Increasing Design Confidence with Model and Code Verification

The Cost of Failure…

Ariane 5

$7,500,000,000

Rocket & payload lost
The Cost of Failure…

USS Yorktown

0 Knots
Top speed

The Cost of Failure…

Therac-25

6 Casualties
due to radiation overdose
Motivation

It is easier and less expensive to fix design errors early in the process when they happen.

Model-Based Design enables:

1. Early testing to increase confidence in your design
2. Delivery of higher quality software throughout the workflow

Gaining Confidence in our Design
Application: Cruise Control
Control speed according to setpoint

![Speedometer with 50 km/h highlighted]

MATLAB EXPO 2016
Application: Cruise Control

**Inputs**
- Cruise on/off
- Brake
- Speed
- Coast set
- Accel reset

**Outputs**
- Engaged
- Target speed
Gaining Confidence in our Design

Ad-hoc Tests

New “Dashboard” blocks facilitate early ad-hoc testing
Gaining Confidence in our Design

Ad-hoc testing

Confidence

Effort / Time

MATLAB EXPO 2016

Finding Design Errors: Dead Logic

Compute target speed CRUISE
.on.*[after(incdec/holdrate...|*10,tick)]

Transition: Transition trigger expression F DEAD LOGIC
Transition: Transition trigger expression T ACTIVE LOGIC

MATLAB EXPO 2016
Gaining Confidence in our Design

Effort / Time

Confidence

Ad-hoc testing

Design error detection

Simulation Testing Workflow

Requirements

Did we meet requirements?

Review functional behavior

Did we completely test our model?

Structural coverage report

MATLAB EXPO 2016
Did We Completely Test our Model?

Potential causes of less than 100% coverage:
- Missing requirements
- Over-specified design
- Design errors
- Missing tests

MATLAB EXPO 2016

Requirements Based Functional Testing with Coverage Analysis

- All 14 requirements based test cases pass
- By analyzing model coverage results we find:
  - Missing test cases for vehicle speed exit conditions, and
  - Missing requirements (and test cases) for “hold” or continuous speed button input
Functional Testing with Added Requirements & Test Cases

- Added 2 new requirements for the “hold” case for speed setting input buttons
- Added 5 test cases to the original 14 requirements based test cases
  - 3 test cases for the 2 new requirements
  - 2 test cases for the missing test cases for the vehicle speed exist conditions
- 4/5 new functional test cases pass
  - Failed test case showed overshoot beyond target speed limits
  - Coverage analysis highlighted transitions with design errors
  - Fixed comparison operators, (<) \(\rightarrow\) (<=), and (>) \(\rightarrow\) (>=)

- Now all (19) functional test cases pass with 100% model coverage!
Gaining Confidence in our Design

Confidence

- Ad-hoc testing
- Design error detection
- Functional & structural tests

Effort / Time

Model Advisor – Model Standards Checking

Model Advisor Report for Step_01_inputs

Web Browser - Model Advisor Report for Step_01_inputs

Model Advisor Report for Step_01_inputs

See Also
Gaining Confidence in our Design

- Ad-hoc testing
- Design error detection
- Functional & structural tests
- Modeling standards

Effort / Time

MATLAB EXPO 2016

Equivalence Testing:
Model vs SIL or PIL Mode Testing

Model Testing

SIL or PIL Mode Testing

Coverage → 100%

MATLAB EXPO 2016
Code Generation with Model-to-Code Traceability

Code Equivalence Check Results: Model vs Code

Code Coverage

<table>
<thead>
<tr>
<th>File Contents/Complexity</th>
<th>Code Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D1</td>
</tr>
<tr>
<td>CruiseControl.c</td>
<td>22.97%</td>
</tr>
<tr>
<td>CruiseControlInit</td>
<td>100%</td>
</tr>
<tr>
<td>CruiseControl</td>
<td>20.97%</td>
</tr>
<tr>
<td>CruiseControl_unalyze</td>
<td>100%</td>
</tr>
</tbody>
</table>

Decisions analyzed:

<table>
<thead>
<tr>
<th>Condition</th>
<th>True</th>
<th>False</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CruiseControl_DW.temporalCounter_S1 &lt; MAX_uint32_T</td>
<td>399/399</td>
<td>420/420</td>
<td>820/820</td>
</tr>
</tbody>
</table>

MATLAB EXPO 2020

MathWorks
**Code Equivalence Check Results: Model vs Code**

- Re-used full coverage test vectors and harnesses from Model Verification testing
- Ran test vectors on generated code using Model Reference SIL mode
- Equivalence test performed in Simulink Test, including test execution, evaluation and presentation of the results
- Compared Model Coverage to Code Coverage using the SIL Code Coverage Report
- Successfully demonstrated code behavior matches model behavior!

<table>
<thead>
<tr>
<th>File Contents/Complexity</th>
<th>D1</th>
<th>C1</th>
<th>MCDC</th>
<th>Stmt</th>
</tr>
</thead>
<tbody>
<tr>
<td>CruiseControl.c</td>
<td>22</td>
<td>97</td>
<td>99%</td>
<td>100%</td>
</tr>
<tr>
<td>CruiseControl_Init</td>
<td>3</td>
<td>100</td>
<td>99%</td>
<td>100%</td>
</tr>
<tr>
<td>CruiseControl</td>
<td>30</td>
<td>97</td>
<td>99%</td>
<td>100%</td>
</tr>
</tbody>
</table>

---

**Gaining Confidence in our Design**

- Design error detection
- Functional & structural tests
- Modeling standards
- Model & code equivalence checks
- Code integration analysis

---

MATLAB EXPO 2016
**Code Integration Analysis**

```plaintext
Inputs
- Cruise_onoff
- Brake
- Speed
- Coast set
- Accel reset
- EGO Sensor
- MAP Sensor

ECU system
- Cruise Control Module (MBD)
- Fuel Rate Control Module
- Shift Logic Control Module

Outputs
- Gear
- Engaged
- Target speed
- Fuel Rate

MATLAB EXPO 2016
```
Finding Dead Code During Integration

Inputs
- Cruise_onoff
- Brake
- Speed
- Coast set
- Accel reset
- EGO Sensor
- MAP Sensor

Inaccurate scaling for speed

Inputs
- ECU system

Cruise Control Module (MBD)

Outputs
- Gear
- Engaged
- Target speed
- Fuel Rate

Finding Dead Code with Polyspace

Target speed parameter propagated to “Cruise_ctrl.c” [0 … 40]

Maximum target speed = 90

```
/* Entry 'STANDBY': '<SS>:52' */
*try_Engaged = false;
}
else if (trv_Speed > maxtarget)

if (trv_Speed < mintarget)
/* Transition: '<SS>:111' */
```
Polyspace Code Analysis

Start with C/C++ source code

```c
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;
    }
}
```

Source code painted in green, red, gray, orange

- **Green**: reliable safe pointer access
- **Red**: faulty out of bounds error
- **Gray**: dead unreachable code
- **Orange**: unproven may be unsafe for some conditions
- **Purple**: violation MISRA-C/C++ or JSF++ code rules

Range data tool tip
Gaining Confidence in our Design

Model Verification
Discover design errors at design time

Code Verification
Gain confidence in the generated code

Workflow approved by TÜV SÜD for development of safety-critical software in accordance with ISO 26262 (automotive), IEC 61508 (industrial), EN 50128 (railway), IEC 62304 (medical devices)
Conclusion

It is easier and less expensive to fix design errors early in the process when they happen.

Model-Based Design enables:

1. Early testing to increase confidence in your design
2. Delivery of higher quality software throughout the workflow

MATLAB EXPO 2016