MATLAB EXPO 2018

Verification & Validation: Automating Best Practices to Improve Design Quality

Chuck Olosky
Growing Complexity of Embedded Systems

- Emergency Braking
- Body Control Module
- Voice Recognition
- Power Window
- Vehicle-to-Infrastructure
- Power Liftgate
- Power Seat
- Back-up Camera
- Long-Range Radar
- All-Wheel Drive
- Stability Control
- Adaptive Cruise Control
- Automatic Parking
- Smart Junction Box
- Battery Management
- Instrument Panel
- Airbag
- DC/DC Converter
- Propulsion Motor Control
- Navigation Management
- Transmission Control
- Forward Camera
- Adaptive Front Lighting
- HVAC Control
- Vehicle-to-Vehicle
- E-Call
- Keyless Entry
- Short-Range Radar

McKendrick, J. “Cars become ‘datacenters on wheels’, carmakers become software companies,” ZDJNet, 2013
Growing Complexity Challenges the Traditional Development Process

- Find requirements defects *later* in the process
- Find specification issues *later* in the process
- Find design issues *later* in the process
Using Simulink Models for Specification

- Find requirements defects earlier in the process
- Find specification issues earlier in the process
- Find design issues later in the process
Complete Model Based Design

- Find requirements defects earlier in the process
- Find specification issues earlier in the process
- Find design issues earlier in the process

Simulink Models

Requirements → Executable Specification → Model used for production code generation

C/C++ → Generated Code

Design Verification → Code Verification

MATLAB EXPO 2018
Model Based Design Verification Workflow

- Requirements
- Executable Specification
- Model used for production code generation
- Simulink Models
- Generated Code

Component and System Level Testing
Design Review and Static Analysis
Equivalence Testing
Equivalence Checking

Design Verification
Code Verification

MATLAB EXPO 2018
Key Takeaways

- Author, manage requirements in Simulink
- Early verification to find defects sooner
- Automate static and dynamic verification
- Workflow that conforms to safety standards

“Reduce costs and project risk through early verification, shorten time to market on a certified system, and deliver high-quality production code that was first-time right”  Michael Schwarz, ITK Engineering

MATLAB EXPO 2018
Why do 71% of Embedded Projects Fail?

Poor Requirements Management

Sources: Christopher Lindquist, Fixing the Requirements Mess, CIO Magazine, Nov 2005
Challenges with Requirements

Where are they implemented?

Are they consistent with the design?

How are they tested?

Requirements → Executable Specification → Model used for production code generation → Generated Code

Simulink Models

C/C++
Gap Between Requirements and Design

Requirements
- IBM Rational DOORS

Simulink Models
- Executable Specification
- Model used for production code generation

C/C++

Generated Code

MATLAB EXPO 2018
Simulink Requirements

Link & Track

Author

#31: Increment mode

IMPLEMENTS

opMode.Increment

Manage Updates

Issue: Destination Changed.

Stored:  Revision: 15

Actual:  Revision: 18

Clear Issue
To create a new requirement set to store requirements, click New Requirement Set. Save the requirement set to assign a name.

To add a requirement to a requirement set, select the requirement set and click Add Requirement. In the Properties pane, enter details for the requirement.

To add a child requirement, right-click a requirement and select Add Child Requirement.

To link a requirement to a block in your model, select the block, then right-click the requirement and select Link from "object name" (object type). A link appears in the Links pane.

For information on linking using the Requirements Perspective, see Getting Started in the documentation.

To view a list of links, select Links from the View dropdown list in the toolbar.

Change the source - destination relationship by selecting a link, and choosing a Type from the dropdown list in the Properties pane.
To create a new requirement set to store requirements, click New Requirement Set. Save the requirement set to assign a name.

To add a requirement to a requirement set, select the requirement set and click Add Requirement. In the Properties pane, enter details for the requirement.

To add a child requirement, right-click a requirement and select Add Child Requirement.

To link a requirement to a block in your model, select the block, then right-click the requirement and select Link from "object name" (object type). A link appears in the Links pane.

For information on linking using the Requirements Perspective, see Getting Started in the documentation.

To view a list of links, select Links from the View dropdown list in the toolstrip.

Change the source - destination relationship by selecting a link, and choosing a Type from the dropdown list in the Properties pane.
Import Requirements from External Sources

- **Import Requirements from External Sources**
  - IBM Rational DOORS
  - 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.
Requirements Perspective View of Model
Requirements Perspective View of Model
REQ 3.1 ENABLING CRUISE CONTROL
Cruise control is enabled when …..

ENABLE SWITCH DETECTION
If the Enable switch is pressed …..

Generated Code

Design Model

Test Case

High Level Reqs

Low Level Req

Implemented

By

Verified

By

REQ 3.1 ENABLING CRUISE CONTROL
Cruise control is enabled when .....

ENABLE SWITCH DETECTION
If the Enable switch is pressed ....
### Track Implementation and Verification Status

**Implementation Status**
- **Implemented**
- **Justified**
- **Missing**

**Verification Status**
- **Passed**
- **Failed**
- **No Result**
- **Missing**

#### Requirements - crs_controller

<table>
<thead>
<tr>
<th>Index</th>
<th>ID</th>
<th>Summary</th>
<th>Implemented</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>crs_req_func_spec</td>
<td>--</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>#1</td>
<td>Driver Switch Request Handling</td>
<td></td>
<td>Passed</td>
</tr>
<tr>
<td>2</td>
<td>#19</td>
<td>Cruise Control Mode</td>
<td></td>
<td>Passed</td>
</tr>
<tr>
<td>2.1</td>
<td>#20</td>
<td>Disable Cruise Control system</td>
<td></td>
<td>Passed</td>
</tr>
<tr>
<td>2.2</td>
<td>#24</td>
<td>Operation mode determination</td>
<td></td>
<td>Passed</td>
</tr>
</tbody>
</table>
Respond to Requirements 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.

issue: Destination Changed.
Verify Design to Guidelines and Standards

Is the design built right?

Too complex?

Ready for code generation?

Review and static analysis

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code
Automated Static Analysis of the Design

Check the model for

- Readability and Semantics
- Performance and Efficiency
- Clones
- And more……

Model Advisor Analysis

Simulink Models

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

MATLAB EXPO 2018
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(^1), Yes(^2)</td>
<td>No</td>
</tr>
<tr>
<td>sldemo_fuelsys/Throttle Command</td>
<td>Repeating table</td>
<td>Yes(^3)</td>
<td>No</td>
</tr>
</tbody>
</table>

Model Advisor Analysis

Simulink Models:

- Requirements
- Executable Specification
- Model used for production code generation
- C/C++
- Generated code
Guidance Provided to Address Issues w/ Auto-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.

MathWorks
Modeling Guidelines for High-Integrity Systems

- Leverage industry-best practices and MathWorks tool expertise when developing high-integrity systems.
- Modeling Guidelines with corresponding Model Advisor checks.
- Mapped to the modeling standards and guidelines objectives of industry standards like ISO 26262 and MISRA-C.

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 Static Analysis

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code
Generate Reports for Reviews and Documentation

Model Advisor Analysis

Model Advisor Reports

Simulink Models

Requirements

Executable Specification

Model used for production code generation

C/C++

Generated code

MATLAB EXPO 2018
Checks for Standards and Guidelines are often Performed Late

Simulink Models

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

Rework → Static Analysis
Shift Verification Earlier with Edit-Time Checking

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

Executable Specification

Model used for production code generation

C/C++

Generated code
Find Compliance Issues while you Design
Modeling Standards Checking with Simulink Check

- Static analysis of models against a set of checks

- Modeling Standards Checks
  - MAAB Style Guidelines V3.0
  - ISO 26262
  - MISRA C:2012

- Additional Checks
  - Model Metrics
  - Tool Bug Reports (Cert Kit)
  - Requirements Consistency
Assess Quality with Metrics Dashboard

• Consolidated view of metrics
  • Size
  • Compliance
  • Complexity

• Identify where problem areas may be
Grid View for Metrics Analysis

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

Legend:
- Red: Fail
- Orange: Warning
- Green: Pass
- Gray: Not run
Static Analysis for Detecting Design Errors

- Find run-time design errors
- Generates a test case to reproduce the issue for debugging
Functional Testing

Does the design meet requirements?

Is it functioning correctly?

Is it completely tested?

Component and System Level Testing

Simulink Models

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

Test Case

Inputs

- MAT file (input)
- Signal Builder
- Test Sequence

Assessments

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

Excel file (input)

Excel file (baseline)

MATLAB EXPO 2018
Test Execution and Results Analysis
Coverage Analysis to Measure Test Completeness

- 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
Prove 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
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?

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

Generated Code is integrated with Hand Code
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
Equivalence Testing

Is the code functionally equivalent to model?

Is all the code tested?

Equivalence Checking

Equivalence Testing

Simulink Models

Requirements

Executable Specification

Model used for production code generation

C/C++

Generated code

MATLAB EXPO 2018
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
- Check for equivalent outputs model to code
- Collect code coverage, compare to model coverage
Qualify tools with IEC Certification Kit and DO Qualification Kit

- Qualify code generation and verification tools
- 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

Kostal’s electronic steering column lock module.

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

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

MATLAB EXPO 2018
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
Summary

1. Author and manage requirements within Simulink
2. Find defects earlier
3. Automate static and dynamic verification
4. Reference workflow that conforms to safety standards
Learn More

Visit MathWorks Verification, Validation and Test Solution Page:
mathworks.com/solutions/verification-validation.html
Thank You!