MATLAB EXPO 2018

Verification and Validation of High-Integrity Systems

Chethan CU, MathWorks
Vaishnavi HR, MathWorks
Growing Complexity of Embedded Systems

- Engine Management
- Transmission Control
- Forward Camera
- Electric Power Steering
- Smart Junction Box
- Battery Management
- Propulsion Motor Control
- DC/DC Converter
- Stability Control
- Infotainment
- Navigation
- Vehicle-to-Vehicle Management
- Transmission Control
- Forward Camera
- Adaptive Front Lighting
- HVAC Control
- Vehicle-to-Infrastructure
- E-Call
- Keyless Entry
- Short-Range Radar

Lines of Code:
- 2000: 2-3M
- 2015: 16M
- 2015: 6M

McKendrick, J. “Cars become ‘datacenters on wheels’, carmakers become software companies,” ZDJNet, 2013
Model-Based Design, Verification and Validation

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

Simulink Models

Requirement based Model → Standards Compliant Model → Design Error free and Functionally correct Model → Code Generation ready Model

Verified code ready for target deployment
Key Takeaways

- Author, manage requirements in Simulink
- Early verification to find defects sooner
- Automate manual verification tasks
- Workflow that conforms to safety standards
- Static Source code verification

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

Poor Requirements Management

Sources: Christopher Lindquist, Fixing the Requirements Mess, CIO Magazine, Nov 2005
Challenge with Traditional Development Process
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
- Generated code

Simulink Models

Component and system testing
Review and static analysis
Equivalence testing
Equivalence checking

MathWorks
MATLAB EXPO 2018
Challenges with Requirements

Where are requirements implemented?

Is design and requirements consistent?

How are they tested?

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

Simulink Models
Gap Between Requirements and Design

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

Author

Track

Manage

MATLAB EXPO 2018
Requirements Editor

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

Microsoft Word

Simulink Requirements Editor

IBM Rational DOORS

ReqIF

MATLAB EXPO 2018

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.

Dashboard image

Keywords:
- Revision information:
- Links

Show in document
Requirements Perspective
Requirements Perspective
REQ 3.1 ENABLING CRUISE CONTROL
Cruise control is enabled when …..
REQ 3.1 ENABLING CRUISE CONTROL
Cruise control is enabled when ..... 

ENABLE SWITCH DETECTION
If the Enable switch is pressed ......
REQ 3.1 ENABLING CRUISE CONTROL

Cruise control is enabled when ..... 

ENABLE SWITCH DETECTION

If the Enable switch is pressed ...... 

Implemented By

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

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

Implemented By

Verified By

Test Case
Track Implementation and Verification

![Implementation and Verification Diagram](image)

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

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

Issue: Destination Changed.
Verify Design to Guidelines and Standards

Is the design built right?

Is it too complex?

Is it ready for code generation?

Simulink Models

Requirements → Requirement based Model → Standards Compliant Model → Design Error free and Functionally correct Model → Code Generation ready Model → C/C++
Automate verification with static analysis

Check for:
- Readability and Semantics
- Performance and Efficiency
- Clones
- And more......

Model Advisor Analysis

Simulink Models

Requirements → Requirement based Model → Standards Compliant Model → Design Error free and Functionally correct Model → Code Generation ready Model → C/C++

MATLAB EXPO 2018
Generate reports for reviews and documentation

Model Advisor Analysis

Model Advisor Reports

Simulink Models

Requirements

Requirement based Model

Standards Compliant Model

Design Error free and Functionally correct Model

Code Generation ready Model

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

Simulink Models

Requirements → Requirement based Model → Standards Compliant Model → Design Error free and Functionally correct Model → Code Generation ready Model → C/C++
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 → Requirement based Model → Standards Compliant Model → Design Error free and Functionally correct Model → Code Generation ready Model → C/C++

- My Custom Checks
  - My Company's Modeling Standards
    - Check state machine type of Stateflow charts
    - Check safety-related solver settings for simulation time
    - Check usage of Stateflow constructs
  - My Company's Metrics
  - My Company's Guideline Checks
  - Modeling Standards for IEC 61508
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

Simulink Models

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

MathWorks

MATLAB EXPO 2018
Find Compliance Issues as you Edit with Edit-Time Checking
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

Simulink Models

- Requirements
- Requirement based Model
- Standards Compliant Model
- Design Error free and Functionally correct Model
- Code Generation ready Model

C/C++
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
- Requirement based Model
- Standards Compliant Model
- Design Error free and Functionally correct Model
- Code Generation ready Model

MATLAB EXPO 2018
Functional Testing

Does the design meet requirements?

Is it functioning correctly?

Is it completely tested?

Simulink Models

Requirements → Requirement based Model → Standards Compliant Model → Design Error free and Functionally correct Model → Code Generation ready Model → C/C++
Systematic Functional Testing

Test Case

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

Assessments
- MAT file (baseline)
- MATLAB Unit Test
- MATLAB Code Check
- Excel file (baseline)

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

- Identify testing gaps
- Missing requirements
- Unintended Functionality
- Dead Logic
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
Model-Based Design, Verification and Validation

Simulink Models

- Requirement based Model
- Standards Compliant Model
- Design Error free and Functionally correct Model
- Code Generation ready Model

Simulink Models

- Requirements
- Executable Specification
- Model used for production code generation

Verified code ready for target deployment

Generated code

Target

MATLAB EXPO 2018
Equivalence Testing

Is the code functionally equivalent to model?

Is all the code tested?

Requirements

Executable Specification

Model used for production code generation

Simulink Models

C/C++

Generated code
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

Benefits
- Re-use tests developed for model to test code
- Collect code coverage
- Generate artefacts for IEC 61508, ISO 26262, EN 50128, and DO-178 certification
- Early verification and defect detection
Static Code Analysis

- Is interface between generated and other code fully tested?
- Is integrated code free of run-time errors?
- Is the code compliant to MISRA?

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

**Generated code**
- C/C++

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

Results from Polyspace Code Prover
Code Proving with Polyspace
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
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
5. Static Code verification using Polyspace
# MathWorks V&V Product Capabilities

<table>
<thead>
<tr>
<th>Category</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Simulink Requirements* <em>(New in R2017b)</em></td>
</tr>
<tr>
<td>Standards Compliance</td>
<td>Simulink Check* <em>(New in R2017b)</em></td>
</tr>
<tr>
<td>Testing</td>
<td>Simulink Test</td>
</tr>
<tr>
<td>Formal Verification</td>
<td>Simulink Design Verifier</td>
</tr>
<tr>
<td>Coverage Analysis</td>
<td>Simulink Coverage* <em>(New in R2017b)</em></td>
</tr>
<tr>
<td>Static Code Analysis</td>
<td>Polyspace Bug Finder, Polyspace Code Prover</td>
</tr>
<tr>
<td>SIL, PIL</td>
<td>Simulink Test</td>
</tr>
</tbody>
</table>

* Customers with Simulink V&V licenses will automatically receive these new products
KOSTAL Asia R&D Center Receives ISO 26262 ASIL D Certification for Automotive Software Developed with Model-Based Design

Challenge
Develop automotive electronic steering column lock software and certify it to the highest-level functional safety standard

Solution
Use Model-Based Design to design, implement, and verify the application software via back-to-back PIL testing required for ISO 26262 ASIL D certification

Results
- Development and certification time cut by 30%
- 80% of errors identified in modeling phase
- PIL test framework for ISO 26262 established

"Using Model-Based Design to design, implement, and verify our software for the highest functional safety standard enabled our team to save costs, increase efficiency, and ensure software quality. Without Model-Based Design, more engineers would be needed to complete the project in the same time frame."
– Cheng Hui, KOSTAL
Miele Proves Absence of Run-Time Errors in Control Software Across Its Entire Product Line

Challenge
Maintain a reputation for producing quality appliances and other products by minimizing defects in the control software

Solution
Integrate Polyspace Code Prover and Polyspace Bug Finder into the development process to prove the absence of run-time errors in the software and enforce standard coding rules

Results
- Hundreds of source files analyzed daily
- Developer focus on core functionality enabled
- Reusable, trusted components proven free of run-time errors

“We have embedded static code analysis with Polyspace products deeply into our quality assurance processes. It is much better to find run-time errors as development begins than to find them at the end of development—or worse, after the product is delivered.”
- Stefan Trampe, Miele

MATLAB EXPO 2018
Learn More

Visit MathWorks Verification, Validation and Test Solution Page: mathworks.com/solutions/verification-validation.html
Training Services

Exploit the full potential of MathWorks products

Flexible delivery options:

- Public training available in several cities
- Onsite training with standard or customized courses
- Web-based training with live, interactive instructor-led courses

More than 48 course offerings:

- Introductory and intermediate training on MATLAB, Simulink, Stateflow, code generation, and Polyspace products
- Specialized courses in control design, signal processing, parallel computing, code generation, communications, financial analysis, and other areas

www.mathworks.in/training
Verification and Validation of Simulink Models

This one-day course describes techniques for testing Simulink model behavior against system requirements.

Topics include:

- Identifying the role of verification and validation in Model-Based Design
- Creating test cases for Simulink models
- Analyzing simulation results to verify model behavior
- Automating testing activities and managing results
- Formally verifying model behavior
- Automatically generating artifacts to communicate results
Polyspace for C/C++ Code Verification

This two-day course discusses the use of Polyspace Bug Finder™ and Polyspace Code Prover™ to prove code correctness, improve software quality metrics, and ensure product integrity.

Topics include:
- Creating a verification project
- Reviewing and understanding verification results
- Emulating target execution environments
- Handling missing functions and data
- Managing unproven code (color-coded in orange by Polyspace® products)
- Applying MISRA C® rules
- Reporting
Thank You!