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

Bill Potter
Technical Marketing
April 17, 2008
Safety-Critical Model-Based Design Workflow

Requirements

- Validate
- Simulink® & Stateflow®
- Trace: RMI
- Conformance: Model Advisor
- Verify: SystemTest™
- Embedded IDE Link™ XXX

Model

- Real-Time Workshop® Embedded Coder™
- Trace: Model/Code Trace Report
- Conformance: PolySpace™ Products
- Model Coverage
- Verify: SLDV Property Proving

Source Code

- Embedded IDE
- Model/Code Trace Report
- SLDV Test Generation
- Embedded IDE Link™ XXX

Object Code
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
### 2.4 Integral Control and Limit

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.

<table>
<thead>
<tr>
<th>ID</th>
<th>Software requirements for a reusable attitude controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>[Simulink reference: attitude_controller/Rate Limit (Saturate)]</td>
</tr>
<tr>
<td>20</td>
<td>2.4 Integral Control and Limit</td>
</tr>
<tr>
<td>63</td>
<td>[Simulink reference: attitude_controller_harness/Signal Builder (Subsystem)]</td>
</tr>
<tr>
<td>39</td>
<td>[Simulink reference: attitude_controller/Int Gain (Gain)]</td>
</tr>
<tr>
<td>38</td>
<td>[Simulink reference: attitude_controller/Not engaged (Logic)]</td>
</tr>
<tr>
<td>37</td>
<td>[Simulink reference: attitude_controller/Integrator (DiscreteIntegrator)]</td>
</tr>
<tr>
<td>21</td>
<td>2.5 Surface Command and Limit</td>
</tr>
</tbody>
</table>
Requirements trace example – view from Simulink to DOORS

Chapter 2. System - attitude_controller

Table 2.1. Block Requirements Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cmd Limit</td>
<td>2.1.2 Parameters 00000122 #23</td>
</tr>
<tr>
<td></td>
<td>2.5 Surface Command and Limit 0000022 #21</td>
</tr>
<tr>
<td>Disp Gain</td>
<td>2.1.2 Parameters 00000122 #23</td>
</tr>
<tr>
<td></td>
<td>2.2 Attitude Control and Limit 0000022 #18</td>
</tr>
<tr>
<td>Disp Limit</td>
<td>2.1.2 Parameters 00000122 #23</td>
</tr>
<tr>
<td></td>
<td>2.2 Attitude Control and Limit 0000022 #18</td>
</tr>
<tr>
<td>Disp_Cmd</td>
<td>2.1.1 Inputs 00000122 #22</td>
</tr>
</tbody>
</table>

ARP-4754/DD-178B Demonstration
Copyright 1990-2007 The MathWorks Inc.
Requirements based test trace example – view from Simulink Signal Builder block to DOORS
Model coverage report example

Discrete integrator block "Integrator"

Parent: [attitude_controller]

Metric Coverage
Cyclomatic Complexity 3
Decision (C1) 100% (8/8) decision outcomes

Decisions analyzed:

<table>
<thead>
<tr>
<th>Decision</th>
<th>true</th>
<th>false</th>
<th>reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>integration result &lt;= lower limit</td>
<td>50% 50% 50% 50% 50% 50% 50% 50% 50% 100% 100% 100% 100% 50% 50% 50% 50% 100% 100%</td>
<td>401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 2334232 38923861</td>
<td>50% 50% 50% 50% 50% 50% 50% 50% 50% 100% 100% 100% 100% 50% 50% 50% 50% 100% 100%</td>
</tr>
<tr>
<td>integration result &gt;= upper limit</td>
<td>50% 50% 50% 50% 50% 50% 50% 50% 50% 100% 100% 100% 100% 50% 50% 50% 50% 100% 100%</td>
<td>401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 401401 2334232 38923861</td>
<td>50% 50% 50% 50% 50% 50% 50% 50% 50% 100% 100% 100% 100% 50% 50% 50% 50% 100% 100%</td>
</tr>
</tbody>
</table>

Logic block "Not engaged"

Parent: [attitude_controller]

Metric Coverage
Cyclomatic Complexity 0
Condition (C1) 100% (2/2) condition outcomes
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
Model dependency viewer

Dependency view: do178b_dhc2
02-May-2007 10:26:59

Library: library
Model: model

- do178b_dhc2
- pitch_ap
- roll_ap
- yaw_damper
- Altitude_Mode
- attitude_controller
- Heading Mode
Example Model Advisor report

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 Import 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
SystemTest – example report

Data Plotting and expected results comparisons

Summary of results
Signal Builder and Assertion Blocks
Model coverage report example – signal ranges

Signal Ranges:

<table>
<thead>
<tr>
<th>Hierarchy</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Test 5</th>
<th>Test 6</th>
<th>Test 7</th>
<th>Test 8</th>
<th>Test 9</th>
<th>Test 10</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>attitude_controller</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-3</td>
<td>3</td>
</tr>
<tr>
<td>Integrator</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-3</td>
<td>3</td>
</tr>
<tr>
<td>Not engaged</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Cmd Limit</td>
<td>-1</td>
<td>1</td>
<td>-9</td>
<td>9</td>
<td>-10</td>
<td>10</td>
<td>-1</td>
<td>1</td>
<td>-4</td>
<td>4</td>
<td>-5</td>
</tr>
<tr>
<td>Disp Limit</td>
<td>-1</td>
<td>1</td>
<td>-9</td>
<td>9</td>
<td>-10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rate Limit</td>
<td>-1</td>
<td>1</td>
<td>-9</td>
<td>9</td>
<td>-10</td>
<td>10</td>
<td>-1</td>
<td>1</td>
<td>-4</td>
<td>4</td>
<td>-5</td>
</tr>
<tr>
<td>Disp Gain</td>
<td>-1</td>
<td>1</td>
<td>-9</td>
<td>9</td>
<td>-10</td>
<td>10</td>
<td>-1</td>
<td>1</td>
<td>-4</td>
<td>4</td>
<td>-6</td>
</tr>
<tr>
<td>Int Gain</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>Rate Gain</td>
<td>-1</td>
<td>1</td>
<td>-9</td>
<td>9</td>
<td>-10</td>
<td>10</td>
<td>-1</td>
<td>1</td>
<td>-4</td>
<td>4</td>
<td>-5</td>
</tr>
<tr>
<td>Sum</td>
<td>-1</td>
<td>1</td>
<td>-9</td>
<td>9</td>
<td>-10</td>
<td>10</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>Sum1</td>
<td>-1</td>
<td>1</td>
<td>-9</td>
<td>9</td>
<td>-10</td>
<td>10</td>
<td>-1</td>
<td>1</td>
<td>-4</td>
<td>4</td>
<td>-5</td>
</tr>
<tr>
<td>Sum2</td>
<td>-1</td>
<td>1</td>
<td>-9</td>
<td>9</td>
<td>-10</td>
<td>10</td>
<td>-1</td>
<td>1</td>
<td>-4</td>
<td>4</td>
<td>-5</td>
</tr>
</tbody>
</table>

Done
Simulink Design Verifier – Coverage Test

Model

Test Report

Test Case 7

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

Objectives Reached At:

<table>
<thead>
<tr>
<th>Step</th>
<th>Time</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.01</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>0.02</td>
<td>5 11 16 18</td>
</tr>
<tr>
<td>9</td>
<td>0.08</td>
<td>10 12 14 16 18</td>
</tr>
<tr>
<td>13</td>
<td>0.12</td>
<td>15</td>
</tr>
</tbody>
</table>

Generated Input Data:

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>0.01</th>
<th>0.02</th>
<th>0.07</th>
<th>0.08</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>raw</td>
<td>128</td>
<td>1</td>
<td>128</td>
<td>0</td>
<td>128</td>
</tr>
</tbody>
</table>
Simulink Design Verifier – Objective Test

Model with Constraints and Objectives

Test Report

Chapter 3. Test Cases / Counterexamples

Table of Contents

Test Case 1

Summary
Length: 0.13 Seconds (4 sample periods)
Objective Count: 2

Objectives Reached At:

<table>
<thead>
<tr>
<th>Step</th>
<th>Time</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0.06</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>0.12</td>
<td>1</td>
</tr>
</tbody>
</table>

Generated Input Data:

<table>
<thead>
<tr>
<th>Time</th>
<th>0.01</th>
<th>0.07</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>raw</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Done
Simulink Design Verifier – Property Proving

Model with Assumption and Objective

Report

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

Verify True Output

Objectives of: Assertion

# | Status | Test Cases | Description
--- | --- | --- | ---
1 | Undecidable | None | assert
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

Requirements

Simulink & Stateflow

Model

Conformance: Model Advisor

Trace: RMI

Verify: SystemTest SLDV Property Proving Model Coverage
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 ensures stability of a project’s code.

Degree of dependency checking is configurable.
Add Links to Requirements

Requirements appear in the code

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

Traceability Report for attitude_controller

Table of Contents
1. Eliminated / Virtual Blocks
2. Traceable Simulink Blocks / Stateflow Objects / Embedded MATLAB Scripts
   - attitude_controller
   - attitude_controller/private.h
   - attitude_controller-types.h

Eliminated / Virtual Blocks

<table>
<thead>
<tr>
<th>Block Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>这类/Disp_Cmd</td>
<td>Import</td>
</tr>
<tr>
<td>这类/Disp_R</td>
<td>Import</td>
</tr>
<tr>
<td>这类/Disp_1_8</td>
<td>Import</td>
</tr>
<tr>
<td>这类/Disp_1_15</td>
<td>Import</td>
</tr>
<tr>
<td>这类/Disp_ED</td>
<td>Masked SubSystem</td>
</tr>
<tr>
<td>这类/Disp_Cmd</td>
<td>Outport</td>
</tr>
<tr>
<td>cles/EmptySubsystem</td>
<td>Empty SubSystem</td>
</tr>
</tbody>
</table>

Traceable Simulink Blocks / Stateflow Objects / Embedded MATLAB Scripts

Root system: attitude_controller

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Code Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>这类/Disp_Cmd</td>
<td>attitude_controller.c:129, 130</td>
</tr>
<tr>
<td>这类/Disp_R</td>
<td>attitude_controller.c:81, 85</td>
</tr>
<tr>
<td>这类/Disp_1_8</td>
<td>attitude_controller.c:75, 82, 89</td>
</tr>
<tr>
<td>这类/Disp_1_15</td>
<td>attitude_controller.c:133, 144</td>
</tr>
<tr>
<td>这类/Disp_ED</td>
<td>attitude_controller.c:139, 140</td>
</tr>
<tr>
<td>cles/Disp_Cmd</td>
<td>attitude_controller.c:40, 42, 45, 49, 127, 140</td>
</tr>
<tr>
<td>cles/EmptySubsystem</td>
<td>attitude_controller.c:40, 42, 45</td>
</tr>
</tbody>
</table>
Simulink Integration with PolySpace Products

**Input1**
- Entries
- Varying from -500 to 500

**K1 and K2**
- Constants
- Can be tuned from -297 to 303

**Overflow?**

**Math operations**
- Divide, add, min/max, product, substract, sum...

**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 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

Execution
- on host and target
- non-real-time

Communication via one of
- data link e.g. serial, CAN, TCP/IP
- debugger integration with MATLAB
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) or Tom Erkkinen (tom.erkkinen@mathworks.com)