IN 10 SCHRITTEN ZUR ISO-26262-KONFORMITÄT MIT MODELLBASIERTER SW-ENTWICKLUNG

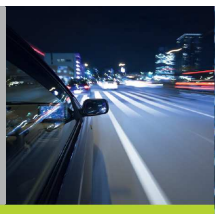
Hartmut Lackner

Leiter MES Test Center

9.02.2017

LÄUFT DIE SOFTWARE, FÄHRT DAS AUTO.

SOLUTIONS FOR INTEGRATED QUALITY ASSURANCE OF EMBEDDED AUTOMOTIVE SOFTWARE



OUR EXPERTISE

MES TEST CENTER	MES QUALITY TOOLS	MES ACADEMY
Full-scale quality assurance services for safeguarding embedded software in the car	Professional tools that simplify development, improve quality and ensure software safety	Knowledge transfer of methods, tools, and best practices for automotive software development

MOTIVATION

Geht es heute nicht um das Thema Testen? Doch!

- ☐ Die ISO-26262 stellt zahlreiche Anforderungen an die Qualitätssicherung
- ☐ Viele davon können wir mit Testmethoden erfüllen.

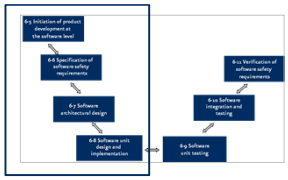
Wie muss sich ein bestehender SW-Prozess ändern?

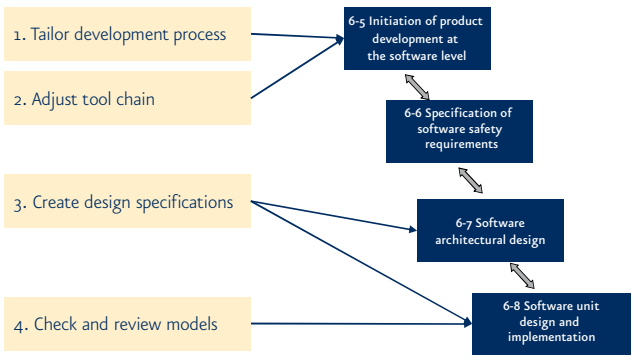
- ☐ Ein modellbasierter SW-Entwicklungsprozess ist bereits definiert
- ☐ Die Toolkette ist auch definiert
- ☐ Das Budget ist limitiert

Priorisierte Anforderungen zur Adaption der ISO-26262

10 SCHRITTE ZUR ISO-26262-KONFORMITÄT

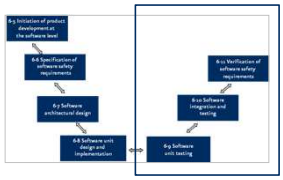
… folgen dem V-Modell nach ISO-26262:





10 SCHRITTE ZUR ISO-26262-KONFORMITÄT

MES MODEL ENGINEERING SOLUTIONS



5. Test focus - requirements

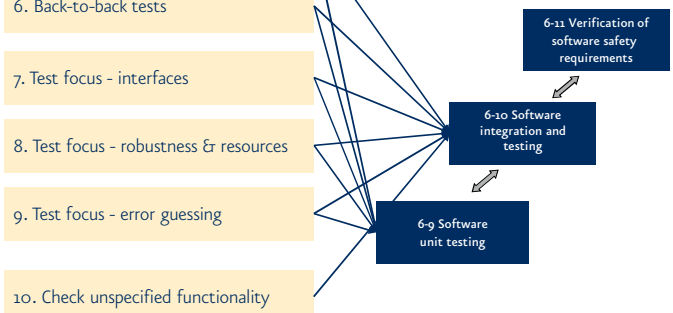
6. Back-to-back tests

7. Test focus - interfaces

8. Test focus - robustness & resources

9. Test focus - error guessing

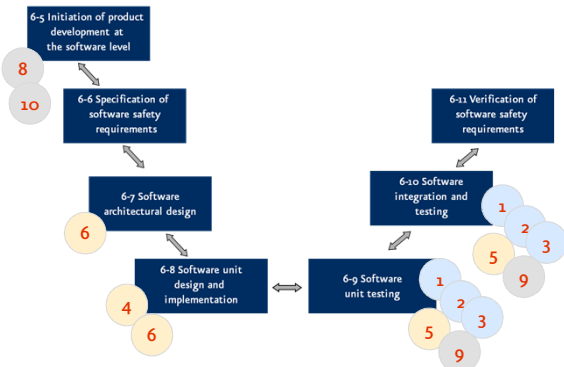
10. Check unspecified functionality



10 SCHRITTE ZUR ISO-26262-KONFORMITÄT

MES MODEL ENGINEERING SOLUTIONS

… und wie wir sie in der Projektrealität leben:



Detect Failures

1. Test focus - requirements
2. Back-to-back tests
3. Test focus - interfaces

Avoid Failures

4. Check and review models
5. Test focus - error guessing
6. Create design specifications

Compliant process

7. Check unspecified functionality
8. Adjust tool chain
9. Test focus – robustness/resources
10. Tailor development process

STEP 1: TEST FOCUS – REQUIREMENTS

TO DO	Automation	ISO
Anforderungsbasiertes Testen auf Unit- und Intergrationsebene ▪ Anforderungsmanagement: Anforderungen mit einzigartigen IDs ▪ Test Management: Testspezifikation mit einzigartigen IDs, Unit-Tests verlinkt mit Anforderungen ▪ Definieren der Testumgebung (MiL/SiL/PiL)		6-9.4, 6-10.4, T10-1a, T11, T13-1a, T15
Zeigen der Anforderungsabdeckung auf Unit-Ebene ▪ Messen der Abdeckung anhand verlinkter Testspezifikationen		6-9.4.5
Strukturelle Abdeckung messen ▪ Messen der Statement Coverage (C0) bzw. Branch Coverage (C1) in der SiL/PiL Testumgebung		6-9.4.5 T12

BENEFIT

- Entdecken fehlerhafter Implementierungen oder unerwartetem Verhalten wegen unvollständiger, inkonsistenter oder mehrdeutiger Anforderungen auf **systematische** Weise
- Bewertung der Testspezifikation auf Unit-Ebene

STEP 4: CHECK & REVIEW MODELS

TO DO	Automation	ISO
Anwendung von Modellierungsrichtlinien ▪ Auswählen der Modellierungsrichtlinien ▪ Einhaltung der Richtlinien prüfen		6-5.4.7-T1, 6-8.4.4-T8
Manuelles Review ▪ Modell-Review als **formales Review** (Inspektion) durchführen – unabhängig vom ASIL Level ▪ Erstellen einer Review-Checkliste		6-8.4.5-T9

BENEFIT

- Verbessert Les- und Wartbarkeit
- Vermeiden fehleranfälliger Modellelemente und Modellierungsmuster
- Entdecken potenzieller schon Fehler vor dem Test, z.B. implizite Typkonvertierungen

STEP 10: TAILOR DEVELOPMENT PROCESS

TO DO	Automation	ISO
Tailoring des ISO-26262 Referenzprozesses ▪ Definieren der Prozessphasen, Aktivitäten und Verantwortlichkeiten unter Berücksichtigung der Werkzeugkette ▪ Dokumentieren der Prozessdefinition		6-5.4.1- 6-5.4.6

BENEFIT

- Effizienter Entwicklungsprozess durch definierte und geprüfte Aktivitäten, Arbeitsergebnisse und Verantwortlichkeiten
- Verhindert Abfließen von Prozesswissen
- Beschleunigt die Integration neuer Mitarbeiter

WERKZEUGUNTERSTÜTZUNG

	SCHRITT	AUTOM.	WERKZEUGE
1	Test focus - requirements		DOORS, Polarion, Reqtify, MES Test Manager, TPT, Embedded Tester
2	Back-to-back tests		MES Test Manager, TPT, Embedded Tester
3	Test focus - interfaces		MES Test Manager, TPT, Embedded Tester
4	Check and review models		MES Model Examiner, Model Advisor
5	Test focus - error guessing & static code analysis		MES Test Manager, TPT, Embedded Tester Polyspace, PCLint, Astrée
6	Create design specifications		Enterprise Architect, Matlab SL/SF
7	Check unspecified functionality		-
8	Adjust tool chain		-
9	Test focus - robustness and resources		MES Test Manager, TPT, Embedded Tester
10	Tailor development process		-

Automatic Manual

DER WEG ZUR VOLLSTÄNDIGEN ISO-26262-6 KONFORMITÄT

Noch zu erfüllende Anforderungen	Details	ISO
Anforderungen und Empfehlungen zur Software Architektur	SW safety analysis	Kapitel 6-7
Anforderungen und Empfehlungen zur Spezifikation von Software Safety Requirements	Refinement of technical safety requirements	Kapitel 6-6
Anforderungen und Empfehlungen zur Validierung von Software Safety Requirements	Safety validation	Kapitel 6-11

ZUSAMMENFASSUNG: VORGEHEN

1. Erhebung der aktuellen Entwicklungspraxis
2. Identifikation von ISO-26262-Lücken im Entwicklungsprozess durch Vergleich mit ISO-Referenzprozess
   - Phasen und Aktivitäten
   - Anforderungen an Aktivitäten
   - Arbeitsergebnisse
3. Priorisierung der ISO-26262-Lücken
4. Füllen der Lücken

MES ISO-26262 TRAINING

Modellbasierte Entwicklung eingebetteter Software nach ISO 26262 - Herausforderungen und bewährte Lösungen

- 20.-21. März 2016 bei MES in Berlin
- Expertenwissen zum Thema ISO-26262
- Zielgruppe:
  - Entwickler, Tester, Qualitätsmanager, Projektmanager, Teamleiter

MODEL ENGINEERING SOLUTIONS GMBH

Mauerstraße 79
10117 Berlin
Germany

T: +49 30 2091 6463-0
F: +49 30 2091 6463-33

info@model-engineers.com
www.model-engineers.com

STEP 2: BACK-TO-BACK TESTS

TO DO	Automation	ISO
Back-to-Back test MiL versus SiL versus PiL tests ▪ Compare results of MiL/SiL/PiL tests ▪ Testing on PiL level requires evaluation board		6-9.4.3 T10-1e, 6-10.4.3 T13-1e
Show requirements coverage on **integration** level ▪ Measure coverage of requirements with test specifications ▪ Integration on model level i.e., integration belongs partly to MBSE ▪ Separate requirements for SW components required		6-10.4.5

BENEFIT

- Understand discretization effects (change from floating-point to fixed-point simulation)
- Validate the code generator (comparison MiL against SiL or PiL)
- Assess quality of test specification for integration level

STEP 3: TEST FOCUS - INTERFACES

TO DO	Automation	ISO
Extend the test specification ▪ Interface test ▪ Analysis of boundary values ▪ Generation and analysis of equivalence classes		6-9.4.3 T10-1b, T11 6-10.4.3 T13-1b, T14
Measure structural coverage ▪ Measure function / call coverage		6-10.4.6

BENEFIT

- Detect failures at external and internal interfaces due to incorrectly implemented interfaces or insufficiently analyzed value ranges signals
- Assess quality of test specification on integration level

STEP 5:
TEST FOCUS - ERROR GUESSING & STATIC CODE ANALYSIS

TO DO	Automation	ISO
Extend the test specification ▪ Error guessing		6-9.4.4, T11, 6-10.4.4,T14
Measure structural coverage ▪ Measure MC/DC coverage		6-9.4.5 T12
Analyze run time behavior ▪ Select rules for static code analysis ▪ Perform static code analysis and analyze results		6-8.4.5 T9

BENEFIT

- Detect failures by exploiting know-how and experiences gained in previous projects
- Assess quality of test specification
- Detect specific run time failures which are difficult to find with tests
  e.g. overflow/underflow of arithmetic operations, access to uninitialized variables

STEP 6: CREATE DESIGN SPECIFICATIONS

TO DO	Automation	ISO
Create software architectural design specification ▪ ASIL determination of SW components/units ▪ Estimation of resources (memory, runtime) ▪ Design principles for architecture		6-7.4.1-6-7.4.5 T3
Create SW unit specification ▪ Select notation (Simulink model with additional documentation) ▪ Define guidelines and best practices		6-8.4.2-6-8.4.4 T8

BENEFIT

- Avoid failures resulting from implicit/incorrect assumptions on overall software by using a software architectural design document on which all stakeholders agree
- Improve maintainability of SW units by providing sufficient and consistent documentation

STEP 7: CHECK UNSPECIFIED FUNCTIONALITY

TO DO	Automation	ISO
Check functions included in embedded SW ▪ Measure structural coverage metrics ▪ Show that all specified functions are implemented ▪ Show that deactivated functionality contained in SW has no impact on safety goals		6-10.4.7

BENEFIT

- Detect missing functionality
- Detect configurations that lead to unintendedly activated functionality

STEP 8: ADJUST TOOL CHAIN

TO DO	Automation	ISO
Review, optimize, document, qualify tool chain ▪ Define criteria for tool selection ▪ Integrate tools ▪ Document guidelines on tool usage (e.g. provide information on code generator options, classify options as required / optional / forbidden)		6-5.4.5

BENEFIT

- Reduce amount of manual work
  by using seamless tool chain for modeling, guideline checking, metrics calculation and testing
- Ensure efficient work flow
  by defining appropriate order of activities (e.g. guideline checking before manual review and test)
- Avoid failures
  resulting form incompatible tools, tool versions, or tool configurations within the team and in particular for distributed development

STEP 9: TEST FOCUS - ROBUSTNESS AND RESOURCES

TO DO	Automation	ISO
Extend test specification ▪ resource usage ▪ fault injection		6-9.4.3, T10-1b,1c 6-10.4.3, T13-1b,1c

BENEFIT

- Detect inefficient implementation and insufficient resources
- Detect insufficient or missing failure handling
  memory corruption, transmission failures, improper use