MATLAB EXPO 2018

Automating Best Practices to Improve Design Quality

Daniel Martins
Why do 71% of Embedded Projects Fail?

Poor Requirements Management

Sources: Christopher Lindquist, Fixing the Requirements Mess, CIO Magazine, Nov 2005
Key Takeaways

- Author, manage requirements in Simulink
- Early verification to find defects sooner
- Automate manual verification tasks
- Workflow that conforms to safety standards
Challenge with Traditional Development Process

1. Requirements
2. Specification
3. C/C++
4. Hand code

MATLAB EXPO 2018
Simulink Models for Specification

Requirements → Executable Specification → C/C++ → Hand code
Complete Model Based Design

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code

Code Generation
Model Based Design Verification Workflow

- Requirements
  - Executable Specification
  - Model used for production code generation
  - C/C++
  - Generated code

Component and system testing

Review and static analysis

Equivalence testing

Equivalence checking
Challenges with Requirements

Where are requirements implemented?

Is design and requirements consistent?

How are they tested?

Component and system testing

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code
Gap Between Requirements and Design

Requirements

Executable Specification

Model used for production code generation

Simulink Models

C/C++

Generated code

MATLAB EXPO 2018
Simulink Requirements

Author

Track

Manage

MATLAB EXPO 2018
Import Requirements from External Sources

**IBM Rational DOORS**

**Microsoft Word**

**Simulink Requirements Editor**

**3.1 Enabling cruise control**

Cruise control is enabled when the following conditions are met:

- Vehicle speed is within the target speed range (40km/h – 100km/h).
- Key position is ON.
- Gear position is Drive.
- Cruise button is pushed while the cruise control mode is disabled.
REQ 3.1 ENABLING CRUISE CONTROL
Cruise control is enabled when .....
Track Implementation and Verification
Respond to Change

Original Requirement

If the switch is pressed and the counter reaches 50 then it shall be recognized as a long press of the switch.

Updated Requirement

If the switch is pressed and the counter reaches 75 then it shall be recognized as a long press of the switch.
Verify Design to Guidelines and Standards

Is the design built right?

Is it too complex?

Is it ready for code generation?

Review and static analysis

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code
Automate verification with static analysis

Check for:
- Readability and Semantics
- Performance and Efficiency
- Clones
- And more……
Generate reports for reviews and documentation

Requirements

Executable Specification

Simulink Models

Model used for production code generation

C/C++

Generated code

Model Advisor Analysis

Model Advisor Reports
Navigate to Problematic Blocks

<table>
<thead>
<tr>
<th>Block</th>
<th>Block Type</th>
<th>Code generation support</th>
<th>Recommendation for C/C++ production code deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../Intake Manifold/p0 [0.589 bar</td>
<td>Integrator</td>
<td>Yes() 2</td>
<td>No</td>
</tr>
<tr>
<td>sldemo_fuelsys/Throttle Command</td>
<td>Repeating table</td>
<td>Yes()</td>
<td>No</td>
</tr>
</tbody>
</table>

Simulink Models

- Requirements
- Executable Specification
- Model used for production code generation
- C/C++
- Generated code
Guidance Provided to Address Issues or Automatically Correct

Recommended Action
Although Embedded Coder supports these blocks, they are not recommended for C/C++ production code deployment. Review the support notes for these blocks and follow the given advice.
Built in checks for industry standards and guidelines

- DO-178/DO-331
- ISO 26262
- IEC 61508
- IEC 62304
- EN 50128

- MISRA C:2012
- CERT C, CWE, ISO/IEC TS 17961
- MAAB (MathWorks Automotive Advisory Board)
- JMAAB (Japan MATLAB Automotive Advisory Board)
Configure and customize analysis

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code
Checks for standards and guidelines are often performed late
Shift Verification Earlier With Edit-Time Checking

- Highlight violations as you edit
- Fix issues earlier
- Avoid rework

Requirements \(\rightarrow\) Executable Specification \(\rightarrow\) Model used for production code generation \(\rightarrow\) Generated code

Simulink Models

Edit-Time Checking

Requirements

Executable Specification

Model used for production code generation

C/C++

Generated code
Assess Quality with Metrics Dashboard

- Consolidated view of metrics
  - Size
  - Compliance
  - Complexity

- Identify where problem areas may be
Grid Visualization for Metrics

- Visualize Standards
- Check Compliance
  - Find Issues
  - Identify patterns
  - See hot spots

Legend:
- Red: Fail
- Orange: Warning
- Green: Pass
- Gray: Not run
Detect Design Errors with Formal Methods

- Find run-time design errors:
  - Integer overflow
  - Dead Logic
  - Division by zero
  - Array out-of-bounds
  - Range violations

- Generate counter example to reproduce error
Prove That Design Meets Requirements

- Prove design properties using formal requirement models
- Model functional and safety requirements
- Generates counter example for analysis and debugging

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code
Functional Testing

Does the design meet requirements?

Is it functioning correctly?

Is it completely tested?

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code
Systematic Functional Testing

Test Case

Inputs
- MAT file (input)
- Group 1
- Signal 1
- Signal Builder
- Test Sequence

Assessments
- MAT file (baseline)
- MATLAB Unit Test
- Test Assessment

Test Harness

Main Model

MATLAB EXPO 2018
Manage Testing and Test Results
Coverage Analysis to Measure Testing

- Identify testing gaps
- Missing requirements
- Unintended Functionality
- Design Errors
Test Case Generation for Functional Testing

- Specify functional test objectives
  - Define custom objectives that signals must satisfy in test cases

- Specify functional test conditions
  - Define constraints on signal values to constrain test generator
Static Code Analysis

Is the code compliant to MISRA?

Is integrated code free of run-time errors?

Is interface between generated and other code fully tested?

Executable Specification

Model used for production code generation

Simulink Models

Other code

C/C++ Generated code

The Generated Code is integrated with Other Code (Handwritten)
Static Code Analysis with Polyspace

- **Code metrics and standards**
  - Comment density, cyclomatic complexity,…
  - MISRA and Cybersecurity standards
  - Support for DO-178, ISO 26262, ….

- **Bug finding and code proving**
  - Check data and control flow of software
  - Detect bugs and security vulnerabilities
  - Prove absence of runtime errors

**Results from Polyspace Code Prover**

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
<th>Example Code</th>
<th>Tool Tip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>reliable safe pointer access</td>
<td>static void pointer_arithmetic (void) {</td>
<td></td>
</tr>
<tr>
<td>Red</td>
<td>faulty out of bounds error</td>
<td>for (i = 0; i &lt; 100; i++) {</td>
<td></td>
</tr>
<tr>
<td>Gray</td>
<td>dead unreachable code</td>
<td>if (get_bus_status() &gt; 0) {</td>
<td></td>
</tr>
<tr>
<td>Orange</td>
<td>unproven may be unsafe for some conditions</td>
<td>else {</td>
<td></td>
</tr>
<tr>
<td>Purple</td>
<td>violation MISRA-C/C++ or JSF++ code rules</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Range data tool tip</td>
<td>(p - i)^2 - 10;</td>
<td></td>
</tr>
</tbody>
</table>
Equivalence Testing

Is the code functionally equivalent to model?

Is all the code tested?

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code

Equivalence checking

Equivalence testing
Equivalence Testing

- **Software in the Loop (SIL)**
  - Show functional equivalence, model to code
  - Execute on desktop / laptop computer

- **Processor in the Loop (PIL)**
  - Numerical equivalence, model to target code
  - Execute on target board

- Re-use tests developed for model to test code
- Collect code coverage
Qualify tools with IEC Certification Kit and DO Qualification Kit

- Qualify code generation and verification products
- Includes documentation, test cases and procedures

KOSTAL Asia R&D Center Receives ISO 26262 ASIL D Certification for Automotive Software Developed with Model-Based Design

BAE Systems Delivers DO-178B Level A Flight Software on Schedule with Model-Based Design

Kostal’s electronic steering column lock module.

Primary flight control computers from BAE Systems.
Lear Delivers Quality Body Control Electronics Faster Using Model-Based Design

Challenge
Design, verify, and implement high-quality automotive body control electronics

Solution
Use Model-Based Design to enable early and continuous verification via simulation, SIL, and HIL testing

Results
- Requirements validated early. Over 95% of issues fixed before implementation, versus 30% previously
- Development time cut by 40%. 700,000 lines of code generated and test cases reused throughout the development cycle
- Zero warranty issues reported

“We adopted Model-Based Design not only to deliver better-quality systems faster, but because we believe it is a smart choice. Recently we won a project that several of our competitors declined to bid on because of its tight time constraints. Using Model-Based Design, we met the original delivery date with no problem.”
- Jason Bauman, Lear Corporation
Customer References and Applications

Airbus Helicopters Accelerates Development of DO-178B Certified Software with Model-Based Design
Software testing time cut by two-thirds

LS Automotive Reduces Development Time for Automotive Component Software with Model-Based Design
Specification errors detected early

Continental Develops Electronically Controlled Air Suspension for Heavy-Duty Trucks
Verification time cut by up to 50 percent

More User Stories: [www.mathworks.com/company/user_stories.html](http://www.mathworks.com/company/user_stories.html)
Summary

1. Author and manage requirements within Simulink
2. Find defects earlier
3. Automate manual verification tasks
4. Reference workflow that conforms to safety standards