Verify, Validate and Document Models and Code

Dr. Michael Kitz
Senior Applications Engineer, Physical Modelling and Control Design
Applications Engineering Group
MathWorks Switzerland
June 09, 2015
The Challenge

• Product Innovation

• ‘Going Smart’, ‘Internet of Things’

→ Software everywhere

How will you know your system works?
Three Key Takeaways

- Find problems and bugs early in the design and code
- Use mathematical analysis methods to prove software correctness
- Reproduce field issues via property proving
Issues during testing

1) Wasting a couple of days on the test bench to find a code integration issue without success

2) While going downhill, target speed increase with “reduce speed” button
Verification and Validation in Context

Confidence

Effort / Time

Ad-hoc tests
Design error checks
Functional tests/ Model coverage
Equivalence tests

REPORT & VERIFICATION
INTEGRATION
IMPLEMENTATION
ANALYSIS
– SPECIFICATION - DESIGN

MODEL

Architecture
Algorithms
Schematics
Environment
Constraints
Physical Domains

IMPLEMENTATION

C, C++
VHDL, Verilog
Structured Text

MCU
DSP
FPGA
ASIC
PLC
PAC

TEST CASES
Ad-hoc testing
Customizable Reports
Verification and Validation in Context

Confidence

Effort / Time

- Design error checks
- Functional tests/Model coverage
- Equivalence tests

Ad-hoc tests

- Verification and validation in context
- Design error checks
- Functional tests/Model coverage
- Equivalence tests

- Confidence vs. Effort/Time

- RESEARCH
- REQUIREMENTS

ANALYSIS – SPECIFICATION - DESIGN

- MODEL
  - Architecture
  - Environment
  - Algorithms
  - Constraints
  - Schematics
  - Physical Domains

IMPLEMENTATION

- C, C++
- VHDL, Verilog
- Structured Text
- MCU
- DSP
- FPGA
- ASIC
- PLC
- PAC

INTEGRATION

TEST & VERIFICATION

TEST CASES

MATLAB CONFERENCE 2015
Finding Unintended Behavior

Converting floating-point model to integer calibrations, signals…

- Dead logic due to “uint8” operation
Finding Unintended Behavior

- Dead logic due to “uint8” operation on incdec/holdrate*10

- Fix change the order of operation 10*incdec/holdrate

Condition can never be false
Finding Unintended Behavior

- Dead logic due to "uint8" operation on `incdec/holdrate*10`
  - Fix: change the order of operation to `10*incdec/holdrate`

Condition can never be false
Verification and Validation in Context

Confidence

Effort / Time

Ad-hoc tests
Design error checks
Functional tests/Model coverage
Equivalence tests

RESEARCH

ANALYSIS – SPECIFICATION – DESIGN

MODEL

Architecture
Algorithms
Schematics
Environment
Constraints
Physical Domains

IMPLEMENTATION

TEST CASES

C, C++, VHDL, Verilog, Structured Text

MCU, DSP, FPGA,ASIC, PLC, PAC

INTEGRATION

TEST & VERIFICATION

TEST CASES
Simulation Testing Workflow

Did we completely test our model?

Design

Did we meet requirements?

Structural coverage report

Review functional behavior

Functional

Structural
Verification and Validation in Context

- Ad-hoc tests
- Design error checks

Effort / Time

Confidence

Equivalence tests

Functional tests/ Model coverage

- RESEARCH
- REQUIREMENTS

ANALYSIS – SPECIFICATION - DESIGN

MODEL
- Architecture
- Algorithms
- Schematics

Environment
Constraints
Physical Domains

IMPLEMENTATION

- C, C++
- VHDL, Verilog
- Structured Text

MCU
DSP
FPGA
ASIC
PLC
PAC

TEST CASES

TEST & VERIFICATION
Equivalence Testing

Model

Speed Cruise Control System Using Simulink

Generated Code

Speed Cruise Control System Using Simulink

Increment Controller

Increment
Decrement
Set
Resume

Increment
Decrement
Set
Resume

Controller (SIL)

Target Speed
Equivalence Testing

Model

Speed Cruise Control System: Using Simulink Design Verifier

Configuration Parameters: plcdemo_cruise_control/Configuration (Active)

Code Generation Report for 'Controller'

Summary

Code generation for model "Controller"

Model version: 1.189
SimulinkCoder version: 8.8 (R2015a) 09-Feb-2015
C source code generated on: Thu May 28 12:18:09 2015

Configuration settings at the time of code generation: click to open
Code generation objectives: Unspecified
Validation result: Not run

MATLAB CONFERENCE 2015
Verification and Validation in Context

Confidence

Effort / Time

Ad-hoc tests

Design error checks

Functional tests/ Model coverage

Equivalence tests

Verification and Validation in Context

RESEARCH

REQUIREMENTS

ANALYSIS – SPECIFICATION- DESIGN

MODEL

- Architecture
- Algorithms
- Schematics

Environment
Constraints
Physical Domains

IMPLEMENTATION

C, C++
VHDL, Verilog
Structured Text

MCU
DSP
FPGA
ASIC
PLC
PAC

TEST & VERIFICATION

TEST CASES

TEST CASES

INTEGRATION
Verification and Validation in Context

Confidence

Effort / Time

Ad-hoc tests

Design error checks

Functional tests/ Model coverage

Equivalence tests

RESEARCH

REQUIREMENTS

ANALYSIS – SPECIFICATION- DESIGN

MODEL

Architecture

Environment

Algorithms

Constraints

Schematics

Physical Domains

IMPLEMENTATION

C, C++

VHDL, Verilog

Structured Text

MCU

DSP

FPGA

ASIC

PLC

PAC

TEST CASES

TEST VERIFICATION

TEST CASES

TEST & VERIFICATION

INTEGRATION

ANALYSIS

SPECIFICATION

DESIGN

ARCHITECTURE

ENVIRONMENT

ALGORITHMS

CONSTRAINTS

SCHEMATICS

PHYSICAL DOMAINS

C, C++

VHDL, Verilog

Structured Text

MCU

DSP

FPGA

ASIC

PLC

PAC
Application Example: Cruise Control

System Inputs → Engine Control Unit → Cruise Control Module (MBD) → Outputs

- Fuel Rate Control Module
- Shift Logic Control Module

Legacy code
## Checking Source Code

<table>
<thead>
<tr>
<th>What Polyspace does</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checks coding rule conformance (MISRA-C/C++, JSF++, Custom)</td>
</tr>
<tr>
<td>Provides metrics (Cyclomatic complexity etc)</td>
</tr>
<tr>
<td>Quickly finds potential errors</td>
</tr>
<tr>
<td>Proves the existence of errors</td>
</tr>
<tr>
<td>Proves the absence of errors</td>
</tr>
<tr>
<td>Indicates when you’ve reached the desired quality level</td>
</tr>
</tbody>
</table>

- No test cases
- No compilation
Identify Run-Time Error in Integrated Code

![Screenshot of MATLAB interface showing code snippet with error message]

**Error Message:**
"ID 1: Illegally dereferenced pointer
Error: pointer is outside its bounds
This check may be a path-related issue, which is not dependent on input values
 Dereference of parameter 'rty_Throttle' (pointer to float 64, size: 64 bits):
 Pointer is null.
 Points to 8 bytes at offset 512 in buffer of 8 bytes, so is outside bounds.
 Pointer may point to variable or field of variable: 'gvar_B_ECU_system'.

```c
rty_Throttle = 0.02 * localsB->Sum1 + 0.01
```

Probable cause for 'Illegally dereferenced pointer': Assignment to local pointer 'throttle'
Identify Run-Time Error in Integrated Code

<table>
<thead>
<tr>
<th>Family</th>
<th>Information</th>
<th>File</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run-time Check</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Red Check</td>
<td></td>
<td>1</td>
<td>18 45 1640</td>
</tr>
<tr>
<td>Illegally dereferenced pointer</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Gray Check</td>
<td></td>
<td></td>
<td>Cruise_ctrl.c</td>
</tr>
<tr>
<td>Orange Check</td>
<td></td>
<td>1</td>
<td>Cruise_ctrl()</td>
</tr>
<tr>
<td>Green Check</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Global Variable</td>
<td></td>
<td>1</td>
<td>28 1640</td>
</tr>
<tr>
<td>Not shared</td>
<td></td>
<td>1</td>
<td>28 1640</td>
</tr>
<tr>
<td>MISRA C:2012</td>
<td></td>
<td>1</td>
<td>38</td>
</tr>
<tr>
<td>5 Identifiers</td>
<td></td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>8 Declarations and definitions</td>
<td></td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>11 Pointer type conversions</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>18 Pointers and arrays</td>
<td></td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>Code Metric</td>
<td></td>
<td></td>
<td>302</td>
</tr>
<tr>
<td>Project Metrics</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>File Metrics</td>
<td></td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>Function Metrics</td>
<td></td>
<td></td>
<td>273</td>
</tr>
</tbody>
</table>
Identify Run-Time Error in Integrated Code

```
if (*rtty_Engaged) {
    /* Sum: '<S5>/Sum' incorporates:
       * DiscreteIntegrator: '<S5>/Discrete-Time Integrator'
       * Gain: '<S5>/Kp'
       * Gain: '<S5>/Kp1'
    */
    *rtty_Throttle = 0.02 * locales->Sum1 + 0.01 *
        localDW->DiscreteTimeIntegrator_DSTATE;
}
/* End of Outputs for SubSystem: '<S1>' */

/* Update for atomic system: '<Root>/Cruise ctrl'
void Cruise_ctrl_Update(boolean_T *rtty
    rtDW_Cruise_ctrl *localDW)
{
    /* Update for Enabled SubSystem: '<S1>
       * Update for EnablePort: '<S5>/Enab
```

```
! ID 1: Illegally dereferenced pointer
Error: pointer is outside its bounds
Dereference of parameter 'rtty_Throttle' (pointer to float 64, size: 64 bits):
  Pointer is not null.
  Points to 8 bytes at offset 512 in buffer of 64 bytes, so is outside bounds.
  Pointer may point to variable or field of variable:
    'gvar_B_ECU_system'.

268  *rtty_Throttle = 0.02 * localB->Sum1 + 0.01 *
```
Verification and Validation in Context

Confidence

Effort / Time

Is there more?

RESEARCH

REQUIREMENTS

ANALYSIS – SPECIFICATION - DESIGN

MODEL

Architecture

Environment

Algorithms

Constraints

Schematics

Physical Domains

IMPLEMENTATION

C, C++

VHDL, Verilog

Structured Text

MCU

DSP

FPGA

ASIC

PLC

PAC

INTEGRATION

TEST CASES

TEST & VERIFICATION

TEST CASES
Using Model-Based Design to Reproduce Field Issue

- **Problem:** While going downhill, target speed increases with “reduce speed” button and assumes random values
  - Functional tests pass for model
  - No redundancies in model (100% coverage achieved)
  - Nominal signal and parameter values worked in simulation
Construct a model of field issue:
- Constrain inputs to represent field issue
- Create model of field issue behavior
- Ask tool to produce a test case
Generated Test Case to Reproduce Field Issue

- Automatic generation of harness model
Debugging

- Going downhill, actual speed could increase (say to 25mph)

- If coast button (reduce speed) is set again, target speed takes on actual speed value of 25mph (jumps up from 20mph)
Verification and Validation Best Practice

Confidence

Ad-hoc tests
Design error checks
Functional tests/Model coverage
Requirement proofs
Equivalence tests
Integration tests
Field tests

Effort / Time

Require-ment proofs

Confidence
Three Key Takeaways

- Find problems and bugs early in the design and code
- Use formal methods to prove software correctness
- Reproduce field issues via property proving