# Ein Beitrag zur Terminologie für den szenarienbasierten Testansatz automatisierter Fahrfunktionen

Markus Steimle; Gerrit Bagschik; Till Menzel;
Jan Timo Wendler; Markus Maurer

Technische Universität Braunschweig, Institut für Regelungstechnik

{steimle,bagschik,menzel,wendler,maurer}@ifr.ing.tu-bs.de

**Kurzfassung**

Für die Markteinführung automatisierter Fahrfunktionen stellen aktuell deren Absicherung und Freigabe zentrale Herausforderungen dar. Als mögliche Lösung wird der szenarienbasierte Testansatz verfolgt, z. B. in den Forschungsprojekten PEGASUS und aFAS sowie in den Projekten ENABLE-S3 und Ko-HAF. Für die Anwendung, Weiterentwicklung und Standardisierung dieses Ansatzes bildet eine einheitliche Terminologie die Basis. Diese existiert aktuell nicht. In dieser Veröffentlichung leisten die Autoren einen Beitrag zur Etablierung einer konsistenten Begrifflichkeit für eine einheitliche, deutsche Terminologie für den szenarienbasierten Testansatz automatisierter Fahrfunktionen. Indem die Autoren relevante Begriffe für einen Überblick des szenarienbasierten Testansatzes ermitteln und in Relation zueinander setzen, schlagen sie ein Grundvokabular und eine Gliederung für eine Terminologie vor. Um Widersprüche und Ungenauigkeiten innerhalb der Terminologie aufzulösen, werden die Begriffe mithilfe von UML-Diagrammen in Relation gesetzt.

## 1 Einleitung

Fahrerassistenzsysteme und automatisierte Fahrfunktionen sind zentrale Bestandteile der Forschung im Automobilbereich (Bundesministerium für Verkehr und digitale Infrastruktur, 2017). So hat die Bundesregierung bereits im September 2015 die „Strategie automatisiertes und vernetztes Fahren“ beschlossen (Bundesministerium für Verkehr und digitale Infrastruktur, 2015, S. 12). Durch diese Strategie soll Deutschland seine

Position als Leitanbieter automatisierter Fahrzeugtechnologien weiter festigen und sich als Leitmarkt etablieren. Um dieses Ziel zu forcieren, initiierte das Bundesministerium für Verkehr und digitale Infrastruktur eine ausgeprägte Förderlandschaft für deutsche Institutionen und Unternehmen. (Bundesministerium für Verkehr und digitale Infrastruktur, 2017)

Für die Markteinführung einer automatisierten Fahrfunktion stellt nicht nur die Entwicklung der Funktion eine Herausforderung dar, sondern insbesondere deren Absicherung und Freigabe (Gasser u. a., 2015, S. 12). Eine herkömmliche, streckenbasierte Absicherung ist aufgrund der zu fahrenden Kilometer aus Zeit- und Kostengründen nicht anwendbar und wurde von Wachenfeld & Winner (2015) als Freigabefalle für das autonome Fahren formuliert (Wachenfeld & Winner, 2015, S. 462 f.). Zur Freigabe automatisierter Fahrfunktionen sind folglich neue Methoden nötig. Vielversprechend erscheint der szenarienbasierte Testansatz (Schuldt, 2017, S. 210). Durch diesen können zur Überprüfung der Fahrfunktion relevante Anwendungsszenarien entlang des Entwicklungsprozesses detailliert sowie gezielt variiert und durchgeführt werden. Deshalb beschäftigen sich derzeit verschiedene Projekte mit der szenarienbasierten Absicherung automatisierter Fahrfunktionen, wie z. B. die Forschungsprojekte PEGASUS (PEGASUS Projektbüro, 2018) und aFAS (aFAS Projektbüro, 2018) sowie die Projekte ENABLE-S3 (ENABLE-S3 PROJECT COORDINATION, 2018) und Ko-HAF (Ko-HAF Projektbüro, 2018).

Zum Erreichen der Ziele aus der „Strategie automatisiertes und vernetztes Fahren“ und um vergleichbare Ergebnisse verschiedener Herangehensweisen szenarienbasierter Test- und Freigabeprozesse sicherzustellen, wird eine einheitliche, deutsche Terminologie benötigt. Nach Recherche der Autoren existiert aktuell keine einheitliche Terminologie für den szenarienbasierten Testansatz automatisierter Fahrfunktionen. Dies führt bei Abstimmungen häufig zu Missverständnissen und erhöht so den Zeit- und Kostenaufwand. Deshalb gehört zur Weiterentwicklung des Forschungsfeldes zum szenarienbasierten Testen auch die Definition einer einheitlichen Fachsprache, um die Grundlage für einen präzisen Forschungsdialog zu legen.

In dieser Veröffentlichung leisten die Autoren einen Beitrag zur Etablierung einer konsistenten Begrifflichkeit für eine einheitliche, deutsche Terminologie für den szenarienbasierten Testansatz automatisierter Fahrfunktionen. Dazu ermitteln sie relevante Begriffe für einen Überblick des szenarienbasierten Testansatzes, setzen diese in Relation zueinan-

der und schlagen dadurch ein Grundvokabular und eine Gliederung für eine Terminologie vor. Mit einer Literaturrecherche wird analysiert, welche Begriffe im szenarienbasierten Testansatz bereits definiert wurden und welche Begriffe aus dem klassischen Softwaretest auf den szenarienbasierten Testansatz übertragen oder geschärft werden können. Viele Begriffe können aus vorhandenen Standards zum klassischen Softwaretest (z. B. ISO 29119 (2013) oder IEEE 829 (2008)) übernommen werden. Teilweise unterscheiden sich diese Begriffe aber zum szenarienbasierten Testansatz automatisierter Fahrfunktionen oder sind nur in englischer Sprache definiert. So wird der Begriff Szenario nach IEEE 829 (2008, S. 51) häufig als ein Set von zugehörigen Testfällen definiert, was nicht der Bedeutung eines Szenarios in vielen szenarienbasierten Testansätzen entspricht.

Durch die Darstellung der Begriffe aus dem klassischen Softwaretest und deren definierten Relationen als UML-Diagramm, fielen zudem Widersprüche oder Ungenauigkeiten in den aktuellen Standards auf. Deshalb wird in diesem Beitrag ein besonderes Augenmerk auf diese Darstellungsweise gelegt.

## 2 Vorschlag eines konsistenten Grundvokabulars für den szenarienbasierten Testansatz automatisierter Fahrfunktionen

Der Standard ISO 26262 (2016) ist ein Leitfaden für die Entwicklung von sicherheitskritischen elektrischen / elektronischen Systemen und legt damit auch einen Rahmen für die Entwicklung automatisierter Fahrfunktionen unter dem Aspekt der funktionalen Sicherheit fest. Der in diesem Standard vorgeschlagene Prozess nach dem V-Modell eignet sich nach Meinung der Autoren als Ausgangspunkt für die Analyse und Gliederung einer Terminologie für den szenarienbasierten Testansatz automatisierter Fahrfunktionen. Nach Bagschik u. a. (2017, S. 1) kann „im Entwicklungsprozess der Norm ISO 26262 [...] in mehreren Prozessschritten eine szenarienbasierte Sichtweise zur Generierung der von der Norm geforderten Arbeitsergebnisse für die funktionale Absicherung der Systeme genutzt werden“. Deshalb wird von den Autoren vorgeschlagen, die Terminologie in die Szenarienerstellung, die Design- und Implementierungsphase (linker Ast des V-Modells) sowie den Testprozess (rechter Ast des V-Modells) zu gliedern.

In diesem Abschnitt werden auf Basis des Stands der Technik und Forschung in den verschiedenen Phasen nach Meinung der Autoren relevante Begriffe für einen Überblick des szenarienbasierten Testansat-

zes automatisierter Fahrfunktionen ermittelt. Es wird analysiert, welche Begriffe im szenarienbasierten Testansatz bereits definiert sind und welche Begriffe aus dem klassischen Softwaretest auf den szenarienbasierten Testansatz übertragen oder geschärft werden können. Daraus abgeleitet schlagen die Autoren ein Grundvokabular für eine einheitliche, deutsche Terminologie vor. Zur Kennzeichnung der jeweiligen Definitionen werden Begriffe an den entsprechenden Stellen im Text fett gedruckt. Zur übersichtlichen grafischen Darstellung der Relationen der definierten Begriffe werden diese als UML-Diagramme dargestellt.

## 2.1 Szenarienerstellung

In diesem Unterabschnitt werden bestehende Begriffsdefinitionen im Bereich der Szenarienerstellung analysiert. Daraus abgeleitet schlagen die Autoren relevante Begriffe und deren Definitionen vor, die nach Meinung der Autoren für einen Überblick des szenarienbasierten Testansatzes in dieser Phase notwendig sind.

Autoren dieses Beitrags haben bereits an der Definition verschiedener Begriffe im Zusammenhang mit einem Szenario mitgewirkt (Ulbrich u. a., 2015) und darauf aufbauend diese Definitionen um funktionale, logische und konkrete Szenarien erweitert (Bagschik u. a., 2017). Nach Bagschik u. a. (2017, S. 2) können durch die Definition von Szenarien auf unterschiedlichen Abstraktionsebenen „Szenarien von Beginn des Entwicklungsprozesses an bereits in der Konzeptphase identifiziert werden und im weiteren Verlauf detailliert und konkretisiert werden." Dadurch kann „die Beschreibung von Szenarien helfen[,] Anforderungen zu formulieren, die benötigten Hard- und Softwarekomponenten zu konzeptionieren und das funktionale Sicherheitskonzept im Testprozess zu validieren und verifizieren."

Nachfolgend werden die Begriffe zur Szenarienerstellung, die für den szenarienbasierten Testansatz relevant sind, aufgeführt und als UML-Diagramme dargestellt. Diese Begriffe werden in den folgenden Unterabschnitten in die Design- und Implementierungsphase sowie in den Testprozess eingeordnet.

In Ulbrich u. a. (2015) ist ein Überblick über verschiedene Begriffe im Zusammenhang mit einem Szenario enthalten. Abbildung 1 zeigt die Begriffe und deren Relationen als UML-Diagramm.

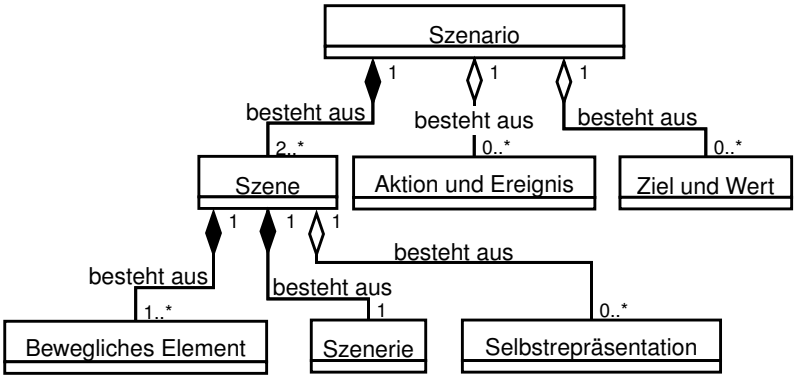


Abbildung 1: Begriffe eines Szenarios dargestellt als UML-Diagramm

Zur Beschreibung des zeitlichen Verlaufes eines **Szenarios** besteht jedes Szenario aus einer Startszene, optionalen Aktionen und Ereignissen sowie optionalen Zielen und Werten (Ulbrich u. a., 2015, S. 10). Eine **Szene** besteht aus beweglichen Elementen, der Szenerie und den Selbstrepräsentationen aller Akteure und Beobachter. Die Szene zu Beginn eines konkreten Szenarios wird als Startszene bezeichnet. (Ulbrich u. a., 2015, S. 2) **Bewegliche Elemente** bewegen sich durch kinetische Energie oder könnten sich durch zur Bewegung vorhandene Energie oder Fähigkeiten bewegen. (Ulbrich u. a., 2015, S. 3) Nach Ulbrich u. a. (2015, S. 4) beschreibt die **Szenerie** „alle räumlich stationären Aspekte einer Szene. Diese sind metrische, semantische und topologische Informationen, z. B. Fahrstreifen, Fahrstreifenmarkierungen, Fahrbahnoberflächen oder der Typ der Verkehrsdomäne. Darüber hinaus umfasst die Szenerie Informationen über Konfliktflächen zwischen Fahrstreifen und deren Verknüpfung untereinander, z. B. an Kreuzungen“. Nach Ulbrich u. a. (2015, S. 4) enthält die **Selbstrepräsentation** aller Akteure und Beobachter „das aktuelle Fertigkeiten-Level wie auch generelle Systemfähigkeiten und Fertigkeiten. Solch eine Selbstrepräsentation kann im einfachsten Fall aus Time-out-Signalen einzelner Komponenten oder Sichtbereichen und Verdeckungen bestehen“. Durch **Aktionen und Ereignisse** werden Szenen in einem Szenario verknüpft (Ulbrich u. a., 2015, S. 10). Im Gegensatz zu Aktionen und Ereignissen kann nach Ulbrich u. a. (2015, S. 10) ein Szenario „auch einzig durch eine Startszene und dem Kommando an alle Akteure, ihre individuellen

Ziele und Werte zu verfolgen, beschrieben sein, ohne dass weitere Szenen vorgegeben werden“. Durch diese **Ziele und Werte** werden Aktionen und Ereignisse von den zugeordneten Akteuren selbstständig ermittelt und ausgeführt. Ziele und Werte können transient (z. B. Mission oder Benutzereingaben) oder permanent (z. B. regulatorisch oder gesellschaftlich) sein (Ulbrich u. a., 2015, S. 8).

In Bagschik u. a. (2017) wurde die Definition eines Szenarios von Ulbrich u. a. (2015) um funktionale, logische und konkrete Szenarien erweitert. Abbildung 2 zeigt die Begriffe, die in Verbindung zu den verschiedenen Abstraktionsebenen von Szenarien stehen und deren Relationen als UML-Diagramm.

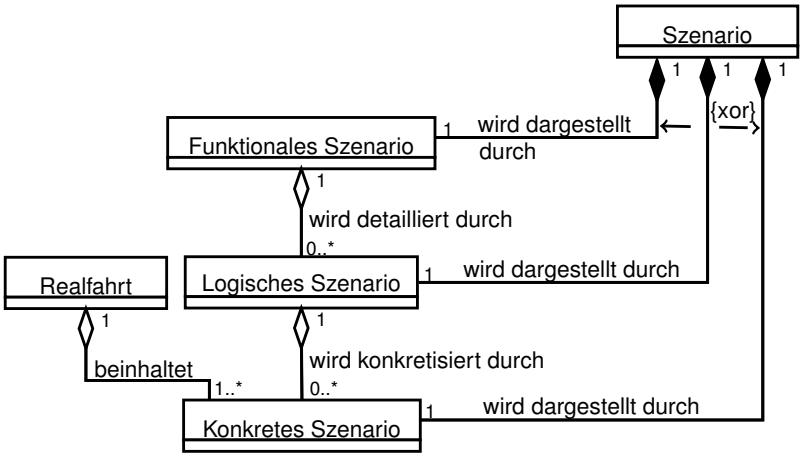


Abbildung 2: Begriffe in Verbindung zu verschiedenen Abstraktionsebenen von Szenarien dargestellt als UML-Diagramm

Ein Szenario kann je nach Abstraktionsebene als funktionales, logisches oder konkretes Szenario dargestellt werden. Nach Bagschik u. a. (2017, S. 7) stellen **funktionale Szenarien** „Betriebsszenarien des Entwicklungsgegenstands auf semantischer Ebene dar. Die Entitäten und Beziehungen zwischen den Entitäten der Anwendungsdomäne werden in sprachlich gefassten Szenarien ausgedrückt. Die Szenarien sind widerspruchsfrei. Das Vokabular der funktionalen Szenarien ist spezifisch für den Anwendungsfall und die -domäne und kann unterschiedliche Detailgrade aufweisen.“ Auf Basis der funktionalen Szenarien werden logische Szenarien abgeleitet. Sie detaillieren funktionale Szenarien im

physikalischen Zustandsraum. Nach Bagschik u. a. (2017, S. 9) stellen **logische Szenarien** „Betriebsszenarien durch Entitäten und Beziehungen dieser Entitäten mithilfe von Parameterbereichen im Zustandsraum dar. Für die einzelnen Parameterbereiche können optional statistische Verteilungen angegeben werden. Zusätzlich können optional die Beziehungen der Parameterbereiche zueinander mithilfe von Korrelationen oder numerischen Bedingungen modelliert werden. Logische Szenarien enthalten eine formale Beschreibung von Szenarien." Auf Basis der logischen Szenarien werden konkrete Szenarien abgeleitet. Nach Bagschik u. a. (2017, S. 9) stellen **konkrete Szenarien** „Betriebsszenarien eindeutig durch Entitäten und Beziehungen dieser Entitäten mithilfe von festen Werten im Zustandsraum dar." Eine **Realfahrt** beinhaltet ein oder mehrere konkrete Szenarien, die jeweils einem übergeordneten logischen Szenario zugeordnet werden können.

## 2.2 Design- und Implementierungsphase

In diesem Unterabschnitt werden bestehende Begriffsdefinitionen im Bereich der Design- und Implementierungsphase analysiert. Daraus abgeleitet schlagen die Autoren relevante Begriffe und deren Definitionen vor, die nach Meinung der Autoren für einen Überblick des szenarienbasierten Testansatzes in dieser Phase notwendig sind. Abbildung 3 zeigt die Begriffe der Design- und Implementierungsphase und deren Relationen als UML-Diagramm.

Nach dem Standard ISO 26262 (2016) erfolgt die Produktentwicklung auf vier Ebenen: Fahrzeugebene, Systemebene, Komponentenebene und Unitebene. Auf allen Ebenen wird jeweils Hardware und Software betrachtet. Viele Autoren, die den klassischen Softwaretest betrachten, beziehen sich hauptsächlich auf Software. Hardware wird nur selten betrachtet, da sie im klassischen Softwarebereich nur einen geringen Fokus besitzt. Meist werden drei Ebenen unterschieden: Systemebene, Integrationsebene (auch als Komponentenebene bezeichnet) und Unitebene (auch als Modulebene bezeichnet). Nach Hoffmann (2013, S. 167) unterscheidet sich die System- von der Integrationsebene, indem der Systemtest die zu untersuchende Software nahezu ausschließlich aus funktionaler Sicht betrachtet. Im Gegensatz dazu berücksichtigen Integrations- oder Modultests die interne Code-Struktur. Nach Hoffmann (2013, S. 159) wird die klare Trennlinie, die in der Literatur zwischen der Modul- und Integrationsebene gezogen wird, in der Praxis häufig durchbrochen.

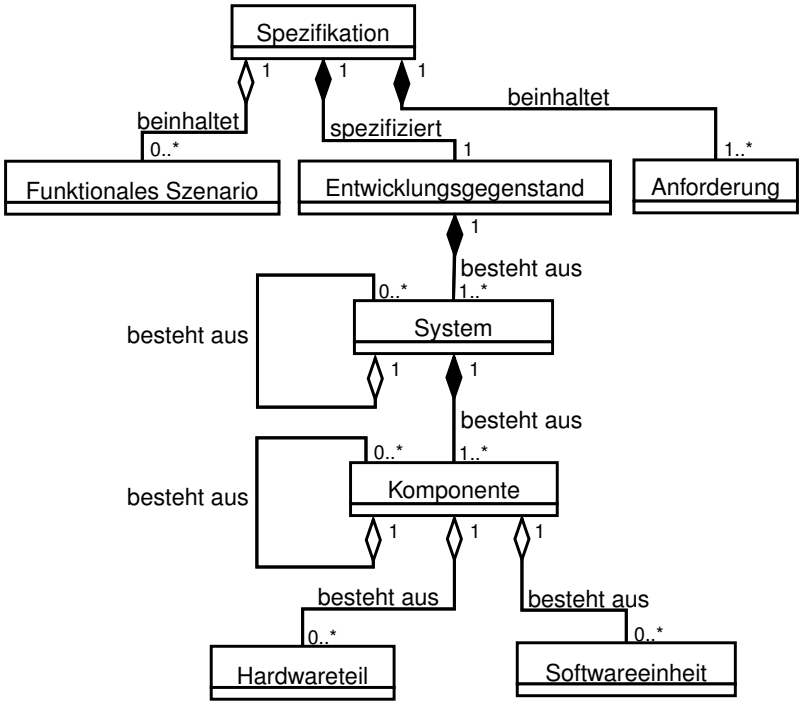


Abbildung 3: Begriffe der Design- und Implementierungsphase dargestellt als UML-Diagramm

Für die in Abbildung 3 enthaltenen Begriffe schlagen die Autoren nachfolgende Definition vor. Die Spezifikation ist Ausgangspunkt für die Design- und Implementierungsphase. Nach ISO 26262 (2016, Teil 3, Abschnitt 5) ist die Spezifikation das Dokument, das die Anforderungen an den Entwicklungsgegenstand enthält und dessen funktionales Verhalten auf Fahrzeugebene spezifiziert. Nach Bagschik u. a. (2017, S. 3) sollen in der Spezifikation „funktionale Konzepte und Systemgrenzen, die Einsatzumgebung, die rechtlichen Rahmenbedingungen und die Abhängigkeiten zu anderen Entwicklungsgegenständen beschrieben werden.“ Für die Autoren ist die **Spezifikation** das Dokument, das die Anforderungen an den Entwicklungsgegenstand und dessen funktionales Verhalten auf Fahrzeugebene spezifiziert. Zur Definition des Funktionsumfangs können (abstrahierte) funktionale Szenarien erstellt

werden und in die Spezifikation aufgenommen werden. Im weiteren Entwicklungsprozess sollen diese funktionalen Szenarien zu logischen Szenarien detailliert und zu konkreten Szenarien konkretisiert werden. Für eine genaue Aufteilung von funktionalen, logischen und konkreten Szenarien auf die verschiedenen Zeitpunkte im Entwicklungsprozess sei auf Bagschik u. a. (2017) verwiesen. Eine **Anforderung** ist eine benötigte Eigenschaft oder Fähigkeit, die ein Entwicklungsgegenstand erfüllen oder besitzen muss (International Software Testing Qualifications Board, 2015, S. 137). Sie muss durch ein Bewertungskriterium überprüfbar sein. Der **Entwicklungsgegenstand** ist ein System oder Verbund aus Systemen. Der Entwicklungsgegenstand implementiert eine Funktion oder den Teil einer Funktion auf Fahrzeugebene. (ISO 26262, 2016, Teil 1, S. 15) Ein **System** ist ein Verbund aus Komponenten und möglichen Teilsystemen, der mindestens einen Sensor, einen Controller und einen Aktor miteinander verbindet. Der zugehörige Sensor oder Aktor kann Teil des Systems oder extern dazu sein. (ISO 26262, 2016, Teil 1, S. 24) Eine **Komponente** ist ein Verbund aus mindestens einem Hardwareteil und / oder aus mindestens einer Softwareeinheit und möglichen Teilkomponenten. Sie setzt eine abgeschlossene Funktionalität um und ist logisch und / oder technisch vom System separierbar. Eine Komponente, die nur aus Hardwareteilen besteht, wird als Hardwarekomponente bezeichnet. (ISO 26262, 2016, Teil 1, S. 5) Eine Komponente, die nur aus Softwareeinheiten besteht, wird als Softwarekomponente bezeichnet (ISO 26262, 2016, Teil 1, S. 24). Ein **Hardwareteil** ist ein Teil einer Hardwarekomponente auf dem ersten Level der hierarchischen Zerlegung (ISO 26262, 2016, Teil 1, S. 13). Eine **Softwareeinheit** ist eine atomare Softwarekomponente der Softwarearchitektur, die einem eigenständigen Test unterzogen werden kann (ISO 26262, 2016, Teil 1, S. 24). Ein **Element** kann abhängig vom Kontext ein System, eine Komponente, ein Hardwareteil oder eine Softwareeinheit sein (ISO 26262, 2016, Teil 1, S. 7). Die Autoren schlagen jedoch vor, den Begriff des Elements im Rahmen einer guten Verständlichkeit nicht zu verwenden, da dieser Begriff nicht eindeutig definiert ist.

## 2.3 Testprozess

In diesem Unterabschnitt werden bestehende Begriffsdefinitionen im Bereich des Testprozesses analysiert. Daraus abgeleitet schlagen die Autoren relevante Begriffe und deren Definitionen vor, die nach Meinung der Autoren für einen Überblick des szenarienbasierten Testansatzes in dieser Phase notwendig sind.

Nach Liggesmeyer (2009, S. 46) können durch die Prüfung auf verschiedenen Ebenen Fehler einfacher erkannt und lokalisiert werden. Eine notwendige Voraussetzung dafür ist Modularisierung (Liggesmeyer, 2009, S. 46). Saust u. a. (2009, S. 4 f.) beziehen diese Voraussetzung auf autonome Fahrzeuge in städtischen Umgebungen und schreiben, dass ein modularer Aufbau des Gesamtsystems eine notwendige Voraussetzung zum Erreichen der notwendigen Testtiefe ist, da nur dann einzelne Module unabhängig voneinander entwickelt, in Betrieb genommen und getestet werden können. Daraus motivieren Saust u. a. (2009, S. 5) ein Testen auf verschiedenen Prüfebenen.

Durch den Entwicklungsprozess nach dem Standard ISO 26262 (2016) ist die Voraussetzung der Modularität gegeben und einzelne Module können entkoppelt vom Gesamtsystem auf unterschiedlichen Prüfebenen (Fahrzeug-, System-, Komponenten- und Unitebene) getestet werden. Auf jeder Prüfebene kann daher ein eigenständiger Testprozess, bestehend aus Testphasen, mit festgelegten Testobjekten, definiert werden. Nach Witte (2016, S. 2) sind die einzelnen Testphasen nicht strikt nacheinander abzuarbeiten, da sich in der Praxis eine zeitliche Überlappung nicht vermeiden lässt. Sie sollen nur einen groben Rahmen zur Orientierung geben. Dieser Rahmen eignet sich nach Meinung der Autoren für die Gliederung der Begriffe des Testprozesses. Dadurch kann der gesamte Testumfang in kleinere Einheiten aufgeteilt und getrennt analysiert werden.

In der Literatur werden unterschiedliche Bezeichnungen und eine unterschiedliche Anzahl der Testphasen verwendet. Nachfolgend werden verschiedene Unterteilungen beispielhaft vorgestellt. Das International Software Testing Qualifications Board (2015, S. 188) und Spillner & Linz (2012, S. 21) unterteilen den fundamentalen Testprozess in fünf Phasen, die allerdings nach Meinung der Autoren auch als neun Phasen gesehen werden können: Planung und Steuerung, Analyse und Design, Realisierung und Durchführung, Bewertung und Berichterstattung sowie Abschluss der Testaktivitäten. Nach dem Standard IEEE 829 (1989, S. iii) können auf Basis der zu erstellenden Artefakte drei Hauptpha-

sen definiert werden: Testkonzept, Testspezifizierung und Testauswertung. Die Testspezifizierung kann weiter in Testentwurfsspezifizierung, Testfallableitung und Testfalldurchführung, die Testauswertung in Testfallübergabe, Testergebnisaufzeichnung und Testabschluss unterteilt werden. Mit dieser feineren Unterteilung ergeben sich sieben Phasen innerhalb des Testprozesses. Horstmann (2005, S. 23 f.) definiert sieben Phasen: Testzieldefinition, Definition der Testspezifikation, Testfallermittlung, Testinstrumentierung, Testdurchführung, Testauswertung und Testdokumentation. Witte (2016, S. 3) unterteilt den grundsätzlichen Testprozess in sechs Phasen: Testplanung, Testdesign, Testspezifikation, Testdurchführung, Testprotokollierung und Testauswertung.

In allen analysierten Quellen ist eine Testplanung zu Beginn des Testprozesses erforderlich, weshalb die Testplanung von den Autoren als erste Phase vorgeschlagen wird. Hier werden unter anderem das Testkonzept und der Testplan erstellt. Im nächsten Schritt sind Testfälle zu spezifizieren. Diese Phase wird von den Autoren als Testspezifizierung bezeichnet. Im Anschluss ist den spezifizierten Testfällen die benötigte Testinfrastruktur zuzuordnen. Diese Phase wird als Testinstrumentierung bezeichnet. Während der nachfolgenden Phase der Testdurchführung werden die in der Testspezifizierung enthaltenen Testfälle entsprechend der Testinstrumentierung durchgeführt. Die Testbewertung kann simultan zur Testdurchführung, aber auch als eigenständige Aktivität nach der Testdurchführung auf Basis aufgezeichneter Daten, stattfinden, weshalb sie als eigenständige Phase betrachtet wird.

Zusammengefasst schlagen die Autoren für die Gliederung der Terminologie folgende fünf Phasen im Testprozess vor: Testplanung, Testspezifizierung, Testinstrumentierung, Testdurchführung und Testbewertung. Aufbauend auf diesen Phasen wird nachfolgend das Grundvokabular zum Testprozess vorgestellt.

### 2.3.1 Testplanung

Die Testplanung im szenarienbasierten Testansatz gleicht entsprechend der Literaturrecherche im Wesentlichen der Testplanung im klassischen Softwaretest, welche beispielsweise in Spillner & Linz (2012, S. 21 ff.) beschrieben ist. In diesem Beitrag soll speziell die Terminologie für den szenarienbasierten Testansatz betrachtet werden, weshalb die Phase der Testplanung nicht weiter betrachtet wird.

### 2.3.2 Testspezifizierung

Nach der Testplanung sind die konkreten Testfälle zu spezifizieren und in die Testspezifikation aufzunehmen. Abbildung 4 zeigt die Begriffe der Testspezifizierung und deren Relationen als UML-Diagramm. Für die darin enthaltenen Begriffe schlagen die Autoren nachfolgende Definition vor.

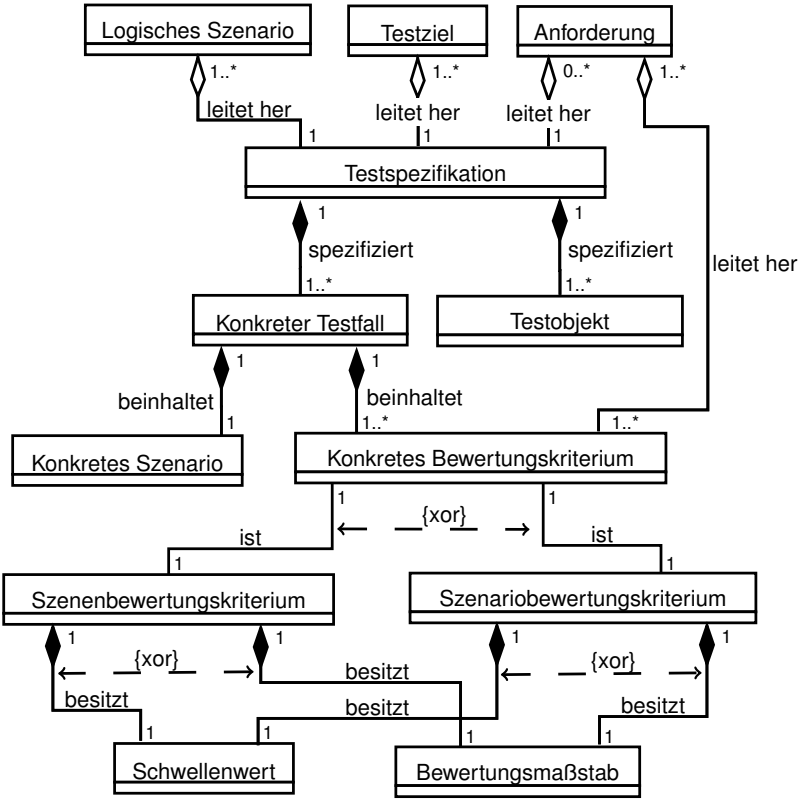


Abbildung 4: Begriffe der Testspezifizierung dargestellt als UML-Diagramm

Nach Horstmann (2005, S. 21) beginnt der Testprozess mit der Definition von Testzielen. In der Literatur gibt es verschiedene Definitionen zum Testziel - von sehr allgemein bis sehr speziell. Laut Hartmann (2001, S. 7) besteht das Testziel darin, Fehler zu finden. Weiterhin

schreibt er, dass Testziele von hierarchischer Natur sind und sich solange in weitere Teiltestziele unterteilen lassen, bis sich auf unterster Ebene einfache Anweisungen ergeben. Jedes Testziel liefert eine Aussage über die Korrektheit und / oder das Verhalten der getesteten Funktion. Ein Test ist abgeschlossen, wenn zu allen für ihn definierten Zielen eine Aussage getroffen ist. (Hartmann, 2001, S. 72 f.) Laut Witte (2016, S. 125) beschreibt das Testziel den Grund des Testens. Es definiert die Erwartungshaltung, die an den Testfall gestellt wird. Zusätzlich begründet es den gewählten Testumfang zum Nachweis der Qualität. Nach dem International Software Testing Qualifications Board (2015, S. 185) und Spillner & Linz (2012, S. 188) ist ein Testziel der Grund oder Zweck für den Entwurf und die Ausführung von Tests. Von Spillner & Linz (2012, S. 79) werden die Testziele auf die verschiedenen Prüfebenen aufgeteilt und auf den Fokus der jeweilige Prüfebene angepasst. Im Komponententest ist das Testziel die Sicherstellung, dass das jeweilige Testobjekt die geforderte Funktionalität entsprechend der Spezifikation korrekt und vollständig realisiert (Spillner & Linz, 2012, S. 48). Im Integrationstest ist das Testziel das Aufdecken von Schnittstellenfehlern (Spillner & Linz, 2012, S. 56). Im Systemtest ist das Testziel die Validierung des fertigen Systems zur Erfüllung der gestellten Anforderungen. Es sollen Fehler und Mängel aufgrund falscher, unvollständiger oder im System widersprüchlich umgesetzter Anforderungen aufgedeckt werden. Zusätzlich sollen undokumentierte oder vergessene Anforderungen entdeckt werden. (Spillner & Linz, 2012, S. 63) Nach dem Standard ISO 29119 (2013, S. 26 f.) wird ein Test durchgeführt, um ein oder mehrere Testziele zu erreichen. Diese Definitionen für den klassischen Softwaretest sind sehr ähnlich formuliert und können inhaltlich auf den szenarienbasierten Testansatz übertragen werden. Für die Autoren sind die **Testziele** der Grund oder Zweck für die Erstellung und die Durchführung von konkreten Testfällen. Testziele begründen den gewählten Testumfang zum Nachweis der Qualität und beantworten die Frage was gefolgert werden kann, wenn der oder die konkreten Testfälle erfolgreich durchgeführt wurden oder fehlgeschlagen sind. Testziele können auf die verschiedenen Prüfebenen aufgeteilt und in unterschiedlichen Detailstufen definiert werden.

Im nächsten Schritt ist die Testspezifikation zu erstellen. Die Basis dafür bilden die logischen Szenarien mit den Anforderungen und den Testzielen. Nach Witte (2016, S. 48) wird die Testspezifikation laufend erweitert, verfeinert und aktualisiert. Laut Spillner & Linz (2012, S. 95) werden beim anforderungsbasierten Testen zur Testfallableitung die An-

forderungen herangezogen. Zu jeder Anforderung wird mindestens ein konkreter Testfall abgeleitet und in der Testspezifikation dokumentiert. Die Testspezifikation wird durch ein Review verifiziert. Nach Horstmann (2005, S. 25) wird die Testspezifikation anhand des Testziels abgeleitet. Sie stellt eine Konkretisierung des Testziels dar. Sie definiert gemeinsam mit den Anforderungen eine eindeutig ableitbare Menge an Testfällen. Nach Witte (2016, S. 3) werden bei der Testspezifikation die Testfälle auf Basis der Anforderungen abgeleitet und im Detail beschrieben. Zusätzlich werden Abhängigkeiten zwischen den Testfällen ermittelt und die einzelnen Testschritte eindeutig beschrieben. In der Praxis soll während des Softwaretests die Testspezifikation laufend erweitert, verfeinert und aktualisiert werden (Witte, 2016, S. 48). Nach dem Standard ISO 29119 (2013, S. 3) ist die Testspezifikation die vollständige Dokumentation der Testfallableitung, der Testfälle und der Testausführungsschritte für ein bestimmtes Testobjekt. Diese Definitionen für den klassischen Softwaretest sind sehr ähnlich formuliert und können inhaltlich auf den szenarienbasierten Testansatz übertragen werden. Allerdings unterscheidet sich das Vorgehen, da logische Szenarien berücksichtigt werden, welche im klassischen Softwaretest in dieser Form nicht existieren. Für die Autoren ist die **Testspezifikation** ein Dokument, das die Testfallableitung, die konkreten Testfälle und die Testobjekte beschreibt. Sie wird auf Basis der logischen Szenarien, der Anforderungen und der Testziele hergeleitet. Das **Testobjekt** ist die Hardware und / oder Software, die überprüft werden soll und dadurch abhängig von der Prüfebene.

Im Vergleich zum klassischen Softwaretest sind folgende Begriffe beim szenarienbasierten Testansatz zusätzlich notwendig bzw. anzupassen. Die Autoren schlagen folgende Definitionen vor. Ein **konkreter Testfall** besteht aus einem konkreten Szenario und einem oder mehreren konkreten Bewertungskriterien. Ein **konkretes Bewertungskriterium** ist ein Szenenbewertungskriterium oder ein Szenariobewertungskriterium, durch welches eine Anforderung mithilfe von beobachtbaren Größen verifiziert werden kann. Es muss während der Testbewertung folglich möglich sein, die Bewertung auf Basis der während der Testdurchführung erzeugten Daten durchzuführen. Mit einem **Szenenbewertungskriterium** wird eine einzelne Szene durch einen Schwellenwert oder Bewertungsmaßstab bewertet. Mit einem **Szenariobewertungskriterium** wird eine Abfolge an Szenen (maximal das vollständige konkrete Szenario durch alle Szenen) durch einen Schwellenwert oder Bewertungsmaßstab bewertet. Ein **Schwellenwert** ist ein

fester Wert anhand dessen die Einhaltung eines Kriteriums getestet werden kann. Es ist folglich nur eine Aussage möglich, ob das Kriterium erfüllt ist oder nicht. Ein **Bewertungsmaßstab** ist ein Maßstab anhand dessen die Einhaltung eines Kriteriums geprüft werden kann. Es ist folglich auch eine Aussage möglich, wie gut das Kriterium erfüllt ist.

### 2.3.3 Testinstrumentierung

Nach der Testspezifizierung ist den spezifizierten Testfällen die benötigte Testinfrastruktur zuzuordnen. Abbildung 5 zeigt die Begriffe der Testinstrumentierung und deren Relationen als UML-Diagramm. Für die darin enthaltenen Begriffe schlagen die Autoren nachfolgende Definition vor.

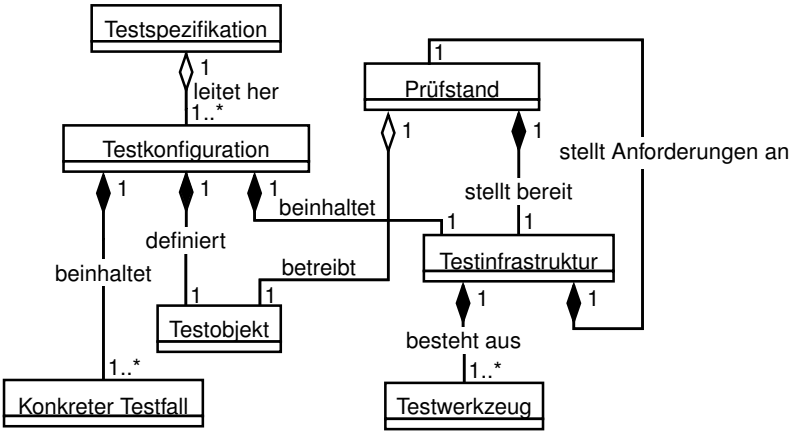


Abbildung 5: Begriffe der Testinstrumentierung dargestellt als UML-Diagramm

Ausgangspunkt für die Testinstrumentierung ist die Testspezifikation. Auf deren Basis werden Testkonfigurationen hergeleitet. Für die Autoren ist die **Testkonfiguration** eine gezielte Zusammenstellung der konkreten Testfälle für ein definiertes Testobjekt mit der zur Testdurchführung notwendigen Testinfrastruktur.

Nach dem International Software Testing Qualifications Board (2015, S. 181) beinhaltet die Testinfrastruktur die organisatorischen Elemente, die für die Durchführung des Testfalls benötigt werden, bestehend aus Testumgebung, Testwerkzeugen, Büroräumen, Verfahren usw. Nach

Spillner & Linz (2012, S. 264) beinhaltet die Testinfrastruktur alle Bestandteile, die notwendig sind, um die geplanten Testaktivitäten durchführen zu können. Mit der Testinfrastruktur kann das Programm unter Benutzung der spezifizierten Testfälle ablaufen (Spillner & Linz, 2012, S. 27). Nach Witte (2016, S. 126) dokumentiert die Testinfrastruktur das Testsystem, die Testdaten und die verwendeten Testhilfsmittel. Die Testinfrastruktur enthält somit eine Auflistung der Bestandteile der Umgebung, die notwendig sind, um die geplanten Tests durchführen zu können (z. B. Testplattformen, Arbeitsplätze für die Tester und technische Ausstattung der Arbeitsplätze) (Witte, 2016, S. 130). Weiterhin beschreibt Witte (2016, S. 126) die Testumgebung als Konfiguration des Testsystems und Festlegung der Bedingungen des Testumfelds. Diese Definitionen für den klassischen Softwaretest sind sehr ähnlich formuliert und können inhaltlich auf den szenarienbasierten Testansatz übertragen werden. Allerdings wird beim szenarienbasierten Testansatz zusätzliche Testinfrastruktur eingesetzt, z. B. Sensormodelle. Für die Autoren umfasst die **Testinfrastruktur** alle Hard- und Softwarekomponenten sowie alle Testwerkzeuge, die benötigt werden, um das Testobjekt betreiben zu können. Sie umfasst Hardware, Instrumentierung, Simulatoren, Softwarewerkzeuge, Treiber zur Generierung der Eingangsdaten für das Testobjekt (Treiber sind z. B. Sensormodelle oder eine Restbussimulation) und andere unterstützende Hilfsmittel.

Nach dem International Software Testing Qualifications Board (2015, S. 194) ist ein Testwerkzeug ein Werkzeug, das eine oder mehrere Testaktivitäten, wie Planung und Steuerung, Spezifikation, Erstellung von Testdaten, Testdurchführung oder Bewertung, unterstützt. Nach Spillner & Linz (2012, S. 209) wird durch Testwerkzeuge die Effizienz und Zuverlässigkeit der Tests verbessert, indem manuelle Tätigkeiten automatisiert werden. Weiterhin können durch sie Tests bewerkstelligt werden, die manuell nicht zu realisieren sind. Diese Definitionen für den klassischen Softwaretest sind sehr ähnlich formuliert und können inhaltlich auf den szenarienbasierten Testansatz übertragen werden. Für die Autoren unterstützt ein **Testwerkzeug** eine oder mehrere Testaktivitäten, wie Planung und Steuerung, Spezifikation, Erzeugung von Testobjekteingabewerten, Testdurchführung oder Testbewertung.

An einem **Prüfstand** können konkrete Testfälle für ein Testobjekt durchgeführt werden. Dazu stellt ein Prüfstand eine bestimmte Testinfrastruktur bereit, mit der das Testobjekt betrieben werden kann. Definierte Test- oder Erprobungsumfänge können an einem Prüfstand durchgeführt werden. Durch die zum Betrieb des Testobjekts benötigte

Testinfrastruktur werden Anforderungen an einen Prüfstand gestellt. Eine Unterscheidung des Prüfstands und der Testinfrastruktur wurde in einer Diskussion mit Bargfrede u. a. (2017) als notwendig erachtet, da in der Praxis der Prüfstand nicht für jeden konkreten Testfall neu aufgebaut wird, sondern entweder vorhanden ist oder für mehrere konkrete Testfälle aufgebaut wird. Durch die für die konkreten Testfälle spezifizierten Testinfrastrukturen, können die konkreten Testfälle auf verschiedene Prüfstände (automatisiert) verteilt werden.

### 2.3.4 Testdurchführung

Die Testdurchführung ist der Prozess zur Durchführung eines oder mehrerer konkreter Testfälle. Hier werden die in der Testspezifizierung enthaltenen Testfälle entsprechend der Testinstrumentierung an einem Prüfstand durchgeführt. Abbildung 6 zeigt die Begriffe der Testdurchführung und deren Relationen als UML-Diagramm. Für die darin enthaltenen Begriffe schlagen die Autoren nachfolgende Definition vor.

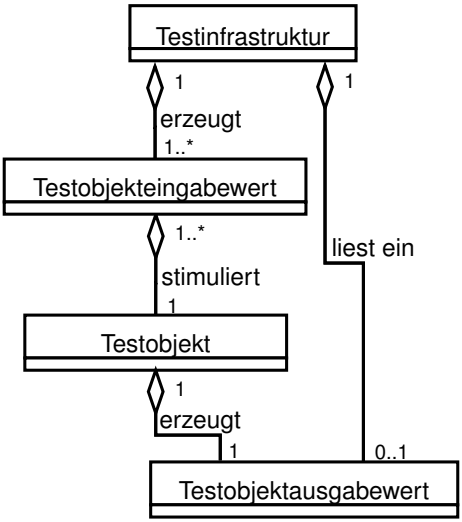


Abbildung 6: Begriffe der Testdurchführung dargestellt als UML-Diagramm

Während der Testdurchführung werden durch die Testinfrastruktur Testobjekteingabewerte für das Testobjekt erzeugt. Im Vergleich zum klassischen Softwaretest sind folgende Begriffe beim szenarienbasierten Testansatz notwendig bzw. anzupassen. Ein **Testobjekteingabewert** ist die Stimulation des Testobjekts bei einer Szene zu einem bestimmten Zeitpunkt $t_s$. Das Testobjekt verarbeitet die Testobjekteingabewerte und erzeugt auf Basis der Szene zum Zeitpunkt $t_s$ einen Testobjektausgabewert zum Zeitpunkt $t_{s+1}$. Ein **Testobjektausgabewert** ist die Szene zum Zeitpunkt $t_{s+1}$, die sich aus der Szene zum Zeitpunkt $t_s$ durch das Verhalten des Testobjekts aufgrund der Testobjekteingabewerte ergibt. Der Testobjektausgabewert kann durch die Testinfrastruktur eingelesen werden und die Testobjekteingabewerte im nächsten Zeitschritt beeinflussen (closed-loop) oder lediglich für die spätere Bewertung genutzt werden (open-loop). Die Testobjektausgabewerte können folglich zur späteren Bewertung aufgezeichnet oder simultan bewertet werden.

### 2.3.5 Testbewertung

Simultan zur Testdurchführung oder nach der Testdurchführung auf Basis aufgezeichneter Daten sind die Testobjektausgabewerte zu bewerten. Abbildung 7 zeigt die Begriffe der Testbewertung und deren Relationen als UML-Diagramm. Für die darin enthaltenen Begriffe schlagen die Autoren nachfolgende Definition vor.

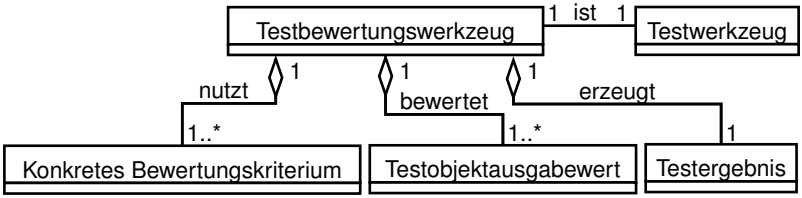


Abbildung 7: Begriffe der Testbewertung dargestellt als UML-Diagramm

Ausgangspunkt der Testbewertung sind die Testobjektausgabewerte. Im Vergleich zum klassischen Softwaretest gibt es beim szenarienbasierten Testansatz Szenen- und Szenariobewertungskriterien. Ein **Testbewertungswerkzeug** ist ein Testwerkzeug, durch das die Testobjektausgabewerte mit einem oder mehreren konkreten Bewertungskriterien bewertet werden. Ein **Testergebnis** fasst die durchgeführten

Bewertungen zu einem Gesamtergebnis zusammen. Auf Basis der Testergebnisse aller durchgeführten Testfälle kann eine Gesamtbewertung der Fahrfunktion durchgeführt werden, die für dessen Freigabe genutzt werden kann. Dadurch kann zum Beispiel ein statistischer Sicherheitsnachweis geführt werden.

## 3 Zusammenfassung und Ausblick

In dieser Veröffentlichung haben die Autoren einen Beitrag zur Etablierung einer konsistenten Begrifflichkeit für eine einheitliche, deutsche Terminologie für den szenarienbasierten Testansatz automatisierter Fahrfunktionen geleistet, indem sie ein Grundvokabular und eine Gliederung der Terminologie vorgeschlagen haben. Dazu wurden für die Autoren relevante Begriffe für den szenarienbasierten Testansatz konsolidiert und ergänzt. Diese Begriffe stellen ein Grundvokabular dar und sollen zukünftig erweitert werden. Zur Wahrung der Übersichtlichkeit und zur Gliederung der Terminologie, wurde die Terminologie entsprechend eines Entwicklungsprozesses nach dem V-Modell in die Szenarienerstellung, die Design- und Implementierungsphase sowie den Testprozess und darin enthaltene Testphasen gegliedert. Die Begriffe jeder Phase wurden grafisch jeweils als UML-Diagramm semiformal dargestellt. Diese Diagramme können zu einer Gesamtdarstellung zusammengefasst werden. Durch diese Darstellung können Zusammenhänge schnell erkannt und neue Begriffe bezüglich der bereits vorhandenen Terminologie in einen Kontext gesetzt und auf diese Weise der Terminologie hinzugefügt werden. Die Terminologie soll zukünftig weiter diskutiert und eingesetzt und somit auf Anwendbarkeit und Erweiterbarkeit überprüft werden. Hier ist zu überprüfen, ob die Darstellung als UML-Diagramm auch noch möglich ist, wenn der Terminologie weitere Begriffe hinzugefügt werden und ob diese Darstellung und Unterteilung noch sinnvoll ist.

## Literatur


aFAS Projektbüro (2018). *Forschungsprojekt aFAS*. URL: https://www.afas-online.de (abgerufen am 08. 01. 2018).

Bagschik, G., Menzel, T., Reschka, A. & Maurer, M. (2017). „Szenarien für Entwicklung, Absicherung und Test von automatisierten Fahrzeugen“. In: *11. Workshop Fahrerassistenzsysteme*. Hrsg. von Uni-DAS e. V., S. 125–135.

Bargfrede, P., Berger, S., Bussler, A. & Schuldt, F. (2017). *Persönliche Diskussionsrunde zu Testbegriffen*.

Bundesministerium für Verkehr und digitale Infrastruktur (2015). *Strategie automatisiertes und vernetztes Fahren: Leitanbieter bleiben, Leitmarkt werden, Regelbetrieb einleiten*. URL: https://www.bmvi.de/SharedDocs/DE/Publikationen/DG/broschuere-strategie-automatisiertes-vernetztes-fahren.pdf?__blob=publicationFile (abgerufen am 06. 09. 2017).

Bundesministerium für Verkehr und digitale Infrastruktur (2017). *BMVI vergibt dreizehn Förderbescheide für neue Forschungsprojekte*. URL: http://www.bmvi.de/SharedDocs/DE/Pressemitteilungen/2017/102-dobrindt-22-millionen-euro-fuer-forschungsprojekte-zum-automatisierten-fahren.html?nn=13326 (abgerufen am 07. 09. 2017).

ENABLE-S3 PROJECT COORDINATION (2018). *Project ENABLE-S3*. URL: https://www.enable-s3.eu/ (abgerufen am 08. 01. 2018).

Gasser, T. M., Schmidt, E. A., Bengler, K., Chiellino, U., Diederichs, F., Eckstein, L., Flemisch, F., Fraedrich, E., Fuchs, E., Gustke, M., Hoyer, R., Hüttinger, M., Jipp, M., Köster, F., Kühn, M., Lenz, B., Lotz-Keens, C., Maurer, M., Meurer, M., Meuresch, S., Müller, N., Reitter, C., Reschka, A., Riegelhuth, G., Ritter, J., Siedersberger, K.-H., Stankowitz, W., Trimpop, R. & Zeeb, E. (2015). *Bericht zum Forschungsbedarf - Runder Tisch Automatisiertes Fahren - AG Forschung*. URL: http://www.bmvi.de/SharedDocs/DE/Anlage/Digitales/bericht-zum-forschungsbedarf-runder-tisch-automatisiertes-fahren.pdf?__blob=publicationFile (abgerufen am 27. 09. 2017).

Ko-HAF Projektbüro (2018). *Projekt Ko-HAF*. URL: http://www.ko-haf.de/startseite/ (abgerufen am 08. 01. 2018).

Hartmann, N. (2001). „Automation des Tests eingebetteter Systeme am Beispiel der Kraftfahrzeugelektronik“. Dissertation. Karlsruhe: Karlsruher Institut für Technologie.

Hoffmann, D. W. (2013). *Software-Qualität*. 2., aktualisierte und korrigierte Auflage. eXamen.press. Berlin Heidelberg: Springer Berlin Heidelberg.

Horstmann, M. (2005). „Verflechtung von Test und Entwurf für eine verlässliche Entwicklung eingebetteter Systeme im Automobilbereich“. Dissertation. Braunschweig: Technische Universität Braunschweig.

IEEE 829 (1989). *IEEE 829: Standard for Software and System Test Documentation*. Standard IEEE 829:1989. Institute of Electrical and Electronics Engineers.

IEEE 829 (2008). *IEEE 829: Standard for Software and System Test Documentation*. Standard IEEE 829:2008. Institute of Electrical and Electronics Engineers.

International Software Testing Qualifications Board (2015). *Standard Glossary of Terms used in Software Testing*. URL: http://glossary.istqb.org (abgerufen am 15. 05. 2017).

ISO 26262 (2016). *ISO 26262: Road vehicles - Functional safety*. Standard ISO 26262:2016. International Organization for Standardization.

ISO 29119 (2013). *ISO 29119: Software and systems engineering - software testing*. Standard ISO 29119:2013. International Organization for Standardization.

Liggesmeyer, P. (2009). *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag.

PEGASUS Projektbüro (2018). *Forschungsprojekt PEGASUS*. URL: http://www.pegasusprojekt.de (abgerufen am 08. 01. 2018).

Saust, F., Müller, T., Wille, J. M. & Maurer, M. (2009). „Entwicklungsbegleitendes Simulations- und Testkonzept für autonome Fahrzeuge in städtischen Umgebungen“. In: *AAET - Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel*. Hrsg. von ITS Niedersachsen, S. 87–107.

Schuldt, F. (2017). „Ein Beitrag für den methodischen Test von automatisierten Fahrfunktionen mit Hilfe von virtuellen Umgebungen“. Dissertation. Braunschweig: Technische Universität Braunschweig.

Spillner, A. & Linz, T. (2012). *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester - Foundation Level nach ISTQB-Standard*. 5. überarbeitete und aktualisierte Auflage. ISQL-Reihe. dpunkt.verlag.

Ulbrich, S., Menzel, T., Reschka, A., Schuldt, F. & Maurer, M. (2015). „Definition der Begriffe Szene, Situation und Szenario für das automatisierte Fahren“. In: *10. Workshop Fahrerassistenzsysteme*. Hrsg. von Uni-DAS e. V., S. 105–117.

Wachenfeld, W. & Winner, H. (2015). „Die Freigabe des autonomen Fahrens“. In: *Autonomes Fahren: Technische, rechtliche und gesellschaftliche Aspekte*. Hrsg. von Maurer, M., Gerdes, J. C., Lenz, B. & Winner, H. Springer Open. Berlin: Springer Vieweg, S. 439–464.

Witte, F. (2016). *Testmanagement und Softwaretest: Theoretische Grundlagen und praktische Umsetzung*. Wiesbaden: Springer Fachmedien Wiesbaden.