Developing Embedded Software with Model-Based Design to Meet Certification Standards

Paul Urban     Jeff Harper
Software safety is important across industries

- Software errors are common
  - Automotive: 19% of all recalls
  - Medical devices: 26% of all recalls

- Recalls are extremely expensive
  - MCAS failure: $18B+
  - Broken hip replacement: $3B+
  - Faulty pedals: $3B+
  - Airbag issues: $25B+

Why functional safety?

Safety critical systems

Best practices and guidelines

Functional safety standard

Certification
Poll: Safety standards across different industries

- **IEC 61508** – All industries
- **ISO 26262, SOTIF** – Automotive
- **DO 178C** – Avionics
- **IEC 62304** – Medical
- **EN 50128** – Rail
- **ISO 25119** – Agriculture and Forestry
- **IEC 61511** – Process Control
- And many others…..
Key Takeaways

Use Reference Workflow with Model-Based Design to Meet Certification Standards

- Create **traceability** across requirements, architecture, design, test and code
- Detect errors **earlier** by continuous testing
- **Reduce coding errors** with automatic code generation
- **Automate** generation of documents and reports for reviews and audits
V-Model for Certifiable Product Development

ISO 26262: 2018

DO 178C, ARP 4754A, ARP 4761
Reference Workflow for Certification

System Requirements → System Architecture → Software Requirements → Software Architecture → Implementation Model → Generated C/C++ Code → Object Code

System Level

Software Level
Reference Workflow for Certification – System Requirements and Architecture
Reference Workflow for Certification – Software Requirements and Architecture
Reference Workflow for Certification – Detailed Design
Reference Workflow for Certification – Model Verification

Model Verification
Discover design errors at design time

Testing at the model level

Review and static analysis at the model level

System Requirements → System Architecture → Software Requirements → Software Architecture → Implementation Model

System Architecture Development

Software Architecture Development

Modeling

System Level

Software Level
Reference Workflow for Certification – Code Generation

Model Verification
*Discover design errors at design time*

Testing at the model level

Review and static analysis at the model level

System Requirements → System Architecture → Software Requirements → Software Architecture → Implementation Model → Generated C/C++ Code

Model Verification

System Architecture Development

Software Architecture Development

Modeling

Code Generation

System Level

Software Level
Reference Workflow for Certification – Static Code Verification

Model Verification
Discover design errors at design time

Code Verification
Gain confidence in the generated code

Testing at the model level

System Requirements → System Architecture

Software Requirements → Software Architecture

Implementation Model → Generated C/C++ Code

System Architecture Development

Software Architecture Development

Modeling

Code Generation

MISRA Compliance
Reference Workflow for Certification – Dynamic Code Verification

Model Verification
Discover design errors at design time

Code Verification
Gain confidence in the generated code

System Requirements → System Architecture
Software Requirements → Software Architecture
Implementation Model → Generated C/C++ Code

Testing at the model level
Review and static analysis at the model level
Code Generation
Compilation and Linking

System Level
Software Level
Reference Workflow for Certification – Atomic Components

**Model Verification**
*Discover design errors at design time*

- Testing at the model level
- Review and static analysis at the model level
- Equivalence testing
- MISRA Compliance

**Code Verification**
*Gain confidence in the generated code*

- Code Generation
- Compilation and Linking
- Generated C/C++ Code
- Object Code

---

**System Level**

1. System Requirements
2. System Architecture
3. Software Requirements
4. Software Architecture

**Software Level**

1. Model Verification
2. Code Verification
3. Implementation Model
4. Generated C/C++ Code
5. Object Code

System Architecture Development
Software Architecture Development
Modeling
Reference Workflow for Certification – Static Code Verification

Model Verification
*Discover design errors at design time*

Testing at the model level

Review and static analysis at the model level

Code Verification
*Gain confidence in the generated code*

Equivalence testing

System Requirements

System Architecture

Software Architecture

Software Requirements

Implementation Model

Generated C/C++ Code

Object Code

MISRA Compliance

System Architecture Development

Software Architecture Development

Modeling

Code Generation

Compilation and Linking

Other C/C++ Code

System Level

Software Level
Reference Workflow for Certification – Integration and Verification

Model Verification
Discover design errors at design time

Code Verification
Gain confidence in the generated code

System Requirements → System Architecture → Software Requirements → Software Architecture → Implementation Model

- Module and integration testing at the model level
- Review and static analysis at the model level
- Equivalence testing
- MISRA Compliance

- Code Generation
- Compilation and Linking

System Level
Software Level
ROI Results

**Continental**
Verification time cut by up to 50 percent

**Leonardo**
Recertification cycle times reduced by more than 90%

**Tessella**
Models reused on follow-on projects, cutting design effort by up to 80%

**Corindus**
Development time halved and engineering effort reduced by 80%

More User Stories: [www.mathworks.com/company/user_stories.html](http://www.mathworks.com/company/user_stories.html)
Poll: What is your biggest challenge in verifying or testing your design? (choose the most important one)

- Tracing requirements to design and test
- Finding requirements errors late in development
- Meeting test coverage goals
- Reusing tests across design phases
- Performing system level validation
- Creating documentation and work products
- Performing manual reviews
- Other (Enter in Chat)
Requirements
Requirement span multiple levels across workflow

- Are all requirements implemented?
- Is each requirement tested?
- What artifacts are impacted by a change?
Simulink Requirements

Work with Requirements, Architecture and Design Together

Import / Export

Author

Analyze

Detect Changes

Trace

Simulink Requirements

External Requirements

Import / Export

Author

Analyze

Detect Changes

Report

Simulink, Stateflow, System Composer, MATLAB Code

Simulink Test

Requirements Management Tools
Review and Analyze Traceability with Traceability Matrix
Review and Analyze Traceability with Traceability Matrix

- Filters to focus view
Review and Analyze Traceability with Traceability Matrix
Review and Analyze Traceability with Traceability Matrix

- Create links to address gaps
Review and Analyze Traceability with Traceability Matrix

- Create links to address gaps
System and Software Architecture

- System Requirements
- System Architecture
- Software Requirements
- Software Architecture
- Implementation Model
- Generated C/C++ Code
- Object Code

System Level

- System Architecture Development
- Software Architecture Development

Software Level

- Code Generation
- Compilation and Linking
- Modeling
Why is architecture important?

- Author and validate requirements earlier
- Appor tion the safety objectives
- Coordinate software and hardware interfaces
- Share requirements without ambiguity
- Determine allocation to software or hardware
- Link to behavioral models
Describe systems using multiple architecture models to organize and check the completeness of the requirements

- **Functional Architecture**
- **Logical Architecture**
- **Physical Architecture**

Stakeholder Needs
- Analyze & Simulate
- Update & Refine

- Functional Requirements
- System Requirements
- Hardware Requirements
Allocate elements from one model to another to create a traceable & analyzable link
Reference Workflow for Certification – Detailed Design
Reference Workflow for Certification – Detailed Design
Reference Workflow for Certification – Detailed Design
Reference Workflow for Certification – Detailed Design
Reference Workflow for Certification – Model Verification

Model Verification
Discover design errors at design time

Module and integration testing at the model level

Review and static analysis at the model level

System Requirements → System Architecture → Software Requirements → Software Architecture → Implementation Model → Generated C/C++ Code → Object Code

System Architecture Development → Software Architecture Development → Modeling → Code Generation → Compilation and Linking
Verify Model Complies with Guidelines and Standards

Static analysis checks for:

- Readability and Semantics
- Performance and Efficiency
- Reusability
- And more……
Built in checks for industry standards and guidelines
Detect Design Errors Using Formal Methods

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

- Generate counter example to reproduce error
Model-Based Design Reference Workflow

Model Verification
*Discover design errors at design time*

Testing at the model level

Review and static analysis at the model level

System Requirements → System Architecture → Software Requirements → Software Architecture → Implementation Model → Generated C/C++ Code → Object Code

System Level

Software Level
Systematic Functional Testing with Simulink Test

Test Case

Model Sim through SIL, PIL and HIL
Scale with Parallel Computing Toolbox and Continuous Integration

Inputs

- Data file (input)
  - Signal Editor
  - Test Sequence
  - MATLAB Code

Assessments

- Data file baseline
- Test Assessment
- Temporal Assessment

Test Harness

Main Model

Actions

- and more!
- and more!

- Stateflow
- Sim through SIL, PIL and HIL
Manage Testing and Test Results
Coverage Analysis to Measure Testing

- Identify testing gaps
- Missing requirements
- Unintended functionality
- Design errors
Challenge: Managing the many activities and artifacts needed to meet certification standards
Model Testing Dashboard: Manage progress, completeness and quality of requirements-based testing

- Central view that summarizes testing data and status
- Identify gaps and respond faster to requirements changes
- View results from a continuous integration system
Code Generation

System Requirements → System Architecture → Software Requirements → Software Architecture → Implementation Model → Generated C/C++ Code → Object Code

System Architecture Development
Software Architecture Development
Modeling
Code Generation
Compilation and Linking

System Level
Software Level
Automatically generate production-quality code that behaves the same way as the model you created in Simulink
Requirement Traceability included in Generated Code
Reference Workflow for Certification – Code Verification

Code Verification
Gain confidence in the generated code

Equivalence testing
MISRA Compliance
Reference Workflow for Certification – Code Verification

Static Code Verification for MISRA Compliance
Model-Based Design Reference Workflow

System Requirements → System Architecture → Software Requirements → Software Architecture → Implementation Model → Generated C/C++ Code → Object Code

System Architecture Development → Software Architecture Development → Code Generation → Compilation and Linking

System Level → Software Level

Code Verification
Gain confidence in the generated code

Equivalence testing
MISRA Compliance
Back-to-Back Testing

- Model
  - Desktop Simulation (on PC)
- Generated Code
- Compiler
- Object File
- Object Code Execution (SIL/PIL)
- Results
- Compare
  - == ?
- Results
Automate Test Creation using Test Manager Wizard
Generate Reports for Reviews and Audits

Leonardo User Story

250,000 pages of interactively linked documentation generated. “Altogether, we generated the equivalent of around 250,000 pages of interactively linked test reports and other documentation using Simulink Test and Simulink Report Generator,” notes Brown. “As the certification authority has carte blanche to review whatever they want, we felt it was easier to be able to provide evidence for everything we had done. If the DER wanted to see any particular result, it was available for them and fully linked to our model, which really built their trust.”
Interactivity Poll: What capabilities presented today could help you improve your development? (choose all that apply)

- Requirements Traceability
- Architecture Analysis for safety
- Standards and Guideline Checking
- Functional Testing at Model level
- Measuring test coverage
- Dashboards and process management
- Automatic code generation
- Back-to-Back or Equivalence Testing
- Report generation
- Other (Enter in Chat)
**Summary: Reference Workflow with Model-Based Design**

- Create traceability across requirements, architecture, design, test and code
- Detect design errors early by continuous testing
- Automatic code generation
- Generate documents and reports for reviews and audits

![Workflow Diagram](image-url)
IEC Certification Kit and DO Qualification Kit

- Qualify code generation and verification products
- Includes documentation, test cases, and procedures to help certify
Learn More

- **Embedded Code Generation**
- **Verification, Validation, and Test Solution Page**
- **Verifying Models and Code for High-Integrity Systems**
- **Using Model Based Design in the ISO 26262:2018 Case Study**
- **Helicopter Flight Control: A Model-Based Design Example for DO-178C and DO-331**
Thank you