Certified Automotive Software Tester

Version 2017 (2.0) vom 31.03.17

German Testing Board e.V.

Deutschsprachige Ausgabe
Herausgegeben durch German Testing Board e.V.

Urheberrecht © 2014, German Testing Board e.V. (GTB)

Eingetragene Marken

Änderungsübersicht

Version	Datum:	Bemerkung
1.0	19.01.2011	Autor: Dr. Hendrik Dettmering, entwickelt im Auftrag gasq GmbH Die Nutzungsrechte wurden vollständig übertragen an German Testing Board e.V
1.1	14.06.2015	Inhaltlicher Überarbeitung und Abgleich mit dem deutschsprachigen ISTQB Certified Tester Foundation Level Lehrplan 2011 V1.0.1 und dem ISTQB Glossar V2.2 Freigabe gem. GTB Arbeitsgruppenmeeting vom 15.03.2015 (München)
2.0	31.03.2017	Lernziele und Inhalte in Anlehnung an Version 1.1 neu erstellt. Freigabe gem. GTB Arbeitsgruppenmeeting vom 31.03.2017 (Frankfurt am Main)

Inhaltsverzeichnis

Dank

Das German Testing Board (GTB), dankt dem Autoren- und Review-Team der deutschsprachigen Fassung 2017, V2.0 (in alphabetischer Reihenfolge):

Graham Bath, André Baumann, Arne Becher, Ralf Bongard (Leitung), Kai Borgeest, Tim Burdach, Mirko Conrad, Klaudia Dussa-Zieger, Matthias Friedrich, Dirk Gebrath, Thorsten Geiselhart, Matthias Hamburg, Uwe Hehn, Olaf Janßen, Jacques Kamga, Horst Pohlmann, Ralf Reißing, Karsten Richter, Ina Schieferdecker, Alexander Schulz, Stefan Stefan, Stephanie Ulrich, Jork Warnecke, Stephan Weißleder.

Zur Geschichte dieses Dokuments

Der Lehrplan 1.0 wurde 2010/2011 im Auftrag des Global Association for Software Quality AISBL (gasq) von Dr. Hendrik Dettmering entwickelt.

Zum Review des Dokuments wurden ausgewählte Experten deutscher OEMs berufen, durch die die Qualität und die Zielsetzung des Lehrplans geprüft und für geeignet bewertet wurden. Damit stellt dieses Dokument den Lehrplan für die Zertifizierung zum Certified Automotive Software Tester dar und ist gleichermaßen die Basis für Schulungsunterlagen als auch für Prüfungsfragen zur Zertifizierung.

Mit dem 01.01.2014 übernimmt die Arbeitsgruppe „Certified Automotive Software Tester“ des German Testing Board (GTB) die Weiterentwicklung des Lehrplans, um der schnellen Entwicklung der Thematik folgen zu können und dem Bedürfnis der Industrie zu entsprechen neben dem branchenunabhängigen CORE-Lehrplan auch die automobilspezifischen Aspekte als Ergänzung zum bewährten ISTQB Foundation Level zur Verfügung zu haben.

Die zum 15.06.2015 veröffentlichte Version 1.1. des Lehrplan war abwärtskompatibel mit der Version 1.0; wobei die redundanten Anteile zum ISTQB Foundation Level 2011 Lehrplan aus der Version 1.1 entfernt wurden.

Einführung[1]

Zweck des Dokuments

Dieser Lehrplan definiert eine Ergänzung zur Basisstufe (Specialist to Foundation Level CORE) des Softwaretestausbildungsprogramms des International Software Testing Qualifications Board (im Folgenden kurz ISTQB® genannt). An Hand des vorliegenden Lehrplans erstellen Ausbildungsanbieter ihre Kursunterlagen und legen eine angemessene Unterrichtsmethodik für die Akkreditierung fest. Die Lernenden bereiten sich anhand des Lehrplans auf die Prüfung vor.

Weitere Informationen über Geschichte und Hintergrund des vorliegenden Lehrplans sind im Anhang A dieses Lehrplans aufgeführt.

ISTQB® Certified Tester, Foundation Level, Automotive Specialist

Die vorliegende Ergänzung zur Basisstufe des Certified Tester Ausbildungsprogramms soll alle in das Thema Softwaretesten involvierten Personen in der Automobilbranche ansprechen. Das schließt Personen in Rollen wie Tester, Testanalysten, Testingenieure, Testberater, Testmanager, Abnahmetester und Softwareentwickler mit ein. Die Basisstufe richtet sich ebenso an Personen in der Rolle Projektleiter, Qualitätsmanager, Softwareentwicklungsmanager, Systemanalytiker (Business Analysten), IT-Leiter oder Managementberater, welche sich ein Basiswissen und Grundlagenverständnis über das Thema Softwaretesten in der Automobilbranche erwerben möchten.

[1] Text wurde in weiten Teilen aus dem ISTQB® CTFL Lehrplan [2] übernommen.

Geschäftlicher Nutzen

In diesem Abschnitt wird der geschäftliche Nutzen (Business Outcomes nach ISTQB®) aufgelistet, den man von Kandidaten mit einer zusätzlichen Zertifizierung als Certified Automotive Software Tester erwarten kann.

Ein Certified Automotive Software Tester kann im Rahmen des Testens von Elektrik/Elektronik[2]-Systemen im automobilen Umfeld …

AUTFL-BO-01 …in einem Testteam effektiv mitarbeiten. („collaborate“)

AUTFL-BO-02 …die aus dem ISTQB Certified Tester Foundation Level (CTFL®) bekannten Testverfahren an die spezifischen Projektbedingungen anpassen. („adapt“)

AUTFL-BO-03 …bei der Auswahl von angemessenen Testverfahren die grundlegenden Anforderungen der relevanten Normen und Standards (Automotive SPICE®, ISO 26262, etc.) berücksichtigen. („select“)

AUTFL-BO-04 …das Testteam bei der risikoorientierten Planung der Testaktivitäten unterstützen und dabei bekannte Elemente der Strukturierung und Priorisierung anwenden. („support & apply“)

AUTFL-BO-05 …die virtuellen Testmethoden (zum Beispiel HiL, SiL, MiL, etc.) in Testumgebungen anwenden. („apply“)

Lernziele/Kognitive Stufen des Wissens

Jeder Abschnitt dieses Lehrplans ist einer kognitiven Stufe zugeordnet:

- K1: sich erinnern
- K2: verstehen
- K3: anwenden
- K4: analysieren

Die Lernziele legen fest, was Sie nach Beenden des jeweiligen Abschnitt/Kapitels/Moduls gelernt haben sollten.

Die Inhalte für als [informativ] gekennzeichnete Lernziele muss der Trainingsprovider in angemessener Zeit vermitteln, sind allerdings NICHT prüfungsrelevant.

Beispiel:

AUTFL-2.2.3.1 Sie können Aufbau und Struktur der ISO 26262 wiedergeben. [informativ]

Begriffe

Alle Begriffe, die im Absatz direkt unter der Überschrift unter „Begriffe“ genannt werden, sollen wiedergegeben werden können (K1), auch wenn das in den Lernzielen nicht explizit angegeben ist. Es gelten die Definitionen des ISTQB Glossars bzw. der nationalen Übersetzung in der jeweils freigegebenen Fassung (inkl. der zusätzlichen Begriffe aus dem vorliegenden Lehrplan).

[2] Für den Begriff „Elektrik/Elektronik“ verwendet dieser Lehrplan im Weiteren das Akronym „E/E“.

Die Prüfung

Auf diesem Lehrplan basiert eine zusätzliche Prüfung für das domänenspezifische Zertifikat CTFL Specialist „Certified Automotive Software Tester“. Eine Prüfungsfrage kann Stoff aus mehreren Kapiteln des Lehrplans abfragen. In der Regel ist jede Prüfungsfrage einem Lernziel zugeordnet mit der Ausnahme der Fragen, die einem Schlüsselbegriff zugeordnet sind. Das Format der Prüfung ist Multiple Choice. Prüfungen können unmittelbar im Anschluss an einen akkreditierten Ausbildungslehrgang oder Kurs, aber auch unabhängig davon (zum Beispiel in einem Prüfzentrum oder einer öffentlich zugänglichen Prüfung) abgelegt werden. Die Teilnahme an einem Kurs stellt keine Voraussetzung für das Ablegen der Prüfung dar. Die vom German Testing Board zugelassenen Prüfungsanbieter sind die für das Zertifizierungsmodul Certified Automotive Software Tester aufgelisteten Prüfungsanbieter.

Voraussetzungen für die Prüfung

Voraussetzung für die Prüfung zum Certified Automotive Software Tester ist das erworbene Zertifikat zum ISTQB Certified Tester Foundation Level (CTFL®) und Interesse der Kandidaten am Testen im Rahmen von automobilen E/E-Entwicklungsprojekten.

Es empfiehlt sich allerdings für die Kandidaten,

- zumindest ein minimales Hintergrundwissen im Bereich Softwareentwicklung oder Softwaretest zu haben (zum Beispiel sechs Monate Erfahrung als System- oder Abnahmetester oder als Entwickler)
- oder einen Kurs besucht zu haben, der nach dem ISTQB® Standard (durch ein ISTQB®-Mitglieds-Board) akkreditiert ist und/oder
- erste Erfahrungen im Testen in E/E- Entwicklungsprojekten in der automobilen Branche gesammelt zu haben.

Akkreditierung

Das German Testing Board akkreditiert Ausbildungsanbieter, deren Ausbildungsunterlagen entsprechend diesem Lehrplan aufgebaut sind. Die Akkreditierungsrichtlinien sind bei diesem nationalen Board erhältlich. Ein akkreditierter Kurs ist als zu diesem Lehrplan konform anerkannt und darf als Bestandteil eine Zusatz-Prüfung enthalten.

Weitere Hinweise für Ausbildungsanbieter sind im Anhang enthalten.

Detaillierungsgrad

Der Detaillierungsgrad dieses Lehrplans erlaubt konsistentes Lehren und Prüfen. Um dieses Ziel zu erreichen, enthält dieser Lehrplan Folgendes:

- allgemeine Lernziele, welche die Intention der (erweiterten) Basisstufe beschreiben
- Inhalte, die zu lehren sind, mit einer Beschreibung und, wo notwendig, Referenzen zu weiterführender Literatur
- Lernziele für jeden Wissensbereich, welche das beobachtbare kognitive Ergebnis der Schulung und die zu erzielende Einstellung des Teilnehmers beschreiben
- eine Liste von Begriffen, welche der Teilnehmer wiedergeben und verstehen soll
- eine Beschreibung der wichtigen zu lehrenden Konzepte, einschließlich Quellen wie anerkannte Fachliteratur, Normen und Standards

Der Lehrplan ist keine vollständige Beschreibung des Wissensgebiets „Testen für softwarebestimmte Systeme in automobilen Elektronikentwicklungsprojekten“. Er reflektiert lediglich den nötigen Umfang und Detaillierungsgrad, welcher für die Lernziele relevant ist.

Lehrplanaufbau

Der Lehrplan besteht aus vier Hauptkapiteln. Jeder Haupttitel eines Kapitels zeigt die anspruchsvollste Lernzielkategorie/höchste kognitive Stufe, welche mit dem jeweiligen Kapitel abgedeckt werden soll und legt die Unterrichtszeit fest, welche in einem akkreditierten Kurs mindestens für dieses Kapitel aufgewendet werden muss.

Beispiel:

Einleitung (K2) - [30 Minuten]

Das Beispiel zeigt, dass in Kapitel „Einleitung (K2)“ K1[3] und K2 (aber nicht K3) erwartet werden und 30 Minuten für das Lehren des Materials in diesem Kapitel vorgesehen sind.

Jedes Kapitel enthält eine Anzahl von Unterkapiteln. Jedes Unterkapitel kann wiederum Lernziele und einen Zeitrahmen vorgeben. Wird bei einem Unterkapitel keine Zeit angegeben, so ist diese im Oberkapitel bereits enthalten.

Geschlechtsneutrale Formulierung

Aus Gründen der einfacheren Lesbarkeit wird auf die geschlechtsneutrale Differenzierung, zum Beispiel Benutzer/innen, verzichtet. Sämtliche Rollenbezeichnungen gelten im Sinne der Gleichbehandlung grundsätzlich für beide Geschlechter.

[3] ein Lernziel einer höheren Taxonomie Stufe impliziert die Lernziele der tieferen Stufen.

1 Einleitung (K2) [30 Min]

Begriffe

Systemlebenszyklus, Freigabe, Freigabeobjekt

Lernziele

AUTFL-1.1.1 Sie können die Herausforderungen einer automobilen Produktentwicklung, die sich aus gegenläufigen Projektzielen und steigender Produktkomplexität ergeben, anhand von Beispielen erläutern. (K2)

AUTFL-1.2.1 Sie können durch Normen und Standards beeinflusste Projektaspekte wie Zeit, Kosten, Qualität und Projekt-/Produktrisiken wiedergeben. (K1)

AUTFL-1.3.1 Sie können die sechs generischen Phasen im Systemlebenszyklus nach ISO/IEC TR 24748-1:2010 [1] benennen. (K1)

AUTFL-1.4.1 Sie können den Beitrag und die Mitwirkung des Testers am Freigabe-Prozess wiedergeben. (K1)

Einführung

Einer der sieben Grundsätze des Softwaretestens lautet: „Testen ist abhängig vom Umfeld“ [2]. Dieser Abschnitt skizziert das Umfeld der E/E-Entwicklung, in dem sich ein Certified Automotive Software Tester[4] bewegt. Zum einen ergeben sich aus gegenläufigen Zielen, steigender Komplexität und hohem Innovationsdruck besondere Herausforderungen. Zum anderen bilden Normen, Standards und der Lebenszyklus von Fahrzeugen den Rahmen, in dem er arbeitet. Am Ende trägt er mit seiner Arbeit zur Freigabe von Software und Systemen bei.

1.1 Anforderungen aus gegenläufigen Projektzielen und steigender Produktkomplexität (K2) [15 Min]

Hersteller und Zulieferer bringen immer mehr Fahrzeugmodelle[5] immer schneller und unter hohem Kostendruck auf den Markt. Dabei spielen folgende Aspekte eine Rolle:

- Steigende Modellzahl & Komplexität:
  Um individuelle Kundenbedürfnisse besser erfüllen zu können, bieten die Hersteller immer mehr Fahrzeugmodelle an. Dadurch sinken aber die Stückzahlen pro Modell. Um die somit steigenden Entwicklungs- und Produktionskosten zu kompensieren, realisieren die Hersteller mehrere Modelle als Varianten einer gemeinsamen Plattform. Plattformentwicklung ist aber wegen der nötigen Beherrschung der Varianz deutlich komplexer als die Entwicklung eines einzelnen Modells.
- Steigender Umfang der Funktionalität:
  Der Kunde fordert immer mehr Innovationen, ohne auf vorhandene Funktionen verzichten zu wollen, weshalb der Funktionsumfang zunimmt.
- Steigende Anzahl der Konfigurationen:

[4] Im Weiteren wird nur der Begriff „Tester“ verwendet. Er ist als Kurzform des „Certified Automotive Software Tester“ zu verstehen.

[5] Beispielhaft aus einer Studie der Unternehmensberatung Progenium: "1990 waren in Deutschland gerade einmal 101 verschiedene Fahrzeugmodelle im Angebot …, im Jahr 2014 waren es schon 453 Stück." [43]

- Der Kunde will sein Fahrzeugmodell individuellen Wünschen anpassen. Das erfordert viele mögliche Konfigurationen für ein Fahrzeugmodell, auch im Funktionsumfang.
- Höhere Qualitätsanforderungen
  Trotz zunehmender Funktionalität bei gestiegener Komplexität erwartet der Kunde die gleiche oder sogar höhere Qualität des Fahrzeugs und seiner Funktionen.

Da die Projektziele Zeit, Kosten und Qualität konkurrieren („magisches Dreieck"), müssen Hersteller und Zulieferer eine effizientere Systementwicklung anstreben, die bei zunehmender Komplexität, steigenden Qualitätsanforderungen und kleineren Budgets dennoch kürzere Entwicklungszeiten erlaubt.

1.2 Durch Normen und Standards beeinflusste Projektaspekte (K1) [5 Min]

Normen und Standards beeinflussen maßgebliche Projektaspekte wie Zeit, Kosten, Qualität, Projekt- und Produktrisiken:

- Sie steigern die Effizienz der Prozesse (um zum Beispiel bei gleichbleibender Qualität die Entwicklungszeit bzw. -kosten zu reduzieren) durch:
  - einheitliche Nomenklaturen
  - bessere Transparenz
  - einfachere Zusammenarbeit (intern und extern)
  - höhere Wiederverwendbarkeit
  - konsolidierte Erfahrungen („Best Practice")
- Sie helfen durch anerkannte Regeln der Technik [2], Risiken und Mängel früh zu erkennen und abzustellen.
- Sie sind die Basis für Audits. Dadurch kann ein Gutachter (Auditor) die Qualität eines Produktes oder Prozesses bewerten. Zugleich kann er prüfen, ob diese die geforderten Vorgaben erfüllen [1].
- Sie sind Teil vertraglicher oder regulatorischer Vorschriften und Vorgaben.

Dieser Lehrplan betrachtet unter anderem folgende Normen und Standards:

- Standards und Normen, wie ISO 29119 [3] oder ASPICE [4], die Prozesse und Methoden standardisieren.
- Standards und Normen, wie AUTOSAR [5], die Produkte standardisieren.

1.3 Die sechs generischen Phasen im Systemlebenszyklus (K1) [5 Min]

Der Systemlebenszyklus eines PKW Modells und aller darin verbauten Komponenten[6] beginnt mit der Produktidee und endet mit der Außerbetriebnahme. Über diese Zeit sind neben den Prozessen der Entwicklung auch betriebswirtschaftliche, logistische und produktionstechnische Prozesse beteiligt. Für reibungsfreie Abläufe sorgen hier Meilensteine mit zuvor definierten Eingangs- bzw. Endekriterien. Diese unterteilen und synchronisieren den Systemlebenszyklus[7] in sechs Phasen [1]. In Klammern stehen die typischen Test-Aktivitäten[8]:

[6] Steuergeräte (Hardware und Software) sowie Software Komponenten.
[7] Der Sicherheitslebenszyklus der ISO 26262 durchläuft vergleichbare Phasen [22].
[8] Test-Aktivitäten siehe auch: fundamentaler Testprozess [2].

- Konzeption (Testplanung und Testentwurf)
- Entwicklung (Testrealisierung mit Durchführung, Analyse und Bericht)
- Produktion (Band-Ende-Test)
- Betrieb (keine Testaktivitäten)
- Wartung (Wartungstest)
- Außerbetriebnahme (Migrationstest)

Der in der Automobilindustrie verbreitete Produktentstehungsprozess (PEP)[9] betrachtet: Konzeption, Entwicklung und Produktion.

1.4 Der Beitrag / die Mitwirkung des Testers am Freigabeprozess (K1) [5 Min]

Im automobilen Umfeld erreicht ein Projekt durch Aussprechen einer Freigabe (engl. Release) ein Ziel oder einen Meilenstein. Ab diesem Zeitpunkt erfüllt das Freigabeobjekt (engl. Release Item) den für den Einsatz und Zweck benötigten Reifegrad.

Der Freigabeprozess (engl. Release Process) soll zur Freigabe des Freigabeobjekts führen. Dieses besteht aus dem Testobjekt (Softwarekonfiguration inklusive Parametrierung, gegebenenfalls auch mit Hardware und Mechanik) und der dazu gehörigen Begleitdokumentation.

Über den Testabschlussbericht liefert der Tester wichtige Informationen für den Freigabeprozess [3]:

- getestete Testobjekte und Leistungsmerkmale inklusive ihrer Version
- bekannte Fehler
- Produktmetriken
- Freigabeempfehlung (bei Erreichen der Testendekriterien) auf Basis der zugrunde gelegten Freigabebestimmung (sprich: Erprobung auf Gelände oder auf öffentlicher Straße, Verbauempfehlung)

Darüber hinaus wirkt der Tester an weiteren, für die Freigabe relevanten Arbeitsergebnissen mit [6]:

- Priorisieren und Mitentscheiden über Änderungen
- Priorisieren von Funktionen (für Implementierungsreihenfolge)

[9] PEP: Ist nur in der überholten „VDA-Empfehlung 4, Teil 3 (1998)“ zur Projektplanung definiert, dennoch ist der PEP heute (immer) noch im automobilen Umfeld etabliert.

2 Normen und Standards für das Testen von E/E-Systemen (K3) [300 Min]

Begriffe

Automotive SPICE (ASPICE)

Prozessverbesserung, Automotive SPICE (ASPICE), Stufendarstellung, Rückverfolgbarkeit, Prozessmodell, Referenzprozesse, Teststrategie, Regressionsteststrategie, Testdokumentation, Verifikationsstrategie, Kriterien zur Verifikation, Softwarequalifikationstest, Systemqualifikationstest

ISO 26262

ASIL, funktionale Sicherheit, Methodentabelle, Sicherheitslebenszyklus

AUTOSAR

AUTOSAR, Integration

Gegenüberstellung

Multisystemtest, Systemintegrationstest

Lernziele

Automotive SPICE (ASPICE)

AUTFL-2.1.1.1 Sie können die zwei Dimensionen von ASPICE benennen. (K1)

AUTFL-2.1.1.2 Sie können die 3 Prozesskategorien und 8 Prozessgruppen von ASPICE benennen [informativ]. (K1)

AUTFL-2.1.1.3 Sie können die Fähigkeitsstufen 0 bis 3 von ASPICE erläutern. (K2)

AUTFL-2.1.2.1 Sie können den Zweck der 5 testrelevanten Prozesse von ASPICE benennen. (K1)

AUTFL-2.1.2.2 Sie können die Bedeutung der vier Bewertungsstufen und der Fähigkeitsindikatoren von ASPICE aus der Testperspektive erklären. (K2)

AUTFL-2.1.2.3 Sie können die Forderungen von ASPICE an die Teststrategie inkl. Regressionsteststrategie erläutern. (K2)

AUTFL-2.1.2.4 Sie können die Forderungen von ASPICE an die Testdokumentation benennen. (K1)

AUTFL-2.1.2.5 Sie können eine Verifikationsstrategie (in Abgrenzung zu einer Teststrategie) auf Basis gegebener Verifikationskriterien für den Software Komponententest entwerfen. (K3)

AUTFL-2.1.2.6 Sie können die unterschiedlichen Rückverfolgbarkeitsanforderungen aus ASPICE erklären. (K2)

ISO 26262

AUTFL-2.2.1.1 Sie können das Ziel von funktionaler Sicherheit für E/E-Systeme erläutern. (K2)

AUTFL-2.2.1.2 Sie können Ihren Beitrag als Tester zur Sicherheitskultur wiedergeben. (K1)

AUTFL-2.2.2.1 Sie können die Rolle des Testers im Rahmen des Sicherheitslebenszyklus nach ISO 26262 darstellen. (K2)

AUTFL-2.2.3.1 Sie können Aufbau und Struktur der ISO 26262 wiedergeben. [informativ]

AUTFL-2.2.3.2 Sie können die für Sie als Tester relevanten Bände der ISO 26262 benennen. (K1)

AUTFL-2.2.4.1 Sie können die Kritikalitätsabstufungen des ASIL benennen. (K1)

AUTFL-2.2.4.2 Sie können den Einfluss des ASIL auf anzuwendende Testentwurfsmethoden und Testarten für statische und dynamische Tests und die daraus resultierenden Testumfänge erläutern. (K2)

AUTFL-2.2.5.1 Sie können ausgesuchte Methodentabellen der ISO 26262 anwenden. (K3)

AUTOSAR

AUTFL-2.3.1 Sie können die Ziele von AUTOSAR benennen. (K1)

AUTFL-2.3.2 Sie können den prinzipiellen Aufbau von AUTOSAR benennen [informativ]. (K1)

AUTFL-2.3.3 Sie können die Einflüsse von AUTOSAR auf die Arbeit des Testers benennen. (K1)

Gegenüberstellung

AUTFL-2.4.1 Sie können die unterschiedlichen Zielsetzungen von ASPICE und der ISO 26262 wiedergeben (K1).

AUTFL-2.4.2 Sie können die Unterschiede von ASPICE und ISO 26262 zum CTFL bezüglich der Teststufen erläutern (K2).

2.1 Automotive SPICE (ASPICE) (K3) [140 Min]

Einführung

Prozessverbesserung verfolgt den Ansatz, dass die Qualität eines Systems von der Qualität der Prozesse in der Entwicklung abhängt. Prozessmodelle bieten hier einen Ansatz für Verbesserungen, indem sie die Prozessfähigkeit einer Organisation gegen das Modell messen. Das Modell dient außerdem als Rahmenwerk für die Verbesserung der Prozesse in einer Organisation anhand der Bewertungsergebnisse [7].

Ab 2001 entwickelten die SPICE[10] User Group und die AUTOSIG (engl. Automotive Special Interest Group) Automotive SPICE (ASPICE). Seit der Veröffentlichung in 2005 ist der Standard zwischenzeitlich in der Automobilindustrie etabliert.

Im Juli 2015 gab der Verband der Automobilindustrie e.V. die ASPICE Version 3.0 frei. Diese löst ab 2017 [8] die etablierte Version 2.5 [4] ab. Alle in diesem Abschnitt getroffenen Aussagen beziehen sich daher auf die Version 3.0 von ASPICE [9][11].

2.1.1 Aufbau und Struktur des Standards (K2) [25 Min]

2.1.1.1 Die zwei Dimensionen von ASPICE

ASPICE definiert ein Bewertungsmodell mit zwei Dimensionen:

In der **Prozessdimension** legt ASPICE die standardisierten Referenzprozesse fest. Diese dienen als Referenz, um die eigenen Prozesse zu vergleichen, zu bewerten und zu verbessern. Zu jedem Prozess legt ASPICE den Zweck und die Ergebnisse fest. Aber auch die geforderten Tätigkeiten (Basispraktiken) und Arbeitsergebnisse (Arbeitsprodukte). Benötigt eine Organisation über ASPICE

[10] Akronym für „Software Process Improvement and Capability Determination"

[11] Die deutschen Übersetzungen der Prozessnamen, Basispraktiken und Arbeitsprodukte wurden an ASPICE:2008 [4] angelehnt.

hinaus weitere Referenzprozesse, kann sie diese zum Beispiel aus der ISO/IEC 12207 [10] oder ISO/IEC 15288 [11] entnehmen.

In der **Fähigkeitsdimension** definiert ASPICE eine Menge an Prozessattributen. Diese liefern die messbaren Eigenschaften der Prozessfähigkeit. Für jeden Prozess gibt es sowohl für den Prozess spezifische als auch Prozess übergreifende (generische) Attribute. Als Basis zur Beurteilung der Prozessfähigkeit dient die ISO/IEC 33020 [12].

Mit Hilfe dieses Modells ist es möglich, die Prozesse (Prozessdimension) bezüglich ihrer Fähigkeit (Fähigkeitsdimension) zu bewerten.

2.1.1.2 Prozesskategorien in der Prozessdimension

ASPICE fasst die Referenzprozesse nach der Art Ihrer Aktivität in 8 Prozessgruppen zusammen. Diese Gruppen sind wiederum einer von 3 Kategorien zugeordnet [8]:

Die **primären Prozesse** umfassen alle Prozesse, die der Wertschöpfung des Unternehmens dienen (sogenannte Kernprozesse):

- Beschaffung (ACQ) von Produkten und/oder Dienstleistungen
- Zulieferung (SPL) von Produkten und/oder Dienstleistungen
- Systementwicklung (SYS)
- Softwareentwicklung (SWE)

Die **unterstützenden Prozesse** umfassen alle Prozesse, die andere Prozesse unterstützen:

- Unterstützende Prozesse (SUP)

Die **organisatorischen Prozesse** umfassen alle Prozesse, die die Unternehmensziele unterstützen:

- Management (MAN) zum Leiten eines Projektes oder Prozesses
- Prozessverbesserung (PIM) zur Verbesserung der Prozesse
- Wiederverwendung (REU) von Systemen und Komponenten

Für den Tester sind die Prozessgruppen Systementwicklung (SYS) und Softwareentwicklung (SWE) von besonderem Interesse. Diese bilden die Prozesse des Automotive SPICE V-Modells ([9] Annex D „Key Concepts“).

2.1.1.3 Fähigkeitsstufen in der Fähigkeitsdimension

Der Assessor bewertet die Prozessfähigkeit mit einem sechsstufigen Bewertungssystem (Stufendarstellung). ASPICE definiert die Fähigkeitsstufen 0 bis 3[12] wie folgt [8]:

- Stufe 0 (unvollständiger Prozess): Der Prozess existiert nicht oder erreicht nicht den Prozesszweck. Beispiel: Der Tester überprüft nur einen geringen Teil der Anforderungen.
- Stufe 1 (durchgeführter Prozess): Der im Projekt gelebte Prozess erreicht den Prozesszweck (wenn auch möglicherweise unsystematisch). Beispiel: Für den Testprozess ist keine vollständige Planung erkennbar. Der Tester kann jedoch den Grad der Anforderungserfüllung aufzeigen.
- Stufe 2 (gesteuerter Prozess): Das Projekt plant und überwacht den Prozess in seiner Durchführung. Unter Umständen passt es das Vorgehen im Zuge der Ausführung auf das Ziel an. Die Vorgaben an die Arbeitsprodukte sind definiert. Ein Projektmitarbeiter prüft die

[12] Die Fähigkeitsstufen 4 und 5 stehen aktuell in der Automobilindustrie nicht im Fokus.

Arbeitsprodukte und gibt diese frei. Beispiel: Der Testmanager definiert die Testziele, plant die Testaktivitäten und überwacht den Fortschritt. Im Falle von Abweichungen reagiert er entsprechend.

- Stufe 3 (etablierter Prozess): Das Projekt wendet einen standardisierten Prozess an. Dabei nutzt es die Erkenntnisse, um diesen fortlaufend zu verbessern. Beispiel: Es existiert für die gesamte Organisation eine übergreifende Teststrategie. Im Rahmen des Testabschlusses (siehe fundamentaler Testprozess) hilft der Testmanager diese weiter zu entwickeln.

2.1.2 Forderungen durch den Standard (K3) [115 Min]

2.1.2.1 Testspezifische Prozesse

ASPICE definiert zu allen Prozessen der Software- und Systementwicklung zugehörige Testprozesse [8]:

- Der Prozess Softwarekomponenten-Verifikation[13] (SWE.4) fordert statische und dynamische Tests. Er bewertet die Komponenten der Software auf Basis des Feindesigns der Software (SWE.3).
- Der Softwareintegrationstest (SWE.5) bewertet die integrierte Software auf Basis der Architektur der Software (SWE.2).
- Der Softwarequalifikationstest (SWE.6) bewertet die integrierte Software auf Basis der Anforderungen an die Software (SWE.1).
- Der Systemintegrationstest (SYS.4) bewertet das integrierte System auf Basis der Architektur des Systems (SYS.3).
- Der Systemqualifikationstest (SYS.5) bewertet das integrierte System auf Basis der Anforderungen an das System (SYS.2).

2.1.2.2 Bewertungsstufen und Fähigkeitsindikatoren

Die Prozessfähigkeit kann ein Assessor über Fähigkeitsindikatoren bewerten. Diese beschreibt ASPICE für 9 Prozessattribute (PA). Für die Fähigkeitsstufen 1 bis 3 sind diese wie folgt definiert (in Klammern beispielhaft für SWE.6) [8]:

- PA 1.1: Prozessdurchführung (der Tester orientiert sich am fundamentalen Testprozess).
- PA 2.1: Management der Prozessdurchführung (der Tester plant, überwacht und steuert unter anderem die Testaktivitäten).
- PA 2.2: Management der Arbeitsprodukte (der Tester prüft unter anderem die Qualität der Testdokumentation).
- PA 3.1: Prozessdefinition (der für den Testprozess Verantwortliche definiert unter anderem eine projektübergreifende Teststrategie).
- PA 3.2: Prozessanwendung / Prozessumsetzung (der Tester wendet die in PA 3.1 definierte Teststrategie an).

Für die Prozessdurchführung (PA 1.1) definiert ASPICE zwei Arten von Indikatoren: Basispraktiken (BP) und Arbeitsprodukte (WP). Für die höheren Fähigkeitsstufen sind zusätzlich generische Praktiken (GP) und Ressourcen definiert. Die Bewertung der Prozessattribute erfolgt auf Basis des Umsetzungsgrads der Indikatoren in vier Bewertungsstufen [8]:

- **N** (None): nicht erfüllt (0% bis ≤ 15%)

[13] In ASPICE wird durchgängig von „Verifikation“ gesprochen. Der Lehrplan verwendet in diesem Abschnitt den Begriff „Verifikation“ konform zu ASPICE (in der deutschsprachigen Fassung) als Synonym des im ISTQB verwendeten Leitbegriffes „Verifizierung“.

- **P** (Partly): teilweise erfüllt (> 15% bis ≤ 50%)
- **L** (Largely): weitgehend erfüllt (> 50% bis ≤ 85%)
- **F** (Fully): vollständig erfüllt (> 85% bis ≤ 100%)

Damit ein Prozess eine bestimmte Fähigkeitsstufe erreicht, müssen die Indikatoren der angestrebten Fähigkeitsstufe „weitgehend erfüllt (L)“ sein. Die Indikatoren der darunterliegenden Fähigkeitsstufen müssen „vollständig erfüllt (F)“ sein.

2.1.2.3 Teststrategie und Regressionsteststrategie

ASPICE fordert als Basispraktik eine Teststrategie[14] für jeden testspezifischen Prozess (siehe 2.1.2.1). Diese erarbeitet der Testmanager im Rahmen der Testplanung des fundamentalen Testprozesses. Testrichtlinien, Projektziele als auch vertragliche und regulatorische Anforderungen bilden dafür die Grundlage.

Der Tester kennt frühes Testen als einen Grundsatz des Testens. Dies gilt auch für das Testen von Software im automobilen Umfeld. Allerdings kommt hier noch der Aspekt hinzu, dass Testumgebungen in höheren Teststufen erheblich kostenintensiver sind. So wird für das Testen auf höheren Stufen auch die speziell dafür entwickelte, einbettende Hardware benötigt (zum Beispiel als Prototyp oder Unikat). Die Teststrategie legt die stufenspezifischen Testumgebungen fest. Aber auch welche Tests auf welchen Testumgebungen der Tester durchführen soll.

Die Regressionsteststrategie ist ein wesentlicher Bestandteil der Teststrategie. Die Herausforderung liegt in der wirtschaftlich sinnvollen Auswahl der Testfälle („Mehrwert des Testens“). Die Regressionsteststrategie legt das Ziel und die Vorgehensweise bei der Auswahl der Regressionstests fest. So kann beispielhaft die Auswahl risikobasiert erfolgen. Eine Auswirkungsanalyse hilft dabei die Bereiche zu identifizieren, die der Tester erneut durch Regressionstests bewerten muss. Der Testmanager kann aber auch fordern, dass der Tester alle automatisierten Testfälle zu jedem Release wiederholen muss.

2.1.2.4 Testdokumentation in ASPICE

Für die Dokumentation der Testaktivitäten fordert ASPICE viele Arbeitsprodukte (WP), die dem Tester aus dem CTFL bekannt sind [8]:

- WP 08-50: Testspezifikation (bestehend aus Testentwurfs-, Testfall- und Testablaufspezifikation)
- WP 08-52: Testkonzept, inklusive Strategie (WP 19-00)
- WP 13-50: Testprotokoll, Fehler-/Abweichungsbericht und Testabschlussbericht

Zu jedem Arbeitsprodukt definiert ASPICE beispielhafte Merkmale und Inhalte. Diese kann ein Assessor stichprobenhaft bewerten. Sie dienen ihm als objektiver Indikator für eine Prozessdurchführung.

Beim Testkonzept greift ASPICE direkt auf die ISO/IEC/IEEE 29119-3[15] zurück. Aber auch für die anderen geforderten Arbeitsprodukte kann der Tester hieraus Vorlagen entnehmen. Diese kann er dann für den Zweck spezifisch anpassen. Er muss jedoch sicherstellen, dass sie im verwendeten Kontext zu dem beabsichtigten Zweck der Prozesse beitragen.

[14] Nach CTFL [2] ist die projektspezifische Teststrategie auch als Testvorgehensweise bekannt.
[15] Diese löst die im ISTQB noch gebräuchliche IEEE 829:1998 bzw. IEEE 829:2008 ab.

2.1.2.5 Verifikationsstrategie und -kriterien in der Softwarekomponenten-Verifikation (SWE.4)

Für die Verifikation der Softwarekomponenten (SWE.4) fordert ASPICE eine Verifikationsstrategie[16]. In den anderen testspezifischen Prozessen (SWE.5/SWE.6/SYS.4/SYS.5) fordert ASPICE „nur" eine Teststrategie (siehe 2.1.2.3). Die Teststrategie betrachtet „nur" dynamische Tests. Diese ergänzt die Verifikationsstrategie: Sie betrachtet ergänzend auch Code-Reviews und statische Analysen. Beide Verfahren sind aus dem CTFL auch als „statische Tests" bekannt.

Der Tester verifiziert die Komponenten mit dem Zweck, die Übereinstimmung mit den nicht-funktionalen Anforderungen und dem Feindesign nachzuweisen. Die Strategie legt dabei fest, wie der Tester hierfür den Nachweis erbringt. So kann der Tester verschiedene statische und dynamische Testverfahren zum Verifizieren der Komponenten verwenden und miteinander kombinieren.

Verändert ein Entwickler eine Komponente, muss der Tester auch diese Änderung bewerten. Aus diesem Grund beinhaltet die Strategie für die Verifikation von Komponenten auch eine Regressionsstrategie. Hierzu gehören die Verifikation des geänderten Codes, die Fehlernachtests sowie das erneute Verifizieren der nicht geänderten Teile (statische und dynamische Regressionstests).

In SWE.4.BP.2 fordert ASPICE das Entwickeln von Kriterien zur Komponentenverifikation. Diese Kriterien definieren, was erfüllt sein muss. Dadurch kann der Tester bewerten, inwieweit die Komponente die nicht-funktionalen Anforderungen erfüllt und mit dem Feindesign übereinstimmt. Als mögliche Kriterien zur Verifikation von Komponenten zählen:

- Komponententestfälle (inklusive Testdaten), die eine Komponente bestehen muss
- Ziele für die Testüberdeckung (zum Beispiel Anweisungsüberdeckung)
- Werkzeuggestützte statische Analysen, die die Einhaltung von Programmierrichtlinien bewerten (wie MISRA-C Programmierrichtlinien, siehe 4.1.1)
- Code-Reviews für Programmierrichtlinien, die nicht durch werkzeuggestützte statische Analysen bewertet werden können

Die Dokumentation der Verifikationsstrategie ist nach ASPICE ein Teil des Testkonzepts ([13] Abschnitt 6.2.7) auf Komponentenebene. Die Inhalte gliedern sich gemäß der ISO 29119-3, erweitert um die Aspekte der statischen Tests.

2.1.2.6 Rückverfolgbarkeit in ASPICE

Wie im CTFL [2] fordert auch ASPICE eine bidirektionale Rückverfolgbarkeit[17]. Erst dadurch ist es dem Tester möglich

- Auswirkungen zu analysieren,
- eine Überdeckung zu bewerten oder
- einen Status zu verfolgen.

Darüber hinaus ermöglicht diese, die Konsistenz zwischen den verknüpften Elementen sicher zu stellen. Sowohl inhaltlich als auch semantisch.

ASPICE unterscheidet zwischen vertikaler und horizontaler Rückverfolgbarkeit [8]:

[16] Bei den Begriffen „Verifikationsstrategie" und „Teststrategie" wird im ASPICE der Begriff „Strategie" im Gegensatz zum ISTQB als projektspezifische „Vorgehensweise" verstanden.

[17] Im Folgenden impliziert der Begriff Rückverfolgbarkeit stets die bidirektionale Rückverfolgbarkeit.

Vertikal fordert ASPICE, die Anforderungen der Stakeholder bis hin zu den Software-Komponenten miteinander zu verknüpfen. Dabei soll das Verknüpfen über alle Stufen der Entwicklung hinweg eine Konsistenz zwischen den verknüpften Arbeitsergebnissen sicherstellen.

Horizontal fordert ASPICE ebenfalls die Rückverfolgbarkeit und Konsistenz. Hier hingegen zwischen den Arbeitsergebnissen der Entwicklung zu den zugehörigen Testspezifikationen und -ergebnissen.

Ergänzend fordert die Basispraktik SUP.10.BP8 die Rückverfolgbarkeit zwischen Änderungen und davon betroffenen Arbeitsprodukten. Ist die Änderung durch eine Fehlerkorrektur initiiert, so fordert diese auch die Rückverfolgbarkeit zwischen der Änderungsanforderung und dem entsprechenden Abweichungsbericht.

Aufgrund der zum Teil großen Anzahl von Verknüpfungen kann eine durchgängige Werkzeugkette hilfreich sein. Diese ermöglicht dem Tester ein effizientes Herstellen und Verwalten der Verknüpfungen.

2.2 ISO 26262 (K3) [125 Min]

2.2.1 Funktionale Sicherheit und Sicherheitskultur (K2) [20 Min]

2.2.1.1 Ziel funktionaler Sicherheit für E/E-Systeme

Die funktionale und technische Komplexität eingebetteter Systeme nimmt stetig zu. Zugleich ermöglichen leistungsstarke softwarebasierte elektrische und elektronische Systeme neue, komplexe Funktionalitäten wie die Automatisierung von Fahrfunktionen im PKW.

Durch die hohe Komplexität steigt das Risiko, dass es bereits während der Entwicklung zu einer Fehlhandlung kommt. Deren Folge kann ein (unerkannter) Fehlerzustand im System sein. Für Systeme mit inhärentem Gefährdungspotential für Leib und Leben muss daher der Sicherheitsverantwortliche potenzielle Gefährdungen analysieren. Liegt eine tatsächliche Gefährdung vor, identifiziert er geeignete Maßnahmen, um ihre möglichen Auswirkungen auf ein akzeptables Risiko zu reduzieren.

Die Methoden zur Durchführung solcher Analysen sind in den Normen für die Funktionale Sicherheit (FuSi) zusammengefasst. Eine grundlegende Norm ist die IEC 61508. Für die Automobilentwicklung hat die Internationale Organisation für Normung (ISO) daraus die ISO 26262 abgeleitet. [ISO 26262], [IEC 61508].

FuSi für E/E-Systeme im Sinne der ISO 26262 bedeutet die Vermeidung nicht tolerierbarer Risiken für Leib und Leben bei Fehlverhalten dieser Systeme. Dabei ist der Begriff von anderen Sicherheitsbegriffen wie Informationssicherheit, Produktsicherheit oder Arbeitssicherheit abzugrenzen.

Arbeitssicherheit oder Informationssicherheit sind nicht im Fokus der ISO 26262. Fehlende Informationssicherheit kann die funktionale Sicherheit gefährden. Funktionale Sicherheit und Informationssicherheit tragen zur Produktsicherheit bei.

2.2.1.2 Beitrag des Testers zur Sicherheitskultur

Im Rahmen der Produktentwicklung nach ISO 26262 genügt es nicht, ausschließlich die eigenen Prozesse im Blick zu behalten. Alle Beteiligten müssen einen prozessübergreifenden Ansatz leben. Jeder muss seinen Einfluss auf das Entwicklungsgeschehen und die Sicherheit des finalen Produkts verstehen. Dies schließt externe Partner und Zulieferer ein.

Die Beteiligten müssen verstehen, dass die eigene Tätigkeit nicht losgelöst von anderen Prozessen stattfindet. Jeder Schritt der Entwicklung stellt einen essenziellen Beitrag zur Einhaltung und Umsetzung der FuSi-relevanten Anforderungen dar. Diese Verantwortung endet dabei *nicht* mit der Produkteinführung. Sie reicht bis zum Ende des Systemlebenszyklus.

Der Tester trägt zur Sicherheitskultur bei, indem er in allen Entwicklungsphasen verantwortungsvoll mitarbeitet und seine Tätigkeit stets mit Blick auf den Gesamtkontext der Produktentwicklung ausübt. [ISO 26262]

2.2.2 Einordnung des Testers in den Sicherheitslebenszyklus (K2) [15 Min]

Der Sicherheitslebenszyklus beschreibt die Phasen einer sicherheitsgerichteten Produktentwicklung. Dieser beginnt mit der ersten Produktidee und der Ermittlung möglicher Risiken. Nach der Spezifikation daraus resultierender Sicherheitsanforderungen erfolgt die Umsetzung in ein konkretes Produkt. Der Zyklus endet mit der Entsorgung des Produkts an seinem Lebensende (siehe auch Kapitel 1.3).

Der Sicherheitslebenszyklus in der Auslegung nach ISO 26262 durchläuft folgende Phasen:

- die Konzeptphase,
- die Produktentwicklung und
- die Phase nach der Produktionsfreigabe (zum Beispiel Produktion, Betrieb, Wartung).

Der Tester ist überwiegend in den ersten beiden Phasen tätig. Änderungen am Produkt im Rahmen der dritten Phase führen je nach Umfang zu einem Rücksprung in die erste oder zweite Phase. Damit ist der Tester auch an Modifikationen beteiligt. Basierend auf sicherheitsgerichteten Anforderungen (siehe Kapitel 2.2.4) entwirft er im Rahmen der Produktentwicklung die Testfälle und Testabläufe zur Verifizierung und Validierung dieser Anforderungen. Diese führt der Tester dann in den jeweiligen Subphasen der Produktentwicklung durch.

Die Aktivitäten der Testplanung finden in der Regel während der Konzeptphase statt. Anpassungen an den dabei entstandenen Dokumenten (zum Beispiel am Testkonzept oder den Testspezifikationen) können aber in jeder Phase erforderlich sein. Die Testdurchführung findet vor allem an den Übergängen zwischen den einzelnen Subphasen der Produktentwicklung statt. Beispielsweise zwischen der Implementierung und der Softwareintegration sowie dann weiterführend zur Hardware-Software-Integration. Darüber hinaus trägt der Tester mit seinen Testaktivitäten zentral zum Übergang zur dritten Phase bei. [ISO 26262]

2.2.3 Gliederung und testspezifische Anteile der Norm (K1) [10 Min]

2.2.3.1 Aufbau und Struktur der Norm [informativ]

Die Norm ISO 26262 besteht aus 10 Bänden:

- Vokabular (Band 1)
- Management der Funktionalen Sicherheit (Band 2)
- Die Phasen des Sicherheitslebenszyklus:
  - Konzeptphase (Band 3)
  - Produktentwicklung für Gesamtsystem, Hardware und Software (Bände 4-6)
  - Produktion und Betrieb (Band 7)
- Unterstützungsprozesse (Band 8),
- ASIL- und sicherheitsgerichtete Analysen (Band 9)
- Leitlinien für die Anwendung der ISO 26262 (Band 10)

Abgesehen von Band 1 und Band 10 enthält jeder Band zunächst Standardinhalte. Hierzu zählen:

- eine allgemeine Einleitung,
- der Geltungsbereich,
- normative Referenzen und
- Anforderungen für die Erfüllung der Norm.

Darauf folgen die spezifischen Themen des jeweiligen Bandes. Die Struktur ihrer Beschreibung ist dabei in jedem Band gleich. Die durchzuführenden Aktivitäten werden in allen Bänden anhand einer wiederkehrenden Struktur beschrieben:

- Ziel
- Allgemeine Informationen
- Eingangsinformationen
  - Vorbedingungen
  - Weitere unterstützende Informationen
- Anforderungen und Empfehlungen
- Arbeitsergebnisse

[ISO 26262]

2.2.3.2 Für den Tester relevante Bände

Für den Softwaretester stehen die Softwareverifizierung und (zumindest anteilig) auch die Systemvalidierung im Vordergrund. Für ihn sind neben Band 1 (Terminologie) auch einige andere Bände von besonderem Interesse: Die Bände 4 und 6 geben ihm detaillierte Hinweise und Anforderungen hinsichtlich empfohlener Maßnahmen der Softwareverifizierung. Dies gilt sowohl für die Auswahl, den Entwurf und die Implementierung, als auch für die Durchführung der jeweiligen Verifizierungsmaßnahmen.

Dabei fokussieren diese Bände auf die test- und verifizierungsspezifischen Aspekte der System- (Band 4, inklusive Systemvalidierung) und Softwareebene (Band 6). Sollten für seine Arbeit auch hardwarespezifische Aspekte relevant sein, findet der Tester diese in Band 5. Aspekte, die sowohl Hard- als auch Software betreffen, werden im Rahmen des Hardware Software Interface betrachtet (Bände 4, 5, 6).

Band 8 der ISO 26262 nimmt eine Sonderstellung ein, da dieser die prozessspezifischen Eigenheiten der Verifizierung auf allen Teststufen beschreibt. Darüber hinaus finden sich dort Anforderungen an für den Tester wichtige unterstützende Prozesse, wie beispielsweise die Dokumentation.

[ISO 26262]

2.2.4 Einfluss der Kritikalität auf die Testumfänge (K2) [20 Min]

2.2.4.1 Die Kritikalitätsabstufungen des ASIL

Der ASIL (im Englischen: „Automotive Safety Integrity Level“) ist ein Maß für die erforderliche Risikominderung durch Maßnahmen der FuSi. Derartige Maßnahmen können beispielsweise eine eigenständige Sicherheitsfunktion zur Überwachung eines E/E-Systems oder der Einsatz spezifischer festgelegter Methoden sein. Für höhere Risiken können dabei aufwändigere Maßnahmen erforderlich sein.

Zu Projektbeginn führt ein Expertenteam die Gefährdungsanalyse und Risikobewertung (G&R) für das Produkt durch. Für *jede* durch diese Analyse identifizierte Gefährdung ermittelt er mit Hilfe einer in der Norm vorgegebenen Methodik einen ASIL. Im nächsten Schritt formuliert er Sicherheitsziele und Sicherheitsanforderungen. Diese besitzen den gleichen ASIL wie die zu Grunde liegende Gefährdung.

Die ISO 26262 definiert vier Ausprägungsgrade: von ASIL A für niedrige, bis ASIL D für hohe Sicherheitsanforderungen.

Ergeben sich aus der G&R Anforderungen unterhalb von ASIL A, so sind diese aus Sicht der Norm *nicht* sicherheitsrelevant. Diese Anforderungen werden durch die Einhaltung des vorhandenen Qualitätsmanagements (QM) abgedeckt. [ISO 26262]

2.2.4.2 Einfluss des ASIL auf Testverfahren, Testarten und Testumfänge

Der ermittelte ASIL beeinflusst unmittelbar die vom Tester zu realisierenden Testumfänge. Abhängig vom jeweiligen Ausprägungsgrad des ASILs empfiehlt die ISO 26262 die Durchführung unterschiedlicher Maßnahmen oder Maßnahmenpakete. Dabei gilt, dass die Norm für höhere ASIL, umfangreichere und detailliertere Maßnahmen empfiehlt. Für niedrigere ASIL ist die Durchführung solcher Maßnahmen hingegen oft optional.

Den Grad der Empfehlung teilt die ISO 26262 in drei Stufen ein: neutral, empfohlen („recommended“) und dringend empfohlen („highly recommended“). Bei „neutral“ gibt die Norm keine Empfehlung für oder gegen die Verwendung der betreffenden Maßnahme. Sie kann bedenkenlos unterstützend durchgeführt werden. Ihre Durchführung ersetzt aber nicht die von der ISO empfohlenen oder dringend empfohlenen Maßnahmen.

Für den Tester bedeutet das, dass ihm die Norm für FuSi-relevante Systeme abhängig vom ASIL konkrete Testentwurfsverfahren und Testarten empfiehlt. Der Tester kann nur insoweit frei entscheiden, wie es die Norm für den konkreten Fall zulässt. So sind beispielsweise Äquivalenzklassenbildung und Grenzwertanalyse für ASIL A empfohlen. Bei einem ASIL B und höher hingegen sind diese Verfahren *dringend* empfohlen (siehe dazu Kapitel 2.2.5).

Der ASIL ist keine Eigenschaft des Gesamtprodukts. Er ist an ein konkretes Sicherheitsziel und die daraus abgeleiteten Sicherheitsanforderungen gebunden. Für *ein* Produkt können sich also für Sicherheitsanforderungen mit unterschiedlichem ASIL unterschiedliche Testaufwände ergeben. Dies ist bei der Planung der Testumfänge durch den Tester zu berücksichtigen.

[ISO 26262]

2.2.5 Anwendung des aus CTFL bekannten Wissens im Kontext der ISO 26262 (K3) [60 Min]

Die ISO 26262 bietet dem Tester konkrete Empfehlungen zu seinen Testaktivitäten in Form der so genannten Methodentabellen. Diese Tabellen finden sich in den Bänden 4, 5, 6 und 8. Sie enthalten neben FuSi-spezifischen Empfehlungen für Prozesse und Tätigkeiten, auch die vom Tester anzuwendenden Verfahren.

Die Norm verwendet in diesem Zusammenhang für alle anzuwendenden Verfahren oder Tätigkeiten den einheitlichen Oberbegriff Methode (im Englischen: „method“). An dieser Stelle weicht die FuSi-Nomenklatur also etwas von der Begriffswelt des ISTQB ab. Für den Tester sind folgende Methoden der ISO 26262 von besonderem Interesse:

- Testentwurfsverfahren (Äquivalenzklassenbildung, Grenzwertanalyse, …)
- Verfahren der Testdurchführung (Simulation, Fahrzeugtests, …)
- Testarten (nicht-funktionale Tests wie Performanztest, Lebensdauertest, …)
- Testumgebungen (HIL, Fahrzeug, …)
- statische Testverfahren (Reviews, statische Analysen, …)

Die Methodentabellen geben vor, bei welchem ASIL die Anwendung einer Methode jeweils von der Norm empfohlen wird.

Die Tabellen sind dabei stets nach dem gleichen Schema aufgebaut:

		ASIL A	**ASIL B**	**ASIL C**	**ASIL D**
1	Methode x	o	+	++	++
2	Methode y	o	o	+	+
3a	Methode z1	+	++	++	++

3b	Methode z2	++	+	o	o

Tabelle 1: Beispiel für Methodentabelle

Für jede Methode ist abhängig vom ASIL dokumentiert, ob ihre Verwendung empfohlen (+) oder sogar dringend empfohlen (++) ist. Für neutral (o) gekennzeichnete Methoden liegt keine Empfehlung der Norm für oder gegen eine Verwendung vor.

Die ISO 26262 nennt in den Tabellen auch gleichwertige alternative Methoden (im obigen Beispiel die Zeilen 3a und 3b). Hier muss der Tester eine geeignete Kombination auswählen, um die zugehörigen Anforderungen ASIL-konform überprüfen zu können. Die gewählte Kombination ist dabei durch den Tester zu begründen.

Im Fall von Methoden ohne Alternativen (im Beispiel die Zeilen 1 und 2) entfällt diese Auswahlmöglichkeit. Hier hat der Tester alle Methoden anzuwenden, die für den jeweiligen ASIL dringend empfohlen sind.

Aus dem obigen Beispiel ergeben sich für den Nachweis einer Anforderung nach ASIL C folgende Methoden:

- Methode x: dringend empfohlen, also üblicherweise anzuwenden, wenn nach ISO 26262 entwickelt wird
- Methode y: empfohlen, also anzuwenden, falls für den Nachweis dienlich,
- Methode z1 und z2: hier ist mindestens Methode z1 auszuwählen, da sie für ASIL C die höhere Einstufung besitzt.

Die ISO 26262 erlaubt dem Tester auch andere als die in den Tabellen aufgeführten Methoden anzuwenden. In einem solchen Fall muss er allerdings Tauglichkeit und Angemessenheit der von ihm alternativ gewählten Methode begründen. [ISO 26262]

2.3 AUTOSAR (K1) [15 Min]

Einführung

AUTOSAR steht als Abkürzung für „AUTomotive Open System ARchitecture“ und für die dahinterstehende Entwicklungspartnerschaft. Diese gründete sich im Jahr 2003 und setzt sich hauptsächlich aus Herstellern und Zulieferern der Automobilindustrie zusammen. Ihr Ziel: Einen frei zugänglichen Standard für eine Softwarearchitektur im Fahrzeugumfeld zu schaffen und zu etablieren. Damit soll der Standard der steigenden Bedeutung und Komplexität der Software Rechnung tragen [14]. Heute ist AUTOSAR ein weltweit verbreiteter Standard für E/E-Systeme. So kommt der Tester zwangsläufig mit Produkten mit AUTOSAR-Architektur in Kontakt. Daher ist es wichtig, dass er die Ziele, den prinzipiellen Aufbau sowie die Berührungspunkte mit seiner Tätigkeit kennt.

2.3.1 Ziele von AUTOSAR (K1) [5 Min]

Die folgenden Projektziele für AUTOSAR werden von der Devise „Zusammenarbeit bei

den Standards, Wettbewerb bei der Umsetzung“ geleitet: [14, 15]:

1. Unterstützt die Übertragbarkeit von Software.
2. Unterstützt die Skalierbarkeit von Systemen für unterschiedliche Fahrzeug- und Plattform-Varianten.
3. Unterstützt eine breite Vielfalt von Domänen.
4. Definiert eine offene Architektur, die sowohl wartbar als auch anpassbar und erweiterbar ist.
5. Unterstützt das Entwickeln von verlässlichen Systemen (gekennzeichnet durch Verfügbarkeit, Zuverlässigkeit, Sicherheit (sowohl funktional als auch im Sinne von Datenschutz, engl.: safety & security), Integrität und Wartbarkeit).

6. Unterstützt ein nachhaltiges Verwenden natürlicher Ressourcen.
7. Unterstützt die Zusammenarbeit zwischen verschiedenen Partnern.
8. Standardisiert die Grundfunktionalität der Software von automobilen Steuergeräten.
9. Ist konform mit internationalen Normen und Standards für Fahrzeuge sowie mit Technologien, die dem Stand der Technik entsprechen.

2.3.2 Prinzipieller Aufbau von AUTOSAR [informativ] (K1) [5 Min]

Die Architektur von AUTOSAR besteht aus drei getrennten Schichten:

- Die von der Hardware unabhängige Schicht mit den AUTOSAR Software-Komponenten (SW-C).
- Die hardwareorientierte Schicht mit standardisierter Basissoftware (BSW).
- Die Abstraktionsschicht mit der AUTOSAR Laufzeitumgebung (RTE). Diese steuert innerhalb und außerhalb der Steuergeräte den Datenaustausch und setzt diesen um. Sowohl zwischen den Software-Komponenten als auch zwischen Software-Komponenten und Basissoftware.

Einen weiteren Aspekt stellt die AUTOSAR-Methodik zum harmonisierten Entwickeln von Steuergeräte-Software dar. Dabei tauschen Hersteller und Lieferanten Informationen über Beschreibungsdateien gemäß AUTOSAR Templates (sogenannte „.arxml-Files“) aus [14, 16]:

- Die „ECU-Konfigurationsbeschreibung“ umfasst Daten zur Integration der SW-C auf dem Steuergerät.
- Die „System-Konfigurationsbeschreibung“ enthält Daten zur Integration aller Steuergeräte in einem Fahrzeug.
- Das „ECU-Extrakt“ beinhaltet aus der „System-Konfigurationsbeschreibung“ die Daten für ein Steuergerät.

2.3.3 Einfluss von AUTOSAR auf die Arbeit des Testers (K1) [5 Min]

AUTOSAR wirkt sich auch auf die Arbeit des Testers aus. Insbesondere in den folgenden Teststufen[18]:

- Softwarekomponententest und Softwareintegrationstest in virtueller Umgebung (zum Beispiel Software in the Loop): Mit einer virtuellen BSW und RTE kann der Tester schon früh die Komponenten der Applikation testen [17, 18].
- Softwaretest und Softwareintegrationstests im realen Steuergerät: Hier erhält der Tester Zugriff auf die Kommunikation auf der RTE. Darüber kann der Tester das Verhalten der SW-C zur Laufzeit messen und stimulieren [19].
- Der AUTOSAR Akzeptanztest ist ein Systemtest der Software, der die Erfüllung der AUTOSAR Funktionalität auf Kommunikations- und Applikationsebene sicherstellt. Die Durchführung des AUTOSAR Akzeptanztests ist optional [20, 21].
- Systemintegrationstest: Funktionales Integrieren und Vernetzen verschiedener Steuergeräte (zum Beispiel auch im Fahrzeug). Durch das Simulieren von fehlenden, eventuell verteilten Funktionalitäten, kann der Tester schon früh das Systemverhalten bewerten [17].

[18] Zu Teststufen: siehe auch 2.4.2

2.4 Gegenüberstellung (K2) [20 Min]

2.4.1 Zielsetzung ASPICE und ISO 26262 (K1) [5 Min]

Es gibt viele Normen und Standards, die Anforderungen an die Produktentwicklung stellen. Diese beleuchten typischerweise jeweils unterschiedliche Aspekte bei der Entwicklung. Die ISO 26262 und ASPICE werden hier bezüglich ihrer Zielsetzung gegenübergestellt.

Die ISO 26262 [22] hat zum Ziel, Risiken aus systematischen Fehlern in der Entwicklung und zufälligen Hardware-Fehlern im Betrieb durch die Vorgabe von geeigneten Anforderungen und Prozessen zu vermeiden. Für die Entwicklung von E/E-Systemen definiert sie Anforderungen an die vom Tester anzuwendenden Prozesse und Methoden. Diese sind abhängig vom ASIL des Produkts.

ASPICE [9] dient u.a. dazu, im Rahmen von Assessments die Fähigkeit des Produktentwicklungsprozesses festzustellen. Hierzu definiert ASPICE bewertbare Anforderungen an diese Prozesse. Diese sind im Gegensatz zur ISO 26262 unabhängig von der Kritikalität und vom ASIL des Produkts.

2.4.2 Gegenüberstellung der Teststufen (K2) [15 Min]

Sowohl die ISO 26262 als auch ASPICE beschreiben Teststufen. Jedoch sind diese nicht vollständig konsistent mit den Teststufen aus dem CTFL [2]. Für eine effiziente und effektive Zusammenarbeit sollten darum die Tester ein gemeinsames Verständnis über alle Teststufen haben.

Der in ASPICE verwendete Begriff „System“ und die in der ISO 26262 verwendeten Begriffe „System“ und „Item“ beziehen sich auf ein Produkt bestehend aus Hardware und Softwarekomponenten. Der CTFL fokussiert sich bei „System“ hingegen in erster Linie auf Software. So lassen sich die Teststufen nach ISTQB [23] den Teststufen in der ISO 26262 und ASPICE wie folgt zuordnen:

ISTQB	**ISO 26262**	**ASPICE**
Abnahmetest	Sicherheitsvalidierung (4-9)[19]	kein Äquivalent
Multisystemtest[20]	Item-Integration und Test (4- 8)[21]	System-Qualifikationstest (SYS.5)
Systemintegrationstest	kein Äquivalent	System-Integrationstest (SYS.4)
Systemtest	Verifizierung der Software-Sicherheitsanforderungen (6-11) Software-Integration und Test (6-10)	Software-Qualifikationstest (SWE.6)

[19] Die Sicherheitsvalidierung deckt nur Teile eines Abnahmetests nach ISTQB ab.

[20] Das Testen von mehreren heterogen verteilten Systemen, sogenannten „Systemen von Systemen“ [23, 41]

[21] Item-Integration und Test umfasst drei Teilphasen: die Integration und den Test von Hardware und Software eines Elements, die Integration und den Test aller zum Item gehörigen Elemente, und die Integration und den Test des Items im Verbund mit anderen Items im Fahrzeug.

Integrationstest	kein Äquivalent	Software-Integrationstest (SWE.5)
Komponententest	Software-Unit-Test (6-9)	Softwarekomponentenverifikation (SWE.4)

Tabelle 2: Zuordnung der Teststufen

Nach ISTQB CTFL (1) sind Testverfahren weitestgehend unabhängig von den Teststufen anwendbar. Auch ASPICE benennt in der Regel keine Verfahren pro Teststufe. Somit überlassen beide die Auswahl den Testern. In der ISO 26262 existieren hingegen zu jeder Teststufe individuelle Methodentabellen (siehe Kapitel 2.2.5 und 2.2.4.2). Diese geben dem Tester unter anderem abhängig vom ASIL Empfehlungen, welche Verfahren er anwenden sollte.

3 Testen in virtueller Umgebung (K3) [160 Min]

Begriffe

XiL-Testumgebung, Model in the Loop (MiL), Software in the Loop (SiL), Hardware in the Loop (HiL), Open-Loop-System, Closed-Loop-System, Umgebungsmodell

Lernziele

AUTFL-3.1.1 Sie können den Zweck/die Motivation einer Testumgebung im automobilen Umfeld benennen. (K1)

AUTFL-3.1.2 Sie können die allgemeinen Bestandteile einer automobilspezifischen Testumgebung aufzählen. (K1)

AUTFL-3.1.3 Sie können die Unterschiede von Closed-Loop-Systemen und Open-Loop-Systemen erläutern. (K2)

AUTFL-3.1.4 Sie können die wesentlichen Funktionen, Datenbasen und Protokolle eines automobilen Steuergerätes benennen. (K1)

AUTFL-3.2.1.1 Sie können den Aufbau einer MiL-Testumgebung wiedergeben. (K1)

AUTFL-3.2.1.2 Sie können die Einsatzgebiete und die Randbedingungen einer MiL-Testumgebung erläutern. (K2)

AUTFL-3.2.2.1 Sie können den Aufbau einer SiL- Testumgebung wiedergeben. (K1)

AUTFL-3.2.2.2 Sie können die Einsatzgebiete und die Randbedingungen einer SiL-Testumgebung benennen. (K1)

AUTFL-3.2.3.1 Sie können den Aufbau einer HiL-Testumgebung wiedergeben. (K1)

AUTFL-3.2.3.2 Sie können die Einsatzgebiete und die Randbedingungen einer HiL-Testumgebung erläutern. (K2)

AUTFL-3.2.4.1 Sie können die Vor- und Nachteile für das Testen anhand von Kriterien der XiL-Testumgebungen (MiL, SiL und HiL) zusammenfassen. (K2)

AUTFL-3.2.4.2 Sie können Kriterien für die Zuordnung eines gegebenen Testumfangs auf eine oder mehrere Testumgebungen anwenden. (K3)

AUTFL-3.2.4.3 Sie können die drei XiL-Testumgebungen (MiL, SiL, HiL) im V-Modell einordnen. (K1)

3.1 Testumgebung allgemein (K2) [30 Min]

3.1.1 Motivation für eine Testumgebung im automobilen Umfeld (K1) [5 Min]

Der Tester steht vor einer besonderen Herausforderung. Auf der einen Seite soll er so früh wie möglich mit dem Testen beginnen, um Fehlerzustände früh im Entwicklungsprozess zu finden. Auf der anderen Seite benötigt er eine realitätsnahe Umgebung, um das System zu testen und die Fehlerzustände zu finden, die im fertigen Produkt auftreten würden. Diesen Konflikt kann der Tester lösen, indem er zweckmäßige Testumgebungen, passend zu den unterschiedlichen Entwicklungsphasen, einsetzt. Damit kann der Tester seine individuellen Testaufgaben realisieren und durchführen bevor das fertig produzierte oder entwickelte Steuergerät zur Verfügung steht. Er kann mit verschiedenen Testumgebungen zum Beispiel Fehler in der Umgebung des Testobjekts einfach und gezielt simulieren, die später im realen Fahrzeug schwer nachstellbar sind. Ein Beispiel solcher Fehler sind Kabelbrüche und Kurzschlüsse im Kabelbaum. [24]

3.1.2 Allgemeine Bestandteile einer Testumgebung (K1) [5 Min]

In den frühen Entwicklungsphasen ist vom zu testenden System bestenfalls die Software vorhanden. Die Hardware des Systems ist meist rudimentär bis gar nicht vorhanden. Damit der Tester seine Aktivitäten durchführen kann, benötigt er eine Testumgebung, in der die fehlenden Bestandteile simuliert werden. Diese Umgebung hilft dem Tester die Eingänge des Testobjektes zu stimulieren und die Reaktion an den Ausgängen zu beobachten. Nach ISO 29119 besteht eine Testumgebung aus folgenden Bestandteilen:

- Hardware der Testumgebung (Steuerrechner, echtzeitfähiger Rechner)
- Software der Testumgebung (Betriebssystem, Simulationssoftware, Umgebungsmodelle)
- Kommunikationsmittel (Netzwerkzugänge, Datenlogger)
- Werkzeuge (Oszilloskope, Messgeräte,)
- Labor (Schutz vor elektromagnetischer Strahlung und Lärm)

Ein wichtiger Bestandteil der Testumgebung ist das Umgebungsmodell. Es bildet die reale Welt ab, wie zum Beispiel den Verbrennungsmotor, das Getriebe und den Fahrer oder die Beschaffenheit der Straße. Die Testumgebung besitzt weiterhin verschiedene Zugriffsstellen. Diese kann der Tester nutzen, um das Testobjekt zu stimulieren und zu beobachten [25].

3.1.3 Unterschiede von Closed-Loop und Open-Loop (K2) [15 Min]

Die Testumgebung stimuliert die Eingangsschnittstellen des Testobjektes. Anschließend wird das Verhalten an den Ausgangsschnittstellen analysiert. Bei einem erfolgreichen Test muss das beobachtete Verhalten mit dem erwarteten Ausgangsschnittstellen übereinstimmen.

Es gibt prinzipiell zwei unterschiedliche Konfigurationen, die die Stimulation in das Testobjekt einspielen: Closed-Loop und Open-Loop Testumgebungskonfigurationen.

3.1.3.1 Open-Loop-System

Die Eingänge des Testobjekts bei einem Open-Loop-System gibt die Testablaufspezifikation direkt vor. Das bedeutet, dass die Ausgaben des Testobjektes in keiner Beziehung zu den Eingaben stehen.

Der Anwendungsfall für Open-Loop- und Closed-Loop-Systeme hängt stark von der Funktionsweise des Testobjektes ab. Hat das Testobjekt ein reaktives Verhalten oder spiegelt es einen Zustandsautomaten wider, ist ein Open-Loop-System zu bevorzugen. In der Innenraum- und Karosserie-Elektronik gibt es viele Beispiele für Open-Loop-Systeme (siehe Lampen und Schalter).

3.1.3.2 Closed-Loop-System

Die Stimulation bei einem Closed-Loop-System (auch in-the-Loop) berücksichtigt die Ausgaben des Testobjektes. Dies erfolgt über ein Umgebungsmodell, welches die Ausgaben erfasst und diese direkt oder indirekt an den Eingang des Testobjektes weiterleitet. Somit entsteht ein Regelkreis in der Testumgebung.

Für das Testen von Reglern ist das Closed-Loop-System zu bevorzugen. Hiermit kann der Tester komplexe Funktionen testen. Dazu zählen Motor- und Getriebesteuerungen sowie Fahrerassistenzsysteme wie das Antiblockiersystem (ABS) oder die Fahrdynamikregelung (ESP). [26, 27]

3.1.4 Wesentliche Schnittstellen, Datenbasen und Kommunikationsprotokolle eines Steuergerätes (K1) [5 Min]

Ein Steuergerät im automobilen Umfeld ist ein eingebettetes System. Es besteht aus Hardware und Software. Das Steuergerät besitzt verschiedene analoge und digitale Eingänge, die permanent Umgebungsdaten in Form von Spannungen und Strömen erfassen. So erhält das Steuergerät Informationen aus der Umwelt. Des Weiteren versorgen Bussysteme das Steuergerät mit weiteren

Informationen. Diese kommen von Sensoren oder anderen Steuergeräten, die sie entweder selber erfassen und verarbeiten oder zur Verfügung stellen. Das Steuergerät verwaltet die Daten im Speicher, um sie zu verarbeiten. Die Ausgabe von Informationen erfolgt ebenfalls über analoge und digitale Ausgänge, Bussysteme oder Diagnoseschnittstellen.

Die Datenbasen sind Datenbanken und definieren die Ein- und Ausgangssignale des Steuergerätes. Diese Daten enthalten ebenfalls Beschreibungen, Einheiten und Umrechnungsformeln der Signale. Die Kommunikationsprotokolle beschreiben den Datenaustausch über die jeweiligen physikalischen Schnittstellen. In diesen Protokollen ist definiert, welche Spannung oder Bitfolge welchen Wert des Signals repräsentiert. Die Auswahl der Datenbasis und des Kommunikationsprotokolls hängt von der Funktion des Steuergerätes ab. Um beispielsweise auf Diagnosefunktionen im Steuergerät zugreifen zu können, benötigt der Tester die Informationen über die verwendete Datenbasis (ASAM MCD2 D; auch „Open Diagnostic Data Exchange“) und das Kommunikationsprotokoll („Unified Diagnostic Services“ nach ISO 14229). Weitere automobilspezifische Datenbasen sind im ASAM Standard definiert [27, 28].

3.2 Testen in XiL-Testumgebungen (K3) [130 Min]

In der Automobilindustrie kommen folgende Arten von XiL-Testumgebungen zur Anwendung:

- Model in the Loop (MiL),
- Software in the Loop (SiL),
- Processor in the Loop[22] (PiL),
- Hardware in the Loop (HiL) und
- Vehicle in the Loop[22] (ViL).

Die wichtigsten Testumgebungen (MiL, SiL und HiL) soll der automobile Tester hier kennen und verstehen lernen. Die nachfolgenden Abschnitte vertiefen den Aufbau und die Einsatzgebiete der verschiedenen Testumgebungen. XiL steht dabei als Oberbegriff für die verschiedenen Testumgebungen.

3.2.1 Model in the Loop (MiL) [20 Min]

3.2.1.1 Aufbau einer MiL-Testumgebung

Bei einer MiL-Testumgebung liegt das Testobjekt als Modell vor. Das Modell ist menschenlesbar und nicht für eine spezielle Hardware kompiliert. Solche Modelle werden von den Entwicklern in speziellen Softwarewerkzeugen implementiert. Damit der Tester diese Modelle ausführen und testen kann, benötigt er eine Testumgebung. Diese ist in der gleichen Entwicklungsumgebung implementiert wie das Testobjekt selbst. Dieses zusätzliche Modell ist das Umgebungsmodell. Der Tester kann über Zugriffsstellen im Umgebungsmodell das Testobjekt stimulieren und das Verhalten beobachten. Die Zugriffsstellen in dieser Testumgebung sind beliebig platzierbar. Das Modell des Testobjektes ist mit dem Umgebungsmodell verbunden und kann sehr einfach als Closed-Loop-System implementiert und genutzt werden.

3.2.1.2 Einsatzgebiete und Randbedingungen einer MiL-Testumgebung

Der Tester ist mit einer MiL-Testumgebung in der Lage bereits den funktionalen Systementwurf zu testen. Mit fortlaufender Entwicklung (analog zum allgemeinen V-Modell) kann der Tester ebenso

[22] Diese Testumgebung wird im Lehrplan nicht betrachtet und ist rein informativ.

Komponenten bis hin zu einem gesamten Steuergerät testen. Um die Tests durchzuführen, benötigt der Tester einen Rechner und die entsprechende Simulationssoftware inklusive des Umgebungsmodells. Das Umgebungsmodell wird mit zunehmendem Funktionsumfang des Testobjektes immer komplexer. Die Abbildung von Realität und Umweltfaktoren ist sehr aufwändig. Auch die Rechenzeit für die Modelle steigt überproportional an. Deshalb lohnt sich ab einer bestimmten Entwicklungsphase der Aufwand eine MiL-Testumgebung einzusetzen nicht mehr.

Mit einer MiL-Testumgebung beschränkt sich der Tester auf Softwarekomponententests und Softwareintegrationstests. Das Umgebungsmodell zu befähigen Bus- und Diagnosefunktionen oder physikalisches Verhalten (wie zum Beispiel Kabelbrüche oder Kurzschlüsse) zu simulieren, ist nicht üblich. Diese Testaufgaben sind mit anderen Testumgebungen einfacher und günstiger realisierbar.

Bei der MiL-Testumgebung ist zu beachten, dass die Testdurchführung nicht in realer Zeit abläuft. Da alle Komponenten als Modell vorliegen, läuft die Testdurchführung in Simulationszeit. Je komplexer ein System ist, desto mehr Rechenzeit oder mehr Leistung braucht ein Rechner, um alle notwendigen Informationen zur Verfügung zu stellen. Die Simulationsdauer ist bei kleinen Systemen geringer als die Ausführung in der realen Zeit.

Ein großer Vorteil ist jedoch, dass der Tester die Simulation zu jedem Zeitpunkt pausieren kann, um detaillierte Analysen und Bewertungen durchzuführen.

3.2.2 Software in the Loop (SiL) (K1) [10 Min]

3.2.2.1 Aufbau einer SiL-Testumgebung

Das Testobjekt ist für eine SiL-Testumgebung kompiliert. Das bedeutet, dass der Quellcode mit einem Softwarewerkzeug für eine bestimmte Rechnerarchitektur kompiliert ist. Dieser Maschinencode ist für den Tester nicht mehr lesbar, da es sich um binäre Datensätze handelt. Damit die Testumgebung auf Signale zugreifen kann, ist ein Wrapper notwendig. Ein Wrapper ist eine zusätzliche Softwarefunktion, die spezielle Zugangsschnittstellen im Maschinencode erzeugt. Somit kann der Tester Signale von außen stimulieren und beobachten. Der Wrapper bestimmt die Zugriffsstellen im Testobjekt und besitzt keine funktionalen Aufgaben des Testobjektes. Die Anzahl der Zugriffsstellen ist durch den Wrapper und das Testobjekt beschränkt.

Für die Simulation ist ein Umgebungsmodell notwendig. Das Testobjekt ist mit Hilfe des Wrappers mit der Testumgebung verbunden. Die Testdurchführung erfolgt auf einem Rechner ohne spezielle Hardware. Die Softwarewerkzeuge müssen in der Lage sein, für das Testobjekt einen Wrapper zu erzeugen, um Zugriffsschnittstellen für das Umgebungsmodell bereitzustellen.

3.2.2.2 Einsatzgebiete und Randbedingungen einer SiL-Testumgebung

Die Signale und Variablen im Testobjekt haben in einer SiL-Testumgebung feste Datentypen und Speicherbereiche. Somit kann der Tester hier Datentypprüfungen und Speicherüberläufe testen. Des Weiteren kann der Tester erste funktionale Komponenten- und Integrationstests durchführen. Übliche Testverfahren sind Grenzwertanalysen und Äquivalenzklassenbildung (vergleiche CTFL Testmethoden). Diese Tests beziehen sich auf Softwarekomponenten, da in der SiL-Testumgebung noch keine Steuergerätehardware zum Einsatz kommt.

Der Tester lässt die Tests, analog zur MiL-Testumgebung, in Simulationszeit durchlaufen. In Abhängigkeit von der Rechentechnik und der Komplexität des Umgebungsmodells kann diese Simulationszeit geringer oder länger als die reale Zeit sein. Der Tester kann dabei die Durchführung jederzeit pausieren, um detaillierte Analysen und Bewertungen durchzuführen. Schnittstellen- und Integrationstests sind zwei Vertreter, die in einer SiL-Testumgebung Anwendung finden. Last- und Stresstests sind untypisch für eine SiL-Testumgebung. Diese Tests lassen sich in anderen Testumgebungen einfacher durchführen.

3.2.3 Hardware in the Loop (HiL) (K2) [20 Min]

3.2.3.1 Aufbau einer HiL-Testumgebung

Ist das Testobjekt als Muster vorhanden oder bereits fertig entwickelt, kann der Tester eine HiL-Testumgebung nutzen, um Test durchzuführen. Die typischen Bestandteile einer HiL-Testumgebung sind:

- Eine Stromversorgung, um verschiedene Versorgungsspannungen einzustellen
- Ein echtzeitfähiger Rechner, auf dem das Umgebungsmodell läuft
- Diverse Realteile, die nicht im Umgebungsmodell implementiert sind
- Eine Signalverarbeitung für die Wandlung von Signalart und Signalamplitude
- Eine elektrische Fehlersimulation für die Simulation von Kabelbruch und Kurzschlüssen
- Eine Breakoutbox als zusätzliche Zugriffsschnittstelle im Kabelbaum
- Eine Restbussimulation für die Simulation der nicht vorhandenen Busteilnehmer

Die Zugangsschnittstellen in einer HiL-Testumgebung sind vielfältig. Der Tester muss die verschiedenen Zugangsschnittstellen und deren Zusammenhänge in der HiL-Testumgebung kennen. Dem Tester muss bewusst sein, dass die Verwendung von falschen Zugängen zum Testobjekt die Testergebnisse nutzlos machen kann. Nur mit diesem Wissen kann der Tester die Tests in guter Qualität realisieren, durchführen und beurteilen.

3.2.3.2 Einsatzgebiete und Randbedingungen einer HiL-Testumgebung

Die HiL-Testumgebung ist auf Grund ihrer vielen Bestandteile komplexer, im Vergleich zu den vorher angesprochenen Testumgebungen (MiL und SiL). Der Tester muss diese Komplexität beherrschen, um seine Testaufgaben lösen zu können. Die HiL-Testumgebung kann bei Komponententests, Integrationstests und Systemtests eingesetzt werden. Ziel ist unter anderem funktionale und nicht funktionale Fehler in der Software und Hardware zu finden.

Mit Hilfe von HiL-Testumgebungen sind verschiedene Integrationsstufen analysierbar. Besteht das Testobjekt aus einem einzigen Steuergerät, handelt es sich um einen Komponenten-HiL. Ist das Testobjekt ein Verbund aus mehreren Steuergeräten, ist es ein System-HiL. Der Tester nutzt den Komponenten-HiL, um Funktionen des Steuergerätes zu testen. Beim System-HiL liegt der Fokus auf das Testen des Datenaustausches zwischen den Steuergeräten und auf dem Systemtest des Gesamtsystems.

Anders als bei den Testumgebungen zuvor (MiL und SiL) läuft die Simulationszeit bei der HiL-Testumgebung immer in realer Zeit. Ein wesentlicher Grund ist, dass die Software auf einer realen Hardware läuft. Ein Pausieren oder Stoppen ist in dieser Testumgebung nicht mehr möglich. Aus diesem Grund beinhaltet die Testumgebung einen echtzeitfähigen Rechner, der es schafft, alle relevanten Signale rechtzeitig zu erfassen und zu bedienen.

3.2.4 Gegenüberstellung der XiL-Testumgebungen (K3) [80 Min]

3.2.4.1 Vor- und Nachteile für das Testen anhand von Kriterien der XiL-Testumgebungen

Der Tester muss die Kriterien der verschiedenen Testumgebungen kennen. So kann er die Vor- und Nachteile für das Testen in der entsprechenden Umgebung verstehen und beurteilen. Die Kriterien sind in Tabelle 3 gegenübergestellt.

Kriterien	MiL-Testumgebung	SiL-Testumgebung	HiL-Testumgebung
Realitätsnähe	Gering	Gering bis Mittel	Hoch
	Realität wird simuliert, viele Eigenschaften werden abstrahiert, der Fokus liegt auf Strukturen und Logik	Kompilierte reale Software ist lauffähig (ohne Hardware)	Integriertes lauffähiges System
Kosten für Fehlerbehebung	Gering	Mittel	Hoch
	Fehler im Modell des Testobjekts gefunden (Modellanpassung)	Fehler in programmierter Software gefunden (Softwareanpassung)	Fehler auf Systemebene gefunden (Systemanpassung)
Aufwand für Inbetriebnahme und Wartung	Gering	Mittel	Hoch
	Umgebungsmodell erstellen	Umgebungsmodell und Wrapper erstellen	Umgebungsmodell erstellen und Hardwarekomponenten verkabeln
Aufwand für Testvorbereitung	Gering	Gering	Hoch
	Umgebung ist schnell einsatzfähig	Umgebung ist schnell einsatzfähig	Einrichtung, Inbetriebnahme und Auswertung der Tests erfordern einen hohen Aufwand
Notwendiger Reifegrad des Testobjektes	Gering	Mittel	Hoch
	Systemmodelle werden simuliert	Erste Funktionen werden mit der Zielsoftware getestet	Ein oder mehrere lauffähige Steuergeräte oder Teilsysteme werden möglichst vollständig getestet
Notwendige Detaillierungstiefe der Testbasis (Spezifikation)	Mittel	Mittel bis Hoch	Hoch
	Ohne vollständige Spezifikation werden Modelle getestet, welche sogar zum Teil zur Konkretisierung der Spezifikation beitragen	Die relevanten Informationen auf SW-Ebene müssen verfügbar sein (detaillierte Komponenten-Spezifikation)	Anforderungen lassen sich am System testen (vollständige System-Spezifikation)

Tabelle 3: Kriterien und deren Auswirkungen für MiL-, SiL- und HiL-Testumgebungen

3.2.4.2 Zuordnung durchzuführender Testfälle auf eine oder mehrere Testumgebungen

In der Teststrategie teilt der Testmanager den Testumfang auf mehrere verschiedene Testumgebungen auf. Die Entscheidung, welche Testumgebung zum Einsatz kommt, hängt davon ab, ob die Testumgebung, gemäß den oben genannten Kriterien (siehe Kap. 3.2.4.1), dazu geeignet ist, Testziele möglichst effizient zu erreichen. In der folgenden Tabelle sind wesentliche Testarten zur Erreichung der Testziele näher beschrieben und für die Testumgebungen bewertet.

Testart	Beschreibung	MiL	SiL	HiL
Kundenanforderungen testen	Korrekte Erbringung der geforderten Funktionalität. Dazu gehören die korrekte Verarbeitung von Eingaben, die korrekte Reaktion auf Eingaben sowie die korrekte Datenausgabe am Ausgang.	O	O	+
Mechanismen zur Fehlererkennung und -behandlung testen	• Erkennung und Behandlung von zufälligen Hardwarefehlern • Erkennung und Behandlung von Softwarefehlern • Übergang in einen sicheren Zustand bei erkannten Fehlern - zum Beispiel Deaktivierung eines Systems	+	+	+
Reaktion auf Konfigurationsdaten testen	Überprüfung des Einflusses von Konfigurationsdaten (wie Parametersätze oder Variantenkodierung) auf das Verhalten des Testobjektes.	O	+	+
Diagnosefunktionen testen	Korrekte Erbringung der geforderten Diagnose-Funktionalität wie zum Beispiel die Fehlererkennung sowie Fehlersetz- und Rücksetzbedingung, die Fehlerablage im Fehlerspeicher (zum Beispiel On-Board-Diagnose oder in der Werkstatt).	-	+	+
Interaktion an Schnittstellen testen	Interne und externe Schnittstellen des Testobjektes prüfen.	O	+	+
Nachweis der Benutzbarkeit führen	Das betrachtete Testobjekt soll wie gefordert und wie vom Benutzer erwartet benutzbar sein.	-	O	+
Legende: + empfohlen, o möglich, - nicht sinnvoll				

Tabelle 4: Vergleich von Testzielen in MiL-, SiL- und HiL-Testumgebungen

Diese Tabelle verdeutlicht, dass Testumgebungen für bestimmte Testziele besser oder schlechter geeignet sind. Diese diversitäre Ausrichtung macht sich insbesondere beim Testen der Mechanismen zur Fehlererkennung und -behandlung deutlich: Nach dem Prinzip des Frontloading[23] sollen grundlegende Anforderungs- und Designfehler bereits frühzeitig durch Testen aufgedeckt werden. Also bei MiL Erkennen von prinzipiellen Entwurfsfehlern, bei SiL von meistens technischen Softwarefehlern und bei HiL von technischen Hardware-/Softwarefehlern. Wichtig ist weiterhin zu beachten, dass bis auf die Nachweise der Robustheit und Zuverlässigkeit, Effizienz und Leistungsfähigkeit sowie der Benutzbarkeit alle Testarten auf die funktionale Eignung des Testobjektes abzielen.

Der Tester kann durch die Kombination der beiden vorher eingeführten Tabellen aus der inhaltlichen Eignung (Tabelle 3) und der Kriterien für die prozessuale, wirtschaftliche Eignung (Tabelle 4) die optimale Testumgebung auswählen.

[23] je früher ein Fehler erkannt wird, desto besser

3.2.4.3 Einordnung der XiL-Testumgebungen (MiL, SiL, HiL) in das allgemeine V-Modell

Auf der linken Seite des allgemeinen V-Modells steht der technische Systementwurf. Der Tester kann diesen Entwurf mit einer MiL-Testumgebung testen. Sind das Testobjekt und die MiL-Testumgebung weiterentwickelt, kann der Tester ebenfalls Komponenten- und Integrationstests mit dieser Testumgebung durchführen.

Der Tester kann eine SiL-Testumgebung nutzen, wenn einzelne Komponenten des Testobjekts programmiert und kompiliert sind. Typische Tests für eine SiL-Testumgebung sind Komponenten- und Integrationstests. Diese sind auf der rechten verifizierenden Seite des V-Modells zu finden.

Bei Systemtests sind bestimmte Funktionalitäten des Testobjekts fertig entwickelt. Der Tester kann den Systemtest mit einer HiL-Testumgebung durchführen. [24]

Mit einer richtigen Einordnung der Teststufen kann der gesamte Testprozess in drei Hinsichten optimiert werden:

Maximierung der Qualität

- Fokussierung der Testintensität gemäß Fehlerrisiko
- Testlücken vermieden
- für jeden Testfall die optimale Teststufe gewählt

Minimierung der Testkosten

- für jedes Testziel die optimale Teststufe gewählt
- Verlagerung von Tests in frühere, kostengünstigere und virtuelle Teststufen
- durchgängige, abgestimmte Tests

Konformität bzgl. Norm ISO 26262

- Teststufen und Testabdeckung konsistent zueinander

4 Spezielle statische und dynamische Testverfahren [230 Min]

Begriffe

Programmierrichtlinie, Qualitätsmerkmal (von Anforderungen), Bedingungstest, Mehrfachbedingungstest, modifizierter Bedingungs-/Entscheidungstest (MC/DC-Test), Bedingungsüberdeckung, Back-to-Back-Test, Fehlereinfügen, Anforderungsbasierter Test

Lernziele

Statische Testverfahren

AUTFL-4.1.1 Sie können anhand von Beispielen den Zweck und die Arten und Verbindlichkeiten der MISRA-C:2012-Programmierrichtlinie erläutern. (K2)

AUTFL-4.1.2 Sie können beim Review von Anforderungen die Qualitätsmerkmale der ISO/IEC 29148:2011 anwenden, die für Tester relevant sind. (K3)

Dynamische Testverfahren

AUTFL-4.2.1 Sie können den modifizierten Bedingungs-/Entscheidungstest anwenden, um Testfälle zu erstellen, die einen definierten Überdeckungsgrad erzielen (K3)

AUTFL-4.2.2 Sie können den Nutzen von Back-to-Back-Tests anhand von Beispielen erläutern. (K2)

AUTFL-4.2.3 Sie können das Prinzip von Fehlereinfügungstests anhand von Beispielen erläutern. (K2)

AUTFL-4.2.4 Sie können das Prinzip von anforderungsbasierten Tests wiedergeben. (K1)

AUTFL-4.2.5 Sie können vom Kontext abhängige Kriterien bei der Wahl geeigneter und notwendiger Testentwurfsverfahren für Softwaretests anwenden. (K3)

4.1 Statische Testverfahren (K3) [75 Min]

Einführung

Statisches Testen ist das Prüfen von Arbeitsprodukten der Softwareentwicklung ohne diese auszuführen. Hierzu gehört die strukturierte Bewertung durch Personen (Review) und die werkzeuggestützte statische Analyse.

4.1.1 Die MISRA-C:2012-Programmierrichtlinien (K2) [15 Min]

Es gehört heute zum Stand der Technik, dass der Entwickler beim Programmieren Programmierrichtlinien einhält. Dies empfiehlt auch die ISO26262 bei sicherheitskritischer Software[24]. Programmierrichtlinien helfen, Anomalien in der Software zu vermeiden, die möglicherweise zu Fehlerwirkungen führen. Zugleich unterstützen sie den Entwickler dabei, die Wartbarkeit und Übertragbarkeit seiner Software zu verbessern.

[24] siehe auch [ISO 26262:2011] Teil 9 Tabelle 6

Der Standard MISRA-C:2012 [29] enthält Richtlinien für die Programmiersprache C. Er definiert zwei Arten von Richtlinien:

- Regeln sind durch statische Analysewerkzeuge prüfbar. Zum Beispiel, dass der Quelltext keine verschachtelten Kommentare enthält.
- Direktiven sind nicht vollständig durch statische Analysewerkzeuge prüfbar. Das liegt daran, dass sie sich eher auf Details des Entwicklungsprozesses oder Dokumente außerhalb der Software beziehen. Zum Beispiel, ob der Entwickler das implementierte Verhalten ausreichend dokumentiert hat.

Jede Richtlinie hat eine der drei Verbindlichkeiten:

- „empfohlene“ Richtlinien sollte der Entwickler befolgen, solange der Aufwand angemessen ist.
- „erforderliche“ Richtlinien darf der Entwickler nur missachten, wenn er dies nachvollziehbar begründen kann.
- „verbindliche“ Richtlinien muss der Entwickler einhalten. Ausnahmen sind nicht erlaubt.

Organisationen können für sich die Verbindlichkeit einer Regel oder Direktive verschärfen, jedoch nie abschwächen.

4.1.2 Qualitätsmerkmale für Reviews von Anforderungen (K3) [60 Min]

Anforderungen sind die Grundlage für die Entwicklung und den Test. Daher führen Fehlerzustände in diesen Anforderungen zu kosten- und zeitintensiven Folgeaktivitäten. Dies gilt vor allem dann, wenn man die Fehler erst in späten Entwicklungsphasen wie dem Abnahmetest oder im Betrieb entdeckt. Reviews sind eine effektive Maßnahme, um Fehler in Anforderungen bereits früh zu finden und in der Folge frühzeitig und kostengünstig beheben zu können.

Während der Testanalyse muss der Tester die Anforderungen an das Testobjekt prüfen [2]. Dabei werden die Anforderungen insbesondere hinsichtlich ihrer Eignung als Testbasis geprüft. Qualitätsmerkmale helfen dem Tester während des Reviews der Anforderungen, seine Aufmerksamkeit zu fokussieren und möglichst viele Fehler zu finden. Die ISO/IEC/IEEE 29148:2011 [30] enthält sowohl Qualitätsmerkmale für einzelne Anforderungen als auch für Mengen von Anforderungen.

Für Tester relevante Qualitätsmerkmale nach ISO/IEC/IEEE 29148:2011

Qualitätsmerkmale für einzelne Anforderungen beziehungsweise eine Menge von Anforderungen:

- verifizierbar: Jede Anforderung lässt sich durch statische oder dynamische Tests verifizieren.
- eindeutig: Jede Anforderung enthält eindeutige Testbedingungen.
- konsistent: Jede Anforderung ist in sich und zu anderen Anforderungen widerspruchsfrei.
- vollständig: Jede Anforderung berücksichtigt alle möglichen Fälle (auch Fehler-, Abbruch- und Ausnahmeszenarien). Zugleich sind alle verwendeten Tabellen und Diagramme beschriftet; verwendete Abkürzungen und Begriffe sind definiert.
- rückverfolgbar: Jede Anforderung ist eindeutig gekennzeichnet (zum Beispiel durch eine ID). Dadurch wird eine Auswirkungsanalyse möglich und die Überdeckung durch Testfälle ist nachvollziehbar.
- abgrenzbar: Es ist klar abgegrenzt, was der zu entwickelnde und damit der zu testende Umfang ist.
- atomar: Keine Anforderung ist weiter in sinnvolle Teilanforderungen zerlegbar.

Als Hilfsmittel für das Review kann der Tester aus den Merkmalen zum Beispiel Review-Checklisten ableiten. Diese Review-Checklisten enthalten dann zu den zuvor genannten Aussagen passende Fragen. Der Tester muss sie nach bestem Wissen und Gewissen beantworten. Die folgende Liste enthält einen Auszug möglicher Fragen, die zu jeder Anforderung beantwortet werden müssen:

- verifizierbar: Ist die Anforderung durch statische oder dynamische Tests auf der entsprechenden Teststufe verifizierbar?
- eindeutig: Lässt die Anforderung keinen Interpretationsspielraum zu bzw. baut sie nicht auf implizitem Wissen oder Erfahrungswissen auf?
- konsistent: Ist die Anforderung in sich und zu anderen Anforderungen widerspruchsfrei?
- atomar: Ist die Anforderung *nicht* in weitere Teilanforderungen zerlegbar, zum Beispiel indem logische Verknüpfungen wie if-then-else-Konstrukte innerhalb der Anforderung aufgelöst und die resultierenden Teilanforderungen separat aufgeschrieben werden?

Nach [30] sollten Anforderungen auch realisierbar, lösungsneutral und notwendig sein. Diese Merkmale kann der Tester allerdings meistens nur schwer bewerten, beziehungsweise sie beeinflussen den Testentwurf nur gering.

4.2 Dynamische Testverfahren (K3) [155 Min]

4.2.1 Bedingungstest, Mehrfachbedingungstest, modifizierter Bedingungs-/Entscheidungstest [60 Min]

Die hier beschriebenen Verfahren gehören zu den White-Box-Testentwurfsverfahren (für Details und Beispiele siehe auch Lehrplan CTAL-TTA). Der Tester leitet die Testfälle direkt aus der Struktur des Testobjekts ab (zum Beispiel aus dem Quellcode).

Im Vergleich zum Entscheidungstest, bei dem der Tester die Testfälle im Hinblick auf die Überdeckung der Entscheidungsausgänge entwirft (siehe [2]), betrachten Bedingungstests die Entscheidungs*eingänge*. Diese Verfahren befassen sich also damit, *wie* eine Entscheidung getroffen wird: Jede Entscheidung besteht aus einer oder aus mehreren atomaren Bedingungen. Führt der Tester einen Testfall aus, ergibt sich für jede dieser Bedingungen der Wert „wahr“ oder „falsch“. Aus der logischen Kombination dieser Werte resultiert dann der Wert der Entscheidung [7].

Besteht eine Entscheidung aus nur einer atomaren Bedingung, sind diese Verfahren mit dem Entscheidungstest identisch. Ansonsten unterscheiden sich diese Verfahren wie folgt [31]:

- (einfacher) Bedingungstest (A): Der Tester entwirft Testfälle mit dem Ziel, die einzelnen Werte der atomaren Bedingungen zu testen. Bei unkluger Wahl der Testdaten (siehe Tabelle 5) testet der Tester bei 100% (einfacher) Bedingungsüberdeckung *nicht* zugleich alle Werte der resultierenden Entscheidung!
- Mehrfachbedingungstest (B): Der Tester entwirft Testfälle mit dem Ziel, die Kombinationen von Werten der atomaren Bedingung zu testen. Wenn jede Kombination von Werten getestet ist, ist damit auch jeder Entscheidungsausgang getestet.
- modifizierter Bedingungs-/Entscheidungstest (MC/DC-Test) (C): Wie bei Mehrfachbedingungstest. Jedoch betrachtet das Verfahren nur Kombinationen, bei denen eine atomare Bedingung unabhängig den Wert der Entscheidung beeinflusst.

Tabelle 5 zeigt anhand eines Beispiels die für jeweils 100% Überdeckungsgrad benötigten Testfälle in Abhängigkeit vom gewählten Verfahren:

	atomare Bedingung		**Entscheidung**	**Verfahren**		
Testfall	B1	B2	E = B1 UND B2	A	B	C
TF 1	WAHR	FALSCH	FALSCH	X	X	X
TF 2	FALSCH	WAHR	FALSCH	X	X	X
TF 3	WAHR	WAHR	WAHR		X	X
TF 4	FALSCH	FALSCH	FALSCH		X	

Tabelle 5: Gegenüberstellung der Verfahren Bedingungstest (A), Mehrfachbedingungstest (B) und modifizierter Bedingungs-/Entscheidungstest (MC/DC-Test) (C)

Das Beispiel zeigt die Grenzen der einzelnen Verfahren: Im Fall des (einfachen) Bedingungstests (A) gelingt es dem Tester trotz einer Bedingungsüberdeckung von 100% nur *einen* Entscheidungsausgang zu testen. Eine geschicktere Auswahl der Testfälle würde hier bereits Abhilfe schaffen (im Beispiel TF 3 und TF 4).

Mit Hilfe des Mehrfachbedingungstests (B) kann der Tester alle Ein- und Ausgänge abdecken. Jedoch ist bei diesem Verfahren die Anzahl der durchzuführenden Tests am höchsten.

Durch Anwendung des modifizierten Bedingungs-/Entscheidungstests (C) kann der Tester eine vollständige Abdeckung aller atomaren Bedingungen und aller Entscheidungen mit einer geringeren Zahl von Tests realisieren.

4.2.2 Back-to-Back-Test (K2) [15 Min]

Der Back-to-Back-Test (auch: vergleichender Test [32]) vergleicht zwei oder mehr Varianten eines Testobjekts. Dabei führt der Tester den gleichen Testfall auf allen Varianten durch und vergleicht die Ergebnisse. Sind die Ergebnisse identisch, so ist der Test bestanden. Weichen die Ergebnisse ab, wird die Ursache der gefundenen Abweichung analysiert.

Die Testobjekte müssen auf inhaltlich gleichen Anforderungen basieren. Nur so können diese ein vergleichbares Verhalten zeigen. Die Anforderungen dienen dabei nicht zwingend als Testbasis für den Testentwurf. Vielmehr sollen durch den Back-to-Back-Test ungewollte kleinste Abweichungen zwischen den Testobjekten aufgezeigt werden. Dieser Test ersetzt daher nicht den anforderungsbasierten Test.

Im einfachsten Fall handelt es sich bei den Testobjekten eines Back-to-Back-Tests um unterschiedliche Versionen der gleichen Software. Hier dient zum Beispiel eine frühere Version des Testobjekts als Testorakel für den Regressionstest [33]. Eine andere Alternative ist der Vergleich eines ausführbaren Modells mit dem (manuell oder automatisch) generierten Code [32]. In diesem Fall handelt es sich um eine Form des modellbasierten Testens, bei dem das ausführbare Modell auch als Testorakel dient [34]. Dieses Verfahren eignet sich aus diesem Grund sehr gut für den automatisierten Testentwurf. Hier leitet der Tester neben dem erwarteten Ergebnis auch Testfälle automatisiert aus dem Modell ab.

4.2.3 Fehlereinfügungstest (K2) [15 Min]

Programmiertechnische Verfahren wie die Fehlerbehandlung dienen dazu, dass das System robust und sicher auf interne und externe Fehler reagiert. Um diese Verfahren zu testen, kann der Tester gezielt Fehler an folgenden Punkten ins System einfügen [34]:

- Fehler in externen Komponenten: Wenn das System zum Beispiel unplausible Werte von Sensoren sicher erkennen muss.
- Fehler an Schnittstellen: Wenn zum Beispiel die Funktion des Systems durch Kurzschlüsse oder verloren gegangene Nachrichten nicht beeinträchtigt sein darf.
- Fehler in der verarbeitenden Einheit: Wenn das System interne Fehler erkennen und behandeln soll.

Beim klassischen Fehlereinfügen fügt der Tester einen Fehler durch Manipulation einer realen Komponente ein.

Externe Fehler (auch Schnittstellenfehler) kann der Tester bereits zur Laufzeit simulieren. Das Einfügen der Fehler erfolgt in der Regel in einer HiL-Testumgebung. Hier dient eine Fehlereinfügungs-Komponente [35] als Treiber für physikalische Fehlfunktionen. Zu diesen Fehlfunktionen zählen vor allem Kurzschlüsse und offene Leitungen. Die Simulation softwareseitiger Schnittstellenfehler kann häufig bereits in einer SiL-Testumgebung erfolgen.

Fehler in der verarbeitenden Einheit können meist nur in der Entwicklungsumgebung zum Beispiel mittels Debugger oder XCP eingefügt werden. Die Durchführung ist daher in der Praxis oft sehr zeitintensiv.

4.2.4 Anforderungsbasierter Test (K1) [5 Min]

Anforderungsbasierte Tests ist ein Ansatz (eine Praktik) zum Testen [23] und im engeren Sinne kein Testverfahren. Der Ansatz verfolgt das Ziel, die Anforderungen mit Testfällen zu überdecken. Dadurch bestimmt der Tester, ob das Testobjekt die Kundenanforderungen erfüllt.

Bei diesem Ansatz analysiert der Tester die Anforderungen, leitet daraus Testbedingungen ab, entwirft Testfälle und führt diese aus. Auf Basis der Analyse der Testergebnisse verfeinert er die Tests. Dabei können auch weitere Testfälle entstehen. Ergänzend wendet der Tester weitere Testpraktiken (wie zum Beispiel erfahrungsbasiertes Testen) an. So kann er beispielhaft durch Regressionstests in Form von explorativen Tests das Risiko maskierter Fehler reduzieren.

Sind die Anforderungen unvollständig oder nicht konsistent, leiden die auf dieser Basis entworfenen Tests an den gleichen Problemen. Andererseits kann der Tester bei sehr detaillierten Anforderungen gegebenenfalls nicht alle testen. Hier ist eine Priorisierung der Testfälle unumgänglich. [3]

4.2.5 Kontextabhängige Auswahl von Testverfahren (K3) [60 Min]

Die ISO 26262 (Teil 6) empfiehlt dem Tester anzuwendende Testentwurfsverfahren (siehe Kapitel 2.2) in Abhängigkeit vom ASIL. Diese enthalten unter anderem die im CTFL und zuvor in Kapitel 4.2 behandelten Verfahren:

- anforderungsbasierter Test
- Äquivalenzklassenbildung
- Grenzwertanalyse
- Anweisungstest
- Entscheidungstest
- Modifizierter Bedingungs-/Entscheidungstest (MC/DC)
- intuitive Testfallermittlung
- Fehlereinfügen
- Back-to-Back Test

Doch ob ein Tester ein Verfahren tatsächlich anwendet, hängt unter anderem von den folgenden Faktoren ab:

Stand der Technik

Entspricht das Verfahren für den Zweck dem aktuellen Stand der Technik? Hier helfen Normen wie die ISO 29119 und die ISO 26262. Die ISO 26262 empfiehlt sogar in Abhängigkeit des ASIL anwendbare Verfahren. Auf Abweichungen von den Empfehlungen der Norm wird in Kapitel 2.2 zur ISO 26262 eingegangen.

Testbasis

Liefert die Testbasis für das Verfahren geeignete Testbedingungen? So kann der Tester zum Beispiel nur Äquivalenzklassen bilden, wenn die Testbasis Parameter oder Variablen enthält. Dabei muss er deren Werte zu sinnvollen Äquivalenzklassen zusammenfassen können. Ähnliches gilt für Grenzwerte. Diese kann er nur testen, wenn der Wertebereich linear geordnet ist.

Risikobetrachtung

Stellt ein möglicher Fehlerzustand ein Risiko dar? Risikoorientiertes Testen bedeutet die Identifikation von Produktrisiken und das Heranziehen der Risikostufe zur Auswahl der Verfahren. So ist der Test eines Grenzwertes nur sinnvoll, wenn die Grenze ein Risiko darstellt.

Teststufe

Ist das Verfahren auf der Teststufe sinnvoll anwendbar? White-Box-Tests eignen sich vor allem dann, wenn der Quellcode oder die interne Struktur als Testbasis dient. Im Idealfall ist der strukturelle Überdeckungsgrad messbar. Und auch für Black-Box-Tests muss das Testobjekt zugänglich und beobachtbar sein. So kann das Testen einer Äquivalenzklasse eines Sensors im Systemtest eventuell effizienter sein als im Komponententest. Ist ein Testentwurfsverfahren auf einer Teststufe nicht anwendbar, so sollte der Tester gemäß der Teststrategie des Testmanagers eine alternative Teststufe oder Maßnahme wählen.

Beispielhafte Auswahl der Testverfahren

Die folgende Tabelle enthält eine Liste von Testentwurfsverfahren ergänzt um eine beispielhafte Bewertung eines Nutzers anhand mehrerer, zuvor genannter Faktoren und die darauf basierende Auswahl der Testentwurfsverfahren.

	Testentwurfsverfahren	**Empfohlen für Anwendung bei ASIL A?**	**Testbasis geeignet?**	**Risiko, wenn Fehler nicht entdeckt wird?**	**Teststufe „Systemtest “ sinnvoll?**	**Auswahl**
1	anforderungsbasierter Test	++	JA	++	JA	X
3	Äquivalenzklassenbildung	+	JA	++	JA	X
4	Grenzwertanalyse	+	NEIN	-	JA	
5	Anweisungstest	++	JA	++	NEIN	
6	Entscheidungstest	+	JA	++	NEIN	
7	MC/DC	+	JA	+	NEIN	
8	intuitive Testfallermittlung	+	NEIN	++	JA	
9	Fehlereinfügen	+	JA	+	NEIN	
10	Back-to-Back Test	+	NEIN	++	JA	

Tabelle 6: Beispiel zur Auswahl eines Testverfahrens

Anhang

Datenbasen und Kommunikationsprotokolle aus der Automobilindustrie

Schnittstellen	Datenbasis	Kommunikationsprotokolle
Speicher	ASAM MCD-2 MC (auch ASAP2 oder A2L)	ASAM MCD-1 XCP (Universal Measurement and Calibration Protocol) ASAM standard CCP (CAN Calibration Protocol)
Bus	ASAM MCD2 NET standard (auch *FIBEX - Field Bus Exchange Format*)	FlexRay (ISO 17458) CAN (Controller Area Network nach ISO 11898-2)
	DBC (Kommunikations-Datenbasis für CAN)	CAN (Controller Area Network nach ISO 11898-2)
Diagnose	ASAM MCD2 D (auch *ODX*) CDD (CANdelaStudio diagnostic description)	*KWP2000 (ISO 14230)* *ISO-OBD (ISO 15031)* *UDS (ISO 14229)*

Tabelle 7: Gängige Datenbasen und Kommunikationsprotokolle aus der Automobilindustrie

AUTOSAR hat ein XML Format standardisiert, das die Datenbasen eines kompletten Fahrzeugs integriert. Dies ist das ARXML Format (AUTOSAR Integrated Master Table of Application Interfaces, XML Schema R3.0).

ASAM steht für “Association for Standardisation of Automation and Measuring Systems”.

Tabellenverzeichnis

Referenzen

[1] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), *ISO/IEC TR 24748-1:2010 Systems and software engineering - Life cycle management - Part 1: Guide for life cycle management,* 2010-10-01.

[2] International Software Testing Qualifications Board (ISTQB) / German Testing Board e.V. (GTB), *ISTQB/GTB Certified Tester Foundation Level (CTFL) Syllabus - Version 2011 1.0.1 - Deutsche Ausgabe,* German Testing Board e.V. (GTB), 2011.

[3] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers (IEEE), *ISO/IEC/IEEE 29119-1:2013 Software and systems engineering - Software testing - Part 1: Concepts and definitions,* 2013-09-01.

[4] Verband der Automobilindustrie e.V. (VDA) / QMC Working Group 13 / Automotive SIG, *Automotive SPICE Process Assessment Model,* Berlin: Verband der Automobilindustrie e. V. (VDA), 2008.

[5] AUTOSAR, „http://www.autosar.org/specifications/," [Online]. [Zugriff am 04 04 2016].

[6] ZVEI, *Best Practice Guideline - Software Release,* Frankfurt am Main,: ZVEI, 2016.

[7] International Software Testing Qualifications Board (ISTQB) / German Testing Board e.V. (GTB), *ISTQB/GTB Certified Tester Advanced Level (CTAL) Syllabus - Technical Test Analyst (TTA) - Deutsche Ausgabe,* German Testing Board e.V. (GTB), 2012.

[8] Verband der Automobilindustrie e.V. (VDA) / QMC Working Group 13, „Status and outlook VDA QMC working group 13 - Automotive SPICE 3.0, Blue-Gold Volume," in *Sixth VDA Automotive SYS Conference*, Berlin, 2016.

[9] Verband der Automobilindustrie e.V. (VDA) / QMC Working Group 13 / Automotive SIG, *Automotive SPICE Process Assessment / Reference Model,* http://www.automotivespice.com/download/, 2015 Version 3.0.

[10] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), *ISO/IEC 12207:2008 Systems and software engineering - Software life cycle processes,* International Organization for Standardization (ISO), 2008-02-01.

[11] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers (IEEE), *ISO/IEC/IEEE 15288:2015 Systems and software engineering - System life cycle processes,* 2015-15-05.

[12] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), *ISO/IEC 33020-03:2015 Informationstechnik – Prozessbewertung – Rahmenwerk für Prozessmessungen zur Beurteilung der Prozessfähigkeit,* 01-03-2015.

[13] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers (IEEE), *ISO/IEC/IEEE 29119-3:2013 Software and systems engineering - Software testing - Part 3: Test documentation,* 2013-09-01.

[14] AUTOSAR, „AUTOSAR - The worldwide Automotive Standard for E/E systems," *ATZ extra,* p. 5, 2013.

[15] AUTOSAR, „Project Objectives AUTOSAR Release 4.2.1," [Online]. Available: https://www.autosar.org/fileadmin/files/standards/classic/4-2/main/auxiliary/AUTOSAR_RS_ProjectObjectives.pdf. [Zugriff am 03 03 2016].

[16] AUTOSAR, „Main Requirements AUTOSAR Release 4.2.1," [Online]. Available: https://www.autosar.org/fileadmin/files/standards/classic/4-2/main/auxiliary/AUTOSAR_RS_Main.pdf. [Zugriff am 03 03 2016].

[17] T. Ringler, C. Dziobek und F. Wohlgemuth, „Tagungsband Modellbasierte Entwicklung eingebetteter Systeme - Chancen und Herausforderungen bei der virtuellen Absicherung verteilter Body&Comfort-Funktionen auf Basis von AUTOSAR - S.83 - 93," [Online]. Available: https://www.in.tu-clausthal.de/fileadmin/homes/GI/Documents/MBEES15Proceedings.pdf. [Zugriff am 27 09 2016].

[18] U. Freund, V. Jaikamal und J. Löchner, „Multilevel System Integration of Automotive ECUs based on AUTOSAR," [Online]. Available: http://papers.sae.org/2009-01-0918/. [Zugriff am 27 09 2016].

[19] Patzer und Zaiser, „Einsatzgebiete für XCP," in *XCP-Das Standardprotokoll für die Steuergeräte Entwicklung*, Stuttgart, Vector Informatik GmbH, 2014.

[20] AUTOSAR, „Acceptance Test Main Requirements AUTOSAR TC Release 1.1.0," [Online]. Available: https://www.autosar.org/fileadmin/files/standards/tests/tc-1-1/general_auxiliary/AUTOSAR_ATR_Main.pdf. [Zugriff am 2016 03 03].

[21] AUTOSAR, „Requirements on Acceptance Test AUTOSAR TC Release 1.1.0," [Online]. Available: http://www.autosar.org/fileadmin/files/standards/tests/tc-1-1/general_auxiliary/AUTOSAR_ATR_Requirements.pdf. [Zugriff am 2016 12 12].

[22] International Organization for Standardization (ISO), *ISO 26262:2011 Road Vehicles - Functional Safety,* Genf, 2011.

[23] International Software Testing Qualifications Board (ISTQB) / German Testing Board e.V. (GTB), *ISTQB/GTB Standardglossar der Testbegriffe Version 3.1,* Erlangen: German Testing Board e.V. (GTB), 13. April 2016.

[24] A. Spillner und T. Linz, Basiswissen Softwaretest [Elektronische Ressource] : Aus- und Weiterbildung zum Certified Tester - Foundation Level nach ISTQB-Standard, Heidelberg: dpunkt.verlag, 2012.

[25] A. Spillner, T. Roßner, M. Winter und T. Linz, Praxiswissen Softwaretest Testmanagement: Aus- und Weiterbildung zum Certified Tester - Advanced Level nach ISTQB-Standard, Heidelberg: dpunkt.verlag, 2008.

[26] G. Baumann, „Was verstehen wir unter Test? Abstraktionsebenen, Begriffe und Definitionen," FKFS 1. AutoTest; Fachkonferenz zum Thema Test und Diagnose in der Automobilentwicklung., Stuttgart, 2006.

[27] H. Wallentowitz, Handbuch Kraftfahrzeugelektronik : Grundlagen, Komponenten, Systeme, Anwendungen ; mit zahlreichen Tabellen, Wiesbaden: Vieweg, 2016.

[28] Measuring, Association for Standardization of Automation and, „http://asam.net/," 2016. [Online]. [Zugriff am 2016].

[29] MISRA Electrical Group MIRA Ltd., *MISRA-C:2012-Programmierrichtlinien – Version 3.,* UK, Warwickshire, 2013.

[30] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers (IEEE), *ISO/IEC/IEEE 29148:2011 - Systems and software engineering - Life cycle processes - Requirements engineering,* 2011-12-01.

[31] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers (IEEE), *ISO/IEC/IEEE 29119-4:2015 Software and systems engineering - Software testing - Part 4: Test techniques,* Bd. 4, 2015.

[32] M. Conrad und G. Sandmann, „A Verification and Validation Worflow for IEC 61508 Applications," *SAE International,* 2009.

[33] H.-W. Wiesbrock, M. Conrad, I. Fey und H. Pohlheim, „Ein neues automatisiertes Auswertverfahren für Regressions- und Back-to-Back-Tests eingebetteter Regelsysteme," *Softwaretechnik-Trends,* Bd. 22, 2002.

[34] C. Hobbs, Embedded Software Development for Safety-Critical Systems, Taylor & Francis Group, 2016.

[35] National Instruments Germany GmbH, „Einsatz von Fault Insertion Units (FIUs) für die Überprüfung elektronischer Steuergeräte," Nr. 25. Juni, 2015.

[36] AUTOSAR, „Glossary AUTOSAR Release 4.2.2," [Online]. Available: https://www.autosar.org/fileadmin/files/standards/classic/4-2/main/auxiliary/AUTOSAR_TR_Glossary.pdf. [Zugriff am 03 03 2016].

[37] K. Borgeest, *Elektronik in der Fahrzeugtechnik,* Springer Vieweg, 2014.

[38] R. Schönfeld, Regelungen und Steuerungen in der Elektrotechnik, Verlag Technik GmbH, 1993.

[39] Verband der Automobilindustrie e.V. (VDA), *Sicherung der Qualität in der Prozesslandschaft,* Bd. Band 4, Verband der Automobilindustrie e.V. (VDA), 2011.

[40] Verband der Automobilindustrie e.V. (VDA), *Entwicklung softwarebestimmter Systeme - Forderungen an Prozesse und Produkte,* Bd. 13, Verband der Automobilindustrie e.V. (VDA), 2004.

[41] M. Winter, M. Ekssir-Monfared, H. M. Sneed, R. Seidl und L. Borner, Der Integrationstest: Von Entwurf und Architektur zur Komponenten- und Systemintegration, München: Carl Hanser Verlag GmbH & Co. KG, 2012.

[42] K. Hoermann, M. Mueller, L. Dittmann und J. Zimmer, Automotive SPICE in Practice in der Praxis – Interpretationshilfe für Anwender und Assessoren, Heidelberg: dpunkt verlag GmbH, 2. Auflage, 2016.

[43] dpa, „www.motor-talk.de," 24 02 2015. [Online]. Available: http://www.motor-talk.de/news/die-zahl-der-modelle-waechst-der-absatz-nicht-t5219608.html. [Zugriff am 12 12 2016].

Definitionen

Die folgenden Begriffe werden in diesem Lehrplan verwendet und sind ergänzend zum ISTQB® Glossar [ISTQB®2015]:

Begriff	Definition / Bedeutung	Referenz
Automotive Open System Architecture	In 2003 gegründete Entwicklungspartnerschaft mit dem Ziel einen offenen Industrie-Standard für eine Software Architektur in der Autoindustrie zu schaffen und zu etablieren.	
Automotive Safety Integrity Level	Maß für die notwendige Risikoreduzierung durch spezielle Maßnahmen der Funktionalen Sicherheit (zum Beispiel durch separate Sicherheitsfunktionen) in einem E/E-System. Der ASIL ergibt sich aus der Gefahrenanalyse und Risikobewertung.	[22]
Basissoftware	(AUTOSAR): standardisierte, hardwarenahe Softwarekomponenten	[36]
Breakoutbox	Eine Messeinrichtung, um physikalische Signale in Leitungen zu analysieren, zu unterbrechen oder zu manipulieren.	[27]
Bussystem	Netzwerk aus mehreren Steuergeräten, die Nachrichten über die gleichen Verbindungen untereinander austauchen.	[37]
Closed-Loop-System	Ein geschlossener Wirkungskreis, der als Regler fungiert.	[38]
Codereview	Eine Eignungsprüfung des Codes gegen den vorgesehenen Zweck und Abweichungsanalyse von vergebenen Spezifikationen und Standards.	[9]
Direktive (MISRA)	Eine Programmierrichtlinie in MISRA-C:2012, die nicht vollständig durch statische Analysewerkzeuge prüfbar ist.	[29]
E/E/PE-System	Funktionales System aus elektrischen, elektronischen oder programmierbar elektronischen Elementen.	[22]
E/E-System	Funktionales System aus elektrischen oder elektronischen Elementen.	[22]
Echtzeitfähiger Rechner	Siehe Echtzeitrechner. Eine Recheneinheit, die die Verarbeitung von Signalen in einem definierten Zeitfenster garantiert.	[27]

ECU-Extrakt	Beinhaltet die Daten für ein Steuergerät aus der System-Konfigurationsbeschreibung.	[36]
ECU-Konfigurationsbeschreibung	Umfasst Daten zur Integration der SW Komponenten auf dem Steuergerät.	[36]
Entwicklung	Phase in der die aus dem Konzept resultierende Aufgabenstellung durch das Entwickeln eines vermarktbaren und produzierbaren Produktes gelöst wird. Im PEP im automobilen Umfeld auch als Produktentwicklung bekannt [39]	[11]
Erstausrüster	Siehe Originalausrüstungshersteller	[4]
Fähigkeitsdimension	In der Fähigkeitsdimension wird eine in Fähigkeitsstufen unterteilte Menge an Prozessattributen definiert. Die Prozessattribute liefern die messbaren Eigenschaften der Prozessfähigkeit.	[9]
Fähigkeitsindikator	Fähigkeitsindikatoren sind Indikatoren, welche zur Durchführung und Begründung einer Prozessfähigkeitsbewertung herangezogen werden können.	[9]
Fähigkeitsstufe	Eine Fähigkeitsstufe besteht aus ein oder mehreren Prozessattributen, bei deren hinreichender Erfüllung eine signifikante Verbesserung der Prozessfähigkeit festgestellt werden kann.	[9]
Fehlereinfügungs-Komponente	(engl. Fault Insertion Unit) Mit einer Fehlereinfügungs-Komponente können physikalische Fehler an den Schnittstellen einer Hardwarekomponente simuliert werden (wie zum Beispiel Kurzschluss und Leerlauf)	
Fehlerliste	Fehlerliste: Welche (bereits bekannten) Fehler wurden behoben, welche nicht.	[6]
Freigabe	Formale Entscheidung auf der Basis der Projektergebnisse über den Reifegrad eines Produktes/Objektes zur offiziellen Übergabe an den nächsten Prozessschritt.	[40]
Freigabebestimmung	Bestimmung (Release Purpose) für den das Freigabeobjekt verwendet werden kann bzw. darf.	[6]
Freigabeempfehlung	Empfehlung durch den Tester bzw. den Testmanager, dass das Freigabeobjekt aufgrund der Testergebnisse freigegeben	[6]

	werden kann (oder nicht).	
Freigabeobjekt	Freigabeobjekt (Release-Item) besteht aus dem Testobjekt sowie der Begleitdokumentation.	[6]
Freigabe-Prozess	Prozess der zur Freigabe führt.	[6]
Funktionsliste	Die Funktionsliste enthält die geplanten Funktionen zum Release, dazu gehören aber auch die bekannten (neuen und alten, nicht behobene) Fehler.	[6]
Komponenten-HiL	Eine Testumgebung für die Abbildung eines einzelnen Steuergerätes.	[27]
Konzeption	Phase in der als Grundlage für ein Projekt aus einem Entwurf bzw. einer Grundidee ein Konzept entsteht. Im PEP im automobilen Umfeld auch als Vorentwicklung bekannt [39]	[11, 1]
Langzeittest	Dauerläufer- oder Dauertest	[22]
Laufzeitumgebung	(AUTOSAR): Laufzeitumgebung ist die Abstraktionsschicht die den Datenaustausch zwischen den AUTOSAR Software Komponenten untereinander sowie zwischen Applikation und BSW steuert und umsetzt, sowohl innerhalb als auch außerhalb der Steuergeräte.	[36]
Methodentabelle	Tabellen in der ISO 26262 mit empfohlenen Techniken und Verfahren	[22]
Modellaufzeit	Laufzeit eines Produkts (zum Beispiel Automodell), vom Produktionsstart (SOP) bis zum Produktionsende (EOP)	[11, 1]
Multisystemtest	Dieser Test hat die Aufgabe, die Erfüllung der Anforderungen eines Systems von Systemen (auch: Multisystem) zu prüfen. Dieser Test ist vergleichbar mit dem Systemtest für ein einzelnes System.	[41]
Open–Loop-System	Übertragungsglieder, die als Steuerung hintereinander als Wirkkette aufgebaut sind.	[38]
Originalausrüstungshersteller	In der Automobilindustrie wird dieser Begriff verwendet, um Fahrzeughersteller zu beschreiben. Siehe auch "Tier 1... n".	[4]
Produktentstehungsprozess (PEP)	Prozess, der alle Tätigkeiten von der ersten Produktidee bis zur Herstellung umfasst.	[40]

Produktion	Erstellung des entwickelten Produktes. Im PEP im automobilen Umfeld auch als Fertigung / Serienfertigung bekannt [39]	[11, 1]
Programmierrichtlinie	Legt die erforderlichen Programmierpraktiken fest.	[7]
Prozessattribut	Ein Prozessattribut beinhaltet messbare Eigenschaften eines Prozesses zur Prozessfähigkeitsbewertung.	[9]
Prozessdimension	In der Prozessdimension werden sämtliche relevanten Prozesse definiert. Die Prozesse werden in Prozesskategorien und auf zweiter Ebene in Prozessgruppen zusammengefasst.	[9]
Regel (MISRA)	Eine Programmierrichtlinie in MISRA-C:2012, die durch statische Analysewerkzeuge prüfbar ist.	[29]
Regressionsteststrategie	Die Regressionsteststrategie legt fest, nach welchen Kriterien bei einer Änderung des Testobjekts die Regressionstestfälle ausgewählt werden.	
Restbussimulation	Virtualisierung der Buskommunikationsschnittstelle von nicht vorhandenen Steuergeräten.	
Sicherheitskultur (engl. Safety Culture)	Die unternehmensweit gelebte Einstellung, gemeinsam ein funktional sicheres Produkt zu entwickeln.	[22]
Sicherheitslebenszyklus	Produktlebenszyklus eines sicherheitsrelevanten Systems. Beginnt mit der Produktidee und endet mit der Entsorgung des Produkts am Ende seiner Lebensdauer.	[22]
Simulationszeit	Die Zeit, die für eine Computersimulation gilt.	[27]
Softwarekomponente	(AUTOSAR): Hardwareunabhängige Softwareschicht, enthält die individuellen Applikationen und Funktionalitäten.	[36]
System-HiL	Eine Testumgebung für die Abbildung eines Steuergeräteverbundes bis hin zum Gesamtfahrzeug.	[27]
System-Konfigurationsbeschreibung	Enthält Daten zur Integration aller Steuergeräte in einem Fahrzeug.	[36]
Systemlebenszyklus	Phasen der Entstehung und des Einsatzes eines Systems bis zu seiner endgültigen Außerbetriebnahme. Geht damit über den PEP	[40]

	hinaus.	
Testobjekt	Testobjekt im automobilen Kontext besteht aus einer Softwarekonfiguration inkl. Grundparametrierung und meist auch einer Hardware und einer Mechanik. Vergleiche Testobjekt im ISTQB Glossar [23]	[6]
Tier 1…n	Mit Tier 1...n werden die Lieferanten in der Lieferkette auf den verschiedenen Ebenen benannt. Die direkten Lieferanten des OEM werden mit Tier 1 bezeichnet, die Zulieferer eines Tier 1 werden mit Tier 2 bezeichnet usw.	[4]
Umgebungsmodell	Abstraktion der Fahrzeugprozesse in einer Echtzeitsimulation.	[27]
Verbauempfehlung	Zusatz zur SW-Freigabe, mit der der Zulieferer gegenüber dem OEM erklärt, dass das Freigabe-Objekt eine uneingeschränkte Freigabe für öffentliche Straßen erhält und dort verwendet/getestet werden darf.	
Verifikationskriterium	Ein Verifikationskriterium definiert qualitative und quantitative Kriterien, welche erfüllt werden müssen um ein Testobjekt erfolgreich zu verifizieren.	[9]
Verifikationsstrategie	In der Verifikationsstrategie wird das allgemeine Vorgehen zur Verifikation des zu verifizierenden Objekts beschrieben und die anzuwendenden Methoden als auch die Verifikationskriterien festgelegt.	[9]

Tabelle 8: verwendete Begriffe

Abkürzungen

Die folgenden Abkürzungen werden in diesem Lehrplan verwendet:

Abkürzung	Definition / Bedeutung (Deutsch)	Definition / Bedeutung (Englisch)	Referenz
ACQ	Beschaffung	Acquisition	[9]
App	Anwendungssoftware	Application	
ASIL		Automotive Safety Integrity Level	[22]
AUTOSAR		Automotive Open System Architecture	[36]
AUTOSIG		Automotive Specific Interest Group	[42]
BP	Basispraktik	Base Practice	[9]
BSW	Basis-Software	Base Software	[36]
CTFL		Certified Tester Foundation Level	
E/E	Elektrik / Elektronik	Electric / Electronic	
ECU	elektronisches Steuergerät	Electronic Control Unit	
EOP	Produktionsende	End-of-Production	
FIU		Fault Insertion Unit	[35]
FuSi	Funktionale Sicherheit		
G&R	Gefahrenanalyse und Risikobewertung		
GP	Generische Praktik	Generic Practice	[9]
HIL		Hardware-in-the-Loop	
HIS	Herstellerinitiative Software		[42]
IEC		International Electrotechnical Commission	
ISO		International Organization for Standardization	
ISTQB		International Software Testing Qualifications Board	
MAN	Management (ASPICE)	Management (ASPICE)	[9]

MC/DC	Modifizierte Bedingungs-/Entscheidungsüberdeckung	Modified Condition/Decision Coverage	
MISRA		Motor Industry Software Reliability Association	
OEM	Erstausrüster	Original Equipment Manufacturer	
PA	Prozessattribut	Process Attribute	[9]
PAM	Prozess-Assessmentmodell	Process Assessment Model	[9]
PEP	Produktentstehungsprozess	Product Evolution Process	[40]
PIM	Prozessverbesserung (ASPICE)	Process Improvement (ASPICE)	[9]
PLC	Produktlebenszyklus	Product Live Cycle	[40]
PRM	Prozess-Referenzmodell	Process Reference Model	[9]
QM	Qualitätsmanagement	Quality Management	
REU	Wiederverwendung (ASPICE)	Reuse (ASPICE)	[9]
RTE	Laufzeitumgebung	Run Time Environment	[36]
SOP	Produktionsstart	Start-of-Production	
SPICE		Software Process Improvement and Capability Determination	[9]
SPL	Zulieferung (ASPICE)	Supply (ASPICE)	[9]
SUP	Unterstützung (ASPICE)	Support (ASPICE)	[9]
SW	Software	Software	
SW-C	Softwarekomponente	Software Component	[36]
SWE	Softwareentwicklung (ASPICE)	Software Engineering (ASPICE)	[9]
SYS	Systementwicklung (ASPICE)	System Engineering (ASPICE)	[9]
VDA	Verband der Automobilindustrie		
WP	Arbeitsprodukt	Work Product	[9]
XCP		Universal Measurement and Calibration Protocol	[19]

Tabelle 9: verwendete Abkürzungen

Index

1