

TESTMETHODEN

Testen in Projekten mit Funktionaler Sicherheit

Die ISO 26262 beschreibt die Aktivitäten, Methoden und Maßnahmen zur Funktionalen Sicherheit für elektrische und elektronische Lösungen in Fahrzeugen bis 3,5 t. Dazu ist der Sicherheitslebenszyklus für das Vorhaben aufzustellen und die einzelnen Schritte vor, während und nach der Entwicklung sind zu planen und umzusetzen. Für die einzelnen Testmethoden müssen die Verantwortlichen der Funktionalen Sicherheit die Übersichtlichkeit und das Verständnis selbst entwickeln und beschreiben. Ansätze dazu bietet der folgende Artikel.

In der Entwicklung werden die Phasen Systementwurf, Hardwareentwicklung und Softwareentwicklung durch ein übersichtliches und verständliches V-Modell beschrieben. Die Anforderungen zum Test sind dabei unterschiedlich auf die verschiedenen Phasen und Stufen des V-Modells verteilt. Die Autoren haben in vielen Beratungsprojekten immer wieder gesehen, wie schwierig es ist, die Anforderungen übersichtlich und verständlich im Zusammenhang zum konkreten Vorhaben zu beschreiben, und so Reviews, Audits und Assessment zur Funktionalen Sicherheit überhaupt erst möglich zu machen.

In den verschiedenen Phasen werden die Testanforderungen auch unterschiedlich interpretiert, dabei sind viele Testanforderungen doch in den einzelnen Phasen wiederkehrend.

Wiederkehrende Testmethoden in der Entwicklung

Die häufig wiederkehrenden Testmethoden werden nachfolgend beschrieben, um eine einheitliche Vorgehensweise in den verschiedenen Phasen der Entwicklung zu unterstützen.

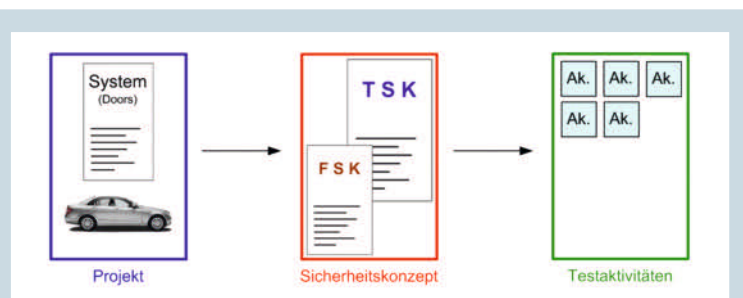


Bild 1: Ableitung von Testaktivitäten in einem Projekt.

Nr.	Methode	Software-Unit	Softwareintegration	HW/SW-Integration	System-Integration	Vehicle-Integration
1	Requirements-basierter Test	Ableitung der Testfälle während der Produktentwicklung der sicherheitsrelevanten Software	Ableitung der Testfälle aus dem Lastenheft der SW oder Beitrag der Software zum FSK/TSK	Ableitung der Testfälle aus dem TSK	Ableitung der Testfälle aus Systemlastenheft oder Beitrag des Systems zum FSK/TSK	Ableitung Testfälle aus FSK/TSK
2	Test of Interfaces (extern, intern)	Test der externen Schnittstellen der Module und damit der Internen der Software	Test der externen Schnittstellen der Software	Interne Schnittstellen (z. B. Hardwaretreiber) und externe z. B. CAN	Interne Schnittstellen (z. B. Datenbus) und externe (HMI oder Datenbus zu anderen Systemen).	Interne Schnittstellen (z. B. Datenbus) und externe (HMI)
3	Back-to-Back	Vergleich zwischen Model und Code	Vergleich zwischen Model und Code	Vergleich Verhaltensmodel (SW/Algorithmen) versus Komponente.	Vergleich Verhaltens Model (SW/ Algorithmen) versus System.	NA
4	Interface Consistency Check	NA	NA	Hier mit Schwerpunkt auf interne Schnittstellen der Funktions- und Basissoftware	Hier Schwerpunkt auf Datenbuskommunikation	NA
5	Generation and Analysis of Equivalence Classes	Erstellung Äquivalenzklassen für die Bereiche der Eingangsgrößen des Moduls	Erstellung Äquivalenzklassen für die Bereiche der Eingangsgrößen der Software	Nicht gefordert, aber mögliche TDT für Interface Tests	Nicht gefordert, aber mögliche TDT für Interface Tests	NA
6	Analysis of Boundary Values	Grenzwertanalyse zu den Äquivalenzklassen der Eingangsgrößen des Moduls	Grenzwertanalyse zu den Äquivalenzklassen der Eingangsgrößen der Software	Nicht gefordert, aber mögliche TDT für Interface Consistency Check	Nicht gefordert, aber mögliche TDT für Interface Consistency Check	NA
7	Control Flow Analysis	Review des Control Flow Graph	Review des Control Flow Graph	NA	NA	NA
8	Fault Injection Tests	NA	Stimulation Fehlererkennungsmechanismen	Elektrische Fehler an PINs, eventuell Manipulation Prozessor oder Datenspeicher. Stimulation Fehlererkennungsmechanismen	Elektrische Fehler, Störung Datenbus, Stimulation Fehlererkennungsmechanismen	Elektrische Fehler, Störung Datenbus, Stimulation Fehlererkennungsmechanismen
9	Error Guessing	NA	Fokus Software	Fokus Komponente	Fokus (EE) System	Fokus Fahrzeug

Tabelle 1: Beschreibung wiederkehrende Testmethoden in den einzelnen Phasen.

Requirementsbasierte Tests

Eine grundlegende Forderung der ISO und jedes guten Testprojektes ist es, Testfälle von den Requirements abzuleiten und auf eine durchgängige Verfolgbarkeit zwischen Testfällen und Requirements zu achten. Das ermöglicht nicht nur auf eine einfache Art die Testabdeckung zu messen, sondern bietet auch die Möglichkeit die Tester bereits mit der Fertigstellung der Sicherheitsanforderungen im FSK und TSK in das Projekt einzubinden. Beim Review der Sicherheitsanforderungen und bei der Ableitung von Testfällen werden Unstimmigkeiten und Qualitätsmängel der Anforderungen früh aufgedeckt.

Schnittstellentest

Eine weitere wichtige Testart ist der Schnittstellentest. Dieser wird in der ISO für verschiedene Teststufen in der

Ausprägung eines externen oder internen Schnittstellen Tests gefordert. Im Idealfall liegt eine Schnittstellenspezifikation im TSK vor, aus der sich die Testfälle ableiten lassen. Typischerweise erstellt man für jeden Schnittstellenparameter mindestens einen Testfall. Durch paarweise Kombination von Parametern lässt sich die Testintensität erhöhen. Führt man eine Grenzwertanalyse durch, so lässt sich ein Schnittstellen Test zum „Interface Consistency Test“ erweitern.

Back-to-Back-Test

Die grundlegende Idee des Back-to-Back-Tests ist es, das Verhalten verschiedener Repräsentationsformen des Testobjektes mit identischen Tests zu überprüfen. Dabei wird ein identisches Verhalten am Ausgang erwartet. Ziel ist dabei der Nachweis der (partiellen) Äquivalenz. Damit lässt

Methods and measures		SW-Unit				SW-Integration				SW/HW-Integration				System-Integration				Vehicle-Integration				
		A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	
1	Requirements-based tests _a	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++
2	Back-to-back tests _b	+	+	++	++	+	+	++	++	+	+	++	++	0	+	+	++					
3	Tests of external interfaces _c	++	++	++	++	+	+	++	++	+	+	++	++	+	++	++	++					
4	Interface consistency check _d					see SW/HW-Integration				+	++	++	++	+	++	++	++					
5	Tests of internal interfaces _e					+	++	++	++	+	++	++	++	+	++	++	++	0	+	+	+	+
6	Communication tests _r					Function coverage								++	++	++	++					
7	Tests of interaction/communication _g					Call coverage								++	++	++	++	++	++	++	++	++
8	Fault injection tests _r	+	+	+	+	+	+	++	++	+	+	++	++	+	+	++	++	+	+	++	++	++
9	Error guessing tests _g	+	+	+	+	+	+	++	++	+	+	++	++	+	+	++	++	+	+	++	++	++
10	Tests derived from field experience _j													0	+	+	++	+	+	++	++	++
11	Long term tests as user tests under real life conditions _g																	+	+	++	++	++
12	Resource usage tests _h	+	+	+	++	+	+	+	++	+	+	+	++	0	+	+	++	0	0	+	+	+
13	Performance tests _i									+	+	+	++	0	+	+	++	0	0	+	+	++
14	Stress tests _j									+	+	+	++	0	+	+	++	0	0	+	+	++
15	Tests for interference resistance/robustness and under certain environmental conditions _n													++	++	++	++	+	+	+	+	++

Tabelle 2: Einsatz wiederkehrender Testmethoden bei unterschiedlichem ASIL.

© automotive

sich die korrekte Umsetzung einer Spezifikation überprüfen. Voraussetzung ist die Unabhängigkeit der Programmiererteams für die verschiedenen Repräsentationen. In der Automobilentwicklung ist diese Voraussetzung jedoch sehr selten erfüllt. Es können jedoch Back-to-Back-Tests zwischen Model und daraus erzeugtem Code durchgeführt werden um die korrekte Funktion des Compilers zu überprüfen, oder man kann einen Vergleich zwischen reinen Softwaretests und Test der Software auf der Zielhardware durchführen.

Error Guessing

Eine wichtige Ergänzung nicht nur für sicherheitsrelevante Funktionen mit hohen ASIL-Stufen, sondern für jedes Testprojekt stellt das Error Guessing dar. Dieses ergänzt die strikte Ableitung von requirementsbasierten Tests um die Erfahrung der Tester und andere Projektbeteiligter. Es empfiehlt sich, Error Guessing in einem oder mehreren forma-

len Workshops durchzuführen. Punkte, die in den Requirements nicht bedacht wurden, oder Schwachstellen die nicht auf den ersten Blick ersichtlich sind, können in diesen Workshops gefunden werden.

Fault Injection

Häufig liefert das Error Guessing Testfälle, die das Einschleusen von Fehlern in das Testobjekt (Fault Injection) notwendig macht. Abhängig von der Sicherheitsintegrität wird diese Art von Tests in der ISO 26262 für verschiedene Teststufen gefordert. Je nach Art des einzuschleusenden Fehlers wird der Aufwand eine entsprechende Schnittstelle in der Software, der Hardware oder dem System zu haben, relativ groß. Außerdem muss darauf geachtet werden, dass die Vergleichbarkeit des Testobjekte mit Fault Injection zum später real im Fahrzeug verwendeten Objekt nicht verloren geht. Oft führt aber insbesondere beim Testen von Sicherheitsanforderungen kein Weg an einer entsprechenden Schnittstelle oder einer geeigneten Manipulation einer Simulationsumgebung vorbei.

Testdesign-Techniken

Neben den Testmethoden schreibt die ISO 26262 auch Testdesign-Techniken zur Ableitung der Testfälle vor.

Äquivalenzklassen und Grenzwertanalyse

Die am häufigsten geforderte Testdesign-Technik und gleichzeitig die, die die geringsten Anforderungen an die Form der Requirements stellt, ist die Bildung von Äquivalenzklassen für die Eingangsgrößen. Ausgangspunkt ist die Annahme von gleichem Verhalten des Testobjektes für Werte innerhalb einer Äquivalenzklasse. Aus einem erfolgrei-

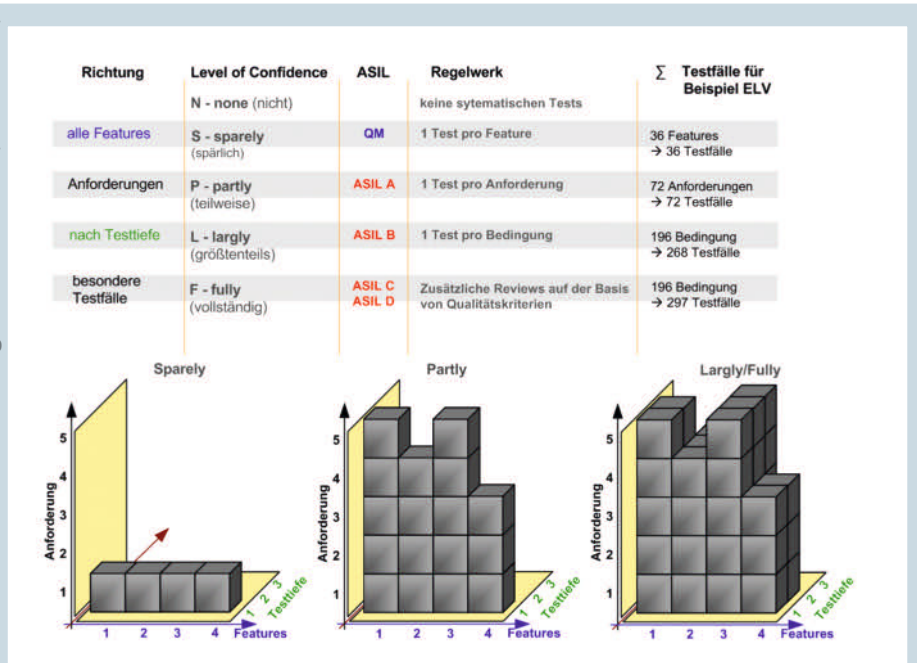


Bild 2: Level of Confidence.

© automotive

Abarbeitung von Testfällen in einem Projekt mit QM, ASIL B, ASIL C und ASIL D

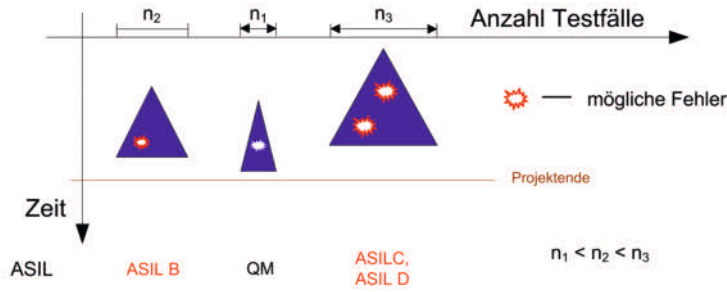


Bild 3: Risiko der Testobjekte bestimmt die Testintensität.

© automotive

ungen als C2 werden in der Praxis kaum verwendet. Eventuell findet auch nur ein Review des Control Flow statt.

Abhängigkeit der Testmethoden von der Integrität

Die im vorigen Abschnitt beschriebenen Testmethoden werden für die Entwicklung von Fahrzeugen schon seit langer Zeit angewendet, auch in Vorhaben ohne Funktionale Sicherheit. Die ISO 26262 nimmt diese Testmethoden auf und stellt zusätzlich in einen Zusammenhang zur geforderten Sicherheitsintegrität her, wie **Tabelle 2** beschreibt.

chen Test für einen Wert dieser Klasse folgt, dass vermutlich alle weiteren Tests mit Werten dieser Klasse ebenfalls erfolgreich sind. Die Testfälle werden durch Kombination der Eingangsgrößen erstellt, wobei darauf geachtet werden muss, welche Randbedingungen das Verhalten des Testobjektes beeinflussen können.

Da es die Erfahrung lehrt, dass sich Fehler an Grenzen von Wertebereichen häufen, kann es sehr lohnend sein, die Äquivalenzklassen um eine Grenzwertanalyse zu ergänzen und nicht nur einen Wert pro Klasse für die Erstellung von Testfällen zu verwenden, sondern einen Wert knapp unterhalb der Grenze, ein Wert auf der Grenze und einen Wert knapp oberhalb der Grenzen des Wertebereiches. Dadurch steigt zwar die Anzahl der Testfälle, aber Fehler durch falsche Operatoren („≥“ anstelle von „>“) und Überläufe von Variablen bei Falschbelegung (Wert „8“ auf eine 3-bit-Größe) lassen sich dadurch aufspüren.

Control Flow Analysis

Der Strukturtest ist eine typische Methode zur Ableitung von White-Box-Tests, da Requirements für Software am häufigsten in Form eines Kontrollflusses vorliegen. Bei der Ermittlung der Testfälle können verschiedene Überdeckungen des Kontrollflusses gewählt werden.

Typische Abdeckungsmaße sind dabei die C1-Abdeckung (jede Aktion wird einmal ausgeführt) und die C2-Abdeckung (jeder Pfad wird einmal ausgeführt). Höhere Abdeck-

Grundsätzlich ändert sich eine Testmethode nicht, wenn diese in Projekten mit Funktionaler Sicherheit zur Anwendung kommt. Die Tabelle aber stellt den Zusammenhang zwischen einer Testmethode und einer geforderten Sicherheitsintegrität dar. Die ISO 26262 beschreibt zwar diesen Zusammenhang nicht näher, man kann aber das Fehlermodell benutzen, wie es für die Hardware im Band 5 der ISO 26262, Tabelle D.1, zur Anwendung kommt. Diesem Fehlermodell folgend sollte eine Testmethode bei höherer Sicherheitsintegrität ein umfangreicheres Fehlermodell berücksichtigen als bei niedriger Sicherheitsintegrität. Dementsprechend sind für niedrigere Sicherheitsintegrität schon Testmethoden ausreichend, die Kurzschluss und Unterbrechung erkennen, während bei höherer Sicherheitsintegrität auch Drift und Offset erkannt werden müssen.

Teststrategie

Die Kontrolle wiederkehrender Testmethoden in den unterschiedlichen Phasen der Entwicklung ist ein praktisches Problem. Tests in den Phasen des Systementwurfs, der Hardwareentwicklung und der Softwareentwicklung werden von unterschiedlichen Teams mit unterschiedlicher Ausbildung und Ausstattung durchgeführt. Für die Phasen Hardwareentwicklung und Softwareentwicklung existieren sehr umfangreiche einzelne Testfallspezifikationen, auf

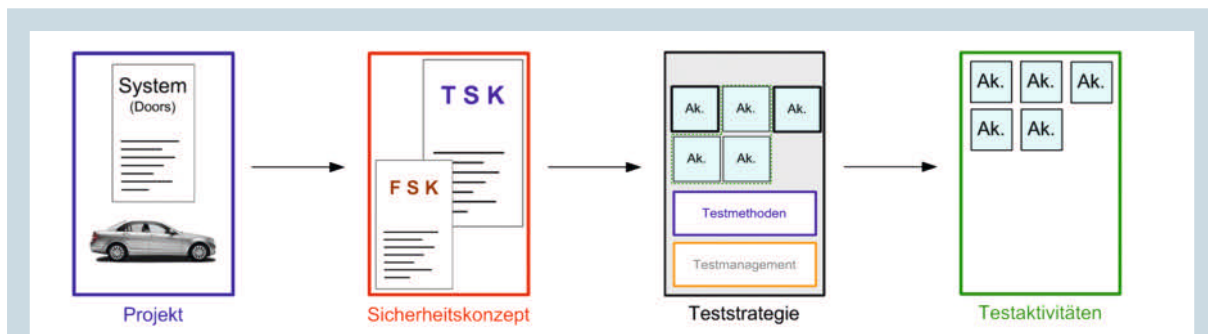


Bild 4: Einsatz einer Teststrategie in einem Projekt.

© automotive

Systemebene häufig Testspezifikationen mit dem Schwerpunkt auf der Kundenlebbbarkeit des Vorhabens. Wenn diese Testfallspezifikationen modifiziert und erweitert werden, um den Anforderungen der Funktionalen Sicherheit zu genügen, entsteht nicht automatisch ein schlüssiges Gesamtkonzept zum Test.

Ein konzeptionelles Vorgehen wird für die Testfallspezifikation auch nicht so gefordert, wie es für das Funktionale Sicherheitskonzept und das Technische Sicherheitskonzept schon umgesetzt wird. Die Autoren haben viele gute Erfahrungen mit der Anwendung von Teststrategien in Projekten mit Funktionaler Sicherheit gesammelt. Die Vorteile sind:

- Zusammenfassende Darstellung aller ausgewählten Testmethoden,
- Zusammenfassende Begründung für die ausgewählten Testmethoden,
- Einheitliche Testprinzipien für alle Phasen der Entwicklung,
- Einheitliche Vorgehensweisen für wiederkehrende Testmethoden,
- Einheitliche Templates, Dokumente und Reviews,
- Festlegung aller Managementaktivitäten zum Test,
- Mögliche Anwendung einer Priorisierung der Testaktivitäten,
- Beschreiben des Zusammenhanges zwischen Sicherheitsdiagnosen und Test.

Für die Teststrategie kann die Testintensität in Projekten mit Funktionaler Sicherheit durch den Level of Confidence beschrieben werden, wie **Bild 2** zeigt. Die Managementaktivitäten zum Test können durch die Anwendung des Levels of Confidence unterstützt werden. Mit dem Level of Confidence wird die notwendige und erreichte Testtiefe anschaulich abgeleitet und dargestellt. Die notwendige Testtiefe kann direkt aus der geforderten Sicherheitsintegrität abgeleitet werden, wie **Bild 3** zeigt.

Die Anwendung des Level of Confidence wird von den Autoren für Projekte mit Funktionaler Sicherheit bei höherer Sicherheitsintegrität unbedingt empfohlen.

Neben dem Test in den Phasen der Systementwicklung, Hardwareentwicklung und Softwareentwicklung wird die Funktionale Sicherheit des Vorhabens nach dem Start der Serienproduktion beim Einsatz des Vorhabens im Fahrzeug (während der Fahrt) überprüft. In dieser Phase des Sicherheitslebenszyklus haben die Sicherheitsdiagnosen eine hohe Bedeutung. Ihre Wirkung kann bereits am Ende der Produktionslinie beginnen. Abhängig von der gewählten Strategie zur Funktionalen Sicherheit können Sicherheitsdiagnosen mit Testmethoden übereinstimmen. Tests, die während der Entwicklung durchgeführt werden sollen, können durch die erforderlichen Sicherheitsdiagnosen bereits abgedeckt werden. Die Sicherheitsdiagnosen sind im FSK oder TSK zu beschreiben. In der Teststrategie kann die entsprechende Anwendung für Testfälle festgelegt werden. (oe)



Dr. Hartmut Paulus ist Senior Test Consultant bei der MBtech Group und beschäftigt sich hauptsächlich mit der Definition und Verbesserung von Testprozessen sowie der Erstellung von Teststrategien für Projekte in der Automotive Branche.



Dr. Frank Eimbeck ist Senior Consultant Functional Safety bei der MBtech Group und u. a. für die Erstellung von Sicherheitskonzepten, Safety Management und SW Projektleitung für verschiedene Entwicklungsprojekte zuständig.

NI VeriStand

Echtzeitprüfumgebung und Simulationssoftware



Offene, konfigurationsbasierte Softwareumgebung zur Erstellung von Echtzeitprüfapplikationen

- Echtzeitprüfsysteme per Mausclick konfigurieren und implementieren
- I/O von NI und Drittherstellern nahtlos integrieren
- Simulationsmodelle aus NI LabVIEW und anderen Modellierungsumgebungen importieren
- Benutzeroberflächen zur Laufzeit erstellen und anpassen
- Das Beste aus beiden Welten: Entwicklungseffizienz und Flexibilität

>> **Kostenfreie Testversion und Demovideos unter:**

ni.com/veristand/d

089 7413130



National Instruments Germany GmbH
Ganghoferstraße 70 b • 80339 München
Tel.: +49 89 7413130 • Fax: +49 89 7146035
ni.com/germany • info.germany@ni.com