Architektur in Simulink & Stateflow

Wie Sie große Softwaremodelle verwalten - 2 Tage

Modelle sind die Kern-Artefakte in der Softwareentwicklung. Im Laufe der Zeit - innerhalb eines einzelnen Entwicklungsprojekts oder auch über mehrere Entwicklungsschritte hinweg - wachsen die Modelle, da sie immer mehr Funktionalität umfassen. Dadurch werden Modelle schwer zu pflegen, kaum noch verständlich und das Fehlerrisiko steigt durch unerwartetes Verhalten. Die Fehleranfälligkeit des Modells erfordert geeignete Gegenmaßnahmen. Diese Schulung befasst sich mit Defiziten, die sich durch große Simulink- und Stateflow-Modelle ergeben und zeigt Wege auf, wie diese Risiken überwunden werden können.

Zielgruppe

Diese Schulung richtet sich an Modellierer*innen, Entwickler*innen, Tester*innen, Qualitätsmanager*innen, Projektmanager*innen und Teamleiter*innen, deren Schwerpunkt die modellbasierte Entwicklung von Embedded Software auf Basis von MATLAB/Simulink für Serienprojekte ist.

Highlights

- Grundlagen von Softwarearchitekturen
- Bewertung von architektonischen Gestaltungsprinzipien in Modellen
- Refactoring von Simulink-Modellen
- Mehrschichtige Anwendungsarchitektur
- Darstellung von Architektur in Modellen

Pawel Malysz, FCA US

„Die Seminare vermitteln Einblicke und Ideen, wie man an große Softwareprojekte systematisch herangeht - mit praktischen Vorschlägen und quantitativen Metriken.”

Sprachen

auf Deutsch oder Englisch

Formate

Piktogramm On-Site-Training

Vor Ort
an einem unserer Schulungsstandorte

Piktogramm Online-Training

Online

wo immer Sie gerade sind

Piktogramm Inhouse Training

Image not found or type unknown

Spezifisch für Ihr Unternehmen
online oder vor Ort in Ihrem Unternehmen

Weitere Details zu Formaten und Standorten

Preise und Geschäftsbedingungen

Die Preise und Geschäftsbedingungen entnehmen Sie bitte der PDF-Datei.

- Image not found or type unknown MES AGB - Schulungen (44,0 KiB)

Anfrage senden

Unsere Trainer*innen

Image not found or type unknown Image not found or type unknown Image not found or type unknown Image not found or type unknown Image not found or type unknown Image not found or type unknown

Agenda

Tag 1

Überblick: Modellbasierte Entwicklung und Qualitätssicherung mit Simulink

- Grundlagen der modellbasierten Entwicklung
- Überblick über Entwicklungs- und Qualitätssicherungsaktivitäten
- Merkmale der ISO 26262-konformen Entwicklung

Analyse und Evaluation von Modellstrukturen

- Einführung in Komplexitätsmetriken
- Berechnung der Modellkomplexität
- Gegenmaßnahmen bei komplexen Modellen
- Beurteilung von Kohärenz in Modellen
- Softwarearchitektur und Modellstruktur am Anwendungsbeispiel

Softwarearchitektur

- Grundlagen der Softwarearchitektur
- Erwartete Eigenschaften einer ISO 26262-konformen Softwarearchitektur
- Prinzipien des Software Unit Designs

Implementierung von Softwarearchitektur in Modellen

- Softwarearchitektur in Modellen
- Prinzipien von Schichtenmodellen
- Interface-Handling in Modellen

Integration von Modellen und verteilte Modellierung

- Vorteile von referenzierten Modellen und Bibliotheken
- Definition von verteilten Parameterdateien

Hands-on: Verbesserung der Modellstrukturen

Tag 2

Refactoring von Simulink-Modellen und deren Struktur

- Modellierungsstile, die das Refactoring erleichtern
- Grundlegende Refactoring-Operationen in Simulink
- Komplexe Refactoring-Operationen

Hands-on: Model Refactoring

Refactoring von Stateflow-Charts

- Herausforderungen der Stateflow-Semantik
- Ein sicherer Modellierungsstil für Stateflow
- Beispielregeln für Refactoring

Hands-on: Refactoring mit den Modellen der Teilnehmer*innen

Regressionstests von Modellen

- Testziele auf verschiedenen Teststufen
- Sicherstellen der funktionalen Eigenschaften von Modell und Code
- Regressions- und Back-to-Back-Tests, MiL – SiL – PiL
- Automatische Testevaluation mit Testassessments

Prozessbelange des Refactorings

- Rollen und Verantwortlichkeiten von Software-Architekt*innen, Software-Entwickler*innen und Testingenieur*innen
- Unterscheidung zwischen Architekturdesign (Top-Down-Ansatz) und Architekturverbesserung (Bottom-Up-Ansatz) von neu entstehenden Architekturen
- Refactoring in agilen Umgebungen
- Refactoring von Legacy-Modellen