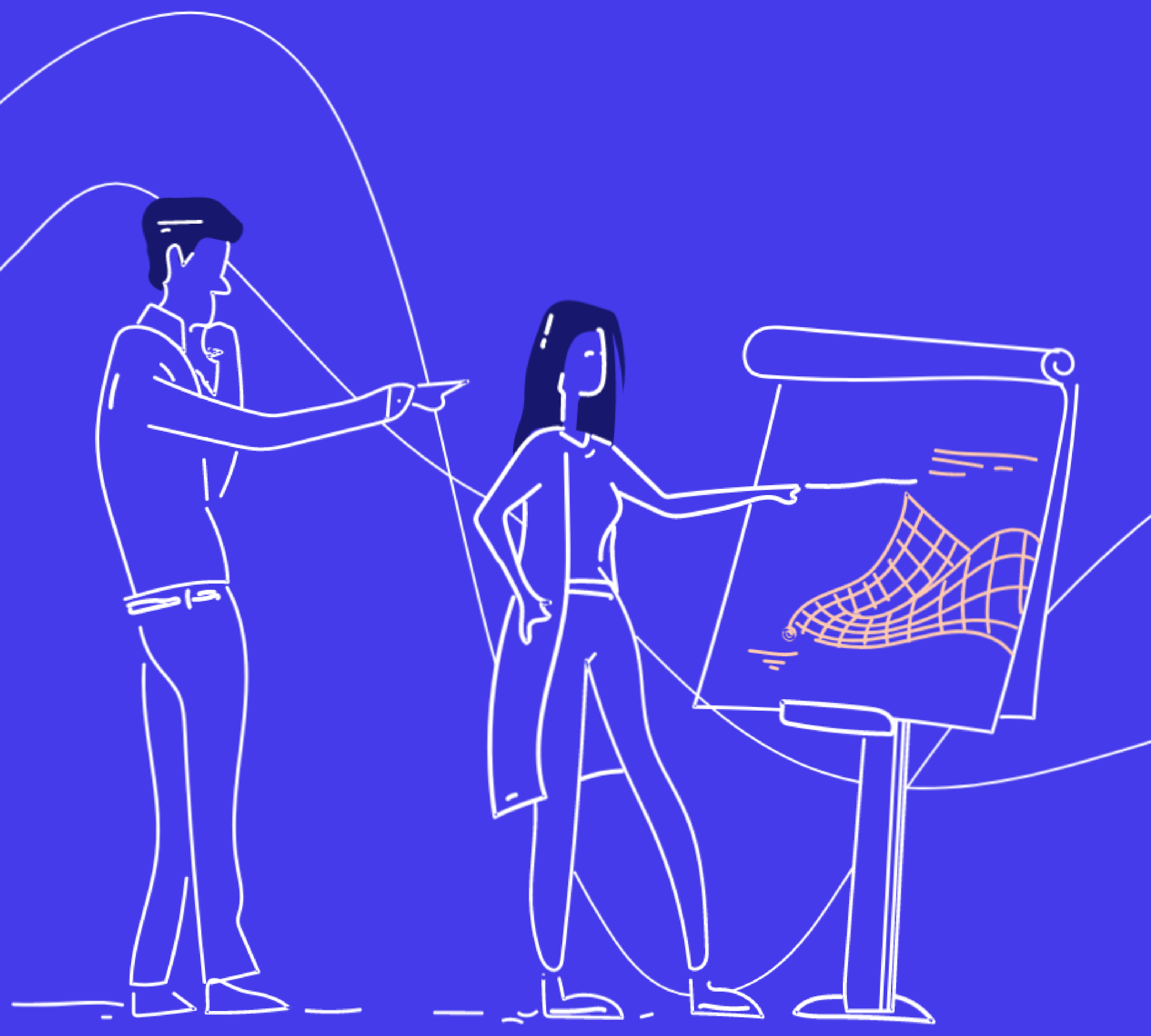
sciengineer

Applying Model-Based Design for ISO 26262



SciEngineer's training courses are designed to help organizations and individuals close skills gaps, keep up-to-date with the industry-accepted best practices and achieve the greatest value from MathWorks® and COMSOL® Products.

Applying Model-Based Design for ISO 26262

This five-day course describes guiding principles for applying Model-Based Design to meet ISO 26262 certification. It enables users to take advantage of the Simulink® environment to synthesize, implement, and validate their software components in a manner consistent with the principles of ISO 26262.

Prerequisites

MATLAB Onramp and Simulink Onramp. This course is intended for intermediate or advanced Simulink users. Knowledge of C programming language is recommended. Knowledge of the ISO 26262 standard is recommended.

DURATION	LEVEL
5 Days	Medium

TOPICS

Day 1

- Overview of ISO 26262 and Model-Based Design
- Model Creation
- Model Timing and Execution
- Modeling Standards

Day 2

- Requirements Management
- Architecture Modeling
- Model Management

Day 3

- Project Management
- Developing Test Cases
- Building Test Suites

Day 4

- Code Generation and Customization
- Generated Code Architecture
- Software Testing and Analysis

Day 5

- Reports
- Software Development Best Practice
- Tool Qualification
- Case Study

TRAINING CONTENT - DAY 1

Overview of ISO 26262 and Model-Based Design

OBJECTIVE: Get an overview of ISO 26262 and its role in the automotive industry. Discuss MathWorks' involvement and level of support within this standard.

- SO 26262 standard
- Required safety level
- Tool confidence level
- Reference workflow
- MathWorks tools certification
- IEC Certification Kit

Model Creation

OBJECTIVE: Create a Simulink model and define its sample time and data types.

- Introduction to the Simulink environment
- System inputs and outputs
- Discrete signals and states
- Simulation and analysis
- Simulation data inspector

Model Timing and Execution

OBJECTIVE: Simulating a single-rate and a multi-rate model. Data transfer considerations.

- Simulink solver overview
- Block execution
- Modeling single-rate systems
- Multi-rate discrete systems
- Rate transitions
- Data integrity and data determinism considerations

Modeling Standards

OBJECTIVE: Explore how to set up and enforce modeling standards, check for common modeling errors, and optimize model performance.

- Modeling standards
- Model Advisor
- Reporting results

TRAINING CONTENT - DAY 2

Requirements Management

OBJECTIVE: Link a Simulink model to system requirements.

- Identifying and writing high-level requirements
- Writing requirements
- Creating requirement sets
- Importing requirements
- Requirements linking

Architecture Modeling

OBJECTIVE: Introduce System Composer for system architecture workflows.

- Introducing System Composer
- Creating architectural elements
- Defining stereotypes for each type of element
- Analyzing architectures
- Component interfaces
- Bus objects
- Data dictionaries
- Creating architectural views
- Linking Simulink models

Model Management

OBJECTIVE: Discuss the pros and cons of the different features used for organizing a Simulink model into separate components.

- System component considerations
- Virtual subsystems
- Atomic subsystems
- Model references
- Subsystems and model referencing
- Model referencing workflow
- Model reference simulation modes
- Model workspaces
- Libraries
- Creating and populating libraries
- Managing library links
- Component variants

Project Management

OBJECTIVE: Discuss how to effectively organize a project (containing models, data, documentation, etc.) and perform configuration management tasks.

- Project setup
- File dependencies and impact
- Source control integration
- File differences

Developing Test Cases

OBJECTIVE: Create time-based and logic-based test cases for a Simulink model.

- Types of verification
- Defining test cases
- Generating test harnesses
- Creating and importing test inputs
- Incorporating logic in tests

Building Test Suites

OBJECTIVE: Create repeatable groups of tests and automatically generate reports from the test results.

- Creating test files
- Discuss simulation, baseline, and equivalence tests
- Performing requirements-based assessments
- Measuring model coverage
- Increasing coverage with automatic test generation
- Viewing and documenting test results

Code Generation and Customization

OBJECTIVE: Configure Simulink models for embedded code generation using optimization and customization options and effectively interpret the generated code.

- Architecture of an embedded application
- Generating code
- Modifying function prototypes
- Reusable function interface
- Setting signal storage classes
- Controlling storage classes with data objects
- Creating reconfigurable data types
- Data dictionaries

Generated Code Architecture

OBJECTIVE: Control the architecture of the generated code using subsystems, model references, and buses.

- Creating reusable model references
- Controlling data type of bus signals
- Generating reusable subsystem code
- Generating variant components

Software Testing and Analysis

OBJECTIVE: Software testing and verification using in-the-loop testing techniques for model references and top-level model.

- Software-in-the-loop testing of generated code
- Profiling generated code
- Model Reference software testing
- Hardware support overview
- Arduino setup
- Validating generated code on target

TRAINING CONTENT - DAY 5

Reports

OBJECTIVE: Discuss the methods of automatically creating reports and documentation from Simulink models.

- Web views
- Standard reports

Software Development Best Practice

OBJECTIVE: Perform static analysis on the generated code to ensure the code is compliant with MISRA C:2012.

- Code verification using Polyspace Bug Finder
- Software MISRA C:2012 compliance
- Analyzing code metrics

Tool Qualification

OBJECTIVE: Use the IEC Certification Kit for ISO 26262 to qualify MathWorks tools to meet compliance with ISO 26262.

- Tool qualification
- IEC Certification Kit for ISO 26262

Case Study

OBJECTIVE: Apply Model-Based Design to implement a control algorithm to showcase the reference workflow.

- Requirements traceability
- Software unit design
- Software unit testing
- Integration testing
- Code generation

sciengineer

Expand your knowledge