

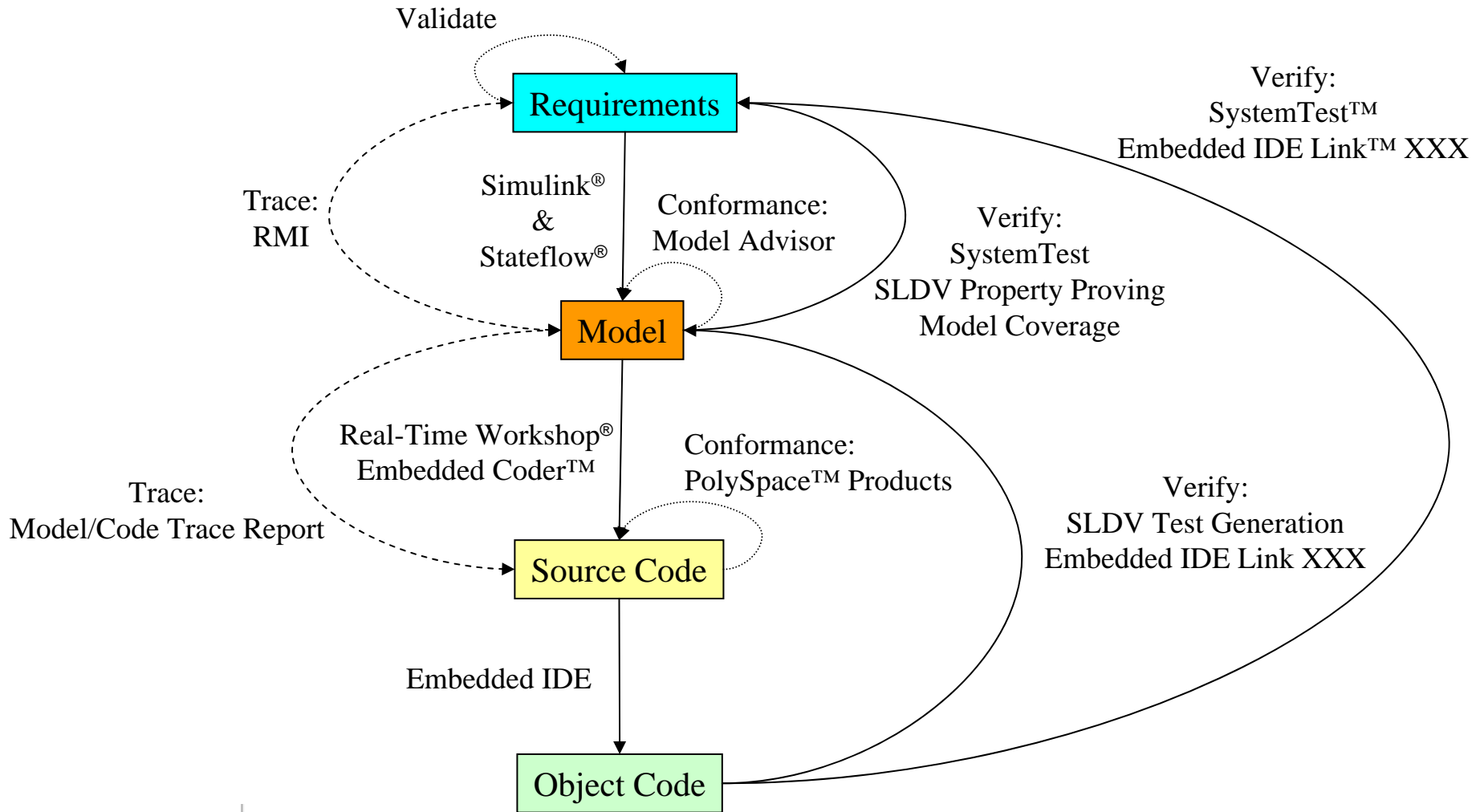
# Model-Based Design for Safety-Critical and Mission-Critical Applications

**Bill Potter**  
**Technical Marketing**  
**April 17, 2008**

MathWorks Symposium

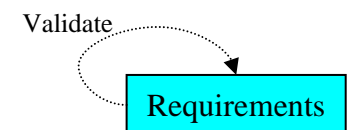
Adopting Model-Based Design  
within Aerospace and Defense

# Safety-Critical Model-Based Design Workflow

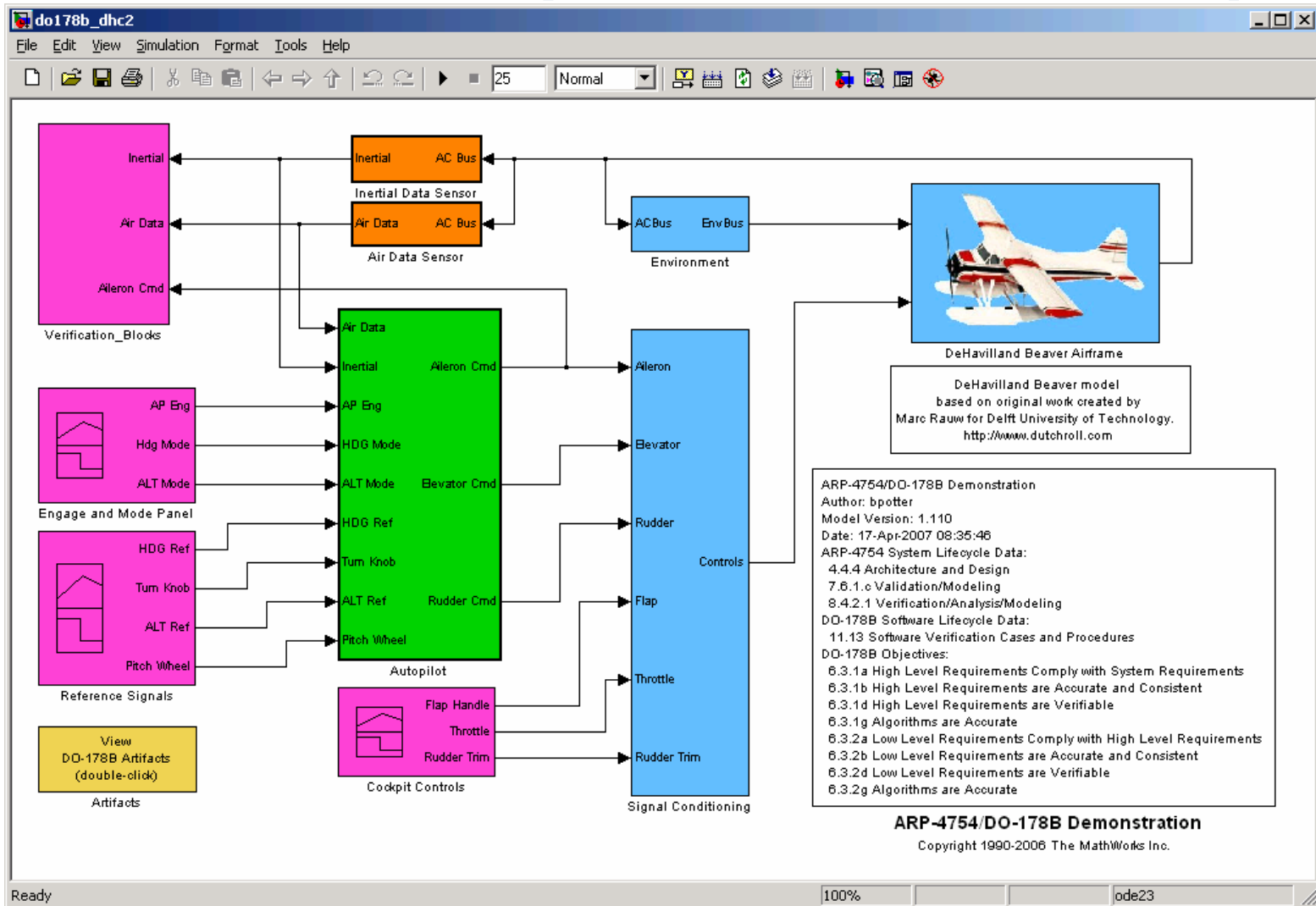


# Requirements Process for Model-Based Design

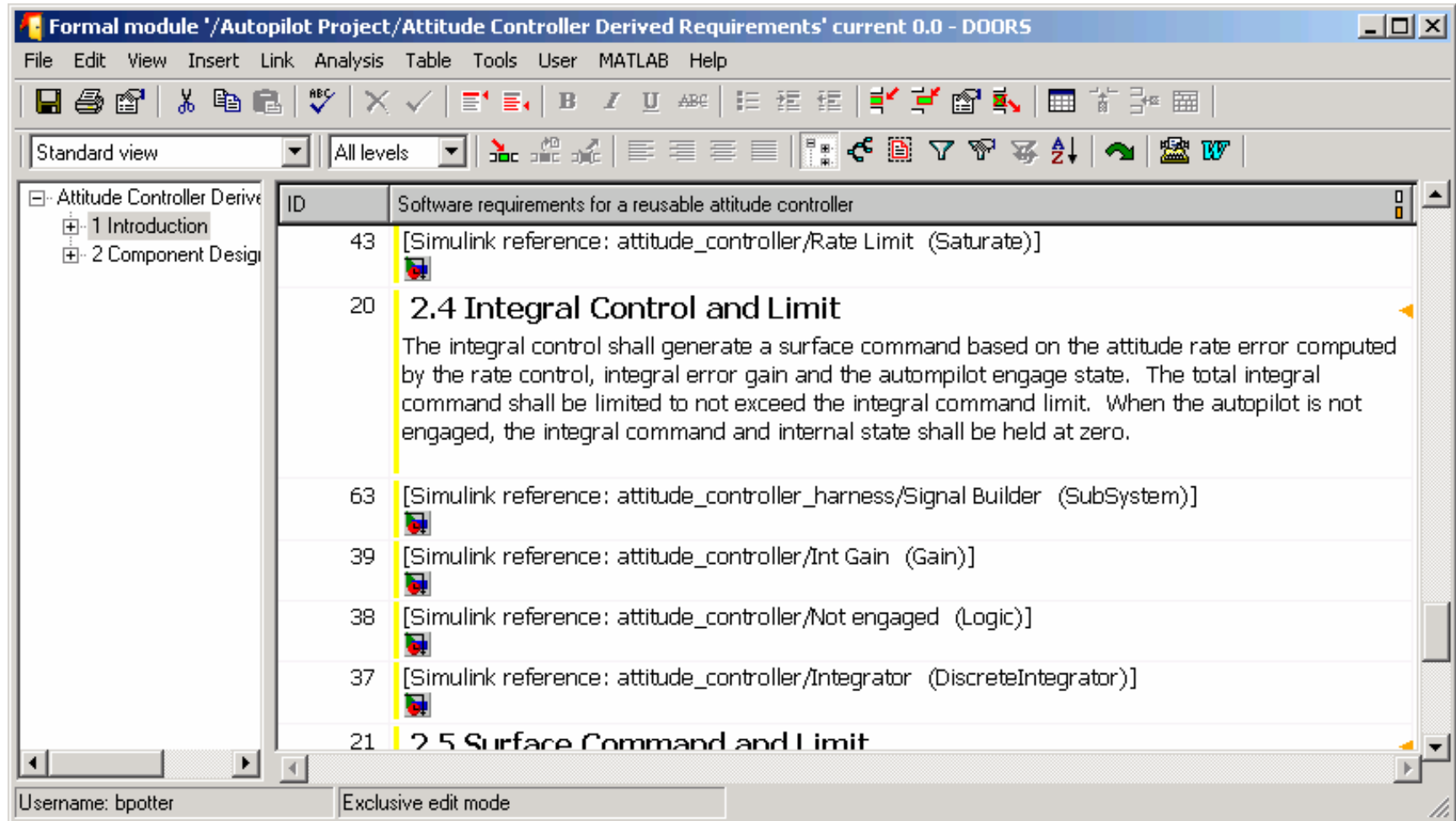
- Functional, operational, and safety requirements
  - Exist one level above the model
  - Models trace to requirements
- Requirements validation - complete and correct
  - Simulation is a validation technique
  - Traceability can identify incomplete requirements
  - Model coverage can identify incomplete requirements
- Requirements based test cases
  - Test cases trace to requirements



# Simulation example – controller and plant



# Requirements trace example – view from DOORS® to Simulink



Formal module: '/Autopilot Project/Attitude Controller Derived Requirements' current 0.0 - DOORS

File Edit View Insert Link Analysis Table Tools User MATLAB Help

Standard view All levels

Attitude Controller Derived Requirements

- 1 Introduction
- 2 Component Design

ID	Software requirements for a reusable attitude controller
43	[Simulink reference: attitude_controller/Rate Limit (Saturate)]
20	<b>2.4 Integral Control and Limit</b> The integral control shall generate a surface command based on the attitude rate error computed by the rate control, integral error gain and the autopilot engage state. The total integral command shall be limited to not exceed the integral command limit. When the autopilot is not engaged, the integral command and internal state shall be held at zero.
63	[Simulink reference: attitude_controller_harness/Signal Builder (SubSystem)]
39	[Simulink reference: attitude_controller/Int Gain (Gain)]
38	[Simulink reference: attitude_controller/Not engaged (Logic)]
37	[Simulink reference: attitude_controller/Integrator (DiscreteIntegrator)]
21	<b>2.5 Surface Command and Limit</b>

Username: bpotter Exclusive edit mode

# Requirements trace example – view from Simulink to DOORS

attitude\_controller Model Requirements Report

File Edit View Go Debug Desktop Window Help

Location: file:///C:/local\_work\_area/demos/autopilot\_R2007a\_mdref/attitude\_controller\_reqreport.html

## Chapter 2. System - attitude\_controller

Attitude Control using displacement rate and integral

1 Displacement Command (Disp\_Cmd) with limit block (Disp Limit -dispLim to dispLim (deg))

2 Displacement Feedback (Disp\_FB)

3 Rate Feedback (Rate\_FB)

4 Engaged/Not engaged status block

intGain (Int Gain)

rateGain (Rate Gain)

Integrator (K Ts,  $\frac{1}{s}$ , -intLim to intLim (deg))

Cmd Limit (Cmd Limit -cmdLim to cmdLim (deg))

Surf\_Cmd

Attitude Controller

Author: bpotter

Version: 1.21

Date: 18-Apr-2007 00:22:44

DO-178B Software Lifecycle Data

11.9 Software Requirements Data

11.10 Design Description

DO-178B Objectives

- 5.2.1a Low Level Requirements are Developed
- 5.2.1a Software Architecture is Developed
- 5.2.1b Derived Low Level Requirements are Developed

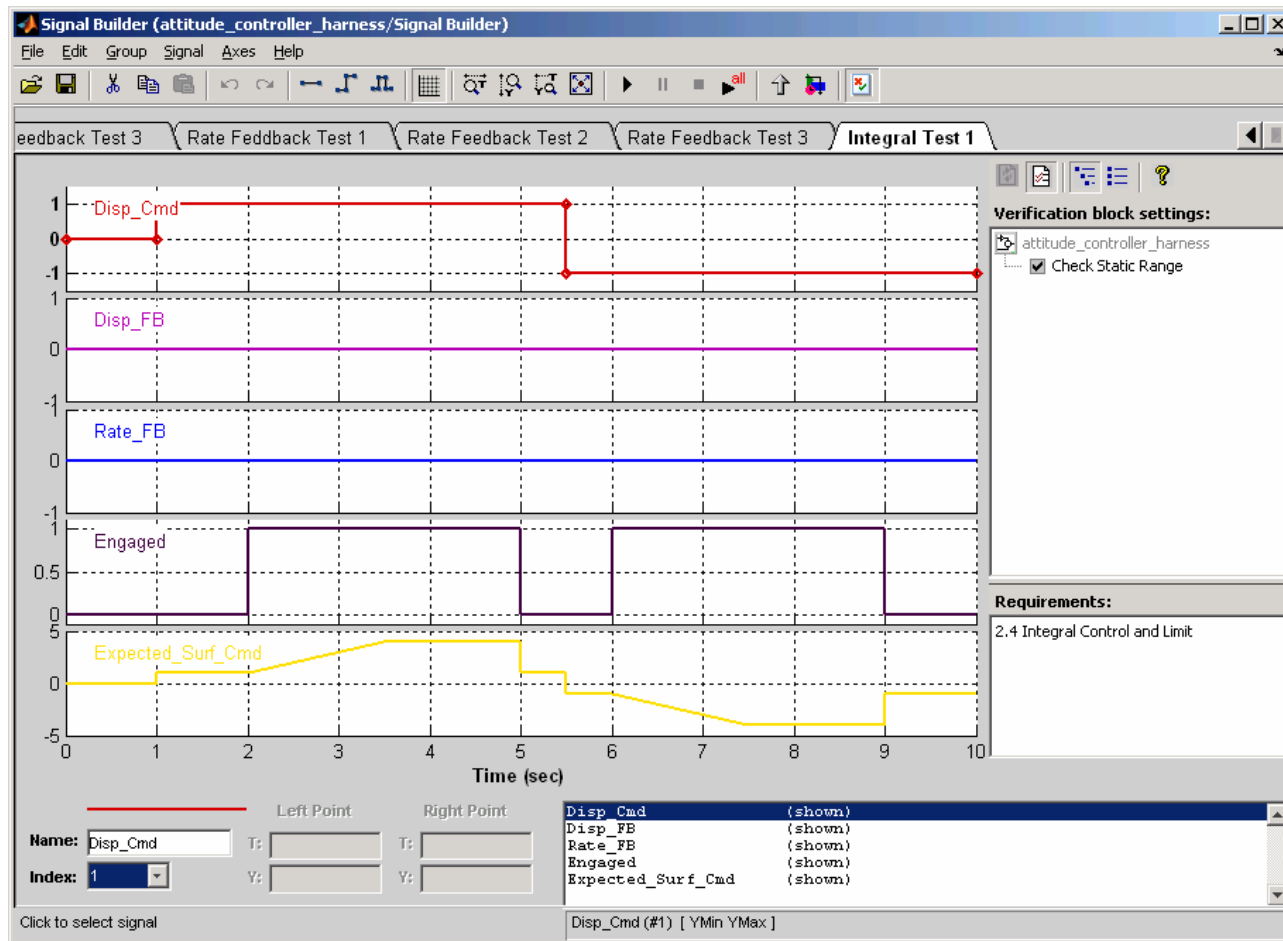
**ARP-4754/DO-178B Demonstration**

Copyright 1990-2007 The MathWorks Inc.

**Table 2.1. Block Requirements Table**

Name	Requirements
Cmd Limit	2.1.2 Parameters 00000022 #23 2.5 Surface Command and Limit 00000022 #21
Disp Gain	2.1.2 Parameters 00000022 #23 2.2 Attitude Control and Limit 00000022 #16
Disp Limit	2.1.2 Parameters 00000022 #23 2.2 Attitude Control and Limit 00000022 #16
Disp_Cmd	2.1.1 Inputs 00000022 #22

# Requirements based test trace example – view from Simulink Signal Builder block to DOORS



# Model coverage report example

attitude\_controller Coverage Report

File Edit View Go Debug Desktop Window Help

Location: file:///C:/local\_work\_area/demos/autopilot\_R2007a\_mdldref/attitude\_control\_results19486407.html

**Discrete integrator block "Integrator"**

Parent: [/attitude\\_controller](#)

**Metric Coverage**

Cyclomatic Complexity 3

Decision (D1) 100% (6/6) decision outcomes

**Decisions analyzed:**

Reset	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
false	201/401	201/401	201/401	201/401	201/401	201/401	201/401	201/401	201/401	240/401	2049/4010
true	200/401	200/401	200/401	200/401	200/401	200/401	200/401	200/401	200/401	161/401	1961/4010
integration result <= lower limit	50%	50%	50%	50%	50%	50%	50%	50%	50%	100%	100%
false	401/401	401/401	401/401	401/401	401/401	401/401	401/401	401/401	401/401	283/342	3892/3951
true	0/401	0/401	0/401	0/401	0/401	0/401	0/401	0/401	0/401	59/342	59/3951
integration result >= upper limit	50%	50%	50%	50%	50%	50%	50%	50%	50%	100%	100%
false	401/401	401/401	401/401	401/401	401/401	401/401	401/401	401/401	401/401	342/401	3951/4010
true	0/401	0/401	0/401	0/401	0/401	0/401	0/401	0/401	0/401	59/401	59/4010

**Logic block "Not engaged"**

Parent: [/attitude\\_controller](#)

**Metric Coverage**

Cyclomatic Complexity 0

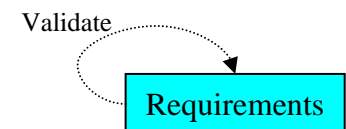
Condition (C1) 100% (2/2) condition outcomes

Done



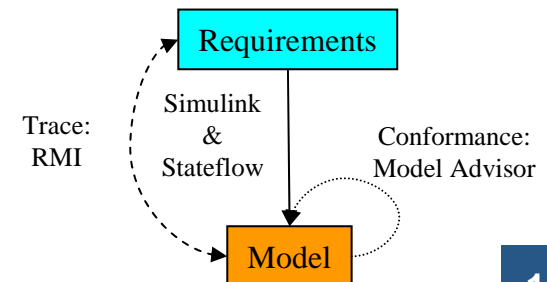
# Requirements Process take-aways

- Early requirements validation
  - Eliminates rework typically seen at integration on projects with poor requirements
- Early test case development
  - Validated requirements are complete and verifiable which results in well defined test cases
- Requirements management and traceability
  - Requirements management interfaces provide traceability for design and test cases



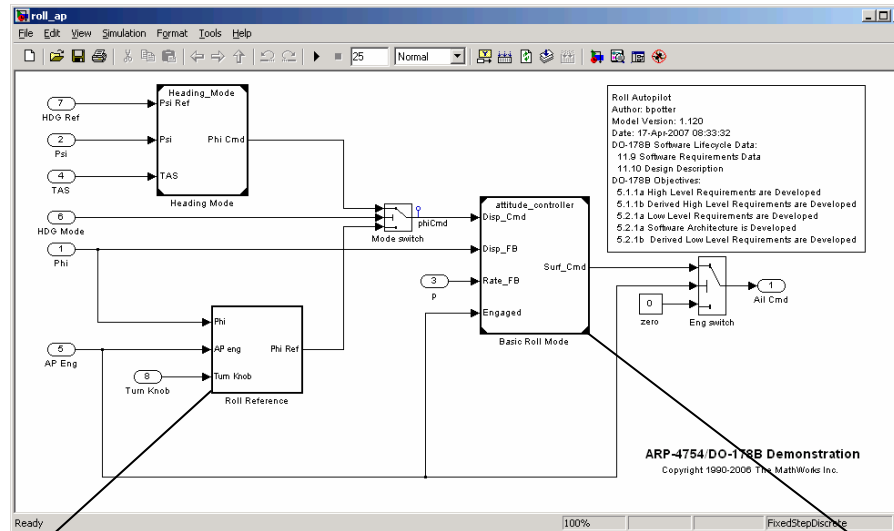
# Design Process for Model-Based Design

- Model-Based Design
  - Create the design - Simulink and Stateflow®
  - Modular design for teams - Model Reference
  - Model architecture/regression analysis - Model Dependency Viewer
  - Documented design - Simulink Report Generator
  - Requirements traceability using Simulink Verification and Validation™
  - Design conforms to standards using Model Advisor

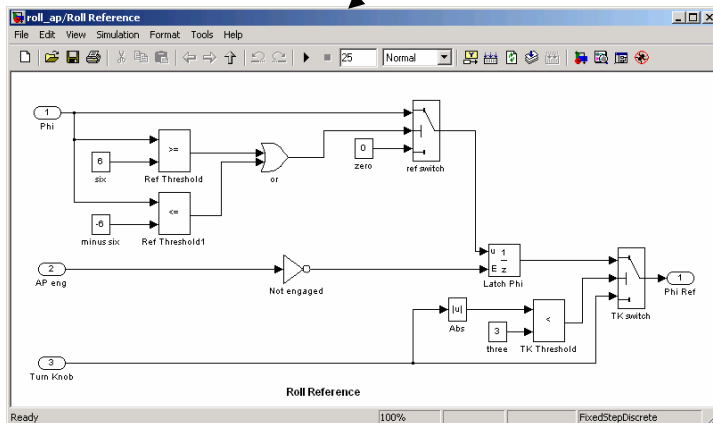


# Example detailed design including model reference and subsystems

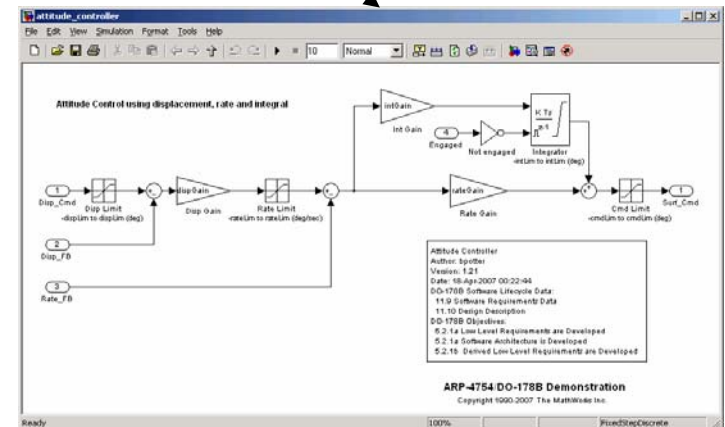
Top Model



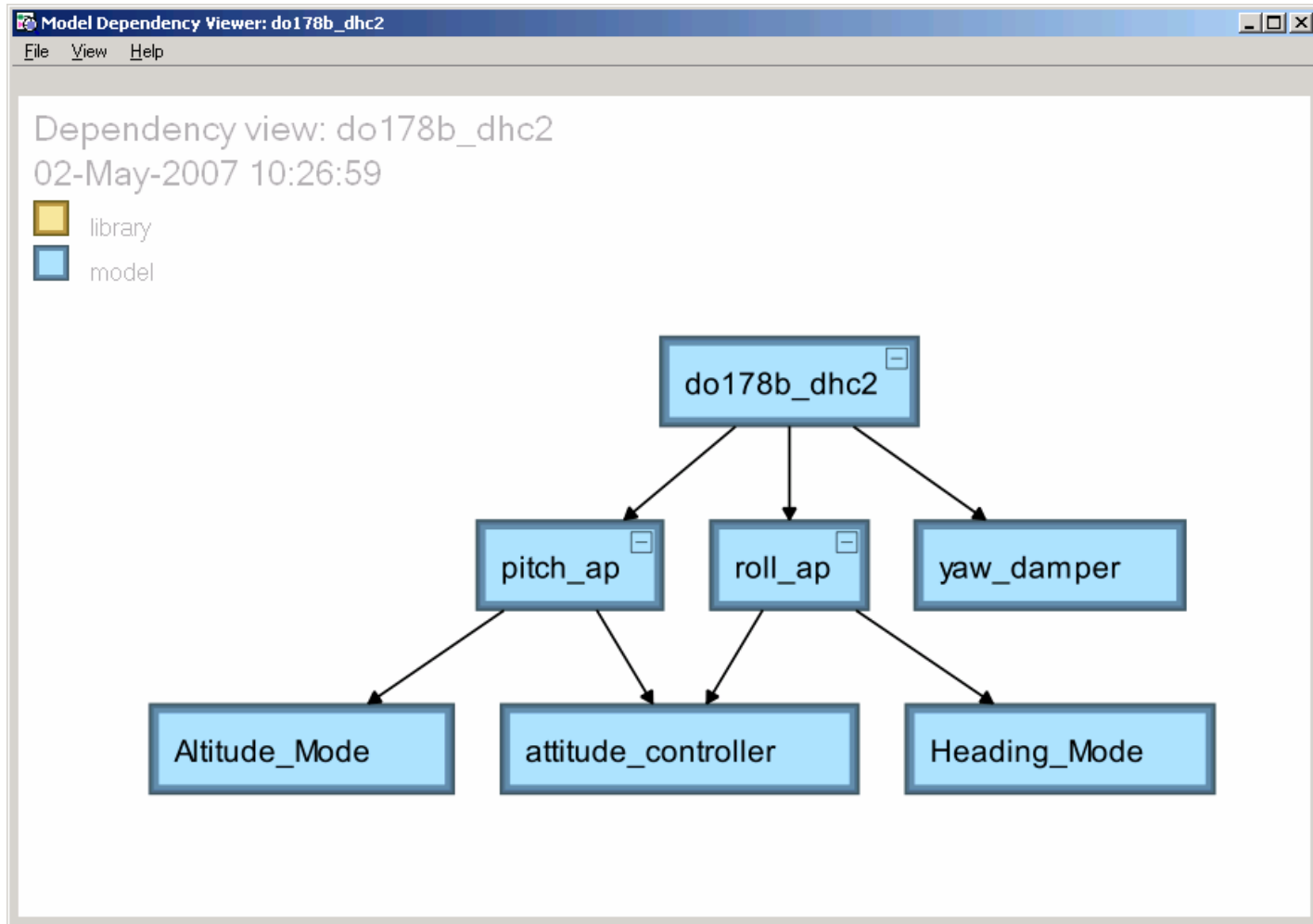
Subsystem



Reference Model



# Model dependency viewer



# Example Model Advisor report

**Model Advisor - attitude\_controller**

File Edit View Help

Select All Deselect All

Task Hierarchy: attitude\_controller

- Model Advisor Task Manager
  - By Product
    - Simulink
      - Real-Time Workshop Embedded Code
      - Simulink Verification and Validation
        - DO-178B and MISRA-C Checks
          - Display software lifecycle data
          - Display model version information
          - Check safety related optimization s
          - Check safety related solver diagno
          - Check safety related sample time c
          - Check safety related data validity -
          - Check safety related data validity -
          - Check safety related data validity -
          - Check safety related data validity -
          - Check safety related type conversi
          - Check safety related connectivity -
          - Check safety related connectivity -
          - Check safety related connectivity -
          - Check safety related compatibility
          - Check safety related model referer

**By Product**

Model Advisor

To process all enabled validations in this folder and generate a r  
To display the report automatically after validations are processe  
To display the last report generated, click the path link listed for '

Analysis

Run Selected Checks

Show report after run

Last Report

From node: By Product

Report: [C:\local\\_work\\_area\demos\autopilot\\_R2007a\\_m](#)

Date/Time: 02-May-2007 10:35:27

Summary:  42  3

---

**Model Advisor Report for 'attitude\_controller'**

Model version: 1.21

Generated on: 02-May-2007 10:35:27

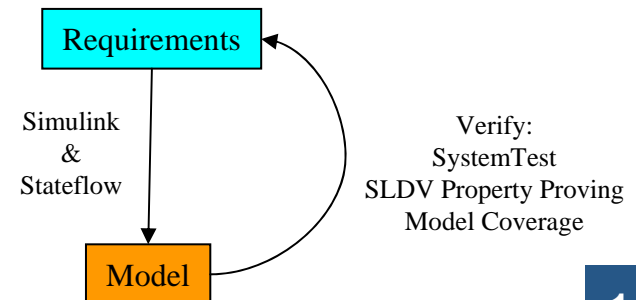
42 of 45 Passed

- Check model, local libraries, and referenced models for known upgrade issues**  
Passed
- Identify unconnected lines, input ports, and output ports**  
Passed
- Check root model Inport block specifications**  
Passed
- Check solver for code generation**  
[Sample times](#) for this model is Unconstrained. If the model does not specify any sample times, consider setting its **Periodic sample time constraint** parameter to Ensure sample time independent. Otherwise, set the parameter to Specified if the model will not be referenced.
- Identify questionable blocks within the specified system**  
Check for blocks not supported by Real-Time Workshop:

Done

# Design Verification for Model-Based Design

- Requirements based test cases
  - Automated testing using SystemTest™ and Simulink Verification and Validation
  - Traceability using Simulink Verification and Validation
- Robustness testing and analysis
  - Built in Simulink run-time diagnostics
  - Formal proofs using Simulink Design Verifier™
- Coverage Analysis
  - Verify structural coverage of model
  - Verify data coverage of model



# SystemTest for requirements based testing

The screenshot displays the SystemTest interface for a test named 'attitude\_control'. The main window is divided into several panes:

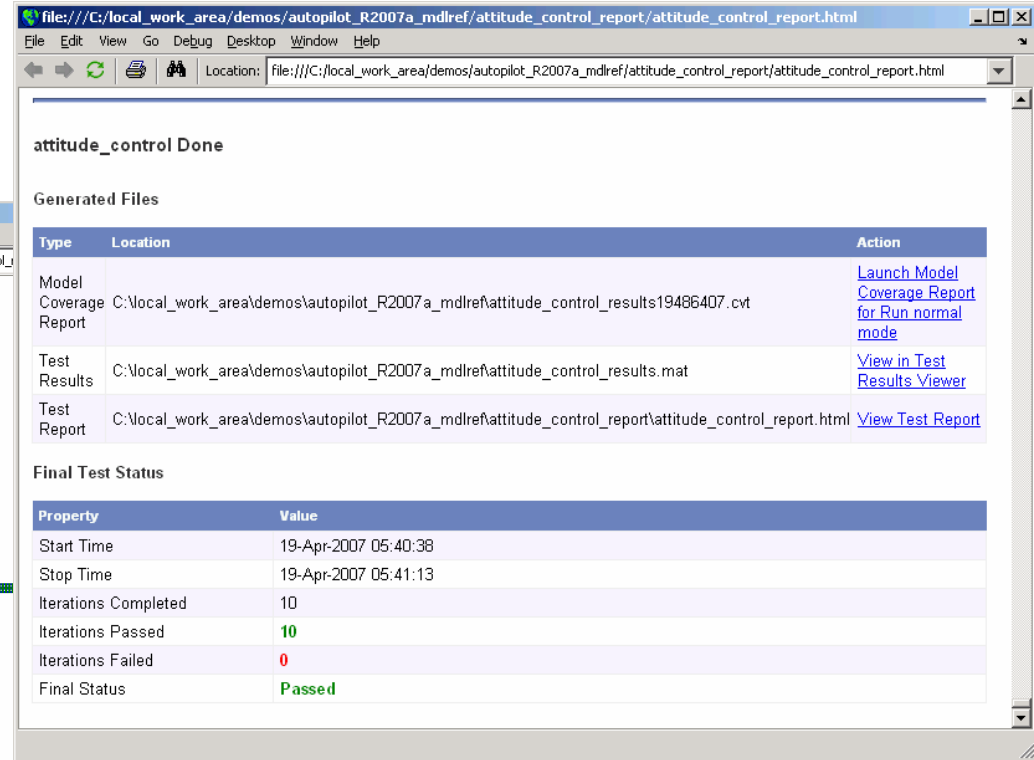
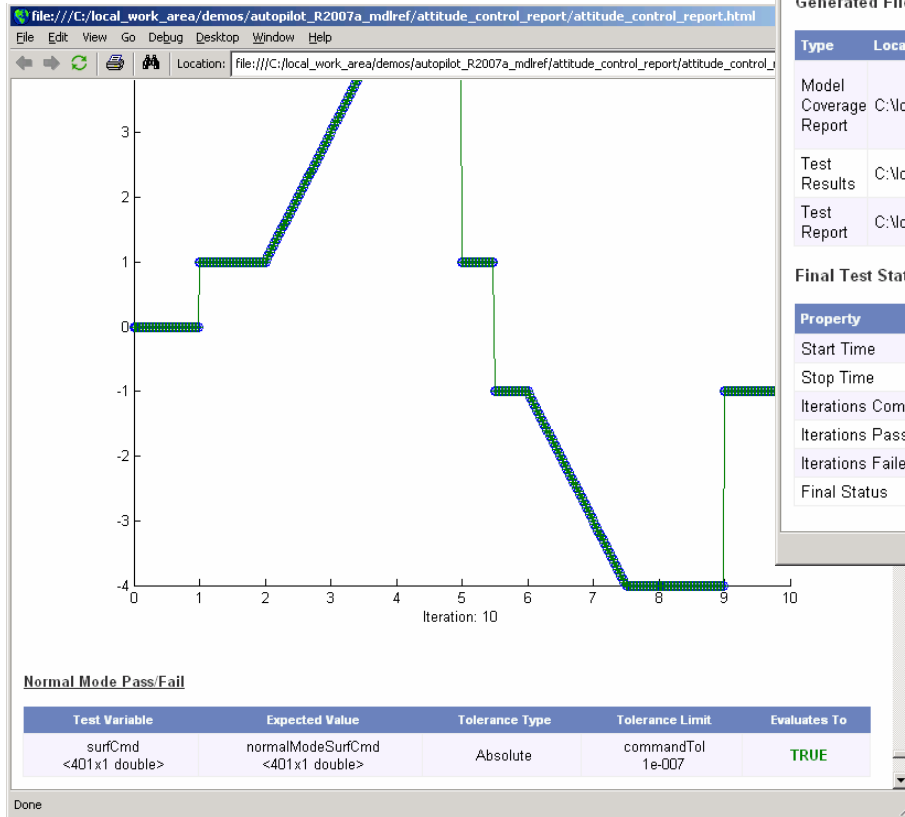
- Test Browser:** Shows a tree view of the test structure, including 'Pre Test', 'Main Test (10 Iterations)', and 'Post Test'. The 'Main Test' section is expanded, showing various test elements like 'Run Reference Model', 'Run test harness', and 'Save Results'.
- Properties - Main Test:** Displays a 'Test Structure' diagram. The diagram shows a flow from 'Pre Test' (pre test element) to 'Main Test' (main test element), which contains a loop of 'ITERATIONS' (test vectors) leading to 'save results', and finally to 'Post Test' (post test element). The flow ends with 'end of test'.
- Test Vectors:** A table with columns for Name, Expression, Vector, It..., and Gr... The table contains one entry: 'index' with expression '1:1:10', vector '[1 2 3 4 5 ...]', iterations '10', and a checkbox.
- Test Variables:** A section for defining test variables, currently empty.
- Test Status:** A section for monitoring the test, showing 'Time Elapsed:' and a progress bar.

Name	Expression	Vector	It...	Gr...
index	1:1:10	[1 2 3 4 5 ...]	10	<input type="checkbox"/>

Total number of iterations: 10

# SystemTest – example report

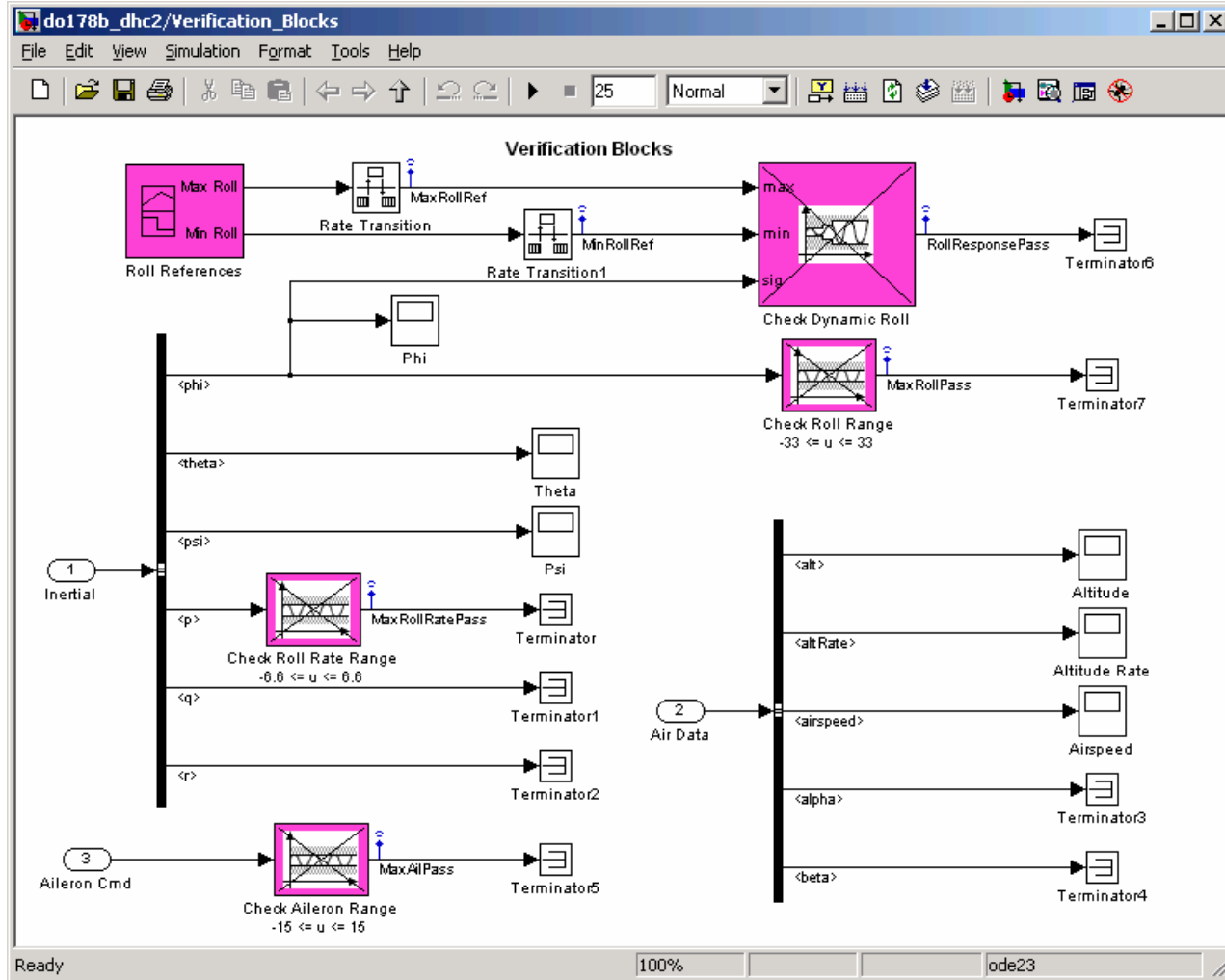
Data Plotting and expected results comparisons



Summary of results



# Signal Builder and Assertion Blocks



# Model coverage report example – signal ranges

attitude\_controller Coverage Report

File Edit View Go Debug Desktop Window Help

Location: file:///C:/local\_work\_area/demos/autopilot\_R2007a\_mdref/attitude\_control\_results19486407.html

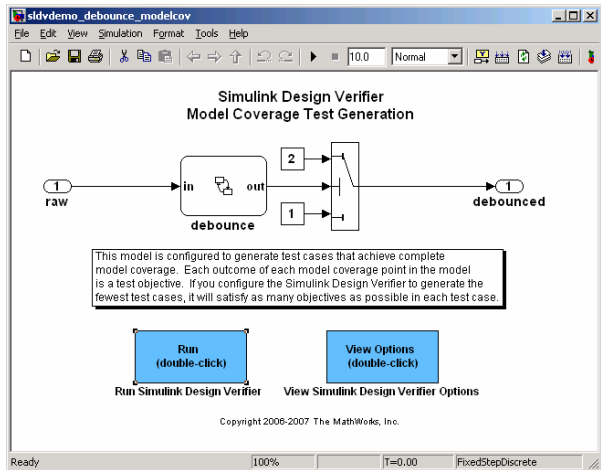
## Signal Ranges:

Hierarchy	Test 1		Test 2		Test 3		Test 4		Test 5		Test 6		Test 7		Test 8		Test 9		Test 10		Overall	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
attitude_controller																						
... <a href="#">Integrator</a>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-3	3	-3	3
... <a href="#">Not engaged</a>	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
... <a href="#">Cmd Limit</a>	-1	1	-9	9	-10	10	-1	1	-4	4	-5	5	-1	1	-7	7	-7.5	7.5	-4	4	-10	10
... <a href="#">Disp Limit</a>	-1	1	-9	9	-10	10	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	-10	10
... <a href="#">Rate Limit</a>	-1	1	-9	9	-10	10	-1	1	-4	4	-5	5	0	0	0	0	0	0	-1	1	-10	10
... <a href="#">Disp Gain</a>	-1	1	-9	9	-10	10	-1	1	-4	4	-6	6	0	0	0	0	0	0	-1	1	-10	10
... <a href="#">Int Gain</a>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-2	2	-2	2
... <a href="#">Rate Gain</a>	-1	1	-9	9	-10	10	-1	1	-4	4	-5	5	-1	1	-7	7	-8	8	-1	1	-10	10
... <a href="#">Sum</a>	-1	1	-9	9	-10	10	-1	1	-1	1	-1	1	0	0	0	0	0	0	-1	1	-10	10
... <a href="#">Sum1</a>	-1	1	-9	9	-10	10	-1	1	-4	4	-5	5	-1	1	-1	1	-1	1	-1	1	-10	10
... <a href="#">Sum2</a>	-1	1	-9	9	-10	10	-1	1	-4	4	-5	5	-1	1	-7	7	-8	8	-4	4	-10	10

Done

# Simulink Design Verifier – Coverage Test

## Model



## Test Report

Simulink Design Verifier Report

Location: file:///H:/Documents/MATLAB/sldv\_output/

### Test Case 7

**Summary**  
 Length: 0.13 Seconds (6 sample periods)  
 Objective Count: 10

**Objectives Reached At:**

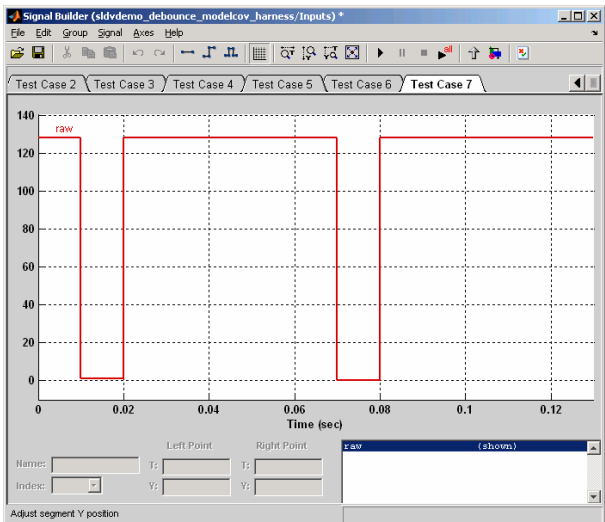
Step	Time	Objectives
1	0	1
2	0.01	7
3	0.02	5 11 16 18
9	0.08	10 12 14
13	0.12	15

**Generated Input Data.**

Time	0	0.01	0.02	0.07	0.08
Step	1	2	3	4	5
raw	128	1	128	0	128

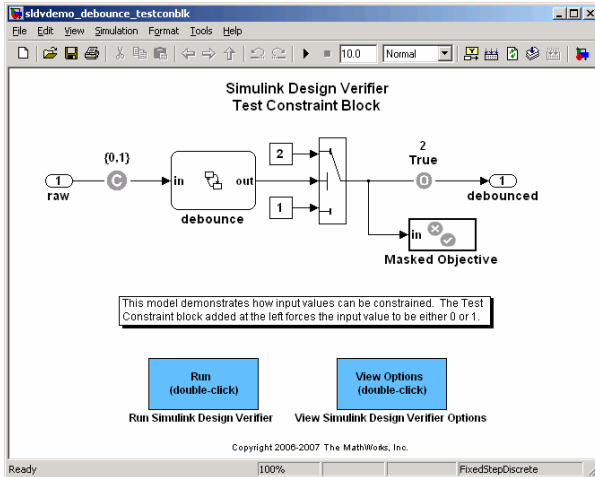


## Generated Test Cases



# Simulink Design Verifier – Objective Test

## Model with Constraints and Objectives



## Test Report

Simulink Design Verifier Report

Location: jvdemo\_debounce\_testconblk\_report.html

### Chapter 3. Test Cases / Counterexamples

Table of Contents

[Test Case 1](#)

### Test Case 1

Summary

Length: 0.13 Seconds (4 sample periods)

Objective Count: 2

Objectives Reached At:

Step	Time	Objectives
7	0.06	<a href="#">2</a>
13	0.12	<a href="#">1</a>

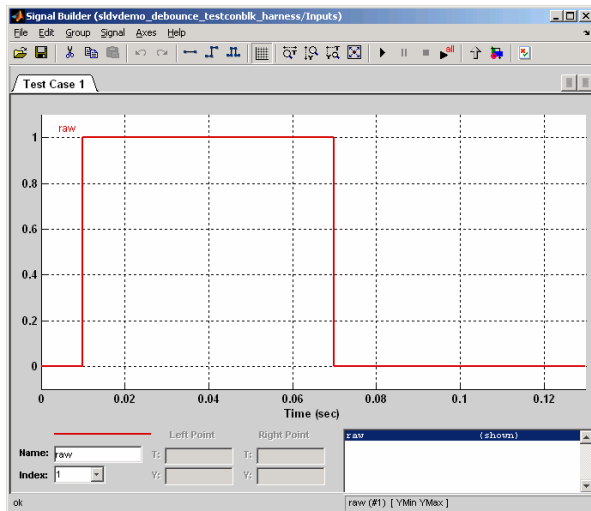
Generated Input Data.

Time	0	0.01	0.07
Step	1	2	3
raw	0	1	0

Done

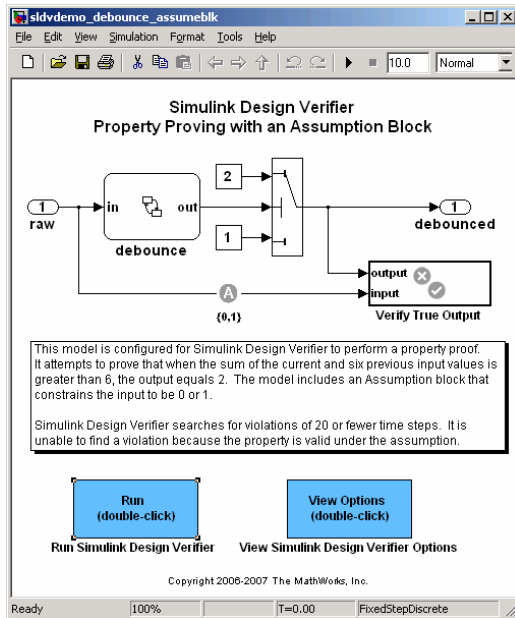


## Generated Test Cases



# Simulink Design Verifier – Property Proving

## Model with Assumption and Objective



## Report

**Simulink Design Verifier Report**

Location: /sldvdemo\_debounce\_assumeblk\_report.html

ReportFileName	\$ModelName\$_report
ReportIncludeGraphics	off
DisplayReport	on

### Chapter 2. Test/Proof Objectives

**Table of Contents**

- [Status](#)
- [Verify True Output](#)

### Status

**Table 2.1. Objectives having No Counterexamples of 20 or Fewer Steps**

#:	Type	Model Item	Description
1	Assert	<a href="#">Assertion</a>	Assertion "Assertion" assert

With the following active constraints:

Name	Constraint
<a href="#">Assumption</a>	{ 0 1 }

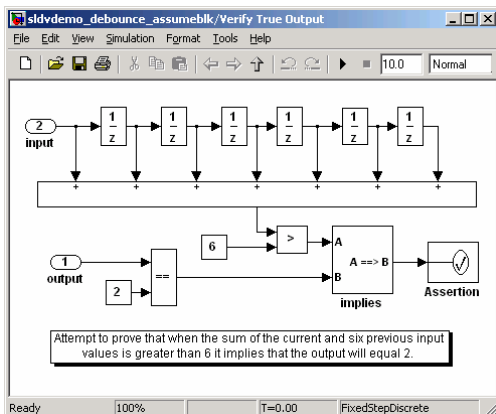
### Verify True Output

Objectives of: Assertion

#:	Status	Test Cases	Description
1	Undecidable	n/a	assert

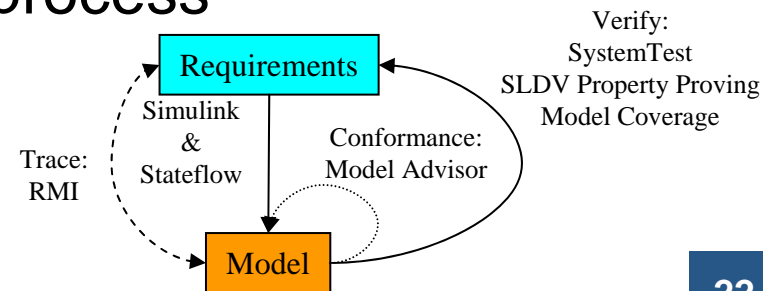
Done

## Property to be proven



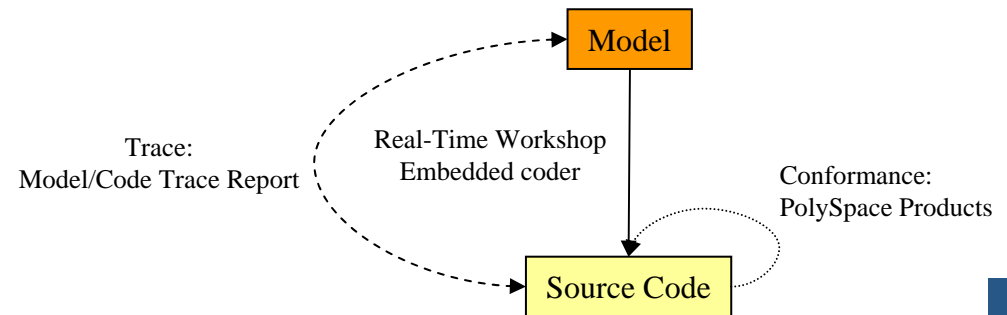
# Design Process take-aways

- Modular reusable implementations
  - Platform independent design
  - Scalable to large teams
- Consistent and compliant implementations
  - Common design language
  - Automated verification of standards compliance
- Efficient verification process
  - Develop verification procedures in parallel with design
  - Coverage analysis early in the process
  - Automated testing and analysis



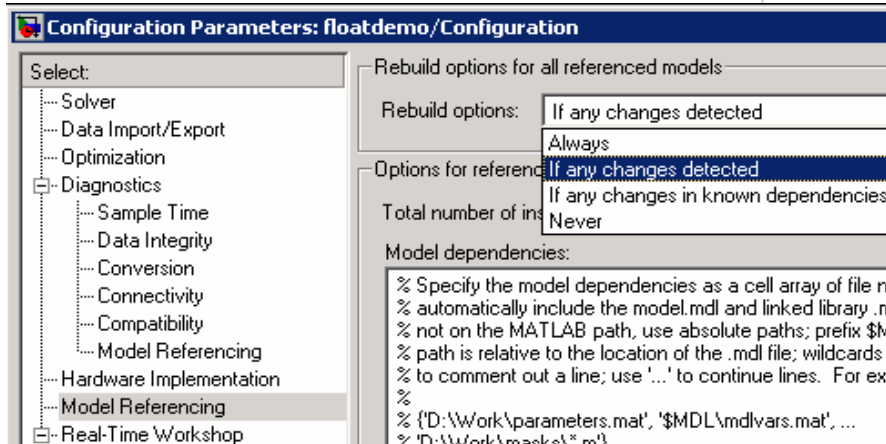
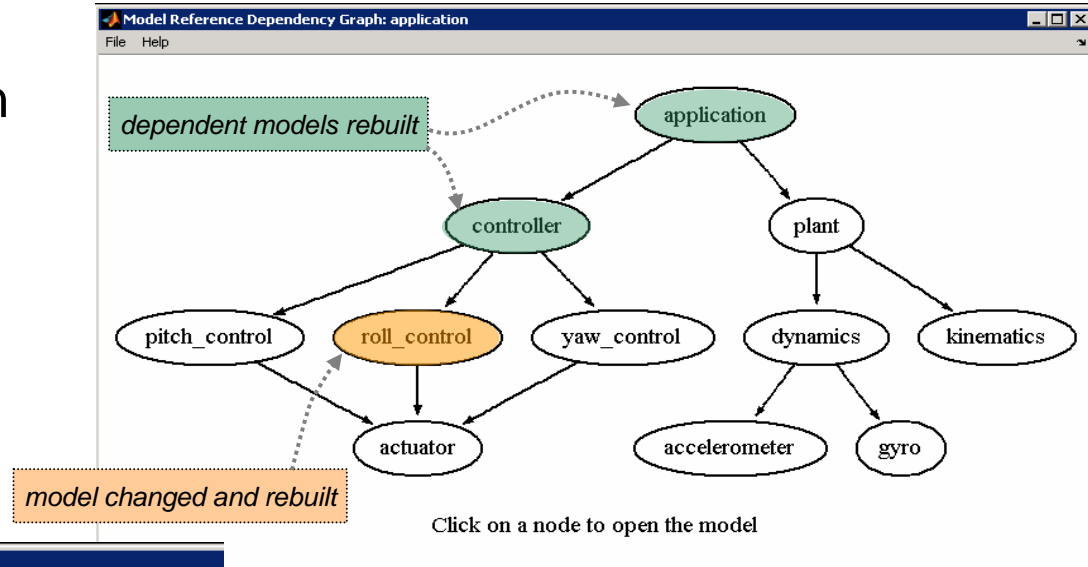
# Coding Process for Model-Based Design

- Automatic code generation
  - Real-Time Workshop Embedded Coder
- Traceability
  - HTML Code Traceability Report
- Source code verification
  - Complies with standards using PolySpace MISRA-C® checker
  - Accurate, consistent and robust using PolySpace™ verifier



# Incrementally Generate Code

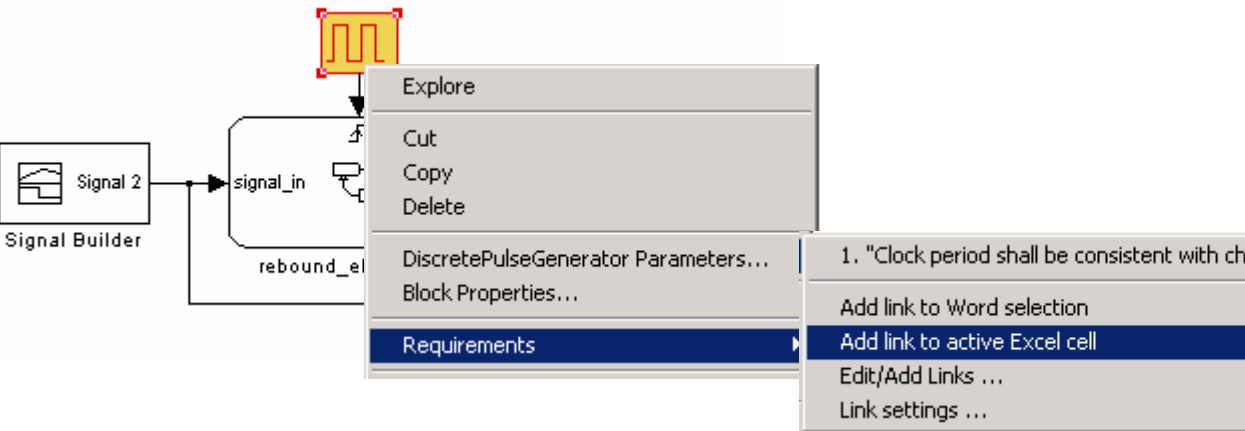
- Incremental code generation is supported via Model Reference
- When a model is changed, only models depending on it are subject to regeneration of their code



- Reduces application build times and ensure stability of a project's code
- Degree of dependency checking is configurable



# Add Links to Requirements



```

94
95  /* DiscretePulseGenerator: '<Root>/clock'
96   *
97   * Requirements for '<Root>/clock':
98   * 1. Clock period shall be consistent with chirp tolerance
99   */
100  rtb_clock =
101    (rtDWork.clockTickCounter < 1.0 &&
102     rtDWork.clockTickCounter >= 0) ?
103    1.0 :
104    0.0;
105  if (rtDWork.clockTickCounter >= 2.0-1) {

```

← Requirements appear in the code

# Code to Model Trace Report

**Real-Time Workshop Report**

Back Forward

**Contents**  
[Summary](#)  
[Traceability Report](#)  
[Subsystem Report](#)  
**Generated Source Files**  
[attitude\\_controller.c](#)  
[attitude\\_controller.h](#)  
[attitude\\_controller\\_private.h](#)  
[attitude\\_controller\\_types.h](#)

## Traceability Report for attitude\_controller

### Table of Contents

- [Eliminated / Virtual Blocks](#)
- [Traceable Simulink Blocks / Stateflow Objects / Embedded MATLAB Scripts](#)
  - [attitude\\_controller](#)
  - [attitude\\_controller/Model Info](#)

### Eliminated / Virtual Blocks

Block Name	Comment
<a href="#">&lt;Root&gt;/Disp_Cmd</a>	Inport
<a href="#">&lt;Root&gt;/Disp_FB</a>	Inport
<a href="#">&lt;Root&gt;/Rate_FB</a>	Inport
<a href="#">&lt;Root&gt;/Enqaqed</a>	Inport
<a href="#">&lt;Root&gt;/Model_Info</a>	Masked SubSystem
<a href="#">&lt;Root&gt;/Surf_Cmd</a>	Output
<a href="#">&lt;S1&gt;/EmptySubsystem</a>	Empty SubSystem

### Traceable Simulink Blocks / Stateflow Objects / Embedded MATLAB Scripts

Root system: [attitude\\_controller](#)

Object Name	Code Location
<a href="#">&lt;Root&gt;/Cmd_Limit</a>	attitude_controller.c:129, 130
<a href="#">&lt;Root&gt;/Disp_Gain</a>	attitude_controller.c:81, 85
<a href="#">&lt;Root&gt;/Disp_Limit</a>	attitude_controller.c:74, 75, 82, 89
<a href="#">&lt;Root&gt;/Int_Gain</a>	attitude_controller.c:138, 144
<a href="#">&lt;Root&gt;/Integrator</a>	attitude_controller.c:19, 20, 48, 49, 137, 140 attitude_controller.h:26, 27 attitude_controller_private.h:40, 45

OK Cancel Help Apply

# Simulink Integration with PolySpace Products

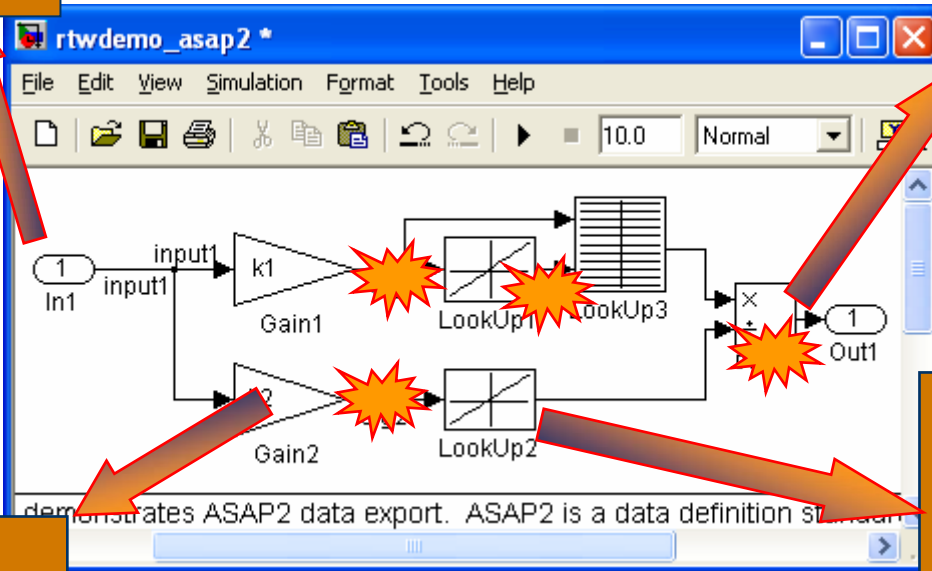
*Overflow?*

**Input1**

- Entries
- varying from -500 to 500

**Math operations**

- Divide, add, min/max, product, subtract, sum...



**K1 and K2**

- Constants
- Can be tuned from -297 to 303

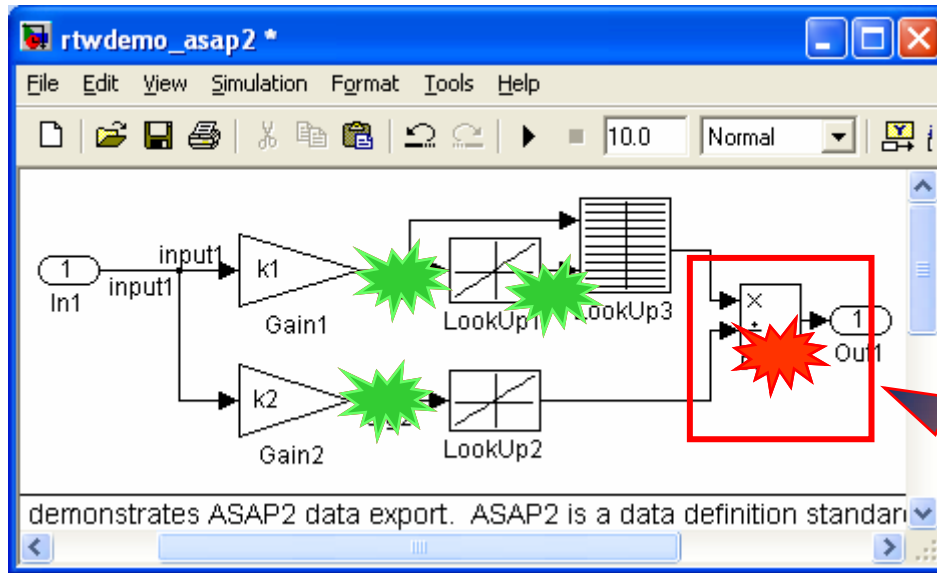
**Lookup tables**

- Maps, surfaces, algorithms, extrapolations
- Adjusted, tuned

*Division by Zero?*

## See results in the model

- Change the model
- Generate the production code
- Run PolySpace software



PolySpace Viewer - C:\PolySpace\_result

File Edit Tools Windows Help

Procedural entities

```

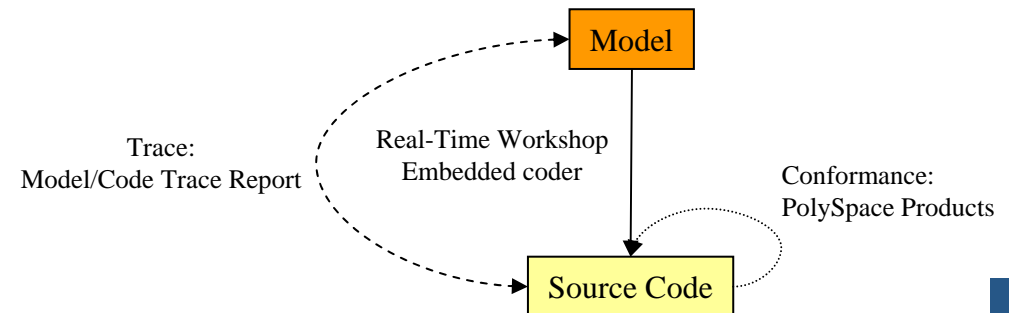
rtwdemo_hyperlinks.c
37  /* Sum: '<Root>/Sum' inc
38  * UnitDelay: '<Root>/2
39  */
40  rtb_Switch = (uint8_T) (
41
42  /* RelationalOperator:
43  rtb_RelOpt = (rtb_Switch
44
45  /* Outport: '<Root>/Out
46  rtY.Out = rtb_RelOpt;
47
48  /* Switch: '<Root>/Switc
49  if (rtb_RelOpt) {

```

*PolySpace detected an error here  
(after having analyzed the generated code)*

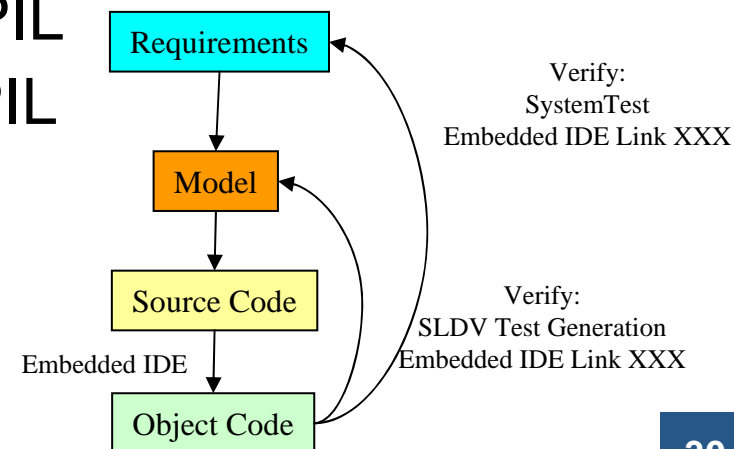
# Coding Process takeaways

- Reusable and platform independent source code
- Traceability
- MISRA-C compliance
- Static verification and analysis



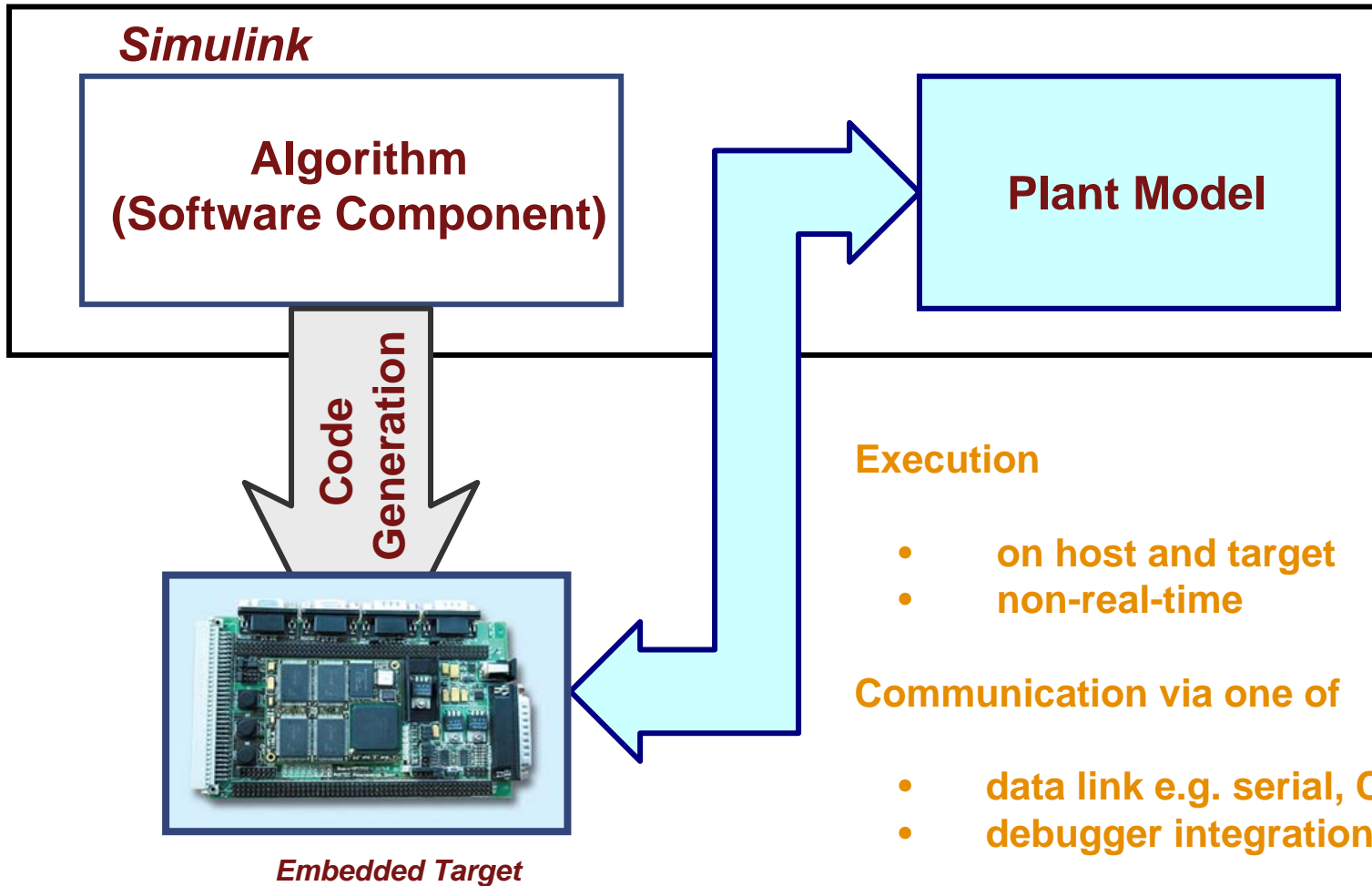
# Integration Process for Model-Based Design

- Executable object code generation
  - ANSI® or ISO® C or C++ compatible compiler
  - Run-time libraries provided
- Executable object code verification
  - Test generation using Simulink Design Verifier
  - Capability to build interface for Processor-In-the-Loop (PIL) testing
  - Analyze code coverage during PIL
  - Analyze execution time during PIL
  - Analyze stack PIL



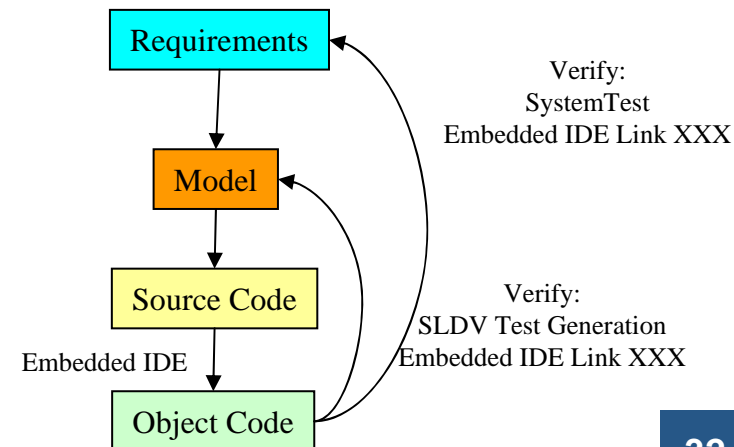
# Processor-in-the-Loop (PIL) Verification

- Execute Generated Code on Target Hardware



# Integration Process Takeaways

- Integration with multiple development environments
- Test cases and harnesses generated automatically
- Efficient processor in-the-loop test capability





## Wrap-up

- Tools to support the entire safety critical development process
- Participation on SC-205/WG-71 committee for DO-178C
- Safety-Critical/DO-178B guideline document
  - Available to licensed customers with Real-Time Workshop Embedded Coder
  - Contact Bill Potter ([bill.potter@mathworks.com](mailto:bill.potter@mathworks.com)) or Tom Erkkinen ([tom.erkkinen@mathworks.com](mailto:tom.erkkinen@mathworks.com))