Lehrplan
Certified Tester
Automotive Software Tester (CT-AuT)

V2.1 DE

International Software Testing Qualifications Board

Urheberschutzvermerk

Unser herzlicher Dank geht an Ursula Zimpfer für ihre wertvolle Unterstützung bei der Bearbeitung der deutschsprachigen Fassung des vorliegenden Lehrplans.

Änderungsübersicht der deutschsprachigen Ausgabe

Version	Datum	Bemerkungen
V2.1 DE	23.11.2025	Deutschsprachige Fassung der englischen Version V2.1 des ISTQB freigegeben
V2.0.2 DE	13.10.2020	Deutschsprachige Fassung der englischen Version V2.0.2 des ISTQB freigegeben
V2.0 DE	31.03.2017	Neuentwickelte Fassung in deutscher Sprache auf Grundlage der Version V1.1 (Grundlage für die englischsprachige Fassung V2.0.2 des ISTQB) freigegeben
V1.1 DE	14.06.2015	Inhaltliche Überarbeitung und Abgleich mit dem deutschsprachigen ISTQB-Lehrplan Certified Tester Foundation Level 2011 V1.0.1 und dem ISTQB-Glossar V2.2 freigegeben
V1.0 DE	19.01.2011	Erste Veröffentlichung des Lehrplans unter der Bezeichnung „Certified Automotive Softwaretester (CAST)“ freigegeben

Inhaltsverzeichnis

Danksagungen

Die Generalversammlung des ISTQB® hat das englische Dokument am 2. Mai 2025 offiziell freigegeben.

Es wurde von einem Team des German Testing Board e. V. erstellt: Ralf Bongard (Leitung), Klaudia Dussa-Zieger, Matthias Friedrich, Thorsten Geiselhart, Horst Pohlmann (Product Owner), Ralf Reißing, Alexander Schulz.

Die folgenden Personen waren am Review, der Kommentierung und der Abstimmung über den englischen Lehrplan beteiligt:

Aktuelle Ausgabe (V2.1): Laura Albert, Ádám Bíró, Darvay Tamás Béla, Yuliia Fomuliaieva, Matthias Hamburg, Jarek Hryszko, Joanna Kazun, Imre Mészáros, Krisztián Miskó, Gary Mogyorodi, Ingvar Nordström, Mirosław Panek, Lukáš Piška, Meile Posthuma, Márton Siska, Klaus Skafte, Radosław Smilgin, Michael Stahl.

Das Team dankt Gary Mogyorodi für sein Technisches Review.

Die folgenden Personen waren am Review, der Kommentierung und der Abstimmung über den lokalisierten Lehrplan beteiligt:

Aktuelle Ausgabe (V2.1): Ralf Bongard (Leitung), Klaudia Dussa-Zieger, Matthias Friedrich, Thorsten Geiselhart, Michael Humm, Horst Pohlmann, Ralf Reißing, Matthias Hamburg.

Das Team dankt Ursula Zimpfer für ihr Lektorat.

0 Einführung in diesen Lehrplan

0.1 Zweck dieses Lehrplans

Dieser Lehrplan bildet die Grundlage für den ISTQB® Certified Tester – Automotive Software Tester. Das ISTQB® stellt diesen Lehrplan folgenden Adressaten zur Verfügung:

1. Nationalen Mitgliedboards, die den Lehrplan in ihre Sprache(n) übersetzen und Schulungsanbieter akkreditieren. Die nationalen Mitgliedboards dürfen den Lehrplan an die Anforderungen ihrer nationalen Sprache anpassen und Referenzen hinsichtlich lokaler Veröffentlichungen berücksichtigen.
2. Zertifizierungsstellen zur Ableitung von Prüfungsfragen in ihrer nationalen Sprache, die an die Lernziele dieses Lehrplans angepasst sind.
3. Schulungsanbietern zur Erstellung von Lehrmaterialien und zur Bestimmung angemessener Lehrmethoden.
4. Zertifizierungskandidaten zur Vorbereitung auf die Zertifizierungsprüfung (entweder als Teil einer Schulung oder unabhängig davon).
5. Der internationalen Software- und Systementwicklungs-Community zur Förderung des Berufsbildes des Software- und Systemtestens und als Grundlage für Bücher und Fachartikel.

0.2 ISTQB® Certified Tester – Automotive Software Tester (CT-AuT)

Die Qualifikation zum Automotive Software Tester richtet sich an alle, die mit dem Testen von Software im Automobilbereich zu tun haben. Dazu gehören beispielsweise Tester, Testanalysten, Testingenieure, Testberater, Testmanager, Abnahmetester und Softwareentwickler. Die Qualifikation zum Automotive Software Tester ist auch für alle geeignet, die ein grundlegendes Verständnis für das Testen von Software im Automobilbereich anstreben, wie z. B. Projektmanager, Sicherheitsmanager, Qualitätsmanager, Softwareentwicklungsmanager, Business-Analysten, IT-Direktoren und Unternehmensberater.

0.3 Karriereweg für Tester

Das ISTQB®-Programm unterstützt Tester in allen Phasen ihrer Laufbahn. Personen, die die ISTQB®-Zertifizierung Certified Tester – Automotive Software Tester (CT-AuT) erlangen, sind möglicherweise auch an den Core Advanced Levels (Test Analyst, Test Automation Engineer und Test Management) interessiert. Für alle, die ihre Fähigkeiten im Bereich des Testens in einer agilen Umgebung ausbauen möchten, kommen die Zertifizierungen ISTQB® Agile Technical Tester oder ISTQB® Agile Test Leadership at Scale in Frage. Darüber hinaus bieten Spezialistenmodule Zertifizierungsprodukte an, die sich auf bestimmte Testtechnologien und -ansätze, bestimmte Qualitätsmerkmale und Teststufen oder das Testen in bestimmten Industriebereichen konzentrieren. Aktuelle Informationen über das ISTQB®-Certified-Tester-Schema finden Sie unter *www.istqb.org*, zu dem Berufsbild Testen des GTB unter *www.gtb.de/der-certified-tester/berufsbild/* oder auf den Seiten der nationalen Boards, wie z. B. *www.gtb.de* (Deutschland), *www.swisstestingboard.org* (Schweiz) oder *www.austriantestingboard.at* (Österreich).

0.4 Geschäftlicher Nutzen

In diesem Abschnitt werden die geschäftlichen Nutzen (Business Outcomes, BO) aufgeführt, die von Kandidaten erwartet werden, die die Zertifizierung zum ISTQB® Certified Tester – Automotive Software Tester (CT-AuT) erlangt haben.

Ein ISTQB® Certified Tester – Automotive Software Tester (CT-AuT) kann ...

AuT-BO1	… effektiv in einem Testteam zusammenarbeiten. ("Zusammenarbeiten")
AuT-BO2	… die aus dem ISTQB® Certified Tester Foundation Level (CTFL®) bekannten Testverfahren an die spezifischen Anforderungen des Projekts anpassen. ("Anpassen")
AuT-BO3	… die grundlegenden Anforderungen der relevanten Normen (z. B. Automotive SPICE® und ISO 26262) bei der Auswahl geeigneter Testverfahren berücksichtigen. ("Auswählen")
AuT-BO4	… das Testteam bei der risikobasierten Planung der Testaktivitäten unterstützen und bekannte Strukturierungs- und Priorisierungselemente anwenden. ("Unterstützen & anwenden")
AuT-BO5	… virtuelle Testumgebungen (d. h. MiL, SiL und HiL) anwenden. ("Anwenden")

0.5 Lernziele und kognitive Wissensstufen

Die Lernziele (Learning Objectives, LO) unterstützen den geschäftlichen Nutzen und dienen zur Ausarbeitung der Prüfungen zum ISTQB® Certified Tester – Automotive Software Tester (CT-AuT).

Die spezifischen Lernziele und ihre Wissensstufen sind zu Beginn jedes Kapitels aufgeführt und wie folgt klassifiziert:

- K1: Erinnern
- K2: Verstehen
- K3: Anwenden

Weitere Einzelheiten und Beispiele für Lernziele finden Sie in Anhang A.

Für alle Begriffe, die als Schlüsselbegriffe direkt unter den Kapitelüberschriften aufgeführt sind, muss die korrekte Bezeichnung und Definition aus dem ISTQB®-Glossar bekannt sein (K1), auch wenn sie nicht ausdrücklich im Lernziel erwähnt werden.

0.6 Zertifizierungsprüfung zum ISTQB® Certified Tester – Automotive Software Tester (CT-AuT)

Die Prüfung zum ISTQB® Certified Tester – Automotive Software Tester (CT-AuT) basiert auf diesem Lehrplan. Zur Beantwortung einer Prüfungsfrage kann die Verwendung von Material aus mehr als einem Abschnitt dieses Lehrplans erforderlich sein. Alle Abschnitte des Lehrplans sind prüfungsrelevant, mit Ausnahme der Einführung und der Anhänge.

Im Lehrplan sind Standards und Fachbücher als Referenzen genannt, ihr Inhalt ist jedoch nicht prüfungsrelevant, abgesehen von dem, was im vorliegenden Lehrplan in zusammengefasster Form enthalten ist.

Weitere Einzelheiten finden sich im Dokument "Exam Structures and Rules" für den ISTQB® Certified Tester – Automotive Software Tester (CT-AuT) V2.1.

Voraussetzung für die Teilnahme an der Zertifizierung zum ISTQB® Certified Tester – Automotive Software Tester (CT-AuT) ist, dass die Kandidaten das ISTQB®-Foundation-Level-Zertifikat und ein Interesse an Softwaretest in Entwicklungsprojekten der Automobilindustrie haben. Es wird jedoch dringend empfohlen, dass die Kandidaten auch:

- über ein Mindestmaß an Erfahrung in der Softwareentwicklung oder im Testen von Software verfügen, z. B. sechs Monate Erfahrung als Softwaretester oder -entwickler, oder
- einen Kurs absolvieren, der nach dem ISTQB®-Standard akkreditiert ist (von einem der nationalen Mitgliedboards des ISTQB®), und/oder
- erste Erfahrungen im Testen von elektrischen/elektronischen (E/E-)Systemen in Entwicklungsprojekten der Automobilbranche gesammelt haben.

Eingangsvoraussetzung für die Teilnahme: Das ISTQB®-Foundation-Level-Zertifikat muss vor der Zertifizierungsprüfung zum Certified Tester – Automotive Software Tester erworben werden.

0.7 Akkreditierung

Ein nationales ISTQB®-Mitgliedboard kann Schulungsanbieter akkreditieren, deren Lehrmaterial diesem Lehrplan entspricht. Schulungsanbieter sollten die Akkreditierungsrichtlinien vom nationalen Mitgliedboard beziehen (in Deutschland: German Testing Board e. V.; in der Schweiz: Swiss Testing Board; in Österreich: Austrian Testing Board) oder von der Stelle, die die Akkreditierung in dessen Auftrag durchführt. Eine akkreditierte Schulung wird als konform mit diesem Lehrplan anerkannt und darf eine ISTQB®-Prüfung als Teil der Schulung enthalten.

Die Akkreditierungsrichtlinien für diesen Lehrplan folgen den allgemeinen Akkreditierungsrichtlinien, die von der ISTQB®-Arbeitsgruppe "Processes Management and Compliance" veröffentlicht wurden.

0.8 Umgang mit Normen und Standards

Internationale Normungsgremien wie IEEE und ISO haben Normen zu Qualitätsmerkmalen und zum Testen von Software herausgegeben. Auf solche Normen wird in diesem Lehrplan verwiesen. Der Zweck dieser Verweise ist es, einen Rahmen zu schaffen (wie bei den Verweisen auf ISO/IEC 25010 bezüglich der Qualitätsmerkmale) oder eine Quelle für zusätzliche Informationen zu bieten, falls der Leser dies wünscht. Bitte beachten Sie, dass in den ISTQB®-Lehrplänen Normdokumente als Referenz verwendet werden. Die Inhalte der Normen sind nicht prüfungsrelevant. Weitere Informationen zu Normen finden Sie im Kapitel 7 „Referenzen“.

0.9 Detaillierungsgrad

Der Detaillierungsgrad dieses Lehrplans ermöglicht international einheitliche Kurse und Prüfungen. Um dieses Ziel zu erreichen, besteht der Lehrplan aus folgenden Bestandteilen:

- Allgemeinen Lehrzielen, die die Absicht des Certified Tester – Automotive Software Tester beschreiben.
- Einer Liste von Begriffen, die die Studierenden kennen müssen.
- Lernzielen für jeden Wissensbereich, die das zu erreichende kognitive Lernergebnis beschreiben.
- Einer Beschreibung der Schlüsselkonzepte, einschließlich Verweisen auf Quellen wie anerkannte Literatur oder Normen.

Der Inhalt des Lehrplans beschreibt nicht das gesamte Wissensgebiet des Softwaretestens im automobilen Kontext. Er spiegelt den Detaillierungsgrad wider, der in den Schulungen zum ISTQB® Certified Tester – Automotive Software Tester (CT-AuT) abgedeckt werden soll. Er konzentriert sich auf Testkonzepte und -verfahren, die für Softwareprojekte im Automobilbereich gelten.

Der Lehrplan verwendet die Terminologie (d. h. den Namen und die Bedeutung), die beim Testen von Software und bei der Qualitätssicherung verwendet wird, gemäß dem ISTQB®-Glossar.

Für die Terminologie verwandter Disziplinen wird auf deren jeweilige Glossare verwiesen: ISO 26262 für funktionale Sicherheit (safety engineering) und das IREB-Glossar für Requirements Engineering.

0.10 Aufbau des Lehrplans

Es gibt vier Kapitel mit prüfbarem Inhalt. Die Überschrift auf der obersten Ebene jedes Kapitels gibt die Zeit für das Kapitel an. Unterhalb der Kapitelebene wird keine Zeitangabe gemacht. Für akkreditierte Lehrgänge fordert der Lehrplan mindestens 12,0 Unterrichtsstunden (720 Minuten), die sich wie folgt auf die vier Kapitel verteilen:

- Kapitel 1: Einführung in das Testen von Software im Automotive-Bereich (30 Minuten)
  - Die Tester lernen die Herausforderungen bei der Entwicklung von Automobilprodukten kennen, darunter gegensätzliche Ziele, zunehmende Komplexität und die Auswirkungen von Normen auf Zeit, Kosten, Qualität und Risiken.
  - Sie lernen auch die sechs Phasen eines Produktlebenszyklus und ihre Rolle bei der Freigabe von Produkten kennen.
- Kapitel 2: Normen für das Testen von elektrischen/elektronischen (E/E-)Systemen (300 Minuten)
  - Die Tester lernen ASPICE mit seinen Teststufen, testrelevanten Prozessen, Anforderungen an die Dokumentation und die Rolle einer Teststrategie sowie die Erwartungen an die Verfolgbarkeit und die Teststrategie von ASPICE kennen.
  - Für ISO 26262 konzentrieren sich die Tester auf die Sicherheitskultur, Automotive Safety Integrity Levels sowie Testverfahren und verstehen ihre Rolle innerhalb des Sicherheitslebenszyklus.
  - Die Tester verstehen auch den Einfluss von AUTOSAR auf das Testen und vergleichen die Ziele von ASPICE, ISO 26262 und CTFL® sowie die jeweiligen Teststufen.
- Kapitel 3: Testen in einer virtuellen Umgebung (160 Minuten)
  - Die Tester lernen den Zweck, die Struktur und die wesentlichen Funktionen von virtuellen Testumgebungen im Automobilbereich kennen, einschließlich Closed-Loop- und Open-Loop-Systemen, und gewinnen ein Verständnis für die verschiedenen XiL-Testumgebungen (d. h. MiL, SiL und HiL), ihre Anwendungen, Vorteile und Grenzen.
  - Sie lernen auch, wie sie jeder Testumgebung einen geeigneten Testumfang zuordnen können.
- Kapitel 4: Statische und dynamische Tests (230 Minuten)
  - Die Tester lernen, wie sie Programmierrichtlinien nach MISRA-C und Qualitätsmerkmale von Anforderungen nach der ISO/IEC/IEEE 29148 erklären und anwenden können.
  - Die Tester erstellen Testfälle für modifizierte Bedingungs-/Entscheidungsüberdeckungen, verstehen Fehlereinfügungstests und Back-to-Back-Tests und wählen geeignete Testverfahren und Testansätze je nach Projektkontext aus.

1 Einführung in das Testen von Software im Automotive-Bereich – 30 Minuten

Schlüsselbegriffe

keine

Lernziele für Kapitel 1 **Der Lernende kann ...**

1.1 Anforderungen aus divergierenden Projektzielen und zunehmender Produktkomplexität

AuT-1.1 (K2) … anhand von Beispielen die Herausforderungen, die sich bei der Produktentwicklung in der Automobilindustrie aus divergierenden Projektzielen und der zunehmenden Produktkomplexität ergeben, erläutern

1.2 Projektaspekte, die von Normen beeinflusst werden

AuT-1.2 (K1) … Projektaspekte, die von Normen beeinflusst werden (z. B. Zeit, Kosten, Qualität, Projektrisiken und Produktrisiken), wiedergeben

1.3 Die sechs generischen Phasen des Systemlebenszyklus

AuT-1.3 (K1) … die sechs generischen Phasen des Systemlebenszyklus nach ISO/IEC/IEEE 24748-1 wiedergeben

1.4 Der Beitrag und die Beteiligung des Testers am Freigabeprozess

AuT-1.4 (K1) … sich an die Rolle (d. h. Beitrag und Mitarbeit) des Testers im Freigabeprozess erinnern

Einführung

Einer der sieben Grundsätze des Softwaretestens lautet "Testen ist abhängig vom Umfeld". Dieses Kapitel skizziert das Umfeld der E/E-Systementwicklung, in dem ein "Automotive Software Tester"[1] arbeitet. Auf der einen Seite führen Zielkonflikte wie Kosteneffizienz, Sicherheitsanforderungen und schnelle Markteinführung, zunehmende Komplexität und die intensive Nachfrage nach innovativen Lösungen zu besonderen Herausforderungen. Auf der anderen Seite bilden Normen und der Lebenszyklus von Fahrzeugen den Rahmen, in dem der Tester tätig ist. Letztlich arbeitet der Tester an der Freigabe von Software und Systemen.

1.1 Anforderungen aus divergierenden Projektzielen und zunehmender Produktkomplexität

Erstausrüster (OEMs) und Zulieferer bringen trotz des zunehmenden Kostendrucks häufiger als früher neue Fahrzeugmodelle[2] auf den Markt. Die folgenden Aspekte beeinflussen diesen Prozess:

- Zunehmende Modellvielfalt und Komplexität:
  Um besser auf die individuellen Bedürfnisse der Endkunden eingehen zu können, bieten die OEMs immer mehr Fahrzeugmodelle an. Dadurch sinken jedoch die Stückzahlen pro Modell. Um den daraus resultierenden Anstieg der Entwicklungs- und Produktionskosten zu decken, entwickeln die Hersteller mehrere Modelle als Varianten einer gemeinsamen Plattform. Die Entwicklung einer Plattform ist jedoch weitaus komplexer als die Entwicklung eines einzelnen Modells, da die vielen möglichen Varianten beherrscht werden müssen.
- Wachsender Umfang der Funktionalität:
  Der Endkunde verlangt immer mehr Innovationen bei gleichzeitiger Beibehaltung der bestehenden Funktionen, wodurch der Funktionsumfang zunimmt.
- Zunehmende Anzahl von Konfigurationen:
  Der Endkunde will sein Fahrzeugmodell an seine individuellen Wünsche anpassen. Dies erfordert eine Vielzahl von Konfigurationen für ein einziges Fahrzeugmodell, auch in Bezug auf die Funktionalität.
- Erhöhte Anforderungen an die Qualität:
  Trotz steigender Funktionalität und Komplexität erwartet der Endkunde die gleiche oder sogar eine höhere Qualität des Fahrzeugs und seiner Funktionen.

Da die Projektziele – Zeit, Kosten und Umfang – miteinander konkurrieren (bekannt als das "Projektmanagement-Dreieck", wobei die Qualität oft als ein Aspekt des Umfangs betrachtet wird), müssen OEMs und Zulieferer eine effizientere Systementwicklung anstreben, die trotz zunehmender Komplexität, steigender Qualitätsanforderungen und kleinerer Budgets kürzere Entwicklungszeiten ermöglicht.

[1] Im Folgenden wird nur noch der Begriff "Tester" verwendet: Er ist als die Kurzform von "Automotive Software Tester" zu verstehen.

[2] Beispiel aus einer Studie der Unternehmensberatung Progenium: "1990 wurden nur 101 verschiedene Automodelle angeboten ..., 2014 waren es bereits 453."
Darüber hinaus bringt "die nächste Dekade der Mobilitätstransformation, die den Verbraucher in den Mittelpunkt stellt" neue Herausforderungen mit sich

1.2 Projektaspekte, die von Normen beeinflusst werden

Normen beeinflussen wichtige Projektaspekte wie Zeit, Kosten, Qualität, Projektrisiken und Produktrisiken:

- Normen erhöhen die Effizienz von Prozessen (z. B. durch Reduzierung der Entwicklungszeit oder -kosten bei gleichbleibender Qualität) durch:
  - einheitliche Namensgebung,
  - bessere Transparenz,
  - einfachere Zusammenarbeit (d. h. intern und extern),
  - erhöhte Wiederverwendbarkeit und
  - konsolidierte Erfahrungen ("Best Practice").
- Mit etablierten Technologierichtlinien helfen sie, Risiken und Fehlerzustände frühzeitig zu erkennen und zu beheben.
- Normen bilden die Grundlage für Audits. So kann ein Auditor die Qualität eines Produkts oder Prozesses beurteilen. Gleichzeitig kann der Auditor prüfen, ob die Normen den Anforderungen entsprechen.
- Normen sind Teil der vertraglichen oder gesetzlichen Bestimmungen und Richtlinien.

Dieser Lehrplan bezieht sich u. a. auf folgende ausgewählte Normen, die für das Testen von Software für die Automobilindustrie relevant sind:

- ISO 26262 und ASPICE, die standardisierte Prozesse und Methoden bereitstellen.
- AUTOSAR, das die Softwarearchitektur und -produkte standardisiert.

1.3 Die sechs generischen Phasen des Systemlebenszyklus

Der Systemlebenszyklus eines Autos und seiner Komponenten[3] beginnt mit der Produktidee und endet mit der Außerbetriebnahme. Er umfasst Entwicklungs-, Geschäfts-, Logistik- und produktionstechnische Prozesse.

Meilensteine mit definierten Eingangs- und Endekriterien sorgen für reife Prozesse und strukturieren den Systemlebenszyklus[4] in sechs Phasen mit typischen Testaktivitäten nach [ISTQB_CTFL]:

- Konzept: Testplanung zur Festlegung der Teststrategie und der Testziele
- Entwicklung: Testanalyse, Testentwurf, Testdurchführung, Testausführung, Testüberwachung, Teststeuerung und Testabschluss sichern die Qualität.
- Produktion: Testen der End-of-Line, um die Bereitschaft zu überprüfen.
- Betrieb: Normalerweise werden keine Testaktivitäten durchgeführt.
- Wartung: Wartungstests zur Sicherstellung der kontinuierlichen Zuverlässigkeit
- Außerbetriebnahme: Migrationstests validieren die Stilllegung oder Datenübertragung.

[3] Steuergeräte (Hardware und Software) sowie Komponenten.
[4] Der Sicherheitslebenszyklus der ISO 26262 besteht aus ähnlichen Phasen.

Der weitverbreitete Produktentwicklungsprozess in der Automobilindustrie fasst diese Phasen oft in drei Hauptstufen zusammen: Konzept, Entwicklung und Produktion, wo die meisten Tests und Freigaben stattfinden.

1.4 Der Beitrag und die Beteiligung des Testers am Freigabeprozess

In der Automobilbranche erreicht ein Projekt einen Meilenstein, indem eine Freigabe erklärt wird, sobald die Ziele als erreicht gelten. Von diesem Zeitpunkt an wird davon ausgegangen, dass das Freigabeobjekt die für seinen Verwendungszweck erforderliche Reife besitzt. Das Freigabeobjekt umfasst die Softwarekonfiguration einschließlich Parametrisierung und ggf. Hardware und Mechanik sowie die begleitende Dokumentation.

Der Tester liefert mit dem abschließenden Testbericht wichtige Informationen für den Freigabeprozess:

- Getestete Elemente
- Bewertete Qualitätsmerkmale (z. B. Antwortzeit und Ressourcenverbrauch)
- Bekannte Fehlerzustände
- Produktmetriken
- Informationen für die Freigabeempfehlung bei Erreichen der Endekriterien auf der Grundlage der Freigabestufe gemäß den Best Practice Guidelines (z. B. Erprobung auf abgesperrtem Gelände, Erprobung auf öffentlichen Straßen und Verbauempfehlungen)

Darüber hinaus beteiligt sich der Tester an der Erstellung weiterer für die Freigabe relevanter Ergebnisse:

- Prioritäten setzen und an der Entscheidung über Änderungen mitwirken
- Priorisierung von Features für die Reihenfolge der Implementierung

2 Normen für das Testen von elektrischen/elektronischen (E/E-)Systemen – 300 Minuten

Schlüsselbegriffe

funktionale Sicherheit, Methodentabelle

Domänenspezifische Schlüsselbegriffe

Automotive Safety Integrity Level (ASIL), Softwareverifizierung, Systemverifizierung

Lernziele für Kapitel 2 **Der Lernende kann ...**

2.1 Automotive SPICE (ASPICE)

AuT-2.1.1.1 (K1) … die zwei Dimensionen von ASPICE wiedergeben

AuT-2.1.1.3 (K2) … die Fähigkeitsstufen 0 bis 3 von ASPICE erläutern

AuT-2.1.2.1 (K1) … sich an den Zweck der testspezifischen Prozesse von ASPICE erinnern

AuT-2.1.2.2 (K2) … die Bedeutung der vier Bewertungsstufen und der Fähigkeitsindikatoren von ASPICE aus Sicht des Testens erläutern

AuT-2.1.2.3 (K2) … die Anforderungen von ASPICE an eine Teststrategie einschließlich der Kriterien für die Regressionsverifizierung erläutern

AuT-2.1.2.4 (K1) … sich an die Anforderungen von ASPICE an Testmittel erinnern

AuT-2.1.2.5 (K3) … Maßnahmen zur Software-Unit-Verifizierung anwenden

AuT-2.1.2.6 (K2) … die Anforderungen an die Verfolgbarkeit von ASPICE aus Sicht des Testens erläutern

2.2 ISO 26262

AuT-2.2.1.1 (K2) … das Ziel der funktionalen Sicherheit für E/E-Systeme erläutern

AuT-2.2.1.2 (K1) … den Beitrag des Testers zur Sicherheitskultur nennen

AuT-2.2.2 (K2) … die Rolle des Testers im Rahmen des Sicherheitslebenszyklus nach ISO 26262 diskutieren

AuT-2.2.3.2 (K1) … die Teile der ISO 26262 nennen, die für den Tester relevant sind

AuT-2.2.4.1 (K1) … die Kritikalitätsstufen des ASIL wiedergeben

AuT-2.2.4.2 (K2) … den Einfluss des ASIL auf die Testverfahren und Testarten für statische und dynamische Tests und den daraus resultierenden Testumfang erläutern

AuT-2.2.5 (K3) … die Methodentabellen der ISO 26262 anwenden

2.3 AUTOSAR

AuT-2.3.1 (K1) … die Projektziele von AUTOSAR wiedergeben

AuT-2.3.3 (K1) … sich an den Einfluss von AUTOSAR auf die Arbeit des Testers erinnern

2.4 Vergleich von ASPICE, ISO 26262 und CTFL®

AuT-2.4.1 (K1) … die unterschiedlichen Ziele von ASPICE und ISO 26262 wiedergeben

AuT-2.4.2 (K2) … die Unterschiede zwischen ASPICE, ISO 26262 und CTFL® in Bezug auf die Teststufen erläutern

2.1 Automotive SPICE (ASPICE)

Einführung

Prozessverbesserung verfolgt den Ansatz, dass die Qualität eines Systems von der Qualität der Prozesse in der Entwicklung beeinflusst wird. Prozessmodelle bieten einen Ansatz für Verbesserungen, indem sie die Prozessfähigkeit einer Organisationseinheit oder eines Projekts im Vergleich zum Referenzmodell messen. Darüber hinaus dient das Modell als Rahmen für die Verbesserung der Prozesse einer Organisationseinheit oder eines Projekts unter Verwendung der Assessment-Ergebnisse.

Seit 2001 haben die SPICE[5] User Group und die Automotive Special Interest Group (AUTOSIG) ASPICE entwickelt. Seit der Veröffentlichung im Jahr 2005 ist der Standard in der Automobilindustrie fest etabliert. Alle Aussagen in diesem Abschnitt beziehen sich auf die ASPICE-Version 4.0.

2.1.1 Aufbau und Struktur des Standards

2.1.1.1 Die zwei Dimensionen von ASPICE

ASPICE definiert ein Bewertungsmodell mit zwei zentralen Dimensionen der Bewertung:

In der **Prozessdimension** definiert ASPICE das Prozessreferenzmodell (PRM). Das PRM dient als Referenz, um die Prozesse der Organisation damit zu vergleichen, so dass sie bewertet und verbessert werden können. Für jeden Prozess definiert ASPICE den Zweck, die Ergebnisse, die erforderlichen Tätigkeiten (d. h. Basispraktiken) und die Informationselemente (d. h. Arbeitsergebnisse und Arbeitsprodukte).

In der **Fähigkeitsdimension** definiert ASPICE mehrere Prozessattribute (d. h. messbare Merkmale), die das Fähigkeitsniveau unterteilen. Für jeden Prozess gibt es generische und prozessspezifische Attribute. Die ISO/IEC 33020 dient als Grundlage für die Bewertung der Prozessfähigkeit.

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

2.1.1.2 Prozesskategorien in der Prozessdimension [informativ]

ASPICE gliedert Prozesse in Gruppen, die wiederum in Kategorien eingeteilt werden:

Zu den **primären Lebenszyklusprozessen** gehören alle Prozesse, die als Schlüsselprozesse dienen:

- Prozessgruppe Beschaffung (ACQ)
- Prozessgruppe Support (SPL)
- Prozessgruppe Systementwicklung (SYS)
- Prozessgruppe Softwareentwicklung (SWE)
- Prozessgruppe Maschinelles Lernen (MLE)
- Prozessgruppe Hardwareentwicklung (HWE)
- Prozessgruppe Validierung (VAL)

[5] Akronym für "Software Process Improvement and Capability Determination".

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

- Unterstützende Prozessgruppe (SUP)

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

- Prozessgruppe Management (MAN)
- Prozessgruppe Prozessverbesserung (PIM)
- Prozessgruppe Wiederverwendung (REU)

Für den Tester sind die Prozessgruppen Systementwicklung (SYS) und Softwareentwicklung (SWE) von besonderem Interesse, eventuell auch die Prozessgruppen MLE, VAL und HWE.

2.1.1.3 Fähigkeitsstufen (Capability Level, CL) in der Fähigkeitsdimension

Der Assessor bewertet die Prozessfähigkeit mit Hilfe eines sechsstufigen Bewertungssystems (d. h. durch Darstellung der Stufen). ASPICE definiert die Fähigkeitsstufen 0 bis 3[6] wie folgt:

- Stufe 0 (unvollständiger Prozess): Der Prozess ist nicht implementiert oder erreicht nicht seinen Prozesszweck. Auf dieser Stufe gibt es wenig oder keine Hinweise auf eine systematische Erreichung des Prozessergebnisses. Beispiel: Der Tester prüft nur einen geringen Teil der Anforderungen.
- Stufe 1 (durchgeführter Prozess): Der im Projekt implementierte Prozess erfüllt seinen Prozesszweck (wird aber möglicherweise uneinheitlich ausgeführt). Beispiel: Es ist keine vollständige Planung für einen Testprozess sichtbar. Der Tester kann jedoch den Erfüllungsgrad der Anforderungen aufzeigen.
- Stufe 2 (gesteuerter Prozess): Das Projekt plant und überwacht den Testprozess bei seiner Testdurchführung. Es passt unter Umständen den Ablauf der Testdurchführung an die Testziele an. Die Anforderungen an die Arbeitsprodukte werden definiert. Ein Projektmitglied reviewt die Arbeitsprodukte und gibt sie frei. Beispiel: Der Testmanager legt die Testziele fest, plant die Testaktivitäten und steuert den Testprozess. Dazu gehört auch, auf Abweichungen entsprechend zu reagieren.
- Stufe 3 (etablierter Prozess): Das Projekt verwendet einen standardisierten Testprozess, und die Befunde werden zur ständigen Verbesserung genutzt. Beispiel: Es gibt eine organisationsweite Teststrategie. Nach Testabschluss arbeitet der Testmanager an deren Weiterentwicklung.

[6] Die Fähigkeitsstufen 4 und 5 sind derzeit nicht im Fokus der Automobilindustrie.

2.1.2 Anforderungen des Standards

2.1.2.1 Testspezifische Prozesse von ASPICE

ASPICE definiert Testprozesse entsprechend allen Prozessen der Software- und Systementwicklung[7]:

- Software-Unit-Verifizierung (SWE.4) fordert statische und dynamische Tests. Sie bewertet die Komponenten der Software auf der Grundlage ihres detaillierten Entwurfs (SWE.3).
- Softwarekomponentenverifizierung und Integrationsverifizierung (SWE.5) bewertet die integrierte Software auf der Grundlage des Softwarearchitekturentwurfs (SWE.2).
- Softwareverifizierung (SWE.6) bewertet die integrierte Software auf der Grundlage der Analyse der Softwareanforderungen (SWE.1).
- Systemintegration und Integrationsverifizierung (SYS.4) bewertet das integrierte System auf der Grundlage des Systemarchitekturentwurfs (SYS.3).
- Systemverifizierung (SYS.5) bewertet das integrierte System auf der Grundlage der Analyse der Systemanforderungen (SYS.2).

2.1.2.2 Bewertungsstufen und Fähigkeitsindikatoren

Ein Assessor kann die Prozessfähigkeit über Fähigkeitsindikatoren bewerten. ASPICE definiert sie für 9 Prozessattribute (PA). Für die Fähigkeitsstufen 1 bis 3 sind sie wie folgt definiert (am Beispiel von SWE.6 in Klammern):

- PA 1.1: Prozessattribut Prozessdurchführung (z. B. der Tester orientiert sich am Testprozess)
- PA 2.1: Prozessattribut Testdurchführung (z. B. der Tester plant, überwacht und steuert die Testaktivitäten)
- PA 2.2: Prozessattribut Arbeitsergebnismanagement[8] (z. B. der Tester überprüft die Qualität der Testmittel/Testdokumentation)
- PA 3.1: Prozessattribut Prozessdefinition (z. B. die für den Testprozess verantwortliche Person definiert eine organisationsweite Teststrategie)
- PA 3.2: Prozessattribut Prozessanwendung/Prozessumsetzung (z. B. der Tester wendet die in PA 3.1 definierte Teststrategie an)

Für die Prozessdurchführung definiert ASPICE zwei Arten von Fähigkeitsindikatoren: Basispraktiken (BP) und Informationselemente (Information Items). Darüber hinaus werden generische Praktiken (GP) und Informationselemente definiert. Die Bewertung der Prozessattribute basiert auf dem Implementierungsgrad der Indikatoren in vier Bewertungsstufen:

- **N** (none): nicht erfüllt (0% bis ≤ 15%)
- **P** (partially): teilweise erfüllt (> 15% bis ≤ 50%)
- **L** (largely): weitgehend erfüllt (> 50% bis ≤ 85%)
- **F** (fully): vollständig erfüllt (> 85% bis ≤ 100%)

[7] Ergänzend zu den hier definierten Testprozesse existiert noch die Prozessgruppe VAL. Der Zweck von VAL.1 besteht darin, den Nachweis zu erbringen, dass das Endprodukt, das eine direkte Interaktion mit dem Endnutzer ermöglicht, die Erwartungen hinsichtlich der beabsichtigten Verwendung in seiner vorgesehenen Betriebsumgebung erfüllt.

[8] Das Arbeitsergebnismanagement gilt für alle dokumentierten Informationen.

Damit ein Prozess einen bestimmten Fähigkeitsgrad erreicht, müssen die Fähigkeitsindikatoren entweder weitgehend erfüllt (L) oder vollständig erfüllt (F) sein.

2.1.2.3 Teststrategie und Kriterien für die Regressionsverifizierung

ASPICE fordert für jeden Testprozess (siehe Abschnitt 2.1.2.1) ab der Teststufe 2 eine Teststrategie (d. h. Verifizierungsmaßnahmen). Sie wird vom Testmanager im Rahmen der konzeptionellen Testplanung entwickelt. Testrichtlinien, Projektziele sowie vertragliche und regulatorische Anforderungen bilden die Grundlage dafür.

Der Tester ist sich des Prinzips des frühen Testens bewusst. Das gilt auch für das Testen von Software in der Automobilumgebung. Allerdings kommt hier ein anderer Aspekt ins Spiel, denn Testumgebungen auf höheren Teststufen sind deutlich teurer. So ist für das Testen auf höheren Stufen speziell entwickelte und eingebettete Hardware notwendig (z. B. als Prototyp oder Unikat). Die Teststrategie definiert die stufenspezifischen Testumgebungen, aber auch, welche Tests der Tester in welchen Testumgebungen durchführen soll.

Die Kriterien für die Regressionsverifizierung sind ein wesentlicher Bestandteil der Teststrategie. Sie legen die Details für die Regressionsverifizierung[9] fest. Die Herausforderung liegt in der wirtschaftlich sinnvollen Auswahl der Testfälle (d. h. im Mehrwert des Testens). Die Kriterien für die Regressionsverifizierung legen das Testziel und den Testansatz für die Auswahl der Regressionstests fest. Die Auswahl kann zum Beispiel risikobasiert erfolgen. Eine Auswirkungsanalyse hilft dabei, die Bereiche zu identifizieren, auf die sich der Tester bei den Regressionstests konzentrieren muss. Der Testmanager kann den Tester aber auch auffordern, alle automatisierten Testfälle für jedes Release zu wiederholen.

2.1.2.4 Testmittel in ASPICE

ASPICE 4.0 stellt keine Anforderungen an Testmittel. Stattdessen werden Anforderungen an die Dokumentation bestimmter Informationen gestellt, die beim Testen anfallen. Für Testaktivitäten fordert ASPICE die folgenden Informationselemente:

- 08-60: Verifizierungsmaßnahmen (z. B. Simulationen, dynamische Tests und statische Tests), typischerweise dokumentiert in einem Testkonzept
- 13-25: Verifizierungsergebnisse (z. B. dokumentierte Testprotokolle, Testberichte und Fehlerberichte)

Für jedes Informationselement definiert ASPICE Beispiele für Informationselementmerkmale (Information Item Characteristics, IIC) und -inhalte. Sie dienen als objektiver Indikator für die Prozessdurchführung.

ISO/IEC/IEEE 29119-3 kann zur weiteren Strukturierung von Informationselementen verwendet werden.

2.1.2.5 Messungen zur Software-Unit-Verifizierung

Der Tester verifiziert die Konformität mit dem Softwarefeindesign und mit den funktionalen und nicht-funktionalen Anforderungen anhand der Verifizierungsmaßnahmen. Die Maßnahmen definieren, wie der Tester den Nachweis erbringt. Der Tester kann Kombinationen von statischen und dynamischen Testverfahren verwenden, um die Software-Units zu verifizieren.

[9] Ähnlich wie die regressionsvermeidende Teststrategie im ISTQB.

In SWE.4.BP1 fordert ASPICE die Definition von Maßnahmen (siehe Abschnitt 2.1.2.4) für die Software-Unit-Verifizierung. Damit kann der Tester beurteilen, inwieweit die Software-Unit ihre Anforderungen erfüllt. Die folgenden Beispiele könnten als Maßnahmen zur Software-Unit-Verifizierung verwendet werden:

- Testfälle der Software-Unit einschließlich Testdaten
- Werkzeuggestützte statische Analyse, die die Konformität mit Programmierrichtlinien bewertet (z. B. MISRA-C, siehe Abschnitt 4.1.1).
- Reviews für Software-Units oder Teile von Softwareeinheiten, die nicht durch eine werkzeuggestützte statische Analyse bewertet werden können.

Wenn ein Entwickler eine Software-Unit ändert, muss der Tester diese Änderung ebenfalls bewerten. Zu den Messungen der Software-Unit-Verifizierung gehören daher auch die Kriterien für die Regressionsverifizierung. Das beinhaltet die Verifizierung des geänderten Codes, den Fehlernachtest und die erneute Verifizierung der nicht geänderten Teile (statische und dynamische Regressionstests).

2.1.2.6 Verfolgbarkeit in ASPICE

Wie im CTFL®-Lehrplan [ISTQB_CTFL] wird auch in ASPICE eine bidirektionale Verfolgbarkeit[10] gefordert. Dies ermöglicht den Testern Folgendes:

- Auswirkungen zu analysieren, z. B. die Auswirkungsanalyse von Änderungen
- Eine Überdeckung zu evaluieren
- Einen Status zu verfolgen

Außerdem können die Tester auf diese Weise die Konsistenz zwischen den verknüpften Elementen sicherstellen, sowohl textlich als auch semantisch.

ASPICE unterscheidet zwischen vertikaler und horizontaler Verfolgbarkeit:

Vertikal: ASPICE fordert, dass die Anforderungen der Stakeholder über alle Ebenen hinweg mit den Software-Units verknüpft werden. Dabei stellt die Verknüpfung über alle Entwicklungsebenen hinweg die Konsistenz zwischen den zugehörigen Arbeitsprodukten sicher.

Horizontal: ASPICE verlangt Verfolgbarkeit und Konsistenz zwischen den Arbeitsergebnissen der Entwicklung und den entsprechenden Testspezifikationen und Testergebnissen.

Darüber hinaus fordert die Basispraxis SUP.10.BP4 eine bidirektionale Verfolgbarkeit zwischen Änderungsanforderungen und den von den Änderungsanforderungen betroffenen Arbeitsprodukten. Da Änderungsanträge häufig durch einen Fehlerzustand ausgelöst werden, wird eine bidirektionale Verfolgbarkeit zwischen Änderungsanträgen und den entsprechenden Fehlerberichten hergestellt.

Aufgrund der mitunter großen Anzahl von Verknüpfungen kann eine konsistente Kette von Werkzeugen hilfreich sein. Dies ermöglicht dem Tester eine effiziente Erstellung und Verwaltung der Abhängigkeiten.

[10] Im Folgenden wird unter dem Begriff Verfolgbarkeit immer die bidirektionale Verfolgbarkeit verstanden.

2.2 ISO 26262

2.2.1 Funktionale Sicherheit und Sicherheitskultur

2.2.1.1 Zielsetzung der funktionalen Sicherheit für E/E-Systeme

Die funktionale und technische Komplexität von eingebetteten Systemen nimmt stetig zu. Gleichzeitig ermöglichen leistungsfähige softwarebasierte elektrische und elektronische Systeme neue komplexe Funktionalitäten, wie z. B. die Automatisierung von Fahrfunktionen in einem Fahrzeug.

Mit der zunehmenden Komplexität steigt auch das Risiko einer Fehlhandlung während der Entwicklung. Die Folge kann ein (unerkannter) Fehlerzustand im System sein. Bei Systemen mit einem inhärenten Risikopotenzial für Leib und Leben muss der verantwortliche Sicherheitsingenieur oder Sicherheitsmanager daher mögliche Risiken analysieren. Liegt ein tatsächliches Risiko vor, müssen geeignete Maßnahmen identifiziert werden, um mögliche Auswirkungen auf ein akzeptables Risiko zu reduzieren.

Die Methoden zur Durchführung solcher Analysen sind in den Normen für die funktionale Sicherheit zusammengefasst. Eine grundlegende Norm ist die IEC 61508. Aus dieser Norm hat die Internationale Organisation für Normung (ISO) die ISO 26262 abgeleitet, die in ihrer zweiten Auflage seit 2018 verfügbar ist.

Bei der funktionalen Sicherheit geht es darum, sicherzustellen, dass E/E-Systeme funktionieren, ohne dass ein unangemessenes Risiko aufgrund von Gefahren durch Fehlverhalten entsteht. In diesem Sinne ist der Begriff von anderen verwandten Begriffen wie Informationssicherheit (oder Cybersicherheit), Produktsicherheit oder Arbeitssicherheit abzugrenzen.

Sicherheit am Arbeitsplatz und Cybersicherheit stehen nicht im Fokus der ISO 26262. Ein Mangel an Cybersicherheit kann jedoch die funktionale Sicherheit gefährden. Sowohl funktionale Sicherheit als auch Cybersicherheit tragen zur Produktsicherheit bei.

2.2.1.2 Beitrag des Testers zur Sicherheitskultur

Bei der Produktentwicklung nach ISO 26262 reicht es nicht aus, nur die Prozesse der eigenen Organisation zu überwachen. Alle Beteiligten müssen einen prozessübergreifenden Ansatz verfolgen. Jeder muss verstehen, welchen Einfluss er auf den Entwicklungsprozess und die Sicherheit des Endprodukts hat. Dies gilt auch für externe Partner und Zulieferer.

Die Beteiligten müssen verstehen, dass ihr eigenes Handeln nicht unabhängig von anderen Prozessen abläuft. Jeder Entwicklungsschritt ist ein wesentlicher Beitrag zur Konformität und Umsetzung der sicherheitsrelevanten Anforderungen. Diese Verantwortung endet nicht mit der Markteinführung. Sie dauert bis zum Ende des Sicherheitslebenszyklus an.

Die Tester tragen zur Sicherheitskultur bei, indem sie verantwortungsbewusst an den Phasen des Softwareentwicklungslebenszyklus teilnehmen und ihre Arbeit unter ständiger Berücksichtigung des Gesamtkontexts der Produktentwicklung ausführen.

2.2.2 Integration des Testers in den Sicherheitslebenszyklus

Der Sicherheitslebenszyklus beschreibt die sicherheitsorientierten Aktivitäten während der Produktentwicklung. Er beginnt mit der ersten Produktidee und der Identifizierung von möglichen Risiken. Nach der Spezifikation der daraus resultierenden Anforderungen an die Sicherheit folgt die Umsetzung in ein konkretes Produkt. Der Sicherheitslebenszyklus endet mit der Entsorgung des Produkts am Ende seiner Lebensdauer.

Der Sicherheitslebenszyklus nach ISO 26262 durchläuft die folgenden Phasen:

- 1. Phase: Konzeptphase
- 2. Phase: Produktentwicklung (diese Phase endet mit der "Freigabe zur Produktion")
- 3. Phase: Produktion, Betrieb, Wartung und Stilllegung

Die Tester sind überwiegend in den ersten beiden Phasen tätig. Änderungen am Produkt innerhalb der dritten Phase führen je nach Umfang zu einer Rückkehr zur ersten oder zweiten Phase. Daher ist der Tester auch an Änderungen beteiligt. Auf der Grundlage der sicherheitsrelevanten Anforderungen werden Testverfahren ausgewählt und Testfälle für die Verifizierung und Validierung dieser Anforderungen entworfen. Die Tester führen diese Aufgaben dann in den jeweiligen Teilphasen der Produktentwicklung durch.

Die Testplanung findet in der Regel während der Konzeptphase statt. Anpassungen an den daraus resultierenden Dokumenten (z. B. im Testkonzept oder den Testspezifikationen) können jedoch in jeder Phase erforderlich sein. Die Testdurchführung findet vor allem an den Übergängen zwischen den einzelnen Phasen der Produktentwicklung statt. Zum Beispiel beim Übergang von der Implementierung zur Softwareintegration und dann zur Hardware-Software-Integration. Darüber hinaus tragen die Tester mit ihren Testaktivitäten zentral zum Übergang in die dritte Phase bei.

2.2.3 Gliederung und testspezifische Anteile der Norm

2.2.3.1 Aufbau und Struktur der Norm [informativ]

Die ISO 26262 besteht aus 12 Teilen:

- Vokabular (Teil 1)
- Management der Funktionalen Sicherheit (Teil 2)
- Die Phasen des Sicherheitslebenszyklus:
  - Konzeptphase (Teil 3)
  - Produktentwicklung auf System-, Hardware- und Softwareebene (Teile 4-6)
  - Produktion und Betrieb (Teil 7)
- Unterstützende Prozesse (Teil 8)
- ASIL-orientierte und sicherheitsorientierte Analyse (Teil 9)
- Leitfaden für die Anwendung der ISO 26262 (Teil 10)
- Leitfaden für die Anwendung der ISO 26262 auf Halbleiter (Teil 11)
- Anpassung für Motorräder (Teil 12)

Abgesehen von Teil 1, Teil 10 und Teil 11 enthält jeder Teil folgende Abschnitte:

- Eine allgemeine Einführung
- Den Geltungsbereich
- Normative Referenzen
- Anforderungen an die Konformität der Norm

Danach folgen die spezifischen Themen des jeweiligen Teils. Die Struktur ihrer Beschreibung ist in jedem Teil gleich. Die auszuführenden Aktivitäten werden durch eine in allen Teilen wiederkehrende Struktur beschrieben:

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

2.2.3.2 Relevante Teile der ISO 26262 für den Tester

Für den Softwaretester ist die Verifizierung der Software und zumindest teilweise die Validierung des Systems von zentraler Bedeutung. Neben Teil 1 (Vokabular) sind auch einige andere Teile von besonderem Interesse: Die Teile 4 und 6 geben detaillierte Hinweise und Anforderungen bezüglich empfohlener Maßnahmen zur Softwareverifizierung. Dies gilt für die Auswahl, den Entwurf, die Implementierung und die Durchführung der jeweiligen Verifizierungsmaßnahmen.

Dabei konzentrieren sich diese Teile auf die test- und verifizierungsspezifischen Aspekte der Systemebene (d. h. Teil 4, einschließlich der Validierung der Sicherheit) und der Softwareebene (d. h. Teil 6). Sollten auch hardwarespezifische Aspekte für diese Arbeit relevant sein, finden die Tester diese in Teil 5. Aspekte, die sowohl die Hardware als auch die Software betreffen, werden im Rahmen des Hardware-Software-Interface (Teile 4-6) berücksichtigt.

Teil 8 kommt eine besondere Rolle zu, da dieser die prozessspezifischen Merkmale der Verifizierung auf allen Teststufen beschreibt. Darüber hinaus sind Anforderungen an unterstützende Prozesse zu finden, die für die Sicherstellung standardisierter Abläufe entscheidend sind, wie z. B. das Konfigurationsmanagement und die Qualifizierung von Werkzeugen.

In Teil 12 finden die Tester angepasste Methodentabellen, die auf der neu eingeführten Bewertung gemäß Motorcycle Safety Integrity Level (MSIL) für Motorradprojekte basieren.

2.2.4 Einfluss der Kritikalität auf die Testumfänge

2.2.4.1 Die Kritikalitätsstufen des ASIL

Der ASIL ist ein Maß für die erforderliche Risikominderung durch Maßnahmen der funktionalen Sicherheit. Derartige Maßnahmen können beispielsweise eine eigenständige Sicherheitsfunktion zur Überwachung eines E/E-Systems oder der Einsatz spezifischer festgelegter Methoden sein. Bei höheren Risiken können aufwendigere Maßnahmen erforderlich sein.

Zu Beginn des Projekts führt ein Expertenteam eine Gefährdungsanalyse und Risikobewertung (Hazard Analysis and Risk Assessment, HARA) für das Produkt durch. Für jedes durch diese Analyse ermitteltes Risiko bestimmt das Team einen ASIL unter Verwendung der in der Norm festgelegten Methoden. Im nächsten Schritt legt das Team Sicherheitsziele und Sicherheitsanforderungen fest. Diese haben denselben ASIL wie die zugrunde liegende Gefährdung.

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

Führen die Ergebnisse der Gefährdungsanalyse und Risikobewertung zu Anforderungen unterhalb von ASIL A, so sind diese im Sinne der Norm nicht sicherheitsrelevant. Für solche Anforderungen sind keine funktionalen sicherheitsrelevanten Maßnahmen erforderlich, und es reicht aus, solche Anforderungen durch bereits bestehende Qualitätsmanagementsysteme (QMS) abzudecken. Aus praktischen Gründen werden diese Anforderungen oft mit "QM" gekennzeichnet, um sie von ASIL-bezogenen Anforderungen zu unterscheiden.

2.2.4.2 Einfluss des ASIL auf Testverfahren, Testarten und Testumfänge

Der ermittelte ASIL hat direkten Einfluss auf den Umfang der von den Testern durchzuführenden Tests. Je nach Höhe des ASIL empfiehlt die Norm ISO 26262 die Durchführung unterschiedlicher Maßnahmen oder Maßnahmenpakete. Dabei rät die Norm bei höherem ASIL zu umfangreicheren und detaillierteren Maßnahmen. Für niedrigere ASIL ist die Durchführung spezifischer Maßnahmen oft optional.

Die ISO 26262 legt drei Empfehlungsstufen fest: keine Empfehlung, empfohlen und dringend empfohlen. Das allgemeine Verständnis dieser Empfehlungsstufen ist, dass eine Methode, die als "dringend empfohlen" aufgeführt ist, angewandt werden muss. Während eine Methode, die als "empfohlen" aufgeführt ist, angewandt werden sollte oder eine Begründung für die Nichtanwendung angegeben werden muss. Im Falle von "keine Empfehlung" gibt die Norm keine Empfehlung für oder gegen die Anwendung der betreffenden Maßnahme. Sie kann ohne Bedenken als unterstützende Maßnahme verwendet werden. Ihre Durchführung ersetzt jedoch nicht die von der ISO 26262 empfohlenen oder dringend empfohlenen Maßnahmen.

Für die Tester bedeutet dies, dass die Norm je nach ASIL spezifische Testverfahren und Testarten für sicherheitsrelevante Systeme empfiehlt. Die Tester können nur so weit frei entscheiden, wie es die Norm für den konkreten Fall zulässt. So werden beispielsweise für ASIL A lediglich die Äquivalenzklassenbildung und die Grenzwertanalyse empfohlen. Für ASIL B oder höher werden diese Techniken hingegen dringend empfohlen (siehe Abschnitt 2.2.5).

Der ASIL ist keine Eigenschaft des Gesamtprodukts. Er ist an ein konkretes Sicherheitsziel und daraus abgeleitete Sicherheitsanforderungen gebunden. Für ein und dasselbe Produkt können Sicherheitsanforderungen mit unterschiedlichem ASIL daher zu einem signifikanten Unterschied beim Testaufwand führen. Dies muss von den Testern bei der Planung des Testumfangs berücksichtigt werden.

2.2.5 Anwendung des aus CTFL® bekannten Wissens im Kontext der ISO 26262

Die ISO 26262 bietet dem Tester spezifische Empfehlungen für Testaktivitäten in Form von Methodentabellen. Diese sind in den Teilen 4, 5, 6, 8 und 12 zu finden. Neben den für die funktionale Sicherheit spezifischen Empfehlungen für Testprozesse und Testaktivitäten enthalten sie auch die zu verwendenden Testverfahren.

In diesem Zusammenhang verwendet die Norm den Begriff "Methode", der sich auf alle anwendbaren Testverfahren oder Testaktivitäten bezieht. Die Terminologie der funktionalen Sicherheit weicht an dieser Stelle leicht von den Begriffen des ISTQB® ab. Für den Tester sind die folgenden Methoden der ISO 26262 von besonderem Interesse:

- Testverfahren (z. B. Äquivalenzklassenbildung und Grenzwertanalyse)
- Verfahren zur Testdurchführung (z. B. Simulation oder Prototyp einer Komponente oder eines Systems)
- Testarten (z. B. nicht-funktionale Tests wie Performanztests und Lebensdauertests)
- Testumgebungen (z. B. Hardware-in-the-Loop und Fahrzeuge)
- Statische Testverfahren (z. B. Reviews und statische Analyse)

Die Methodentabellen definieren die von der Norm empfohlenen Methoden für jeden ASIL.

Die Tabellen sind 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 eine Methodentabelle

Für jede Methode ist in Abhängigkeit zum ASIL dokumentiert, ob ihre Verwendung empfohlen (+) oder sogar dringend empfohlen (++) wird. Für Methoden, die mit “keine Empfehlung” (o) gekennzeichnet sind, gibt es in der Norm keine Empfehlung für oder gegen ihre Verwendung.

Die ISO 26262 führt in den Methodentabellen auch gleichwertige alternative Methoden mit Buchstabensuffixen auf (im obigen Beispiel die Zeilen 3a und 3b). Hier müssen die Tester eine geeignete Kombination wählen, um die relevanten Anforderungen ASIL-konform überprüfen zu können. Die Tester müssen die Zweckmäßigkeit der gewählten Kombination begründen.

Bei alternativlosen Methoden (z. B. Zeile 1 und 2) entfällt diese Wahlmöglichkeit. Hier müssen die Tester alle Methoden anwenden, die für den jeweiligen ASIL dringend empfohlen werden.

Bei der Überprüfung einer Anforderung nach ASIL C ergeben sich z. B. aus Tabelle 1 die folgenden Methoden:

- Methode x: dringend empfohlen. Sie ist also bei der Entwicklung nach ISO 26262 anzuwenden.
- Methode y: empfohlen. Sie sollte angewendet werden oder es muss eine Begründung gegeben werden, wenn die Methode nicht verwendet wird.
- Methoden z1 und z2: Hier ist mindestens die Methode z1 zu wählen, da sie für ASIL C die höhere Empfehlung aufweist.

ISO 26262 erlaubt den Testern auch, andere als die in den Methodentabellen aufgeführten Methoden zu verwenden. In diesem Fall muss die gewählte Methode hinsichtlich ihrer Tauglichkeit und Angemessenheit begründet werden.

Generell erlaubt die ISO 26262 die Anpassung der Sicherheitsaktivitäten an die spezifischen Bedürfnisse eines bestimmten Projekts ("Tailoring"). Für testbezogene Aktivitäten bedeutet dies, dass die Teststrategie und der Testansatz auf das jeweilige Projekt zugeschnitten werden können.

2.3 AUTOSAR

AUTOSAR steht für "AUTomotive Open System Architecture". AUTOSAR ist sowohl die Architektur als auch die dahinterstehende Entwicklungspartnerschaft. Die AUTOSAR-Partnerschaft wurde 2003 gegründet und umfasst hauptsächlich OEMs und Zulieferer aus der Automobilindustrie. Ziel der Partnerschaft ist die gemeinsame Entwicklung und Etablierung eines offenen Industriestandards für die Softwarearchitektur im Automobilbereich. Heute ist AUTOSAR ein weltweit etablierter Standard für automobile E/E-Systeme. Daher wird der Tester mit Sicherheit auf AUTOSAR-Arbeitsprodukte stoßen. Für den Tester ist es wichtig, die Projektziele von AUTOSAR, die allgemeine Struktur der AUTOSAR-Architektur und deren Einfluss auf die Arbeit des Testers zu kennen.

2.3.1 Projektziele von AUTOSAR

Im Folgenden sind die Projektziele von AUTOSAR aufgeführt:

- Unterstützung der Übertragbarkeit von Software
- Unterstützung der Skalierbarkeit über verschiedene Architekturen und Hardwarevarianten hinweg
- Unterstützung verschiedener funktionaler Domänen
- Unterstützung des Datenaustauschs mit Nicht-AUTOSAR-Systemen
- Definition einer offenen Architektur für Automobilsoftware
- Unterstützung der Entwicklung von zuverlässigen Systemen, die sich durch Verfügbarkeit, Zuverlässigkeit, Sicherheit, Integrität, Gebrauchstauglichkeit und Wartbarkeit auszeichnen
- Unterstützung der Zusammenarbeit zwischen Partnern
- Unterstützung anwendbarer internationaler Automobilstandards und modernster Technologien

2.3.2 Allgemeine Struktur von AUTOSAR [informativ]

In AUTOSAR gibt es zwei Architekturvarianten: Classic und Adaptive. Der Einfachheit halber konzentriert sich dieser Lehrplan auf AUTOSAR Classic. Ein AUTOSAR-System besteht typischerweise aus mehreren elektronischen Steuergeräten (ECUs). Auf jedem Steuergerät besteht die Softwarearchitektur von AUTOSAR Classic aus drei Schichten:

- Die oberste Anwendungsschicht ist hardwareunabhängig. Eine Komponente dieser Schicht wird als AUTOSAR-Softwarekomponente (Software Component, SW-C) bezeichnet.
- Die untere hardwareorientierte Schicht ist die standardisierte Basissoftware (Basic Software, BSW).
- Die dazwischen liegende Abstraktionsschicht ist die AUTOSAR-Laufzeitumgebung (Runtime Environment, RTE). Als eine Art Middleware implementiert sie den Datenfluss zwischen SW-Cs sowie zwischen SW-Cs und BSW.

In der AUTOSAR-Methodik zur Entwicklung automobiler Systeme tauschen OEMs und Zulieferer Systeminformationen über Beschreibungsdateien auf Basis von AUTOSAR-Templates (sogenannte "arxml-Dateien") aus:

- Die "ECU-Konfigurationsbeschreibung" enthält Daten für die Integration der SW-Cs auf einem einzelnen Steuergerät.

- Die "Systemkonfigurationsbeschreibung" enthält Daten für die Integration aller Steuergeräte eines Fahrzeugs.
- Der "ECU-Auszug" enthält die Daten aus der "Systemkonfigurationsbeschreibung" für ein einzelnes Steuergerät.

2.3.3 Einfluss von AUTOSAR auf die Arbeit des Testers

AUTOSAR beeinflusst die Arbeit des Testers, insbesondere auf den folgenden Teststufen[11]:

- Softwarekomponententest und Softwareintegrationstest in einer virtuellen Umgebung (z. B. Software-in-the-Loop): Mit Hilfe einer Simulation der BSW und der RTE kann der Tester eine SW-C frühzeitig testen.
- Softwarekomponententest und Softwareintegrationstest im realen Steuergerät: Hier erhält der Tester Zugriff auf die Kommunikation in der RTE. So kann der Tester das Verhalten einer SW-C zur Laufzeit beobachten und steuern.
- Der AUTOSAR-Akzeptanztest ist ein Test des Softwaresystems, der die Konformität der AUTOSAR-Funktionalität auf der Kommunikations- und Anwendungsebene sicherstellt. Die Durchführung des AUTOSAR-Akzeptanztests ist optional.
- Systemintegrationstest: Integration verschiedener Steuergeräte, z. B., um Funktionalität zu testen, deren Implementierung auf verschiedene Steuergeräte verteilt ist. Durch die Simulation von noch nicht implementierter Funktionalität kann der Tester das Systemverhalten frühzeitig beurteilen.

2.4 Vergleich von ASPICE, ISO 26262 und CTFL®

2.4.1 Zielsetzung von ASPICE und ISO 26262

Es gibt mehrere Normen, die Anforderungen für die Produktentwicklung vorschlagen. In der Regel beleuchten diese unterschiedliche Aspekte bei der Entwicklung. In diesem Unterabschnitt werden ISO 26262 und ASPICE hinsichtlich ihrer Ziele verglichen.

ISO 26262 hat das Ziel, Risiken durch Fehlerwirkungen sowohl bei Hardware als auch bei Software zu vermeiden, indem geeignete Anforderungen und Prozesse bereitgestellt werden. Für die Entwicklung von E/E-Systemen definiert sie die Anforderungen an die Testprozesse und -methoden[12], die vom Tester anzuwenden sind. Diese sind abhängig vom ASIL des Items.

ASPICE dient dazu, die Fähigkeit des Produktentwicklungsprozesses im Rahmen von Assessments zu ermitteln. Dazu definiert ASPICE bewertbare Kriterien für diese Prozesse. Diese sind im Gegensatz zur ISO 26262 unabhängig vom ASIL des Produkts.

2.4.2 Vergleich der Teststufen zwischen ASPICE, ISO 26262 und CTFL®

Sowohl ISO 26262 als auch ASPICE beschreiben Teststufen. Diese stimmen jedoch nicht vollständig mit den Teststufen des CTFL® überein. Für eine effiziente und effektive Zusammenarbeit innerhalb des Entwicklungsteams sollten die Tester daher ein gemeinsames Verständnis aller Teststufen haben.

Der in ASPICE verwendete Begriff "System" sowie die in der ISO 26262 verwendeten Begriffe "System" und "Item" beziehen sich auf ein Produkt, das aus Hardware- und Softwarekomponenten

[11] Für eine Übersicht der Teststufen siehe Abschnitt 2.4.2.
[12] Zu den Methoden der ISO 26262 siehe Abschnitt 2.2.5.

besteht. Der CTFL®-Lehrplan [ISTQB_CTFL] bezieht sich jedoch nur auf Software, wenn er den Begriff "System" verwendet. Daher können die Teststufen des ISTQB® wie folgt auf die Teststufen in ISO 26262 und ASPICE abgebildet werden. Die in Klammern angegebenen Zahlen in der Spalte ISO 26262 beziehen sich auf einen bestimmten Teil der Norm ISO 26262 und auf ein bestimmtes Kapitel darin. In der ASPICE-Spalte steht in der Klammer die jeweilige Prozess-ID.

ISTQB® CTFL	ISO 26262:2018	ASPICE 4.0
Abnahmetests	Sicherheitsvalidierung (4-8)[13]	Validierung (VAL.1)[14]
Testen von Systemen von Systemen[15]	System- und Item-Integration und Test (4-7)[16]	Systemverifizierung (SYS.5)
Systemintegrationstests		Systemintegration und Integrationsverifizierung (SYS.4)
Systemtest	Testen der eingebetteten Software (6-11)[17]	Softwareverifizierung (SWE.6)
Komponentenintegrationstest[18]	Softwareintegration und -verifizierung (6-10)	Softwarekomponentenverifizierung und Integrationsverifizierung (SWE.5)
Komponententest[19]	Software-Unit-Verifizierung (6-9)	Software-Unit-Verifizierung (SWE.4)

Tabelle 2: Zuordnung der Teststufen

Die meisten Testverfahren aus dem CTFL®-Lehrplan [ISTQB_CTFL] können auf verschiedenen Teststufen eingesetzt werden. In ähnlicher Weise ordnet ASPICE Testverfahren nicht generell Teststufen zu. Daher überlassen beide die Wahl den Testern. In der ISO 26262 gibt es für jede Teststufe einzelne Methodentabellen (siehe Abschnitte 2.2.4 und 2.2.5). Diese geben dem Tester Empfehlungen für Testverfahren auf der Grundlage der ASIL-Stufe.

[13] Die Sicherheitsvalidierung umfasst nur Teile eines Abnahmetests nach ISTQB.

[14] Die Validierung (VAL.1) deckt nur Teile eines Abnahmetests nach ISTQB ab.

[15] Das Testen von mehreren heterogenen verteilten Systemen. Im CTFL korrespondieren die Prozesse von ASPICE (SYS.5) und ISO 26262 (4-7) mit dem Systemtest (im Sinne von integrierten Systemen)

[16] Die Integration und der Test eines Items umfasst drei Phasen: Hardware-Software-Integration und -Test eines Elements, Integration und Test aller Elemente, die zu diesem Item gehören, sowie die Integration und der Test des Items in Verbindung mit anderen Items im Fahrzeug.

[17] Test der Anforderungen an die Software für funktionale Sicherheit, ausgeführt auf der Zielhardware.

[18] Die Prozesse von ASPICE (SWE.5) und ISO 26262 (6-10) decken zusätzlich den Komponententest (im Sinne von „component“) im CTFL ab.

[19] Die Prozesse von ASPICE (SWE.4) und ISO 26262 (6-9) decken den Komponententest (im Sinne von „unit“) im CTFL ab

3 Testen in einer virtuellen Umgebung – 160 Minuten

Schlüsselbegriffe

Umgebungsmodell, Hardware-in-the-Loop, Model-in-the-Loop, Software-in-the-Loop

Domänenspezifische Schlüsselbegriffe

Closed-Loop-System, Open-Loop-System

Lernziele für Kapitel 3 **Der Lernende kann ...**

3.1 Testumgebung im Allgemeinen

AuT-3.1.1 (K1) … sich an die Motivation für eine Testumgebung in der Automobilentwicklung erinnern

AuT-3.1.2 (K1) … die allgemeinen Bestandteile einer automobilspezifischen Testumgebung wiedergeben

AuT-3.1.3 (K2) … die Unterschiede zwischen Closed-Loop-Systemen und Open-Loop-Systemen erläutern

AuT-3.1.4 (K1) … die wesentlichen Funktionen, Datenbasen und Protokolle eines Steuergeräts wiedergeben

3.2 Testen in XiL-Testumgebungen

AuT-3.2.1.1 (K1) … den Aufbau einer MiL-Testumgebung wiedergeben

AuT-3.2.1.2 (K2) … die Einsatzgebiete und Randbedingungen einer MiL-Testumgebung erläutern

AuT-3.2.2.1 (K1) … den Aufbau einer SiL-Testumgebung wiedergeben

AuT-3.2.2.2 (K1) … die Einsatzgebiete und die Randbedingungen einer SiL-Testumgebung nennen

AuT-3.2.3.1 (K1) … den Aufbau einer HiL-Testumgebung wiedergeben

AuT-3.2.3.2 (K2) … die Einsatzgebiete und die Randbedingungen einer HiL-Testumgebung erläutern

AuT-3.2.4.1 (K2) … die Vor- und Nachteile des Testens anhand von Kriterien für XiL-Testumgebungen zusammenfassen

AuT-3.2.4.2 (K3) … Kriterien für die Zuordnung eines bestimmten Umfangs des Tests zu einer oder mehreren Testumgebungen anwenden

AuT-3.2.4.3 (K1) … die XiL-Testumgebungen im V-Modell skizzieren

3.1 Testumgebung im Allgemeinen

3.1.1 Motivation für eine Testumgebung in der Entwicklung im Automobilbereich

Der Tester steht vor besonderen Herausforderungen. Einerseits wird vom Tester erwartet, dass er so früh wie möglich mit dem Testen beginnt, um Fehlerzustände frühzeitig im Entwicklungsprozess zu finden. Andererseits benötigt der Tester eine realistische Umgebung, um das System zu testen und die Fehlerzustände zu finden, die im fertigen Produkt auftreten würden. Der Tester kann diesen Konflikt lösen, indem er geeignete Testumgebungen verwendet, die zu den verschiedenen Entwicklungsphasen passen. Auf diese Weise kann der Tester einzelne Testaufgaben implementieren und ausführen, bevor das fertig produzierte oder entwickelte Steuergerät zur Verfügung steht. Durch den Einsatz verschiedener Testumgebungen kann der Tester Situationen simulieren, die im realen Fahrzeug nur schwer reproduzierbar sind, wie z. B. Kurzschlüsse und Unterbrechungen in Kabelbäumen oder Überlastungen der Netzwerkkommunikation.

3.1.2 Allgemeine Teile einer Testumgebung

Damit der Tester die Tests durchführen kann, benötigt er eine Testumgebung, in der die fehlenden Teile simuliert werden. Diese Umgebung hilft dem Tester, die Eingaben des Testobjekts zu steuern und die Ausgänge zu beobachten, auch "Point of Control" (PoC) und "Point of Observation" (PoO) genannt. Nach ISO/IEC/IEEE 29119-1 besteht eine Testumgebung aus den folgenden Teilen:

- Hardware der Testumgebung (z. B. ein Steuerrechner, ggf. ein echtzeitfähiger Rechner, ein Prüfstand und ein Entwicklungskit)
- Software der Testumgebung (z. B. Betriebssystem, Simulationssoftware und Umgebungsmodelle)
- Kommunikationsmöglichkeiten (z. B. Netzwerkzugang und Datenlogger)
- Werkzeuge (z. B. Oszilloskop und Messgeräte)
- Labor (z. B. Schutz vor elektromagnetischer Strahlung und Lärm)

Ein wichtiger Bestandteil der Testumgebung ist das Umgebungsmodell. Modelle sind Schlüsselelemente der virtuellen Testumgebung. Sie repräsentieren Aspekte der realen Welt wie den Verbrennungsmotor, Getriebe, Fahrzeugsensoren und Steuergeräte oder auch den Fahrer und die Straßenbedingungen. Die Testumgebung bietet auch verschiedene Zugangspunkte. Diese erlauben es dem Tester, das Testobjekt zu beobachten, zu messen und es bei Bedarf zu beeinflussen oder zu steuern.

3.1.3 Unterschiede zwischen Closed-Loop- und Open-Loop-Systemen

Die Testumgebung dient dazu, die Eingangsschnittstellen des zu testenden Geräts zu steuern (PoC) und seine Ausgabe über die Ausgangsschnittstellen zu überwachen (PoO). Anschließend wird das Verhalten an den Ausgangsschnittstellen analysiert. Ein erfolgreiches Testergebnis ist gegeben, wenn die beobachteten Ausgaben mit den erwarteten Ergebnissen übereinstimmen.

Im Allgemeinen gibt es zwei Arten von Regelsystemen: Closed-Loop-Systeme und Open-Loop-Systeme. Der Unterschied liegt in der Art und Weise, wie das Steuergerät auf seine Umgebung reagiert. Dies führt zu unterschiedlichen Anforderungen an die virtuelle Testumgebung.

3.1.3.1 Closed-Loop-System

Die Stimulation in einem Closed-Loop-System, auch In-the-Loop genannt, berücksichtigt die Ausgaben des Testobjekts. Dies geschieht über ein Umgebungsmodell, das die Ausgaben sammelt und sie direkt oder indirekt an den Eingang des Testobjekts weiterleitet. Es entsteht ein Regelkreis in der Testumgebung.

Closed-Loop-Systeme werden häufig zum Testen von Reglern eingesetzt. Damit lassen sich komplexe Funktionen wie Motor- und Getriebesteuerungen und Fahrerassistenzsysteme wie das Antiblockiersystem (ABS®) oder die Fahrdynamikregelung (Vehicle Dynamic Control, VDC®) testen.

3.1.3.2 Open-Loop-System

Bei einem Open-Loop-System stehen die Ausgänge des Systems in keinem Zusammenhang mit den Eingängen. Das System verfügt über keine Rückkopplungsschleife. In diesem Fall werden die Eingänge des Testobjekts direkt vom Tester im Testablauf definiert.

Der Anwendungsfall für Open-Loop-Systeme und Closed-Loop-Systeme hängt stark vom Verhalten des Testobjekts ab. Closed-Loop-Systeme haben Ausgänge, die die Eingänge beeinflussen und so eine Rückkopplung erzeugen, während Open-Loop-Systeme diesen Rückkopplungsmechanismus nicht besitzen. Wenn das Testobjekt ein reaktives Verhalten aufweist oder einen Zustandsautomaten widerspiegelt, wird ein Open-Loop-System bevorzugt. In der Innenraum- und Fahrwerkselektronik gibt es viele Beispiele für Open-Loop-Systeme (z. B. Leuchten und Schalter).

3.1.4 Datenbasen und Kommunikationsprotokolle eines Steuergeräts

Ein Steuergerät in der Automobilumgebung funktioniert als eingebettetes System, das aus Hardware und Software besteht. Das Steuergerät empfängt verschiedene analoge und digitale Eingaben, die ständig Umgebungsdaten in Form von Spannung, Stromstärke und Temperatur erfassen. Bussysteme dienen der Kommunikation und stellen dem Steuergerät weitere Informationen zur Verfügung, die von Sensoren oder anderen Steuergeräten stammen. Diese Informationen werden vom Steuergerät entweder selbst empfangen und verarbeitet oder generiert und versendet. Das Testobjekt verwaltet die Daten im Speicher, um die ausgegebenen Aktionen, Informationen oder Daten zu verarbeiten. Die erzeugten Ausgaben werden auch über analoge und digitale Ausgangspins, Bussysteme oder Diagnoseschnittstellen ausgeführt.

Die Datenbasen dienen als beschreibende Referenz und definieren die Eingangs- und Ausgangssignale des Steuergeräts. Zu diesen Daten gehören auch Beschreibungen, Einheiten und Umrechnungsformeln der Signale.

Die Kommunikationsprotokolle beschreiben den Datenaustausch über die entsprechenden physikalischen Schnittstellen. Diese Protokolle legen fest, welche Spannung oder Bitfolge welchen Wert des Signals repräsentiert.

Die Auswahl der Datenbasen und Kommunikationsprotokolle hängt von der Funktion des Steuergeräts ab. Für den Zugriff auf Diagnosefunktionen im Steuergerät benötigt der Tester beispielsweise Informationen über die verwendete Datenbasis (z. B. Application Programming Interface Specification (ASAM[20] MCD-3 D) und serviceorientierte Fahrzeugdiagnose (ASAM SOVD) sowie das Kommunikationsprotokoll (z. B. Unified Diagnostic Services nach ISO 14229 und AUTOSAR- Spezifikation (SOME/IP)). Weitere automobilspezifische Datenbasen sind in den ASAM- und AUTOSAR-Standards definiert.

[20] Association for Standardization of Automation and Measuring Systems

3.2 Testen in XiL-Testumgebungen

In der Automobilindustrie werden die folgenden Arten von XiL-Testumgebungen[21] verwendet:

- Model-in-the-Loop (MiL)
- Software-in-the-Loop (SiL)
- Processor-in-the-Loop[22] (PiL)
- Hardware-in-the-Loop (HiL)
- Vehicle-in-the-Loop[23] (ViL)

Der Tester sollte sich mit den Testumgebungen (d. h. MiL, SiL und HiL) vertraut machen und sie verstehen. In den folgenden Abschnitten werden der Aufbau und die Anwendungsbereiche der verschiedenen Testumgebungen näher erläutert.

3.2.1 Model-in-the-Loop (MiL)

3.2.1.1 Aufbau einer MiL-Testumgebung

In einer MiL-Testumgebung ist das Testobjekt als Modell verfügbar. Dieses Modell ist ausführbar, aber nicht für spezielle Hardware kompiliert. Die Entwickler erstellen die Modelle mit speziellen Modellierungswerkzeugen. Der Tester benötigt eine Testumgebung, um diese Modelle ausführen und testen zu können. Diese wird in der Regel in der gleichen Entwicklungsumgebung implementiert wie das Testobjekt selbst. Diese Testumgebung kann zusätzlich ein Umgebungsmodell enthalten. Der Tester kann das Testobjekt über Zugangspunkte steuern und beobachten. Die Zugangspunkte können sowohl im Modell des Testobjekts als auch im Umgebungsmodell beliebig platziert werden. Das Modell des Testobjekts ist mit dem Umgebungsmodell verbunden und lässt sich leicht als Closed-Loop-System implementieren und nutzen.

3.2.1.2 Anwendungsbereiche und Randbedingungen einer MiL-Testumgebung

Mit einer MiL-Testumgebung kann der Tester den funktionalen Systementwurf testen. Während der Entwicklung (z. B. dem allgemeinen V-Modell folgend) kann der Tester auch einzelne Komponenten bis hin zu einem gesamten System testen. Zur Durchführung des Tests benötigt der Tester einen Computer und die entsprechende Simulationssoftware einschließlich des Umgebungsmodells. Das Umgebungsmodell wird mit zunehmendem Funktionsumfang des Testobjekts immer komplexer. Die Abbildung der Realität und der Umweltfaktoren ist sehr komplex. Auch die Ausführungszeiten für die Modelle steigen überproportional an. Daher lohnt sich der Aufwand für die Implementierung einer MiL-Testumgebung ab einer bestimmten Phase der Entwicklung nicht mehr.[24]

[21] Der Buchstabe X in XiL ist ein Platzhalter für die spezifischen Testumgebungen in der angegebenen Liste.
[22] Diese Testumgebung wird in diesem Lehrplan nicht berücksichtigt und ist rein informativ.
[23] Diese Testumgebung wird in diesem Lehrplan nicht berücksichtigt und ist rein informativ.
[24] Dies gilt auch für alle anderen XiL-Testumgebungen.

Durch den Einsatz einer MiL-Testumgebung kann der Tester bereits in einer frühen Phase der Entwicklung die funktionale Eignung von Modellen über alle Entwicklungsstufen hinweg testen (d. h. die linke Seite des V-Modells. Es ist aber nicht üblich, das Umgebungsmodell in die Lage zu versetzen, Bus- oder Diagnosefunktionen oder physikalisches Verhalten wie Kabelbrüche oder Kurzschlüsse zu simulieren. Diese Aufgaben können mit anderen Testumgebungen einfacher und kostengünstiger durchgeführt werden.

In einer MiL-Testumgebung findet die Testdurchführung nicht in Echtzeit statt. Da alle Komponenten als Modell zur Verfügung stehen, läuft die Testdurchführung in Simulationszeit ab. Je komplexer ein System ist, desto mehr Ausführungszeit bzw. Leistung benötigt der Computer, um alle notwendigen Informationen bereitzustellen. Die Dauer der Simulation ist bei kleineren Systemen kürzer als die Ausführung in Echtzeit. Ein großer Vorteil ist, dass der Tester die Simulation jederzeit für eine detaillierte Analyse und Bewertung pausieren kann.

3.2.2 Software-in-the-Loop (SiL)

3.2.2.1 Aufbau einer SiL-Testumgebung

Das Testobjekt wird für eine bestimmte SiL-Testumgebung kompiliert. Das bedeutet, dass der Quellcode für eine bestimmte Rechnerarchitektur kompiliert wurde. Dieser Maschinencode ist für die Testumgebung lesbar, da er aus binären Datensätzen besteht. Damit die Testumgebung auf die Signale zugreifen kann, ist ein WrapperEin Wrapper ist eine zusätzliche Software, die den Zugang zu Eingängen und Ausgängen erlaubt, die möglicherweise nicht zugänglich sind, wenn das Testobjekt direkt und nicht über den Wrapper mit der Umgebung kommuniziert. So kann der Tester Softwaresignale steuern und beobachten. Der Wrapper definiert die Zugangspunkte zum Testobjekt, übernimmt aber nicht dessen funktionale Aufgaben.

Für die Simulation wird ein Umgebungsmodell benötigt. Mit Hilfe des Wrappers wird das Testobjekt mit der Testumgebung verbunden. Die Testdurchführung erfolgt auf einem Computer ohne spezielle Hardware. Der Tester benötigt ein Software-Tool, das einen Wrapper für das Testobjekt mit Zugängen zur Testumgebung erstellen kann.

3.2.2.2 Anwendungsbereiche und Randbedingungen einer SiL-Testumgebung

Wenn der Entwickler Quellcode auf der Grundlage eines Modells generiert, kann sich das tatsächliche Verhalten der Software von dem erwarteten Verhalten unterscheiden. Dies kann durch unterschiedliche Datentypen im Modell, meist Gleitkomma, und im kompilierten Softwarecode, meist Festkomma, aber auch durch unterschiedliche Speicherbereiche verursacht werden. Diese Anomalien im erwarteten Verhalten können zum ersten Mal in einer SiL-Testumgebung getestet werden. Der Tester kann einen Testansatz wie Back-to-Back-Tests (siehe Abschnitt 4.2.2) verwenden, um das Verhalten zu vergleichen.

Der Tester führt die Tests, analog zur MiL-Testumgebung, in Simulationszeit aus. Je nach Berechnungsverfahren und Komplexität des Umgebungsmodells kann diese Simulationszeit kürzer oder länger als in Echtzeit sein. Der Tester kann die Testdurchführung jederzeit für eine detaillierte Analyse und Bewertung anhalten. Funktionale Tests, Schnittstellentests und Regressionstests sind sehr häufige Testarten, die in einer SiL-Testumgebung durchgeführt werden. Performanz-, Effizienz- und Zuverlässigkeitstests sind dagegen in SiL-Testumgebungen eher unüblich. Diese Qualitätsmerkmale werden meist von der Zielhardware beeinflusst.

3.2.3 Hardware-in-the-Loop (HiL)

3.2.3.1 Aufbau einer HiL-Testumgebung

Steht das Testobjekt als Prototyp zur Verfügung oder ist es bereits fertig entwickelt, kann der Tester eine HiL-Testumgebung zur Durchführung von Tests nutzen. Die typischen Bestandteile einer HiL-Testumgebung sind:

- Ein Netzteil zum Einstellen verschiedener Versorgungsspannungen
- Ein echtzeitfähiger Computer, auf dem das Umgebungsmodell ablaufen soll
- Mehrere reale Teile, die nicht im Umgebungsmodell implementiert sind
- Eine Signalverarbeitungseinheit für Signalart und Signalamplitude
- Eine Fault Insertion Unit (FIU) (siehe Abschnitt 4.2.3) für die Simulation von Kabelbrüchen und Kurzschlüssen
- Eine Breakout-Box als zusätzliche Zugangsschnittstelle im Kabelbaum
- Eine Restbussimulation der nicht vorhandenen Busteilnehmer

3.2.3.2 Einsatzgebiete und Randbedingungen einer HiL-Testumgebung

Die Zugangspunkte in einer HiL-Testumgebung sind vielfältig. Der Tester muss sich darüber im Klaren sein, dass die Verwendung der falschen Zugangspunkte zum Testobjekt die Testergebnisse unbrauchbar machen kann. Die Kenntnis der verschiedenen Zugangspunkte und ihrer Verbindungsfähigkeiten in der HiL-Testumgebung ermöglicht es, effektive Tests zu implementieren, durchzuführen und auszuwerten.

Die HiL-Testumgebung ist aufgrund ihrer Vielschichtigkeit komplexer als die zuvor genannten Testumgebungen (d. h. MiL und SiL). Der Tester muss diese Komplexität beherrschen, um Testaufgaben zu bewältigen. Die HiL-Testumgebung kann für Komponententests, Integrationstests und Systemtests eingesetzt werden. Ziel ist es unter anderem, funktionale und nicht-funktionale Fehlerzustände in der Soft- und Hardware zu finden.

Mit Hilfe von HiL-Testumgebungen können verschiedene Teststufen analysiert werden. Handelt es sich bei dem Testobjekt um ein einzelnes Steuergerät, so spricht man von einem Komponenten-HiL[25]. Handelt es sich bei dem Testobjekt um eine Kombination aus mehreren Steuergeräten, spricht man von einem System-HiL. Mit dem Komponenten-HiL testet der Tester Funktionen des Steuergeräts. Beim System-HiL steht das Testen des Datenaustauschs zwischen den Steuergeräten und der Systemtest im Vordergrund.

Im Gegensatz zu den zuvor genannten Testumgebungen (z. B. MiL und SiL) läuft die Simulationszeit in einer HiL-Testumgebung immer in Echtzeit ab. Der Grund dafür ist, dass die Software auf realer Hardware läuft. Ein Pausieren oder Anhalten ist in dieser Testumgebung nicht mehr möglich. Zur Testumgebung gehört daher ein echtzeitfähiger Rechner, der alle relevanten Signale innerhalb einer vorgegebenen Zeitspanne erfassen und ausgeben kann.

3.2.4 Vergleich der XiL-Testumgebungen

3.2.4.1 Vor- und Nachteile des Testens in den XiL-Testumgebungen

Der Tester versteht die Eigenschaften der verschiedenen Testumgebungen. Auf diese Weise kann der Tester die Vor- und Nachteile des Testens in den einzelnen Testumgebungen verstehen und beurteilen. Die Kriterien sind in Tabelle 3 aufgeführt.

Kriterien	XiL-Testumgebung	Auswirkung	MiL	SiL	HiL
Realitätsnähe	Die Realität wird simuliert, viele Merkmale werden abstrahiert, der Schwerpunkt liegt auf den Strukturen und der Logik.	Geringe	+	o	o

[25] Der Begriff Komponente wird in diesem Fall für ein Steuergerät im Kontext eines E/E-Systems verwendet.

Kriterien	XiL-Testumgebung	Auswirkung	MiL	SiL	HiL
	Kompilierte reale Software kann ohne spezielle Hardware ausgeführt werden.	Niedrig bis mittel	o	+	o
	Integriertes System, ausführbar	Hoch	o	o	+
Zeit und Aufwand für Debugging	Fehlerzustände werden im Modell des Testobjekts gefunden (Modellanpassung).	Niedrig	+	o	o
	Fehlerzustände werden in der programmierten Software gefunden (Softwareanpassung).	Mittel	o	+	o
	Fehlerzustände werden auf der Systemebene gefunden (Systemanpassung).	Hoch	o	o	+
Aufwand für Implementierung und Wartung	Erstellen eines Umgebungsmodells	Gering	+	o	o
	Umgebungsmodell und Wrapper erstellen	Mittel	o	+	o
	Erstellen des Umgebungsmodells und Verdrahten der Hardwarekomponenten	Hoch	o	o	+
Aufwand für die Vorbereitung des Tests	Die Umgebung kann schnell eingerichtet werden.	Niedrig	+	o	o
	Die Umgebung kann schnell eingerichtet werden.	Mittel	o	+	o
	Entwurf, Umsetzung und Auswertung der Tests erfordern einen hohen Aufwand.	Hoch	o	o	+
Erforderlicher Reifegrad des Testobjekts	Systemmodelle werden simuliert.	Niedrig	+	o	o
	Erste Funktionen werden mit der Zielsoftware getestet.	Mittel	o	+	o
	Ein oder mehrere lauffähige Steuergeräte oder Teilsysteme werden möglichst vollständig getestet.	Hoch	o	o	+
Erforderlicher Detaillierungsgrad der Testbasis (Spezifikation)	Es werden Modelle getestet, die teilweise eine unvollständige Spezifikation abdecken.	Mittel	+	o	o
	Die relevanten Informationen auf der Softwareebene müssen verfügbar sein (z. B. detaillierte Spezifikation der Komponenten).	Mittel bis hoch	o	+	o

Kriterien	XiL-Testumgebung	Auswirkung	MiL	SiL	HiL
	Anforderungen können auf der Systemebene mit einer vollständigen Systemspezifikation getestet werden.	Hoch	o	o	+
Zugänglichkeit zum Testobjekt	Alle Signale in einem Modell können beobachtet und gesteuert werden.	Hoch	+	o	o
	Nur die im Wrapper verfügbaren Signale können beobachtet und gesteuert werden.	Mittel	o	+	o
	Es können nur die in der Hardware oder den Kommunikationsprotokollen verfügbaren Signale beobachtet und gesteuert werden.	Niedrig	o	o	+

Tabelle 3: Kriterien und ihre Auswirkungen auf MiL-, SiL- und HiL-Testumgebungen

3.2.4.2 Anwendung von Kriterien für die Zuordnung eines bestimmten Testumfangs zu einer oder mehreren Testumgebungen

In der folgenden Tabelle werden die Testziele detailliert beschrieben und den geeigneten Testumgebungen zugeordnet.

Testziel	Beschreibung anhand von Beispielen	MiL	SiL	HiL
Test der Kundenanforderungen	Korrekte Bereitstellung der geforderten Funktionalität. Dazu gehören die korrekte Verarbeitung von Eingaben, die korrekte Reaktion auf Eingaben und die korrekte Datenausgabe am Ausgang.	o	o	+
Testmechanismen zur Erkennung und Behandlung von Fehlerzuständen	• Erkennung und Behandlung von zufälligen Fehlerzuständen der Hardware • Erkennung und Behandlung von Fehlerzuständen der Software • Überführung in einen sicheren Zustand nach Erkennung von Fehlerzuständen, z. B. Deaktivierung eines Systems	+	+	+
Test der Reaktion auf Konfigurationsdaten	Überprüfung des Einflusses von Konfigurationsdaten auf das Verhalten des Testobjekts	o	+	+
Testen von Diagnosefunktionen	Korrekte Bereitstellung der erforderlichen Diagnosefunktionalität, wie Fehlererkennung, Fehleraktivierung und Rücksetzanforderung sowie Fehleraktivierung im Fehlerspeicher (z. B. On-Board-Diagnose oder in der Werkstatt)	-	+	+

Testziel	Beschreibung anhand von Beispielen	MiL	SiL	HiL
Testen des Zusammenspiels an Schnittstellen	Überprüfung der internen und externen Schnittstellen des Testobjekts	o	+	+
Gebrauchstauglichkeit nachweisen	Das Testobjekt sollte die Gebrauchstauglichkeitsanforderungen des Benutzers erfüllen.	-	o	+
Schlüssel: + empfohlen, o möglich, - nicht sinnvoll				

Tabelle 4: Vergleich der Testarten in MiL-, SiL- und HiL-Testumgebungen

Tabelle 4 zeigt, dass Testumgebungen für bestimmte Testziele geeignet sind. Dieser diversifizierte Ansatz wird insbesondere beim Testen der Mechanismen zur Fehlererkennung und -behandlung deutlich. Gemäß Shift-Left lautet die allgemeine Schlussfolgerung, dass grundlegende Anforderungs- und Fehlerzustände bereits durch das Testen frühzeitig erkannt werden. Daher wird MiL für die Erkennung von allgemeinen Designfehlern, SiL hauptsächlich für technische Softwarefehler und HiL für technische Hardware-/Softwarefehler verwendet. Weiterhin ist zu beachten, dass bei allen Testarten neben dem Nachweis von Zuverlässigkeit, Performanz-Effizienz und Gebrauchstauglichkeit die funktionale Eignung des Testobjekts im Vordergrund steht.

In der Teststrategie ordnet der Testmanager den Testumfang mehreren unterschiedlichen Testumgebungen zu. Die Tester sollten sich regelmäßig abstimmen, um effizientes und genaues Testen über verschiedene Testumgebungen hinweg zu gewährleisten. Gegebenenfalls empfiehlt es sich, die Testziele abzustimmen, Doppelarbeit zu vermeiden und sicherzustellen, dass jeder Test in der für seine Entwicklungsphase am besten geeigneten Testumgebung durchgeführt wird. Diese Zusammenarbeit trägt dazu bei, Ressourcenkonflikte zu vermeiden, eine konsistente Überdeckung zu gewährleisten und das Risiko des Testens auf unangemessenen Teststufen zu verringern. Durch die Förderung der Kommunikation und den Einsatz gemeinsamer Tools oder Frameworks können Teams ihre Teststrategien optimieren und einen nahtlosen Verlauf des Lebenszyklus von Tests aufrechterhalten. Durch die Kombination der Kriterien aus den Tabellen 3 und 4 kann der Testmanager die optimale Testumgebung auswählen.

3.2.4.3 Einordnung der XiL-Testumgebungen in das V-Modell

Der technische Systementwurf befindet sich auf der linken Seite des V-Modells. Der Tester kann diesen Entwurf mit einer MiL-Testumgebung testen. Wenn das Modell und die Testumgebung weiterentwickelt werden, kann der Tester auch Komponententests und Integrationstests mit dieser Testumgebung durchführen.

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

Bei Systemtests sind bestimmte Funktionalitäten des Testobjekts vollständig entwickelt worden. Der Tester kann den Systemtest mit einer HiL-Testumgebung durchführen.

Mit einer korrekten Zuordnung der Testumgebung zu den Teststufen kann der gesamte Testprozess nach den folgenden drei Gesichtspunkten optimiert werden:

Minimierung der Produktrisiken

- Auffinden von teststufenspezifischen Fehlerwirkungen (z. B. Performanz-Effizienz auf Systemebene innerhalb einer HiL-Umgebung)

Minimierung der Kosten für den Test

- Für jede Testart ist die optimale Teststufe gewählt.
- Tests werden auf frühere, weniger kostspielige und virtuelle Teststufen verlagert.

Konformität mit Normen

- In den Methodentabellen der Norm ISO 26262 werden Testumgebungen in Abhängigkeit von ASIL empfohlen.

4 Statische und dynamische Tests – 230 Minuten

Schlüsselbegriffe

Back-to-Back-Test, Fehlereinfügung, anforderungsbasierter Test

Lernziele für Kapitel 4 **Der Lernende kann ...**

4.1 Statischer Test

AuT-4.1.1 (K2) … Zweck und Anforderungen der MISRA-C-Programmierrichtlinie anhand von Beispielen erläutern

AuT-4.1.2 (K3) … ein Anforderungsreview anhand der für den Tester relevanten Qualitätsmerkmale der Norm ISO/IEC/IEEE 29148 durchführen

4.2 Dynamischer Test

AuT-4.2.1 (K3) … Testfälle entwerfen, um eine modifizierte Bedingungs-/Entscheidungsüberdeckung zu erreichen

AuT-4.2.2 (K2) … den Einsatz von Back-to-Back-Tests anhand von Beispielen erläutern

AuT-4.2.3 (K2) … Testen mit Fehlereinfügung anhand von Beispielen erläutern

AuT-4.2.4 (K1) … die Prinzipien des anforderungsbasierten Tests wiedergeben

AuT-4.2.5 (K3) … kontextabhängige Kriterien für die Auswahl geeigneter und notwendiger Testverfahren und Testansätze anwenden

4.1 Statischer Test

Einführung

Statische Tests untersuchen Arbeitsprodukte der Softwareentwicklung, ohne sie auszuführen. Dazu gehören die Bewertung durch Gutachter (Reviewer) und die werkzeuggestützte statische Analyse.

4.1.1 Die MISRA-C-Programmierrichtlinien

Es ist Stand der Technik, dass Quellcode konform zu Programmierrichtlinien sein soll. Auch die Norm ISO 26262 empfiehlt die Einhaltung von Programmierrichtlinien für sicherheitsrelevante Software[26]. Programmierrichtlinien helfen, Anomalien im Code zu vermeiden, die zu Fehlerzuständen führen können. Gleichzeitig unterstützen sie Wartbarkeit und Übertragbarkeit.

MISRA-C definiert Programmierrichtlinien für die Programmiersprache C und wird üblicherweise in Softwareprojekten für die Automobilindustrie verwendet. Es definiert zwei verschiedene Arten von Richtlinien: Regeln und Direktiven.

- Regeln sind mit Hilfe von Werkzeugen der statischen Analyse überprüfbar. Beispiel einer Regel: "Der Quellcode darf keine verschachtelten Kommentare enthalten."
- Direktiven können nicht vollständig von Werkzeugen der statischen Analyse verifiziert werden. Sie beziehen sich auf Details des Entwicklungsprozesses oder auf Dokumente außerhalb der Software. Beispiel einer Direktive: "Der Entwickler muss das implementierte Verhalten ausreichend dokumentieren."

Jede Richtlinie wird eine der folgenden drei Verbindlichkeitsstufen zugewiesen:

- "Advisory"-Richtlinien müssen vom Entwickler befolgt werden, wenn der Aufwand zumutbar ist.
- "Required"-Richtlinien müssen vom Entwickler befolgt werden; Ausnahmen sind möglich, müssen aber offiziell genehmigt werden (z. B. durch das Qualitätsmanagement).
- "Mandatory"-Richtlinien müssen vom Entwickler befolgt werden; es gibt keine Ausnahmen.

Organisationen können die Verbindlichkeit für eine Richtlinie individuell anheben, dürfen sie aber niemals absenken.

4.1.2 Qualitätsmerkmale für Anforderungsreviews

Spezifikationen sind die Grundlage für Entwicklung und Testen. Fehlerzustände in Spezifikationen führen daher zu kosten- und zeitintensiver Nacharbeit. Dies gilt insbesondere dann, wenn Fehlerzustände erst in späten Entwicklungsphasen wie Abnahmetests oder im Betrieb festgestellt werden. Reviews sind eine effektive Maßnahme, um Fehlerzustände in Spezifikationen frühzeitig zu finden und folglich auch frühzeitig und kostengünstig zu beheben.

Bei der Testanalyse muss der Tester die Spezifikationen für das Testobjekt überprüfen, insbesondere auf ihre Eignung als Testbasis. Qualitätsmerkmale helfen dem Tester beim Review der Spezifikationen, sich zu fokussieren und möglichst viele Fehlerzustände zu finden. Die ISO 29148:2018 enthält Qualitätsmerkmale für einzelne Anforderungen und für ganze Anforderungsspezifikationen.

[26] Siehe auch ISO 26262:2018, Teil 6, Abschnitt 5.4.3.

Zu den für Tester relevanten Qualitätsmerkmalen der ISO 29148:2018 gehören:

- Überprüfbar: Jede Anforderung kann nachweislich korrekt umgesetzt werden.
- Eindeutig: Jede Anforderung enthält klare Testbedingungen.
- Konsistent: Jede Anforderung ist in sich und mit allen anderen Anforderungen widerspruchsfrei.
- Vollständig: Jede Anforderung lässt keinen Interpretationsspielraum zu und baut nicht auf implizitem Wissen oder Erfahrungswissen auf.
- Atomar: Keine Anforderung kann in sinnvolle Teilanforderungen zerlegt werden.

Aus diesen Qualitätsmerkmalen kann der Tester zur Vorbereitung eines Anforderungsreviews eine Review-Checkliste mit Checklistenpunkten ableiten. Diese Checklistenpunkte müssen spezifischer sein als die einfache Benennung der Qualitätskriterien, die zu abstrakt sind. Ein Checklistenpunkt für Konsistenz könnte beispielsweise lauten: "Werden Begriffe über alle Anforderungen hinweg konsistent verwendet?" Beim Review der Spezifikation muss der Tester die Checklistenpunkte nach bestem Wissen und Gewissen beantworten.

Laut ISO 29148:2018 sollen Anforderungen auch realisierbar, angemessen, korrekt, konform, verständlich und notwendig sein. Allerdings ist es für den Tester meist schwierig, diese Qualitätsmerkmale zu bewerten.

4.2 Dynamischer Test

4.2.1 Modifizierter Bedingungs-/Entscheidungstest

Die in diesem Kapitel beschriebenen Testverfahren sind Teil der White-Box-Testverfahren. Der Tester leitet dabei die Testfälle unmittelbar aus der Struktur des Testobjekts (z. B. des Quellcodes) ab. Für detaillierte Informationen wird auf den Lehrplan zum Technical Test Analyst [ISTQB_CTAL-TTA] verwiesen.

Beim Entscheidungstest werden die Testfälle so spezifiziert, dass sie alle möglichen Entscheidungsergebnisse ausführen, d. h. die "wahren" und "falschen" Ergebnisse jeder Entscheidung. Eine Entscheidung besteht aus einer oder mehreren Bedingungen, die jeweils als "wahr" oder "falsch" bewertet werden können. Das Ergebnis der Entscheidung wird durch die logische Kombination dieser Bedingungswerte bestimmt.

Wenn eine Entscheidung nur aus einer Bedingung besteht, erreichen Entscheidungstests und Bedingungstests die gleiche Überdeckung. Für Entscheidungen mit mehreren Bedingungen stehen ausführlichere Testverfahren zur Verfügung, die im Folgenden beschrieben werden:

- Bedingungstest (Testverfahren A in Tabelle 5): Der Tester entwirft Testfälle mit dem Ziel, die Wahr/Falsch-Ergebnisse jeder einzelnen Bedingung abzudecken. Bei einer schlechten Auswahl der Testdaten (siehe Tabelle 5) kann eine 100%ige Bedingungsüberdeckung erreicht werden, obwohl keine vollständige Entscheidungsüberdeckung erzielt wurde. Für die beiden Testfälle TF 1 und TF 2 werden die einzelnen Bedingungen B1 und B2 sowohl als "wahr" als auch als "falsch" bewertet, doch die Entscheidungsergebnisse werden in beiden Fällen als "falsch" bewertet.
- Mehrfachbedingungstest (Testverfahren B in Tabelle 5): Der Tester entwirft Testfälle mit dem Ziel, alle Wertekombinationen in Bezug auf die einzelnen Bedingungen abzudecken. Wenn jede Wertekombination getestet wird, wird auch jedes Entscheidungsergebnis getestet.
- Modifizierter Bedingungs-/Entscheidungstest (MC/DC) (Testverfahren C in Tabelle 5): Dieses Testverfahren ist vergleichbar mit dem Mehrfachbedingungstest (B). Es berücksichtigt jedoch nur Kombinationen, bei denen einzelne Bedingungen (B1 und B2) das Entscheidungsergebnis unabhängig voneinander beeinflussen. Im Fall von TF 4 ändert sich das Ergebnis der Entscheidung nicht, wenn B1 oder B2 einzeln von "falsch" auf "wahr" geändert wird (d. h., das Entscheidungsergebnis bleibt "falsch"). Eine 100%ige MC/DC-Überdeckung kann mit den Testfällen TF 1, TF 2 und TF 3 erreicht werden; TF 4 muss nicht berücksichtigt werden.

Tabelle 5 zeigt ein Beispiel für die notwendigen Testfälle für eine 100%ige Überdeckung in Abhängigkeit vom gewählten Testverfahren:

	Einzelne Bedingungen		**Entscheidungsergebnis für den Ausdruck**	**Testverfahren**		
Testfall	B1	B2	E = B1 UND B2	A	B	C
TF 1	B1=WAHR	B2=FALSCH	E=FALSCH	X	X	X
TF 2	B1=FALSCH	B2=WAHR	E=FALSCH	X	X	X
TF 3	B1=WAHR	B2=WAHR	E=WAHR		X	X
TF 4	B1=FALSCH	B2=FALSCH	E=FALSCH		X	

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

Das Beispiel zeigt die Grenzen dieser Testverfahren: Beim Bedingungstest (A) geht der Tester trotz einer Bedingungsüberdeckung von 100 % das Risiko ein, nur *ein* Entscheidungsergebnis abzudecken. Eine bessere Auswahl der Testfälle würde dies durch den Einsatz der Testfälle TF 3 und TF 4 korrigieren.

Mit Mehrfachbedingungstests (B) kann der Tester alle möglichen Eingaben und Ausgaben abdecken. Allerdings ist die Anzahl der auszuführenden Tests bei diesen Testverfahren am höchsten.

Mit modifizierten Bedingungs-/Entscheidungstests (C) kann der Tester eine vollständige Überdeckung aller Einzelbedingungen und aller Entscheidungen mit einer geringeren Anzahl von Tests im Vergleich zu Mehrfachbedingungstests erreichen.

4.2.2 Back-to-Back-Test

Back-to-Back-Test ist ein Testansatz, bei dem ein Testobjekt mit Hilfe eines Pseudo-Orakels getestet wird, um erwartete Ergebnisse zu erzeugen. Dazu führt der Tester denselben Testfall auf allen Varianten aus und vergleicht die Testergebnisse der Varianten. Sind die Testergebnisse identisch, ist der Test bestanden. Unterscheiden sich die Testergebnisse, wird die Ursache für die festgestellte Differenz analysiert.

Den Testobjekten müssen hierzu die gleichen inhaltlichen Anforderungen zugrunde liegen. Nur dann können sie ein vergleichbares Verhalten zeigen. Diese Anforderungen dienen typischerweise als Grundlage für die Ableitung der Testeingaben, nicht aber als Testorakel zur Festlegung der erwarteten Ergebnisse. Stattdessen wird bei Back-to-Back-Tests das Verhalten eines Testobjekts, mit dem eines anderen verglichen, wobei das eine als Pseudo-Orakel für das andere dient. Auf diese Weise wird erwartet, dass der Test unbeabsichtigte Unterschiede – selbst sehr geringe – zwischen den Testobjekten oder ihren Umgebungen aufdeckt.

Im einfachsten Fall handelt es sich bei den Testobjekten von Back-to-Back-Tests um verschiedene Versionen derselben Software. In diesem Fall dient eine frühere Version des Testobjekts als Pseudo-Orakel für die Back-to-Back-Tests, ähnlich wie bei einem Regressionstest. Eine Alternative ist der Vergleich eines ausführbaren Modells mit einem daraus manuell oder automatisch abgeleiteten Code. In diesem Fall handelt es sich um eine Form des modellbasierten Testens, bei dem das ausführbare Modell ebenfalls als Pseudo-Orakel dient. Dieser Testansatz ist daher sehr gut für den automatisierten Testentwurf geeignet. Hier leitet der Tester nicht nur das erwartete Ergebnis aus dem Modell ab, sondern auch automatisierte Testfälle.

4.2.3 Fehlereinfügungstest

Fehlereinfügungstest ist ein Testansatz, mit dem die Reaktion einer Komponente oder eines Systems auf ungünstige Bedingungen (z. B. einen Fehlerzustand – in diesem Zusammenhang auch als *Fehler* bezeichnet) bewertet werden soll. Programmiertechniken wie die Fehlerbehandlung dienen dem Zweck, das System auf interne und externe Fehlerzustände robust und sicher reagieren zu lassen. Um Fehlereinfügungstests durchzuführen, kann der Tester an folgenden Stellen gezielt Fehlerzustände in das System einbringen:

- Fehlerzustände in externen Komponenten: z. B. das Erkennen unplausibler Werte von Sensoren
- Fehlerzustände in Schnittstellen: z. B. Vermeidung von Schäden durch Kurzschlüsse oder verlorene Nachrichten
- Fehlerzustände in der Anwendung: Erkennung und Behandlung interner Fehlerzustände

Je nach System und Testumgebung können Fehler auf unterschiedliche Weise injiziert werden:

- Bei der klassischen Fehlereinfügung fügt der Tester einen Fehlerzustand durch Manipulation des physischen Testobjekts ein.
- Externe Fehlerzustände oder schnittstellenbezogene Fehler werden typischerweise zur Laufzeit in einer HiL-Testumgebung mit einer FIU simuliert.
- Softwarebasierte Fehlerzustände, wie z. B. Schnittstellen- oder interne Fehler, werden häufig in einer SiL-Testumgebung oder direkt in der Entwicklungsumgebung mit Hilfe von Tools wie Debuggern oder dem Universal Measurement and Calibration Protocol (XCP) simuliert.

4.2.4 Anforderungsbasierter Test

Beim anforderungsbasierten Testen analysiert der Tester typischerweise textuelle Anforderungen, um Testbedingungen aus einzelnen atomaren Anforderungen abzuleiten, er entwirft Testfälle, die die Anforderungen prüfen, und führt diese Testfälle aus. Diese Dekomposition beinhaltet die Identifizierung verschiedener testbarer Aspekte innerhalb einer Anforderung. Jede atomare Anforderung sollte ein einzelnes Verhalten oder eine Bedingung beschreiben, die unabhängig voneinander überprüft werden kann.

Durch die Zerlegung einer komplexen oder High-Level-Anforderung in solch granulare Elemente gewährleistet der Tester eine umfassende Überdeckung und vermeidet, dass versteckte oder implizite Erwartungen übersehen werden. Die Überdeckung der Anforderungen wird gemessen als das Verhältnis zwischen den atomaren Anforderungen, die von mindestens einem Testfall abgedeckt werden, und der Gesamtzahl der atomaren Anforderungen.

Testfälle für das anforderungsbasierte Testen sind in der Regel positive Testfälle, die bestätigen, dass die angegebenen Anforderungen erfüllt sind. Negative Testfälle werden oft durch ergänzende Testverfahren wie Äquivalenzklassenbildung, intuitive Testfallermittlung oder exploratives Testen abgeleitet, obwohl Anforderungen auch explizit solche negativen Bedingungen definieren können.

4.2.5 Kontextabhängige Auswahl

Mehrere Testverfahren und Testansätze sind im Kontext von Automobilsystemen von Bedeutung. Einige davon werden im CTFL®-Lehrplan [ISTQB_CTFL] vorgestellt, während andere in Abschnitt 4.2 diskutiert werden. Dazu gehören:

- Anforderungsbasierter Test
- Äquivalenzklassenbildung
- Grenzwertanalyse
- Anweisungstest
- Zweigtest
- Modifizierter Bedingungs-/Entscheidungstest (MC/DC)
- Intuitive Testfallermittlung
- Fehlereinfügungstest
- Back-to-Back-Test

Die Auswahl der geeigneten Testverfahren und Testansätze hängt von mehreren Faktoren ab:

Stand der Technik

Entspricht das Testverfahren oder der Testansatz dem aktuellen Stand der Technik für diesen Zweck? Hier geben Normen wie ISO/IEC/IEEE 29119 und ISO 26262 wertvolle Hinweise. Insbesondere die ISO/IEC/IEEE 29119-4 bietet einen strukturierten Überblick über standardisierte Testverfahren, einschließlich entsprechender Überdeckungsmaße, unterteilt in Black-Box-Testverfahren, White-Box-Testverfahren und erfahrungsbasierte Testverfahren. Die Norm ISO 26262 empfiehlt darüber hinaus anzuwendende Testverfahren und Testansätze in Abhängigkeit des ASIL, wie in Abschnitt 2.2 erläutert. Abweichungen von den Empfehlungen müssen sorgfältig erwogen und begründet werden.

Testbasis

Liefert die Testbasis geeignete Testbedingungen für das Testverfahren? Beispielsweise kann der Tester nur Äquivalenzklassen bilden, wenn die Testbasis Parameter oder Variablen enthält. Deren Werte müssen sich in Äquivalenzklassen zusammenfassen lassen. Ähnliches gilt für Grenzwerte. Sie können nur getestet werden, wenn die Werte im Wertebereich geordnet sind.

Risikobasiertes Testen

Risikobasiertes Testen bedeutet die Identifizierung von Produktrisiken und die Berücksichtigung der Risikostufe bei der Auswahl der Testverfahren und Testansätze. So ist z. B. das Testen eines Grenzwertes nur dann sinnvoll, wenn das Risiko besteht, dass Grenzwertverletzungen auftreten und die Auswirkungen solcher Verletzungen ein Risiko darstellen.

Teststufe

Kann das Testverfahren bzw. der Testansatz auf der Teststufe eingesetzt werden? White-Box-Tests eignen sich besonders, wenn der Quellcode oder die interne Struktur als Testbasis dient. Im Idealfall ist der strukturelle Überdeckungsgrad messbar. Für Black-Box-Tests muss das Testobjekt verfügbar und beobachtbar sein. Zum Beispiel kann das Testen einer Äquivalenzklasse eines Sensors beim Systemtest effizienter sein als beim Komponententest. Wenn ein Testverfahren oder ein Testansatz auf einer Teststufe nicht anwendbar ist, sollte der Tester in Übereinstimmung mit der Teststrategie eine andere Teststufe wählen.

Beispiele für die Auswahl von Testverfahren und Testansätzen

Tabelle 6 enthält eine Auflistung von Testverfahren und Testansätzen, ergänzt durch ein Beispiel für die Bewertung mehrerer der oben genannten Faktoren durch einen Anwender und die darauf basierende Wahl des Testverfahrens bzw. des Testansatzes.

	Testverfahren / Testansatz	Empfohlen für ASIL A?	Testbasis geeignet?	Risiko, wenn Fehler-zustand nicht erkannt wird?	Teststufe Systemtest sinnvoll?	Auswahl
1	Anforderungsbasierter Test	++	JA	++	JA	X
2	Äquivalenzklassen-bildung	+	JA	++	JA	X
3	Grenzwertanalyse	+	NEIN	-	JA	
4	Anweisungstest	++	JA	++	NEIN	
5	Entscheidungstest	+	JA	++	NEIN	
6	MC/DC	+	JA	+	NEIN	
7	Intuitive Testfallermittlung	+	NEIN	++	JA	
8	Testen der Fehlereinfügung	+	JA	+	NEIN	
9	Back-to-Back-Test	+	NEIN	++	JA	

Tabelle 6: Beispiel für die Wahl des Testverfahrens / bzw. Testansatzes

5 Liste der Abkürzungen

Abkürzung	Definition/Bedeutung
ACQ	Acquisition (ASPICE)
API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
ASAM	Association for Standardization of Automation and Measuring Systems
ASPICE	Automotive SPICE
AUTOSAR	Automotive Open System Architecture
AUTOSIG	Automotive Special Interest Group
BP	Base Practice (ASPICE) (Basispraktik)
BSW	Basic Software (AUTOSAR) (Basissoftware)
CAD	CANdela Diagnostic Description
CAN	Controller Area Network
CTFL	Certified Tester Foundation Level
E/E	elektrisch/elektronisch
ECU	Electronic Control Unit (Steuergerät)
EOP	End of Production (Ende der Produktion)
FIU	Fault Insertion Unit (Fehlereinfügungseinheit)
GP	Generic Practice (ASPICE) (generische Praktik)
HARA	Hazard Analysis and Risk Assessment (Gefährdungsanalyse und Risikobewertung)
HiL	Hardware-in-the-Loop
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
MAN	Management (ASPICE)

Abkürzung	Definition/Bedeutung
MCD	Measurement, Calibration and Diagnostics
MC/DC	Modifizierte Bedingungs-/Entscheidungsüberdeckung
MiL	Model-in-the-Loop
MISRA	Motor Industry Software Reliability Association
MSIL	Motorcycle Safety Integrity Level
OEM	Original Equipment Manufacturer (Erstausrüster)
PA	Process Attribute (ASPICE)
PIM	Process Improvement (ASPICE)
PoC	Point of Control
PoO	Point of Observation
PRM	Prozessreferenzmodell
QM	Qualitätsmanagement
QMS	Qualitätsmanagementsystem
REU	Reuse (ASPICE)
RTE	Run-Time Environment (AUTOSAR) (Laufzeitumgebung)
SiL	Software-in-the-Loop
SOP	Start of Production (Beginn der Produktion)
SOVD	Service-Oriented Vehicle Diagnostics
SPICE	Software Process Improvement and Capability Determination
SPL	Supplier (ASPICE)
SUP	Supporting (ASPICE)
SW-C	Software Component (AUTOSAR) (Softwarekomponente)
SWE	Software Engineering (ASPICE)
SYS	System Engineering (ASPICE)
VAL	Validation (ASPICE)

Abkürzung	Definition/Bedeutung
VDA	Verband der Automobilindustrie e.V.
XCP	Universal Measurement and Calibration Protocol
XiL	X-in-the-loop: Ein allgemeiner Begriff für alle In-the-Loop-Testumgebungen wie MiL, SiL und HiL

6 Domänenspezifische Begriffe

Begriff	Definition
Automotive Open System Architecture	Eine Entwicklungspartnerschaft, die einen offenen Industriestandard für die Softwarearchitektur in der Automobilindustrie etabliert. Hinweis: Der Begriff wird auch für die standardisierte Softwarearchitektur und den damit verbundenen System-/Softwareentwicklungsansatz verwendet.
Automotive Safety Integrity Level	Ein Safety Integrity Level für die funktionale Sicherheit im Automobil, definiert in ISO 26262.
Automotive SPICE	Ein Prozessreferenzmodell und ein zugehöriges Prozess-Assessment-Modell in der Automobilindustrie.
Basissoftware (AUTOSAR)	In AUTOSAR eine Softwareschicht, die aus standardisierten, hardwareorientierten Komponenten besteht.
Breakout-Box	Eine Einrichtung zum Analysieren, Unterbrechen oder Manipulieren von physikalischen Signalen in elektrischen Leitungen.
Bussystem	Ein Netzwerk von Steuergeräten, die über dieselbe Verbindung Informationen austauschen.
Closed-Loop-System	Ein System, in dem eine Steuerungsmaßnahme oder ein Input vom Output oder von Änderungen des Outputs abhängig ist.
Direktive (MISRA-C)	Eine Programmierrichtlinie in MISRA-C, die durch statische Analyse nicht vollständig verifizierbar ist.
Echtzeitfähiger Computer	Ein Computer, der in der Lage ist, Signale in Echtzeit zu verarbeiten.
ECU-Konfigurationsbeschreibung (AUTOSAR)	Die für die Integration von Softwarekomponenten mit einem Steuergerät erforderlichen Daten.

Begriff	Definition
ECU-Auszug (AUTOSAR)	Systemkonfigurationsdaten für ein elektronisches Steuergerät.
Erstausrüster (OEM)	In der Automobilbranche ein Automobilhersteller.
Fähigkeitsdimension	In ASPICE eine Dimension eines Prozess-Assessment-Modells, die die zu bewertenden Prozessattribute definiert.
Fähigkeitsgrad	In ASPICE eine Stufe, die einem Prozess auf der Grundlage der Bewertung der entsprechenden Prozessattribute in einem Prozess-Assessment zugewiesen wird.
Fähigkeitsindikator	In ASPICE ein Indikator für die Prozessfähigkeit, der im Prozess-Assessment verwendet wird.
Freigabe	Dokumentation der Freigabe eines Freigabeobjektes.
Freigabeobjekt	Ein identifizierbares Element mit implementierten Funktionen, Eigenschaften und Verwendungszweck.
Freigabeempfehlung	Eine auf den Testergebnissen basierende Empfehlung, ob ein Testelement freigegeben werden soll.
Freigabeprozess	Ein Prozess, der zu einer Freigabe führt.
Laufzeitumgebung (AUTOSAR)	In AUTOSAR die Abstraktionsschicht, die den Datenaustausch zwischen AUTOSAR-Softwarekomponenten sowie zwischen Anwendungs- und Basissoftware sowohl innerhalb eines Steuergeräts als auch zwischen Steuergeräten steuert und implementiert.
MISRA-C	Eine von MISRA bereitgestellte Programmierrichtlinie für die Verwendung der Programmiersprache C für kritische Systeme.
Open-Loop-System	Ein System, in dem eine Steuerungsaktion oder eine Eingabe unabhängig von der Ausgabe oder von Änderungen der Ausgabe ist.
Produktentwicklungsprozess	Ein Prozess, der alle Aktivitäten von der ersten Produktidee bis zur Produktion umfasst.

Begriff	Definition
Prozessattribut	In ASPICE eine Reihe von messbaren Merkmalen eines Prozesses für ein Prozess-Assessment.
Prozessdimension	In ASPICE eine Dimension eines Prozess-Assessment-Modells, welche die zu bewertenden Prozesse definiert.
Realzeit	Die Verarbeitung von Daten, so dass die Ergebnisse innerhalb eines vorher festgelegten Zeitraums zur Verfügung stehen.
Regel (MISRA-C)	Eine Programmierrichtlinie in MISRA-C, die durch statische Analyse vollständig verifizierbar ist.
Sicherheitslebenszyklus	Der Lebenszyklus eines sicherheitsrelevanten Systems von der Entstehung bis zur Entsorgung.
Simulationszeit	Der Zeitrahmen einer Computersimulation.
Softwarekomponentenverifizierung und Integrationsverifizierung (ASPICE)	In Automotive SPICE eine Verifizierung der integrierten Software auf der Grundlage der Softwarearchitektur.
Software-Unit-Verifizierung (ASPICE)	In Automotive SPICE eine Verifizierung der Software-Units auf Konsistenz mit ihrem detaillierten Entwurf.
Softwarekomponente (AUTOSAR)	In AUTOSAR eine Komponente in der hardwareunabhängigen Anwendungssoftwareschicht.
Softwareverifizierung (ASPICE)	In Automotive SPICE eine Verifizierung der integrierten Software auf Konsistenz mit den Anforderungen an die Software.
Steuergerät	In der Automobilindustrie ein eingebettetes System, das elektrische/elektronische (Sub-)Systeme in einem Fahrzeug steuert.
Systemkonfigurationsbeschreibung (AUTOSAR)	Die Daten, die bei der Integration aller elektronischen Steuergeräte in einem Fahrzeug verwendet werden.
Systemintegration und Integrationsverifizierung (ASPICE)	In Automotive SPICE eine Verifizierung der integrierten Systemelemente auf Konsistenz mit der Systemarchitektur.

Begriff	Definition
Systemlebenszyklus	Die Phasen der Entwicklung eines Systems bis zu seiner Außerbetriebnahme.
Systemverifizierung (ASPICE)	In Automotive SPICE eine Verifizierung des Systems auf Konsistenz mit den Systemanforderungen.

7 Referenzen

7.1 Normen

Automotive SPICE 4.0	(2023), Automotive SPICE Process Assessment / Reference Model
ISO 14229	(2020), Road vehicles – Unified diagnostic services (UDS)
ISO 26262	(2018), Road vehicles – Functional safety
ISO 26262-1	(2018), Road vehicles – Functional safety Part 1: Vocabulary
ISO 26262-2	(2018), Road vehicles – Functional safety Part 2: Management of functional safety
ISO 26262-3	(2018), Road vehicles – Functional safety Part 3: Concept phase
ISO 26263-4	(2018), Road vehicles – Functional safety Part 4: Product development at the system level
ISO 26263-5	(2018), Road vehicles – Functional safety Part 5: Product development at the hardware level
ISO 26263-6	(2018), Road vehicles – Functional safety Part 6: Product development at the software level
ISO 26263-7	(2018), Road vehicles – Functional safety Part 7: Production, operation, service and decommissioning
ISO 26263-8	(2018), Road vehicles – Functional safety Part 8: Supporting processes
ISO 26263-9	(2018), Road vehicles – Functional safety Part 9: Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses
ISO 26263-10	(2018), Road vehicles – Functional safety Part 10: Guidelines on ISO 26262
ISO/IEC 25010	(2023), Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) Product quality model
ISO/IEC 33020	(2019), Information technology – Process assessment Process measurement framework for assessment of process capability
ISO/IEC/IEEE 12207	(2017), Systems and software engineering – Software life cycle processes
ISO/IEC/IEEE 15288	(2023), Systems and software engineering – System life cycle processes
ISO/IEC/IEEE 24748-1	(2018), Systems and software engineering – Life cycle management Part 1: Guide for life cycle management

ISO/IEC/IEEE 29119-1 (2022), Software and systems engineering – Software testing Part 1: General concepts

ISO/IEC/IEEE 29119-2 (2021), Software and systems engineering – Software testing Part 2: Test processes

ISO/IEC/IEEE 29119-3 (2021), Software and systems engineering – Software testing Part 3: Test documentation

ISO/IEC/IEEE 29119-4 (2021), Software and systems engineering – Software testing Part 4: Test techniques

ISO/IEC/IEEE 29148 (2018), Systems and software engineering – Life cycle processes – Requirements engineering

MISRA C (2025), Guidelines for the use of the C language in critical systems

7.2 ISTQB®-Dokumente

[ISTQB_CTAL-TTA] ISTQB® Certified Tester Advanced Level – Technical Test Analyst V4.0, Lehrplan, 2022

[ISTQB_CTFL] ISTQB® Certified Tester Foundation Level V4.0.2, Lehrplan, 2025

7.3 Glossar-Referenzen

Referenzen für die in diesem Lehrplan verwendeten Begriffe:

IREB®-Glossar https://cpre.ireb.org/en/downloads-and-resources/glossary

ISTQB®-Glossar https://glossary.istqb.org/

8 Marken

CTFL® ist eine eingetragene Marke des German Testing Board (GTB) e. V. nur in der EU.

GTB® ist eine eingetragene Marke des German Testing Board (GTB) e. V. nur in der EU.

ISTQB® ist eine eingetragene Marke des International Software Testing Qualifications Board.

Automotive SPICE® ist eine eingetragene Marke des Verbandes der Automobilindustrie (VDA).

9 Anhang A – Lernziele/kognitive Stufen

Die spezifischen Lernziele für diesen Lehrplan sind am Anfang jedes Kapitels aufgeführt. Jedes Thema des Lehrplans wird entsprechend dem jeweiligen Lernziel geprüft.

Die Lernziele beginnen mit einem Aktionsverb, das der jeweiligen kognitiven Stufe[27] entspricht, wie unten aufgeführt.

Stufe 1: Erinnern (K1)

Die Lernenden können sich an einen Begriff oder ein Konzept erinnern, es erkennen und wiedergeben.

Aktionsverben: Erinnern, erkennen, wiedergeben

Beispiele
Sich an die Konzepte der Testpyramide erinnern.
Die typischen Ziele des Testens wiedererkennen.

Stufe 2: Verstehen (K2)

Die Lernenden können die Gründe oder Erklärungen für Aussagen zum Thema auswählen und können das zugehörige Konzept des Softwaretests zusammenfassen, vergleichen, einordnen und Beispiele dafür geben.

Aktionsverben: Klassifizieren, vergleichen, differenzieren, unterscheiden, erläutern, diskutieren, Beispiele nennen, schließen, zusammenfassen

Beispiele	**Anmerkungen**
Testwerkzeuge nach ihrem Zweck und den Testaktivitäten, die sie unterstützen, klassifizieren.	
Die verschiedenen Teststufen vergleichen.	Kann verwendet werden, um nach Gemeinsamkeiten, Unterschieden oder beidem zu suchen.
Zwischen Testen und Debugging differenzieren.	Sucht nach Unterschieden zwischen Konzepten.
Zwischen Projektrisiken und Produktrisiken unterscheiden.	Ermöglicht die separate Klassifizierung von zwei (oder mehr) Konzepten.
Den Einfluss des Kontexts auf den Testprozess erläutern.	
Beispiele nennen, warum Testen notwendig ist.	
Aus einem gegebenen Profil von Fehlerzuständen auf die Grundursache von Fehlern schließen.	

[27] Quelle zu den hier verwendeten kognitiven Stufen von Lernzielen: Anderson, L. W.; Krathwohl, D. R. (Hrsg.). A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom´s Taxonomy of Educational Objectives. Allyn & Bacon, 2001

Beispiele	Anmerkungen
Die Aktivitäten des Prozesses zur Überprüfung eines Arbeitsergebnisses zusammenfassen.	

Stufe 3: Anwenden (K3)

Die Lernenden können ein Verfahren durchführen, wenn sie mit einer vertrauten Aufgabe konfrontiert werden, oder das richtige Verfahren auswählen und es auf einen gegebenen Kontext anwenden.

Aktionsverben: Anwenden, umsetzen, vorbereiten, nutzen

Beispiele	Anmerkungen
Grenzwertanalyse anwenden, um Testfälle aus gegebenen Anforderungen abzuleiten.	Sollte sich auf ein Verfahren / eine Technik / einen Prozess etc. beziehen.
Methoden zur Erfassung von Metriken umsetzen, um technische und Managementanforderungen zu unterstützen.	
Tests zur Installierbarkeit von mobilen Anwendungen vorbereiten.	
Die Verfolgbarkeit nutzen, um den Testfortschritt auf Vollständigkeit und Konsistenz mit den Testzielen, der Teststrategie und dem Testplan zu überwachen.	Könnte in einem Lernziel verwendet werden, bei dem die Lernenden in der Lage sein sollen, eine Technik oder ein Verfahren anzuwenden. Ähnlich wie "anwenden".

10 Anhang B – Verfolgbarkeitsmatrix des geschäftlichen Nutzens (Business Outcomes) mit Lernzielen

In diesem Kapitel wird die Verfolgbarkeit zwischen geschäftlichen Nutzen und den Lernzielen des Lehrplans Certified Tester – Automotive Software Tester aufgeführt.

Geschäftlicher Nutzen: DerAutomotive Software Tester kann ...		BO1	BO2	BO3	BO4	BO5
AuT-BO1	… effektiv in einem Testteam zusammenarbeiten. ("Zusammenarbeiten")	37				
AuT-BO2	… die aus dem ISTQB® Certified Tester Foundation Level (CTFL®) bekannten Testverfahren an die spezifischen Anforderungen des Projekts anpassen. ("Anpassen")		7			
AuT-BO3	… die grundlegenden Anforderungen der relevanten Normen (z. B. Automotive SPICE® und ISO 26262) bei der Auswahl geeigneter Testverfahren berücksichtigen. ("Auswählen")			24		
AuT-BO4	… das Testteam bei der risikobasierten Planung der Testaktivitäten unterstützen und bekannte Strukturierungs- und Priorisierungselemente anwenden. ("Unterstützen & anwenden")				9	

Geschäftlicher Nutzen: DerAutomotive Software Tester kann ...			BO1	BO2	BO3	BO4	BO5
AuT-BO5	… virtuelle Testumgebungen (d. h. MiL, SiL und HiL) anwenden. ("Anwenden")						13
Eindeutige LO	Lernziel	K-Niveau					
1	Einführung						
1.1	Anforderungen aus divergierenden Projektzielen und zunehmender Produktkomplexität						
AuT-1.1	… anhand von Beispielen die Herausforderungen, die sich bei der Produktentwicklung in der Automobilindustrie aus divergierenden Projektzielen und der zunehmenden Produktkomplexität ergeben, erläutern	K2	X				
1.2	Projektaspekte, die von Normen beeinflusst werden						
AuT-1.2	… Projektaspekte, die von Normen beeinflusst werden (z. B. Zeit, Kosten, Qualität, Projektrisiken und Produktrisiken), wiedergeben	K1	X				

Geschäftlicher Nutzen: DerAutomotive Software Tester kann ...			BO1	BO2	BO3	BO4	BO5
1.3	Die sechs generischen Phasen des Systemlebenszyklus						
AuT-1.3	… die sechs generischen Phasen des Systemlebenszyklus nach ISO/IEC/IEEE 24748-1 wiedergeben	K1	X				
1.4	Der Beitrag und die Beteiligung des Testers am Freigabeprozess						
AuT-1.4	… sich an die Rolle (d. h. Beitrag und Mitarbeit) des Testers im Freigabeprozess erinnern	K1	X				
2	Normen für das Testen von elektrischen/elektronischen (E/E-)Systemen						
2.1	Automotive SPICE (ASPICE)						
AuT-2.1.1.1	… die zwei Dimensionen von ASPICE wiedergeben	K1			X		
AuT-2.1.1.3	… die Fähigkeitsstufen 0 bis 3 von ASPICE erläutern	K2			X		
AuT-2.1.2.1	… sich an den Zweck der testspezifischen Prozesse von ASPICE erinnern	K1	X		X		

Geschäftlicher Nutzen: DerAutomotive Software Tester kann ...			BO1	BO2	BO3	BO4	BO5
AuT-2.1.2.2	… die Bedeutung der vier Bewertungsstufen und der Fähigkeitsindikatoren von ASPICE aus Sicht des Testens erläutern	K2			X		
AuT-2.1.2.3	… die Anforderungen von ASPICE an eine Teststrategie einschließlich der Kriterien für die Regressionsverifizierung erläutern	K2	X		X	X	
AuT-2.1.2.4	… sich an die Anforderungen von ASPICE an Testmittel erinnern	K1	X		X	X	
AuT-2.1.2.5	… Maßnahmen zur Software-Unit-Verifizierung anwenden	K3	X		X	X	
AuT-2.1.2.6	… die Anforderungen an die Verfolgbarkeit von ASPICE aus Sicht des Testens erläutern	K2	X		X		
2.2	ISO 26262						
AuT-2.2.1.1	… das Ziel der funktionalen Sicherheit für E/E-Systeme erläutern	K2			X		
AuT-2.2.1.2	… den Beitrag des Testers zur Sicherheitskultur nennen	K1	X		X		
AuT-2.2.2	… die Rolle des Testers im Rahmen des Sicherheitslebenszyklus nach ISO 26262 diskutieren	K2	X		X		

Geschäftlicher Nutzen: DerAutomotive Software Tester kann ...			BO1	BO2	BO3	BO4	BO5
AuT-2.2.3.2	… die Teile der ISO 26262 nennen, die für den Tester relevant sind	K1	X		X		
AuT-2.2.4.1	… die Kritikalitätsstufen des ASIL wiedergeben	K1	X		X		
AuT-2.2.4.2	… den Einfluss des ASIL auf die Testverfahren und Testarten für statische und dynamische Tests und den daraus resultierenden Testumfang erläutern	K2	X	X	X	X	
AuT-2.2.5	… die Methodentabellen der ISO 26262 anwenden	K3	X	X	X	X	
2.3	AUTOSAR						
AuT-2.3.1	… die Projektziele von AUTOSAR wiedergeben	K1			X		
AuT-2.3.3	… sich an den Einfluss von AUTOSAR auf die Arbeit des Testers erinnern	K1	X	X	X		
2.4	Vergleich von ASPICE, ISO 26262 und CTFL®						
AuT-2.4.1	… die unterschiedlichen Ziele von ASPICE und ISO 26262 wiedergeben	K1			X		

Geschäftlicher Nutzen: DerAutomotive Software Tester kann ...			BO1	BO2	BO3	BO4	BO5
AuT-2.4.2	… die Unterschiede zwischen ASPICE, ISO 26262 und CTFL® in Bezug auf die Teststufen erläutern	K2	X	X	X		
3	Testen in einer virtuellen Umgebung						
3.1	Testumgebung im Allgemeinen						
AuT-3.1.1	… sich an die Motivation für eine Testumgebung in der Automobilentwicklung erinnern	K1	X				X
AuT-3.1.2	… die allgemeinen Bestandteile einer automobilspezifischen Testumgebung wiedergeben	K1	X				X
AuT-3.1.3	… die Unterschiede zwischen Closed-Loop-Systemen und Open-Loop-Systemen erläutern	K2	X				X
AuT-3.1.4	… die wesentlichen Funktionen, Datenbasen und Protokolle eines Steuergeräts wiedergeben	K1	X				X
3.2	Testen in XiL-Testumgebungen						
AuT-3.2.1.1	… den Aufbau einer MiL-Testumgebung wiedergeben	K1	X				X

Geschäftlicher Nutzen: DerAutomotive Software Tester kann ...			BO1	BO2	BO3	BO4	BO5
AuT-3.2.1.2	… die Einsatzgebiete und Randbedingungen einer MiL-Testumgebung erläutern	K2	X				X
AuT-3.2.2.1	… den Aufbau einer SiL-Testumgebung wiedergeben	K1	X				X
AuT-3.2.2.2	… die Einsatzgebiete und die Randbedingungen einer SiL-Testumgebung nennen	K1	X				X
AuT-3.2.3.1	… den Aufbau einer HiL-Testumgebung wiedergeben	K1	X				X
AuT-3.2.3.2	… die Einsatzgebiete und die Randbedingungen einer HiL-Testumgebung erläutern	K2	X				X
AuT-3.2.4.1	… die Vor- und Nachteile des Testens anhand von Kriterien für XiL-Testumgebungen zusammenfassen	K2	X			X	X
AuT-3.2.4.2	… Kriterien für die Zuordnung eines bestimmten Umfangs des Tests zu einer oder mehreren Testumgebungen anwenden	K3	X			X	X
AuT-3.2.4.3	… die XiL-Testumgebungen im V-Modell skizzieren	K1	X			X	X

Geschäftlicher Nutzen: DerAutomotive Software Tester kann ...			BO1	BO2	BO3	BO4	BO5
4	Statische und dynamische Tests						
4.1	Statischer Test						
AuT-4.1.1	… Zweck und Anforderungen der MISRA-C-Programmierrichtlinie anhand von Beispielen erläutern	K2	X	X			
AuT-4.1.2	… ein Anforderungsreview anhand der für den Tester relevanten Qualitätsmerkmale der Norm ISO/IEC/IEEE 29148 durchführen	K3	X	X			
4.2	Dynamischer Test						
AuT-4.2.1	… Testfälle entwerfen, um eine modifizierte Bedingungs-/Entscheidungsüberdeckung zu erreichen	K3	X		X		
AuT-4.2.2	… den Einsatz von Back-to-Back-Tests anhand von Beispielen erläutern	K2	X		X		
AuT-4.2.3	… Testen mit Fehlereinfügung anhand von Beispielen erläutern	K2	X		X		
AuT-4.2.4	… die Prinzipien des anforderungsbasierten Tests wiedergeben	K1	X		X		

Geschäftlicher Nutzen: DerAutomotive Software Tester kann ...			BO1	BO2	BO3	BO4	BO5
AuT-4.2.5	… kontextabhängige Kriterien für die Auswahl geeigneter und notwendiger Testverfahren und Testansätze anwenden	K3	X	X	X	X	

11 Anhang C – Versionshinweise

Der ISTQB® Automotive Software Tester Syllabus V2.1 (2025) ist ein Minor Update von V2.0.1 (2017). Die Lernziele selbst sind unverändert, aber der Inhalt einiger Lernziele wurde auf den neuesten Stand gebracht, um die aktualisierten Standards widerzuspiegeln, auf die im Lehrplan verwiesen wird. Zusätzlich wurden Änderungen zwischen dem ISTQB CTFL 3.1 und CTFL 4.0 sowie Aktualisierungen im ISTQB-Glossar berücksichtigt. Außerdem wurden Konsistenz und Verständlichkeit verbessert, wo dies erforderlich war.

Die folgenden aktualisierten Standards wurden berücksichtigt:

- ISO 24748-1:2024
- ISO 26262:2018
- Automotive SPICE 4.0
- AUTOSAR Classic Release R23-11
- MISRA-C:2025
- ISO 29148:2018

Das Benennungsschema der Lernziele wurde auf "AuT" plus Abschnittsnummer geändert.

12 Anhang D – Automotive-Datenbasen und Kommunikationsprotokolle

Schnittstellen	Datenbank	Kommunikationsprotokolle
Speicher	ASAM MCD-2 MC (auch ASAP2 oder A2L)	ASAM MCD-1 XCP (Universelles Protokoll für Messung und Kalibrierung)
		ASAM MCD-1 CCP (CAN-Kalibrierungsprotokoll)
Bus	ASAM MCD-2 NET-Norm (auch *FIBEX – Field Bus Exchange Format*)	FlexRay (ISO 17458)
		CAN (Controller Area Network nach ISO 11898-2)
	DBC (Kommunikationsdatenbank für CAN)	CAN (Steuergerätenetzwerk nach ISO 11898-2)
Diagnose	ASAM MCD-3 D (API-Spezifikation)	KWP2000 (ISO 14230)
		ISO-OBD (ISO 15031)
	ASAM SOVD (Service-orientierte Fahrzeugdiagnose)	UDS (ISO 14229)
		SOME/IP (AUTOSAR-Spezifikation)
	CDD (CANdelaStudio Diagnosebeschreibung)	

Tabelle 7: Gängige Datenbasen und Kommunikationsprotokolle aus der Automobilbranche

AUTOSAR hat ein standardisiertes XML-Format, das die Datenbasen eines kompletten Fahrzeugs integriert. Es wird ARXML-Format genannt.

13 Index