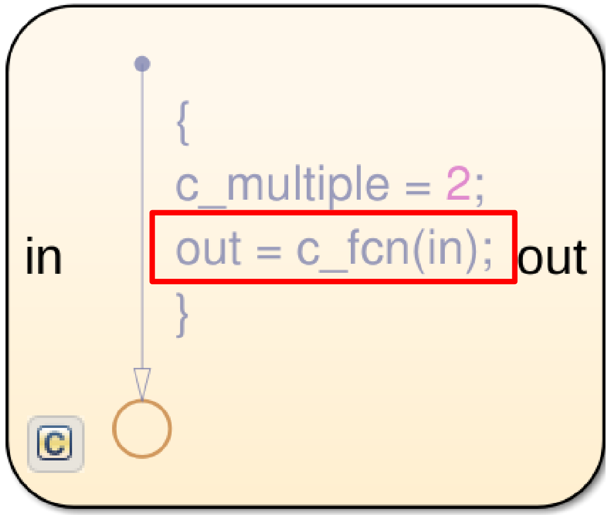
Insert custom C code in generated:

Source file	Header file:
Header file	#include "custom_code.h"
Initialize function	
Terminate function	

Additional build information:

Include directories	Source files:
Source files	custom_code.c
Libraries	
Defines	

Step 2: Put on Config. Set



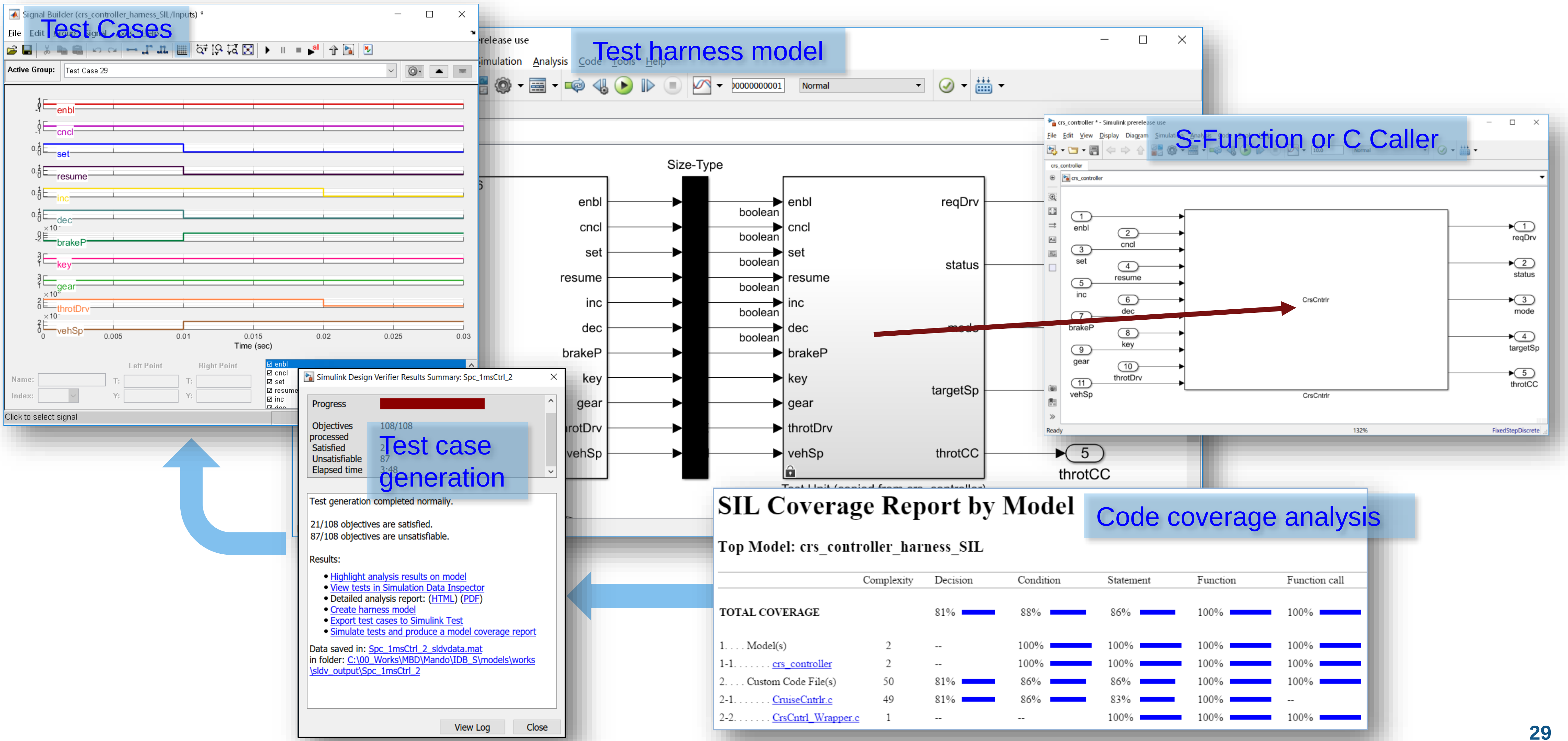
```
{
c_multiple = 2;
out = c_fcn(in);
}
```

Step 3: Use in Stateflow

Agenda

- Overview to V&V in Model-Based Design
- Legacy code integration using Simulink
- Workflow for legacy code verification

Legacy Code Verification using Simulink V&V



Simulink Design Verifier Results Summary: Spc_1msCtrl_2

Progress

Objectives processed 108/108
Satisfied 2
Unsatisfiable 87
Elapsed time 3:48

Test generation completed normally.

21/108 objectives are satisfied.
87/108 objectives are unsatisfiable.

Results:

- Highlight analysis results on model
- View tests in Simulation Data Inspector
- Detailed analysis report: (HTML) (PDF)
- Create harness model
- Export test cases to Simulink Test
- Simulate tests and produce a model coverage report

Data saved in: Spc_1msCtrl_2_sldvdata.mat
in folder: C:\00_Works\MBD\Mando\IDB_S\models\works\sldv_output\Spc_1msCtrl_2

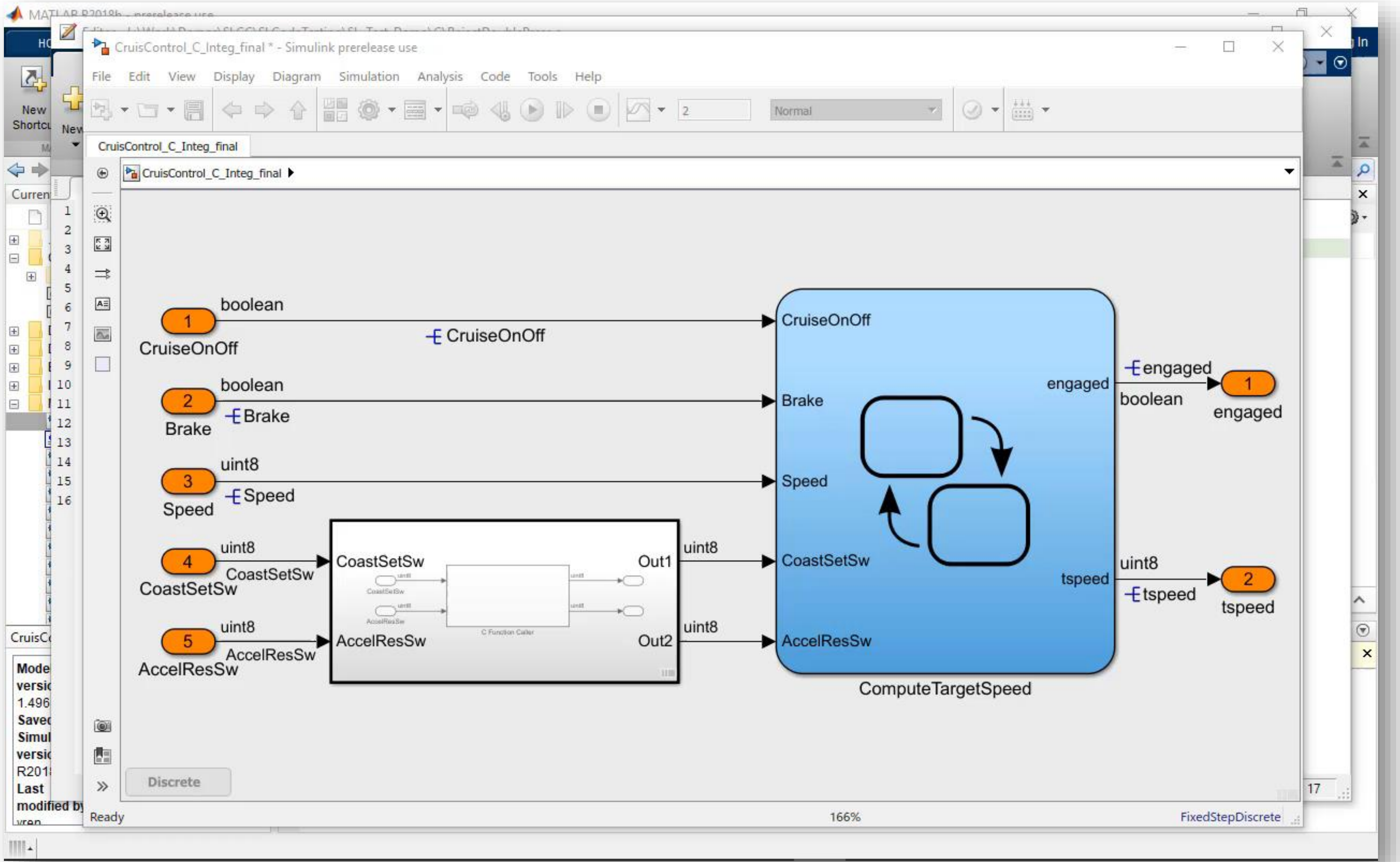
View Log | Close

SIL Coverage Report by Model

Top Model: crs_controller_harness_SIL

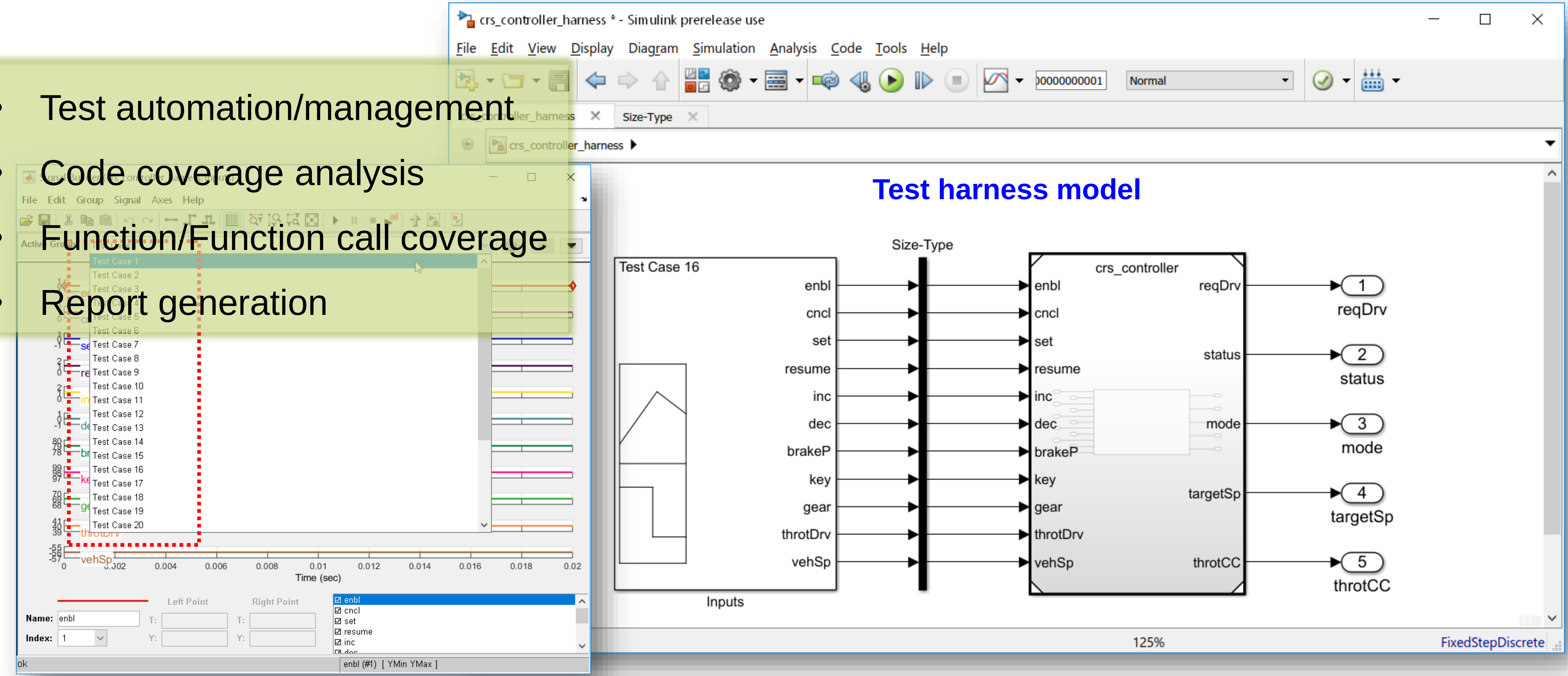
	Complexity	Decision	Condition	Statement	Function	Function call
TOTAL COVERAGE		81%	88%	86%	100%	100%
1. . . . Model(s)	2	--	100%	100%	100%	100%
1-1. . . . . . . crs_controller	2	--	100%	100%	100%	100%
2. . . . Custom Code File(s)	50	81%	86%	86%	100%	100%
2-1. . . . . . . CruiseCntrlr.c	49	81%	86%	83%	100%	--
2-2. . . . . . . CrsCntrl_Wrapper.c	1	--	--	100%	100%	100%

Demo: Legacy C Code Verification



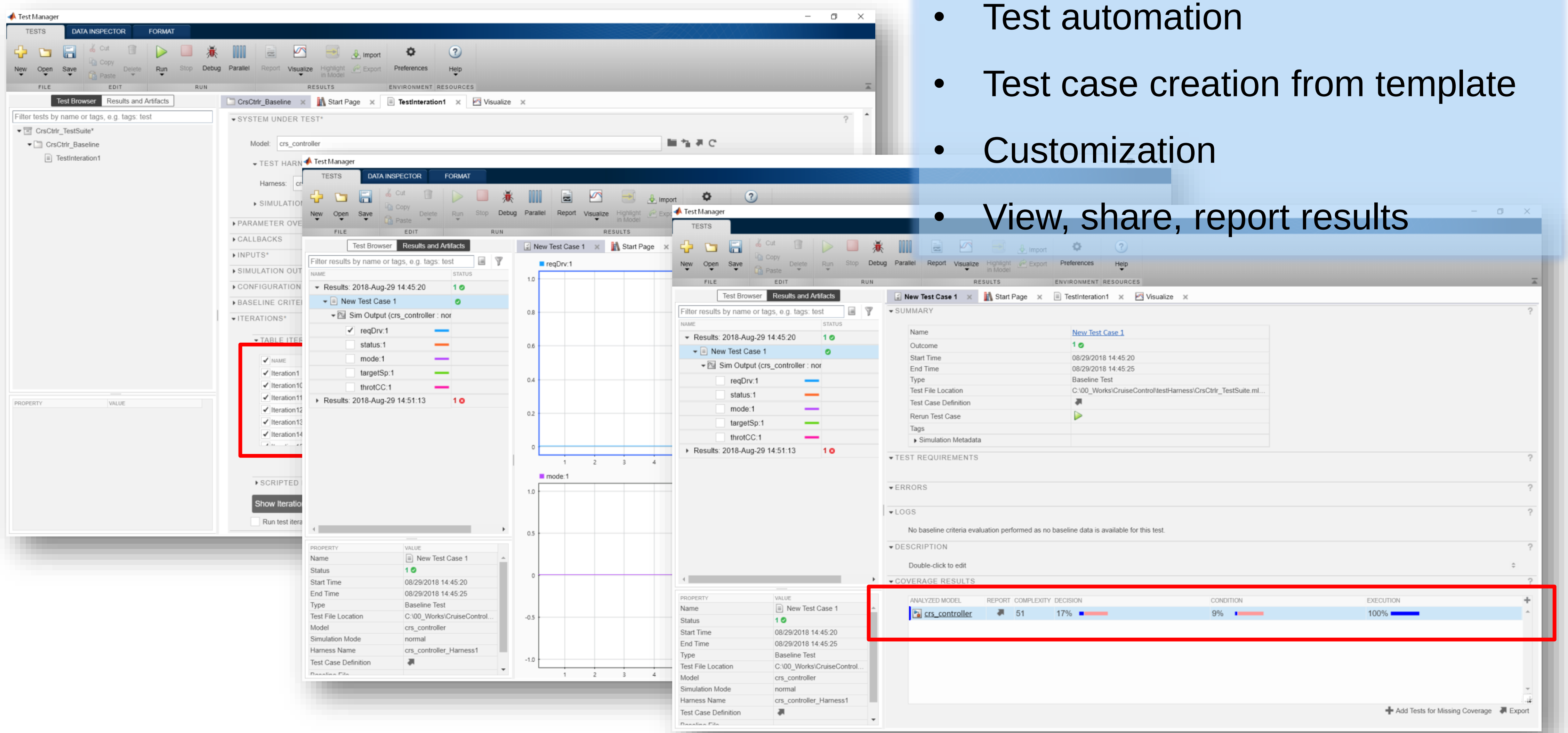
Needs for Test Automation

- Test automation/management
- Code coverage analysis
- Function/Function call coverage
- Report generation



Test Automation with Test Manager

- Test automation
- Test case creation from template
- Customization
- View, share, report results



Filter tests by name or tags, e.g. tags: test

- CrsCtrlr_TestSuite*
  - CrsCtrlr_Baseline
    - TestIteration1

Model: crs_controller

NAME
Iteration1
Iteration10
Iteration11
Iteration12
Iteration13
Iteration14

Filter results by name or tags, e.g. tags: test

NAME	STATUS
Results: 2018-Aug-29 14:45:20	1 ✓
New Test Case 1	✓
Sim Output (crs_controller : nor	
reqDrv:1	
status:1	
mode:1	
targetSp:1	
throtCC:1	
Results: 2018-Aug-29 14:51:13	1 ✗

PROPERTY	VALUE
Name	New Test Case 1
Status	1 ✓
Start Time	08/29/2018 14:45:20
End Time	08/29/2018 14:45:25
Type	Baseline Test
Test File Location	C:\00_Works\CruiseControl...
Model	crs_controller
Simulation Mode	normal
Harness Name	crs_controller_Harness1
Test Case Definition	

SUMMARY

Name	New Test Case 1
Outcome	1 ✓
Start Time	08/29/2018 14:45:20
End Time	08/29/2018 14:45:25
Type	Baseline Test
Test File Location	C:\00_Works\CruiseControl\testHarness\CrsCtrlr_TestSuite.ml...
Test Case Definition	
Rerun Test Case	
Tags	
Simulation Metadata	

TEST REQUIREMENTS

ERRORS

LOGS

No baseline criteria evaluation performed as no baseline data is available for this test.

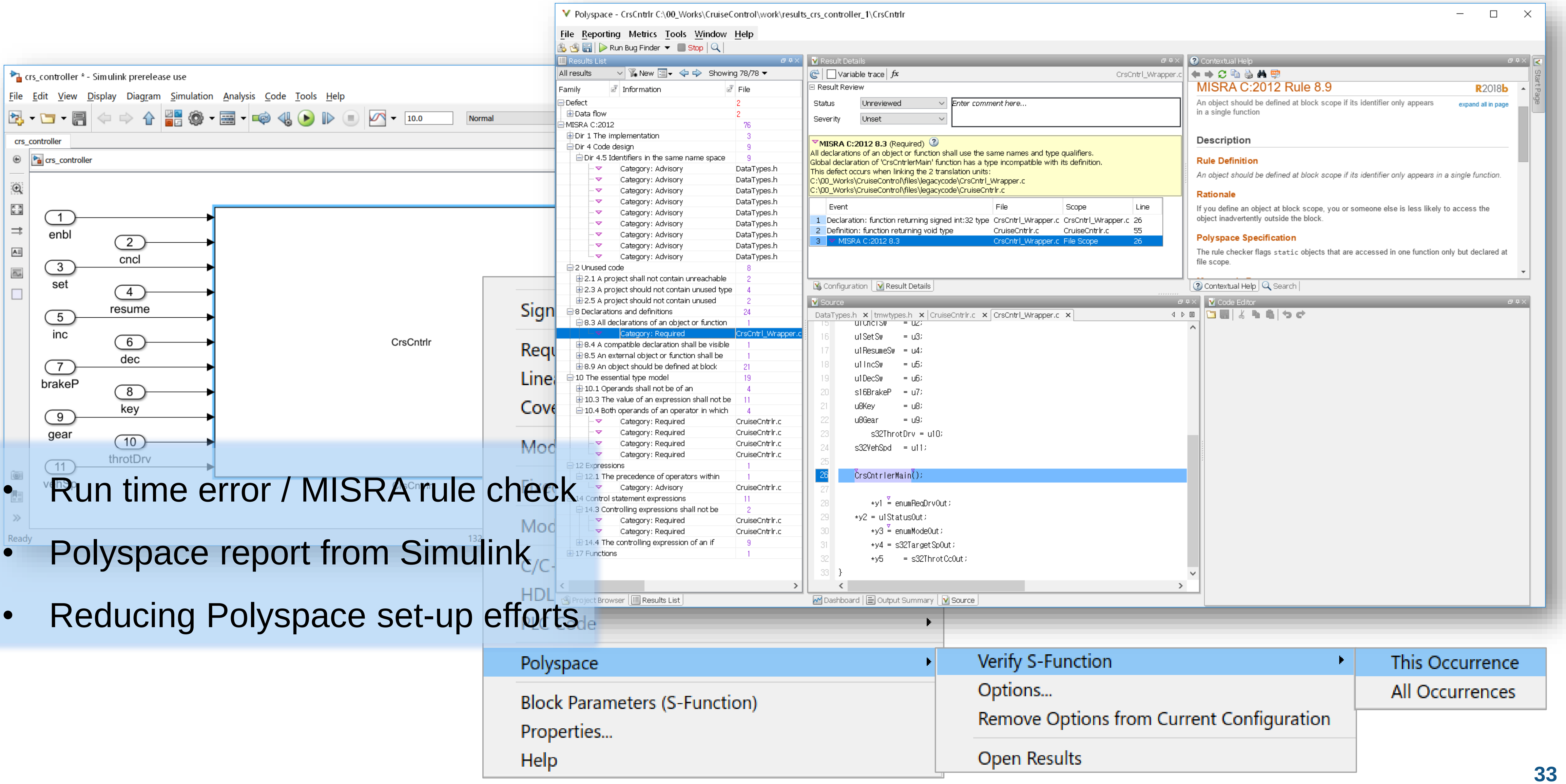
DESCRIPTION

Double-click to edit

COVERAGE RESULTS

ANALYZED MODEL	REPORT	COMPLEXITY	DECISION	CONDITION	EXECUTION
crs_controller		51	17%	9%	100%

Static Code Analysis



Polyspace - CrsCntrlr C:\00_Works\CruiseControl\work\results_crs_controller_1\CrsCntrlr

File Reporting Metrics Tools Window Help

Results List — All results — Showing 78/78

Family	Information	File
Defect		2
Data flow		2
MISRA C:2012		76
Dir 1 The implementation		3
Dir 4 Code design		9
Dir 4.5 Identifiers in the same name space		9
	Category: Advisory	DataTypes.h
	Category: Advisory	DataTypes.h
	Category: Advisory	DataTypes.h
	Category: Advisory	DataTypes.h
	Category: Advisory	DataTypes.h
	Category: Advisory	DataTypes.h
	Category: Advisory	DataTypes.h
	Category: Advisory	DataTypes.h
	Category: Advisory	DataTypes.h
2 Unused code		8
2.1 A project shall not contain unreachable		2
2.3 A project should not contain unused type		4
2.5 A project should not contain unused		2
8 Declarations and definitions		24
8.3 All declarations of an object or function		1
	Category: Required	CrsCntrl_Wrapper.c
8.4 A compatible declaration shall be visible		1
8.5 An external object or function shall be		1
8.9 An object should be defined at block		21
10 The essential type model		19
10.1 Operands shall not be of an		4
10.3 The value of an expression shall not be		11
10.4 Both operands of an operator in which		4
	Category: Required	CruiseCntrlr.c
	Category: Required	CruiseCntrlr.c
	Category: Required	CruiseCntrlr.c
	Category: Required	CruiseCntrlr.c
12 Expressions		1
12.1 The precedence of operators within		1
	Category: Advisory	CruiseCntrlr.c
14 Control statement expressions		11
14.3 Controlling expressions shall not be		2
	Category: Required	CruiseCntrlr.c
	Category: Required	CruiseCntrlr.c
14.4 The controlling expression of an if		9
17 Functions		1

Result Details — Result Review

Status: Unreviewed — Severity: Unset — Enter comment here...

MISRA C:2012 8.3 (Required)
All declarations of an object or function shall use the same names and type qualifiers.
Global declaration of 'CrsCntrlerMain' function has a type incompatible with its definition.
This defect occurs when linking the 2 translation units:
C:\00_Works\CruiseControl\files\legacycode\CrsCntrl_Wrapper.c
C:\00_Works\CruiseControl\files\legacycode\CruiseCntrlr.c

	Event	File	Scope	Line
1	Declaration: function returning signed int:32 type	CrsCntrl_Wrapper.c	CrsCntrl_Wrapper.c	26
2	Definition: function returning void type	CruiseCntrlr.c	CruiseCntrlr.c	55
3	MISRA C:2012 8.3	CrsCntrl_Wrapper.c	File Scope	26

Contextual Help

MISRA C:2012 Rule 8.9 — R2018b

An object should be defined at block scope if its identifier only appears in a single function

Description

Rule Definition

An object should be defined at block scope if its identifier only appears in a single function.

Rationale

If you define an object at block scope, you or someone else is less likely to access the object inadvertently outside the block.

Polyspace Specification

The rule checker flags `static` objects that are accessed in one function only but declared at file scope.

Source — CrsCntrl_Wrapper.c

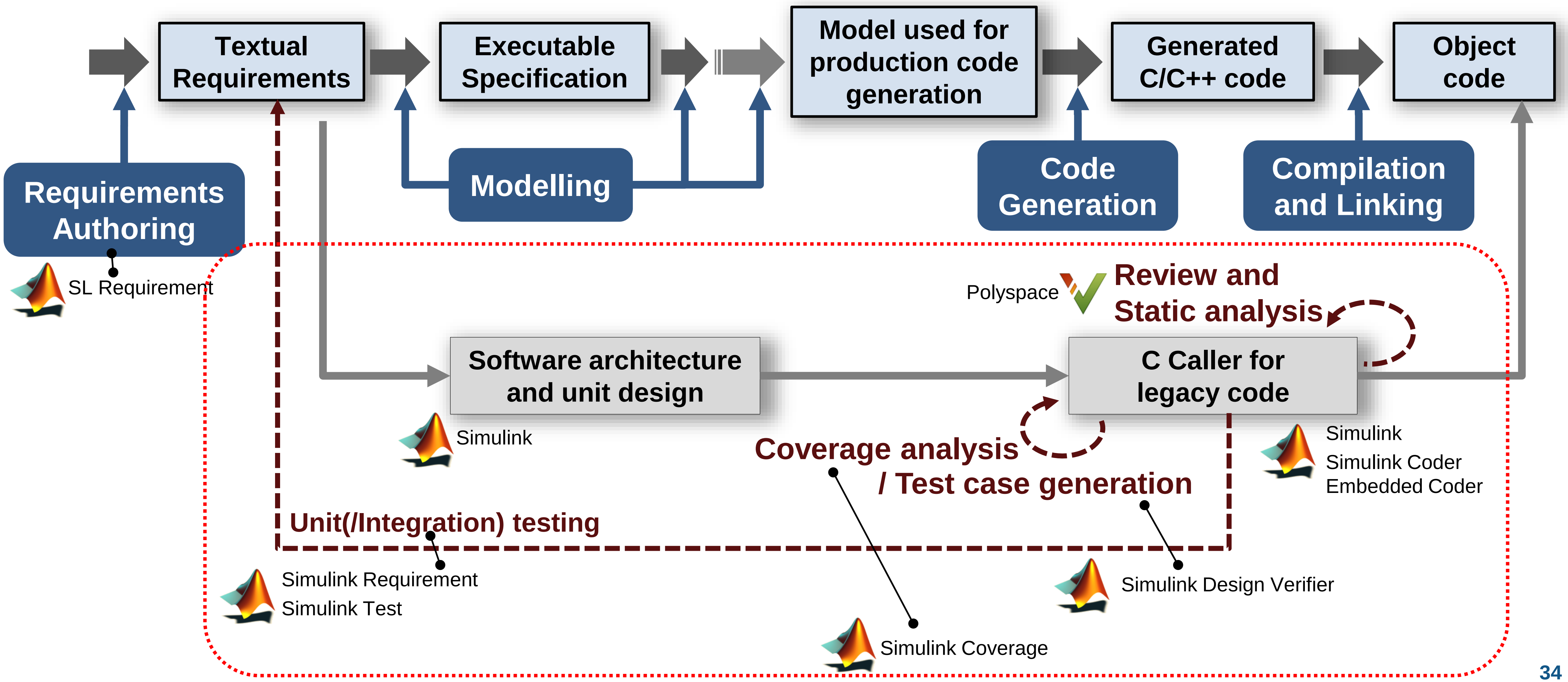
```
    u1SetSw    = u3;
    u1ResumeSw = u4;
    u1IncSw    = u5;
    u1DecSw    = u6;
    s16BrakeP  = u7;
    u8Key      = u8;
    u8Gear     = u9;
    s32ThrotDrv = u10;
    s32VehSpd  = u11;

    CrsCntrlerMain();

    *y1 = enumReqDrvOut;
    *y2 = u1StatusOut;
    *y3 = enumModeOut;
    *y4 = s32TargetSpOut;
    *y5 = s32ThrotCcOut;
}
```

crs_controller * - Simulink prerelease use

File Edit View Display Diagram Simulation Analysis Code Tools Help

- Run time error / MISRA rule check
- Polyspace report from Simulink
- Reducing Polyspace set-up efforts

Key Takeaways



Thank You!