Applying Model-Based Design for ISO 26262

OEM: MathWorks • Code: MW-APPLYING-MODEL-BASED-DESIGN-FOR-ISO-2

COURSE DESCRIPTION

This five-day course describes guiding principles for applying Model-Based Design to meet ISO 26262 and IEC 61508 compliance for safety-related software development. 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 functionals safety guidelines. Design and implement modular software using Simulink subsystems, libraries, and models. Manage traceability between requirements, architecture, subsystems, tests, and code. Practice early verification and validation during software development using model-based and code-based testing. Establish and enforcing software standards across all stages in the development process. Streamline tool qualification using the IEC Certification Kit (for ISO 26262 and IEC 61508).

AUDIENCE PROFILE

- IT professionals and technical staff seeking hands-on training in this course's subject area.

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 or IEC 61508 standard is recommended.

COURSE OBJECTIVES

By the end of this course, participants will be able to:

- Get an overview of ISO 26262 and its role in the automotive industry. Discuss MathWorks' involvement and level of support within this standard.
- Organize project files (models, data, documentation). Familiarize with the project environment.
- Create and simulate a Simulink model for algorithm development. Manage model data using data dictionaries.
- Explore how to set up and enforce modeling standards and check for common modeling errors.
- Link a Simulink model to software requirements.
- Create time-based and logic-based test cases for a Simulink model.
- Generate code for a software unit. Customize the generate code to optimize data storage and execution.
- Create functional partitioning within a software unit using subsystems. Package subsystems into library blocks for reuse. Create partitions in the generated code.
- Introduce rate-based and export function modeling approach. Handle rate transition between rates.

- Create a software architecture model using System Composer. Analyze the software architecture and link to behavioral model.
- Organize software units into an integration model using model referencing. Configure model settings and data dictionaries so they can be shared across different models in the integration stage.
- Testing and verification of the generated code using in-the-loop testing techniques.
- Create repeatable groups of tests and automatically generate reports from the test results.
- Perform static analysis on the generated code to ensure the code is compliant with MISRA C:2012.
- Discuss the methods of automatically creating reports and documentation from Simulink models. Discuss configuration management methods in the project environment.
- Use the IEC Certification Kit (for ISO 26262 and IEC 61508) to qualify MathWorks tools to meet compliance with ISO 26262
- Apply Model-Based Design to implement a control algorithm to showcase the reference workflow.

COURSE OUTLINE

Module 1: Day 1 of 5

Overview of ISO 26262 and Model-Based Design

- ISO 26262 standard
- Model-Based Design overview
- Reference workflow

Project Management

- Project setup
- File shortcuts and labels
- File dependency analysis

Model Creation

- Simulink environment
- Discrete-time models
- Sample time
- Simulation and analysis
- Data dictionary
- Solver selection

Model Compliance

- Modeling standards
- Edit-time checks
- Model Advisor
- Results reporting

Module 2: Day 2 of 5

Requirements Management

- Requirement sets
- Requirements import
- Requirements linking

Software Unit Verification

- Types of verification
- Design error detection
- Test harness creation
- Test inputs
- Logic in tests
- Requirement-based assessments

Code Generation for Software Unit

- Code generation for step function
- Function prototypes
- Data storage optimization
- Data types and storage classes
- Data objects
- Function templates

Module 3: Day 3 of 5

Subsystems

- Subsystems
- Variant subsystems
- Subsystem references
- Masks
- Libraries
- Subsystem code generation

Multirate Modeling

- Block execution
- Single-rate systems
- Multirate systems
- Rate transitions
- Export function models

Architecture Modeling

- Architecture model
- Profiles and stereotypes
- Interface Editor
- Views
- Behavioral model linking

Module 4: Day 4 of 5

System Integration

- System component considerations
- Model references
- Referenced data dictionaries
- Referenced configuration sets
- Code generation for integration model
- Model workspace

In-the-Loop Testing

- Software-in-the-loop testing
- Code profiling
- Model reference software testing
- Processor-in-the-loop testing

Verification Automation

- Test files
- Coverage analysis
- Test results reporting

Module 5: Day 5 of 5

Code Verification

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

Reporting

- Model Testing Dashboard
- Web views
- Standard reports
- Source control integration
- File differences

Tool Qualification

- Tool qualification
- IEC Certification Kit (for ISO 26262 and IEC 61508)