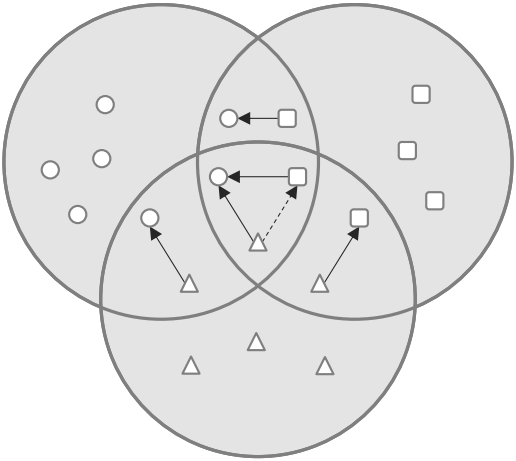
Automatische Verlinkung von Testfällen und Anforderungen

Eine wiederverwendungsorientierte Methode

Thomas Noack



Automatische Verlinkung von Testfällen und Anforderungen

Eine wiederverwendungsorientierte Methode

vorgelegt von
Dipl.-Inf. Thomas Noack
geb. in Bad Muskau

Von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
— Dr.-Ing. —

genehmigte Dissertation

Promotionsausschuss
Prof. Dr. Axel Küpper (Vorsitzender)
Prof. Dr.-Ing. Dr. h.c. Stefan Jähnichen (Gutachter)
Prof. Dr.-Ing. Andreas Spillner (Gutachter)
Prof. Dr.-Ing. Steffen Helke (Gutachter)

Tag der wissenschaftlichen Aussprache:
23.09.2015

Berlin 2015
D 83

Danksagung

Ich möchte allen danken, den Kollegen vom Fachgebiet, den Kollegen vom DCAITI und den Kollegen der Daimler AG. Das vorherrschende freundliche und offene Umfeld trug wesentlich zu der persönlichen Entwicklung bei, die mir als wissenschaftlicher Mitarbeiter zuteil wurde. Besonderer Dank gilt Prof. Stefan Jähnichen. Seine Leitung erlaubt es, Ideen zu entfalten. Ich danke Prof. Andreas Spillner, der während seiner Besuche in Berlin die Zeit fand, mich anregend zu beraten. Großer Dank gebührt Prof. Steffen Helke. In grundlegenden Diskussionen trug er dazu bei, den Kern dieser Arbeit zu schärfen.

Besonderer Dank gilt zudem Dr. Thomas Karbe, der diese Arbeit immer wieder sehr detailliert hinterfragte und dadurch wesentlich zu ihrem Erfolg beitrug. Viele Gespräche mit Dr. Benjamin Wilmes, Quang Minh Tran, Kerstin Hartig, Jonas Winkler und Martin Beckmann führten zu interessanten Erkenntnissen, die nicht immer nur softwaretechnischer Natur waren.

Diese Arbeit entstand in intensiver Zusammenarbeit mit Daimler. In vielen lehrreichen Besuchen in Böblingen und in Sindelfingen konnte ich meine Arbeit im Umfeld der Abteilung von Michael Weber deutlich in Richtung Anwendbarkeit vorantreiben. Für diese Möglichkeit danke ich ihm. Ich danke Dr. Jutta Schneider dafür, dass sie die Ausrichtung meiner Arbeit entscheidend prägte. Für die Starthilfe und fortwährende Unterstützung danke ich Christian Scheidler, Dr. Jacques Kamga, Angela Matz und Daniel Hopp.

Gabi Ambach, Rodger Burmeister und Alexandra Mehlhase lockerten Promotionstiefs auf, die Doktoranden von Zeit zu Zeit anheften. Danke dafür.

Kurzfassung

Beziehungen zwischen Artefakten spielen im Rahmen von Entwicklungsprozessen speziell im Automobilbereich eine große Rolle. Sie werden hergestellt, indem zwei Artefakte miteinander verlinkt werden. Diese Arbeit widmet sich der Verlinkung von Anforderungen und Testfällen nach der Anforderungswiederverwendung.

In der Automobilbranche werden jährlich neue Fahrzeuge bzw. Modelle auf den Markt gebracht. Die neuen Modelle werden nicht von Grund auf neu entwickelt, sondern auf der Grundlage der alten Modelle weiterentwickelt. Dabei werden Entwicklungsartefakte, insbesondere auch Anforderungen und Testfälle wiederverwendet. Da Testfälle mit den Anforderungen der früheren Baureihe verlinkt sind, stellt sich die Frage, wie sie auch systematisch mit den Anforderungen der neuen Baureihe verlinkt werden können. Dieses Problem wird im Kontext dieser Arbeit als WvT-Problem (WvT: Wiederverwendet-Testet) bezeichnet.

In der vorliegenden Arbeit wird die WvT-Verlinkung vorgestellt. Sie löst das WvT-Problem und führt die Verfolgbarkeit von Wiederverwendung (engl.: reuse-based traceability) als neue Kategorie im Bereich der Verfolgbarkeit von Anforderungen (engl.: requirements traceability) ein.

Um die Verlässlichkeit der WvT-Verlinkung zu belegen, wurden Feldstudien mit realen Spezifikationsdokumenten durchgeführt. Dazu wurde der Hauptbeitrag dieser Arbeit in einem Werkzeug umgesetzt. Diese Umsetzung wurde in der Praxis erprobt und eingeführt, wodurch die Verlinkung von Testfällen mit wiederverwendeten Anforderungen erfolgreich automatisiert wurde. Mit der Berücksichtigung der Testplanung und von Verlinkungsregeln wurde die WvT-Verlinkung verfeinert und daher noch effizienter.

Abstract

Relationships between artefacts play a vital role in development processes in the automotive domain. Those relationships are established by trace-linking two artefacts. This work addresses the linking between requirements and test cases after requirements reuse.

In the automotive domain new vehicles are released yearly. Those new vehicles are normally not developed from scratch. Quite the opposite, reuse is daily practice. By reusing requirements, reuse also takes place in the beginning of a new vehicle series project. Because test cases are trace-linked to the requirements of the previous vehicle series, one question arises: How can those test cases be trace-linked to the new requirements systematically? In the context of this thesis, this problem is introduced as the RT-problem (RT: Reuse-Test).

This work proposes the RT-linking technique which solves the RT-problem. It introduces the Reuse-based Test Traceability as a new category in the field of Requirements Traceability.

Field studies with real specification documents have been conducted in order to verify the reliability of the RT-linking. The implementation of the main technique developed in this work has been transfered successfully to the industry. As a further contribution, this work provides insights into how test planning and expert knowledge can be taken into account to improve the efficiency of the automatic linking.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ASIL	Automotive Safety Integrity Level
BR	Baureihe
CBR	Case-Based Reasoning
DOORS	Dynamic Object Oriented Requirements Management System
DXL	DOORS eXtensions Language
KE	Klassifizierende Eigenschaft
SLH	Systemlastenheft
STS	Systemtestspezifikation
T	Testet
TK	Testkonzept
Wv	Wiederverwendet
WvT	Wiederverwendet-Testet

1. Motivation und Aufbau der Arbeit

Automobilbauer streben nach Nachvollziehbarkeit. Wenn ein Softwarefehler in einem Fahrzeug auftritt, soll er buchstäblich auf den Quelltext zeigen, der ihn verursacht. Nachvollziehbarkeit wird durch die Verfolgung von Links (engl.: trace links) ermöglicht, die zwischen Entwicklungsartefakten existieren. In hierarchischen Entwicklungsprozessen entstehen besonders früh Links zwischen Systemanforderungen und Testfällen. Diese Links sind Teil eines Beziehungsgeflechts: Ein Fehler wird durch einen Testfall aufgedeckt. Der Testfall ist mit einer Anforderung verlinkt, die wiederum mit dem Quelltext in Verbindung steht. Jede Art von Verfolgung (engl.: to trace) erfordert die Existenz der Links zwischen den beteiligten Artefakten. Die vorliegende Arbeit beschäftigt sich mit einem Ausschnitt dieses Beziehungsgeflechts - der Verlinkung von Testfällen und Anforderungen. Sie ordnet sich in das Forschungsgebiet der Verfolgbarkeit von Anforderungen (engl.: requirements traceability) ein.

Wiederverwendung. Während der Entwicklung einer neuen Baureihe werden die Entwicklungsartefakte ihres Vorgängers wiederverwendet. Die Wiederverwendung ist verfolgbar, wenn die Artefakte, die in einem Wiederverwendungsverhältnis stehen, miteinander verlinkt sind. Herausforderungen ergeben sich, wenn diese Artefakte mit weiteren Artefakten verlinkt sind und diese Verlinkung nach der Wiederverwendung weiterhin bestehen soll. Diese Herausforderungen gelten zu Beginn eines jeden Baureihenprojekts auch für Anforderungen und Testfälle. Wenn Anforderungen vor ihrer Wiederverwendung mit Testfällen verlinkt waren, muss auch nach ihrer Wiederverwendung gewährleistet sein, dass die richtigen Testfälle mit den richtigen Anforderungen verlinkt sind. Dabei erschweren die neuen, geänderten oder weggefallenen Anforderungen die Verlinkung. Mit Hilfe der in dieser Arbeit beschriebenen WvT-Verlinkung (WvT: Wiederverwendet-Testet) können Testfälle mit Anforderungen verlinkt werden: WENN eine Ziel-Anforderung eine Quell-Anforderung wiederverwendet (Wv) UND WENN ein Testfall diese Quell-Anforderung testet (T), DANN kann der Testfall auch die Ziel-Anforderung testen.

Testplanung. Die ersten Schritte auf dem Weg zu einer neuen Baureihe sind neben ihrer Spezifikation auch durch die Planung ihrer Absicherung gekennzeichnet. Diese als Testplanung bezeichnete Tätigkeit resultiert in einem Testkonzept, in dem definiert ist, für welche Testobjekte (Was?) welche Testziele (Wozu?) in welcher Teststufe (Wann?) abzusichern sind. Im Testkonzept ist somit definiert, welche Testfälle existieren müssen, um die korrekte Umsetzung der Anforderungen einer Baureihe ausreichend abzusichern. Nachvollziehbarkeit wird erreicht, indem das Testkonzept mit den Testfällen und Anforderungen in Beziehung gesetzt wird. Damit sichergestellt ist, dass die richtigen Testfälle mit den richtigen Anforderungen verlinkt sind, muss die WvT-Verlinkung hinsichtlich der Forderungen aus dem Testkonzept erweitert werden: WENN eine Ziel-Anforderung eine Quell-Anforderung wiederverwendet UND WENN ein Testfall mit der Quell-Anforderung verlinkt ist, DANN wird der Testfall GEMÄSS TESTKONZEPT mit der Ziel-Anforderung verlinkt.

Erfahrungsdatenbank. Wiederverwendung bedeutet häufig, dass sich die Ziel-Anforderungen im Vergleich zu den wiederverwendeten Quell-Anforderungen ändern. Diese Änderungen erschweren die Verlinkung von Testfällen mit den Ziel-Anforderungen. Die Erfahrungen der Entwickler spielen hierbei eine wichtige Rolle. Sie erkennen oft intuitiv, nach welchen Änderungen die Verlinkung eines Testfalls hinterfragt werden muss. Um diese Erfahrungen zu berücksichtigen, muss die WvT-Verlinkung erweitert werden: WENN eine Ziel-Anforderung mit einer Quell-Anforderung verlinkt ist UND WENN ein Testfall mit der Quell-Anforderung verlinkt ist, DANN wird der Testfall GEMÄSS ERFAHRUNGS- BZW. FALLDATENBANK mit der Ziel-Anforderung verlinkt.

Verlinkung und Filterung. In dieser Arbeit wird das Zusammenspiel von Wiederverwendung, Testplanung und Erfahrungen aus früheren Projekten betrachtet. Jedes der drei Themen wird in einem eigenen Kapitel behandelt. Kapitel 4 stellt die WvT-Verlinkung und die WvT-Inkonsistenzen vor, um Testfälle mit Ziel-Anforderungen zu verlinken und die Verlinkung zu bewerten. In den aufbauenden Kapiteln 5 und 6 werden Techniken vorgestellt, die die WvT-Verlinkung um Filtermechanismen erweitern. Im Rahmen dieser Arbeit wird das Zusammenspiel der Techniken als 3-schichtige Methode bezeichnet.

Aufbau der Arbeit

Diesem einleitenden Motivationskapitel schließen sich zwei Grundlagenkapitel an. Das erste Grundlagenkapitel 2 bietet eine beispielorientierte Einführung in die *Anforderungs-* und *Testentwicklung* sowie in die *Anforderungs-* und *Testverfolgbarkeit*. Im zweiten Grundlagenkapitel 3 wird das industrielle Umfeld anhand der Spezifikationsdokumente *Systemlastenheft* und *Testspezifikation* beschrieben. Diese Dokumente enthalten *Systemanforderungen* und *Testfälle*. Es wird beschrieben, wie die Beziehungen zwischen den Anforderungen und Testfällen wegen der Anforderungswiederverwendung zum *WvT-Problem* führen. Zudem wird das *Testkonzept* vorgestellt, in dem im Rahmen der Testplanung definiert wird, welche Testfälle mit welchen Anforderungen verlinkt sein müssen, um ein System ausreichend abzusichern. Das zweite Grundlagenkapitel 3 schließt mit dem Aufbau der 3-schichtigen Methode zur Lösung des WvT-Problems und den Zielen dieser Arbeit.

In Kapitel 4 wird die *WvT-Verlinkung* vorgestellt. Sie bildet die Basisschicht der 3-schichtigen Methode, um Anforderungen und Testfälle nach der Anforderungswiederverwendung zu verlinken. Anschließend erfolgt die Beschreibung des *WvT-Diagramms* und der *WvT-Inkonsistenzen*. Mit Hilfe des Diagramms wird die Gesamtverlinkungssituation in den Spezifikationsdokumenten vor bzw. nach der WvT-Verlinkung visualisiert. Die Inkonsistenzen beschreiben suspekte Verlinkungssituationen, in denen sich WvT-Probleme vor bzw. nach ihrer Lösung befinden können. Mit Hilfe des Diagramms und der Inkonsistenzen werden Feldstudien durchgeführt. Dabei wird gezeigt, dass die WvT-Verlinkung zu besseren Ergebnissen als die manuelle Verlinkung führt.

In Kapitel 5 wird die zweite Schicht der 3-schichtigen Methode vorgestellt. Sie erweitert die WvT-Verlinkung um die *testkonzeptgesteuerte Filterung*. Dazu wird der Testfall des WvT-Problems um *klassifizierende Eigenschaften* erweitert. Mit Hilfe dieser Eigenschaften wird die Filtertechnik definiert. Diese erlaubt die Bewertung, ob Anforderungen hinsichtlich des Testkonzepts *vollständig* und *minimal* abgesichert sind. Das Kapitel schließt mit einem Exkurs in die rechtlichen Grundlagen der ISO 26262.

In Kapitel 6 wird die dritte Schicht der 3-schichtigen Methode vorgestellt. Diese Methodenschicht erweitert die WvT-Verlinkung um die *fallbasierte Filterung*. Dazu wird der Begriff der *klassifizierenden Eigenschaft* aus Kapitel 5 wiederaufgegriffen, um neben dem Testfall auch die beiden Anforderungen des WvT-Problems zu klassifizieren. Dies führt zum *vollständig klassifizierten WvT-Problem*. Die fallbasierte Filterung orientiert sich an den Mechanismen des fallbasierten Schließens (engl.: case-based reasoning, CBR). Aus diesem Grund wird der *vollständig klassifizierte WvT-Fall* eingeführt. Mit Hilfe eines *Prüfhinweises* werden im WvT-Fall die Erfahrungen gespeichert, die bei der Lösung eines früher aufgetretenen WvT-Problems gesammelt wurden. Mit Hilfe von *Ähnlichkeitsfunktionen* wird ermittelt, ob sich die klassifizierenden Eigenschaften eines WvT-Falls und eines WvT-Problems ähneln. Ist dies gegeben, unterstützt der Prüfhinweis des WvT-Falls bei der Lösung des aktuell zu lösenden WvT-Problems. In Kapitel 6 werden beispielhaft einige WvT-Fälle vorgestellt. Zudem wird begründet, warum CBR im betrachteten Umfeld momentan noch nicht vollständig umsetzbar ist.

Im letzten Kapitel 7 werden die drei Techniken zur Verlinkung und Filterung von Testfällen und Anforderungen zusammengefasst. Außerdem werden Möglichkeiten für weiterführende Arbeiten aufgezeigt.

Um die eigentliche Arbeit kurz zu halten, wurden Bildschirmabbilder des Anforderungsmanagementwerkzeugs DOORS sowie Aspekte der Kapitel 4 und 6 ausgelagert. In Anhang A werden Bildschirmabbilder zur Umsetzung der WvT-Verlinkung gezeigt. Der Auto-Linker befindet sich aktuell in Version 1.1. Er wurde bereits zur Verlinkung realer Systemlastenhefte in DOORS eingesetzt. Anhang B enthält auch Bildschirmabbilder. Diese zeigen, wie die fallbasierte Filterung aus Kapitel 6 in DOORS umgesetzt werden kann. Anhang C enthält eine zweite Feldstudie, die die Ergebnisse der Feldstudie aus Kapitel 4 bestätigt. In Anhang D wird eine Beispielrechnung durchgeführt, um die Ähnlichkeitsfunktionen aus Kapitel 6 zu veranschaulichen.

2. Grundlagen

Die Spezifikation von Anforderungen und Testfällen ist eine komplizierte Tätigkeit, die im Projekt nicht einfach nebenbei erledigt werden kann. Um dies zu zeigen, werden in diesem Kapitel Anforderungen und Testfälle mit Hilfe des Scheibenwischers eines Fahrzeugs nachträglich spezifiziert. Anhand der Beispiele werden Begriffe und Herausforderungen der Anforderungs- und Testentwicklung (engl.: requirements and test engineering) vorgestellt. Zudem erfolgt eine Einführung in die Thematik der Verfolgbarkeit (engl.: traceability).

2.1. Anforderungsentwicklung

Der zentrale Artefakttyp der Anforderungsentwicklung ist die Anforderung. Anforderungen dienen der Beschreibung von Systemen. Mit ihrer Hilfe wird definiert, über welche Eigenschaften ein System verfügen muss. Anforderungen dienen gleichzeitig der Kontrolle der Systemumsetzung.

2.1.1. Begriffe

Anforderung. In dem Glossary of Software Engineering Terminology [IEEE610.12, S.62] wird der Begriff *Anforderung* definiert als:

1. Eine Bedingung oder Fähigkeit, die von einer Person oder einem System zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.
2. Eine Bedingung oder Fähigkeit, die ein System oder Teilsystem erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen.
3. Eine dokumentierte Repräsentation einer Bedingung oder Eigenschaft gemäß (1) oder (2).

Chris Rupp liefert die obige deutsche Übersetzung in ihrem Standardwerk Requirements Engineering - Ein Überblick [Rup12, S.4].

Arten von Anforderungsartefakten. Die *Systemvision* ist das Wurzelartefakt jeder Systemspezifikation. Sie beschreibt die Grundidee und den Zweck des Systems [Poh08, S.37]. Verschiedene Anforderungsarten dienen der sukzessiven Verfeinerung der Systemvision - von natürlichsprachlichen Anforderungen bis hin zu (semi)formalen Modellen. Klaus Pohl unterscheidet zwischen drei Arten von Anforderungen [Poh08, S.48]:

- *Lösungsneutrale Anforderungen bzw. Systemziele* „abstrahieren sowohl von der Nutzung als auch von [...] der Systemrealisierung“ und liegen in einer definierten Textstruktur in natürlicher Sprache vor.
- *Szenarien* beschreiben das System fachlich durch konkrete Beispiele aus der Perspektive einer Rolle. Sie werden mittels (semi)formaler Modelle wie Anwendungsfällen oder Sequenzdiagrammen dargestellt.
- *Lösungsorientierte Anforderungen* definieren die technischen Rahmenbedingungen, denen eine Systemrealisierung unterliegt. Wegen ihrer technischen Natur werden sie beispielsweise durch konkrete Datenmodelle, Entitätsmodelle oder Algorithmen dargestellt.

Die natürlichsprachlichen Systemanforderungen entsprechen im Kontext dieser Arbeit den lösungsneutralen Anforderungen aus der Standardliteratur. Unabhängig von der Anforderungsart identifiziert Manfred Broy die Verfeinerung von Anforderungen mit der gebotenen Lösungsneutralität als eine der großen Herausforderungen in der Automobilindustrie [Bro06, S.36f].

Systemanforderung und Systemlastenheft. Im Bereich des Automobilbaus beschreiben Systemanforderungen Systeme als Verbund aus Steuergeräten, Sensoren, Aktuatoren und Software sowie die Wechselwirkung zwischen diesen Komponenten [SZ10, S.141ff]. Damit enthalten Systemlastenhefte im Gegensatz zu Softwarelastenheften Anforderungen an das Gesamtsystem. Ein Softwarelastenheft dient der Verfeinerung des Systemlastenhefts. Unabhängig vom Fokus gilt, dass ein „Lastenheft eine Definition der Systemvision enthält, eine Beschreibung der wesentlichen Systemziele (Funktionen und Qualitäten) und wichtige Kontextaspekte [...] sowie ihre Beziehungen zur Vision und den definierten Systemzielen benennt“ [Poh08, S.232].

2.1.2. Anforderungsentwicklungsprozess

Systemanforderungen unterliegen einem systematischen Entwicklungsprozess, der in dem Standardwerk Requirements Engineering - Grundlagen, Prinzipien, Techniken wie folgt definiert ist [Poh08, S.43]: „Das Requirements Engineering ist ein kooperativer, iterativer, inkrementeller Prozess, dessen Ziel es ist zu gewährleisten, dass

- alle relevanten Anforderungen bekannt und in dem erforderlichen Detaillierungsgrad verstanden sind (Gewinnung),
- die involvierten Stakeholder eine ausreichende Übereinstimmung über die bekannten Anforderungen erzielen (Übereinstimmung),
- alle Anforderungen konform zu den Dokumentationsvorschriften dokumentiert bzw. konform zu den Spezifikationsvorschriften spezifiziert sind (Dokumentation)."

Der Spezifikationsprozess besteht in dem oben zitierten Standardwerk aus den drei Kernaktivitäten *Gewinnung*, *Übereinstimmung* und *Dokumentation* sowie den beiden Querschnittsaktivitäten *Validierung* und *Management*.

Gewinnung. Das Ziel der Gewinnungsaktivität ist das inhaltliche Verständnis von Anforderungen [Poh08, S.323ff]. Dazu werden zunächst die Anforderungsquellen identifiziert. Anforderungsquellen können Personen, Altsysteme, normative Dokumente oder auch frühere Anforderungsdokumente sein. Das inhaltliche Verständnis wird je nach Quelle mit Hilfe von verschiedenen Anforderungsgewinnungstechniken vorangetrieben. Beispiele für solche Techniken sind strukturierte Gespräche, Fragebögen, Workshops und spezielle Lesetechniken zur Ermittlung relevanter Abschnitte in langen Texten.

Übereinstimmung. Das Ziel der Übereinstimmungsaktivität ist eine abgestimmte Sichtweise der verschiedenen Interessenvertreter zu den Anforderungen [Poh08, S.393ff]. In diesem Zusammenhang werden zunächst Konflikte aufgedeckt. Dies können beispielsweise sich widersprechende Anforderungen oder unterschiedliche Auffassungen zur Priorisierung sein. Techniken zum Konfliktmanagement dienen der Auflösung solcher Konflikte [Moo03].

Dokumentation. In der Dokumentationsaktivität werden die gewonnenen Anforderungen gemäß den Vorschriften dokumentiert oder spezifiziert [Poh08, S.215ff]. In Dokumentationsvorschriften wird festgelegt, welche Schablonen und Vorlagen verwendet werden müssen, um Anforderungen zu dokumentieren. Beispiele für Anforderungsschablonen liefern Chris Rupp und die SOPHISTen [Rup13]. Zusätzlich wird in den Dokumentationsvorschriften definiert, welche Dokumentationsformen zu verwenden sind. Beispiele hierfür sind natürlichsprachliche Anforderungen oder (semi)formale Modelle.

Spezifikationsvorschriften detaillieren die Dokumentationsvorschriften. Durch sie wird festgelegt, ob eine Normsprache verwendet werden muss und welche Qualitätsmerkmale für die spezifizierten Anforderungen gelten. Gängige Beispiele für Qualitätsmerkmale sind Vollständigkeit, Widerspruchsfreiheit, Eindeutigkeit, Aktualität oder Atomarität [Poh08, S.222f].

Validierung. Die Validierung dient als Querschnittsaktivität der Überwachung der drei Kernaktivitäten *Gewinnung*, *Übereinstimmung* und *Dokumentation* [Poh08, S.415ff]. Der Begriff *Validierung* wird in der Anforderungsentwicklung im Sinne des „Nachweises der Zweckmäßigkeit von Anforderungen“ verwendet [Poh08, S.422]. Das wesentliche Ziel ist es dabei, ein Anforderungsartefakt für die weitere Entwicklung freizugeben. Dazu werden die Ergebnisse aller drei Kernaktiväten mit Hilfe von Qualitätstoren (engl.: quality gates) überprüft:

- Sind die Anforderungen inhaltlich ausreichend durchdrungen?
- Sind alle Beteiligten einverstanden mit den Anforderungen?
- Wurden Dokumentations- und Spezifikationsvorschriften eingehalten?

Die Validierung von Spezifikationsvorschriften mit Hilfe von messbaren Qualitätsmerkmalen ist ein beliebtes Forschungsthema. Fadi Chabarek beschreibt in seiner Doktorarbeit Interaktive Vervollständigung von Szenariospezifikationen eine Methode zur Bewertung der Vollständigkeit von Anforderungen mit Hilfe von Szenariospezifikationen [Cha11]. Nadya Stoyanova beschreibt in ihrer Dissertation Verbesserung der Qualität natürlichsprachlicher Spezifikationen eine linguistische Methode zur Analyse der Qualität von natürlichsprachlichen Systemanforderung [Sto13].

Management. Auch das Management von Anforderungen ist eine Querschnittsaktivität, welche die drei Kernaktivitäten stützt [Poh08, S.491ff]. Klaus Pohl identifiziert hier drei zentrale Managementaktivitäten:

- Anforderungen müssen priorisiert werden. Eine in der Automobilbranche besonders wichtige Priorisierungskategorie ist die funktionale Sicherheit und die damit einhergehende ASIL-Einstufung. Moderne Anforderungsentwicklungstechniken zur Unterstützung automobiler Entwicklungsprozesse müssen die funktionale Sicherheit berücksichtigen.
- Die Änderung von Anforderungen muss verwaltet werden. Dies betrifft sowohl die Änderung während eines Projekts als auch die Änderung nach ihrer Wiederverwendung im Folgeprojekt. Die Änderung von Anforderungen nach ihrer Wiederverwendung wird für die Problemstellung dieser Arbeit noch eine zentrale Rolle spielen.
- Die Verfolgbarkeit von Anforderungen (engl.: requirements traceability) ist ein Teilgebiet des Anforderungsmanagements. In diesem Zusammenhang werden Beziehungen zwischen Artefakten explizit mit Hilfe von Links (engl.: trace links) verwaltet. Die Verfolgbarkeit der Wiederverwendung von Systemanforderungen stellt in Kombination mit der Verfolgbarkeit von Systemtests (engl.: test traceability) den Kern der Problemstellung dieser Arbeit dar.

Weil Wiederverwendung und Verfolgbarkeit den Kern dieser Arbeit ausmachen, ordnet sie sich in den Bereich des Anforderungsmanagements ein.

2.1.3. Beispiel und Herausforderungen

In Gesprächen mit Anforderungsspezifikateuren und auch in der Literatur werden einige Herausforderungen im Rahmen der angewandten Systemspezifikation immer wieder herausgestellt. Drei besonders häufig erwähnte Herausforderungen werden in den nächsten Abschnitten anhand eines Beispiels vorgestellt: Die Einführung einer Funktionshierarchie als Schicht zwischen Systemen und Anforderungen, die Unterscheidung zwischen funktionalen und nichtfunktionalen Anforderungen sowie die Lösungsneutralität.

Das folgende Beispiel soll vermitteln, in welcher Struktur und Abstraktion natürlichsprachliche Systemanforderungen spezifiziert werden. Es dient zugleich der Erläuterung der zuvor genannten Herausforderungen. In diesem Zusammenhang enthält es einige verbesserungswürdige Aspekte, auf die in den nachfolgenden Abschnitten eingegangen wird.

```
System
Scheibenwischen

Systemzweck
Anf-1: Der Scheibenwischer ermöglicht eine sichere Fahrt

Systemeigenschaften
Anf-2: Der Scheibenwischer funktioniert von -30°C bis +60°C



Funktion: Tippwischen (Front)
Bedingungen:
   Anf-3:  Klemme 15 aktiv
   Anf-4:  Während des Motorstarts ist das Scheibenwischen
           grundsätzlich gesperrt
   Anf-5:  Während der Funktion Intervallwischen ist das
           Tippwischen gesperrt

Auslösung:
   Anf-6:  Das Tippwischen wird durch Antippen des
           Lenkstockschalters ausgelöst

Beschreibung:
   Anf-7:  Das Tippwischen vollzieht den Wischzyklus einmal
   Anf-8:  Ein Wischzyklus ist eine vollständige Vor- und
           Zurückbewegung des Wischarms
   Anf-9:  Durch Tippen (tief) wird zusätzlich zum Wischen
           die Funktion Frontscheibenwaschen ausgelöst
   Anf-10: Das Frontscheibenwaschen kann auch während des
           Intervallwischens ausgelöst werden
   Anf-11: Ein Wischzyklus wird immer zu Ende geführt
```

Die obigen Anforderungen wurden nachträglich aus einem Mercedes-Benz C 180 Elegance der Baureihe W202 aus dem Baujahr 1997 ermittelt. Dazu wurde die Funktion *Tippwischen* mittels Lenkstockschalter ausgeführt. Der Lenkstockschalter ist der Hebel, der in der verwendeten C-Klasse links vom Lenkrad zur Ansteuerung von Licht- und Scheibenwischerfunktionen dient. Anhand der Beobachtung des Verhaltens wurden dann die Ist-Anforderungen im Sinne des *Reverse Requirements Engineering* spezifiziert [FC07].

Funktionsorientierung. Wie in der allgemeinen Softwaretechnik ist der Verfeinerungsgedanke auch in der automobilen Softwaretechnik verbreitet. Ein Fahrzeug besteht aus vielen Systemen, welche wiederum aus Komponenten bestehen. Beispielsweise besteht das System *Scheibenwischen* unter anderem aus einem Steuergerät inklusive Software, Wischerarmen, Wischermotor, Wasserdüsen, Wassertank, Wasserpumpe und Lenkstockschalter. Jede dieser Komponenten wird im Systemlastenheft benannt und durch Systemanforderungen beschrieben. Die Verfeinerung findet in Komponentenlastenheften statt.

Die Vielzahl an Systemanforderungen muss systematisch katalogisiert werden. In diesem Zusammenhang hat sich in der Automobilbranche die Funktionsorientierung etabliert [SZ10, S.143]. Moderne Fahrzeuge haben mehr als 2000 Funktionen [Bro06, S.36]. Das Beispiel zeigt die *Fahrzeugfunktion Tippwischen*, die durch *Systemanforderungen* verfeinert wird. Die reale Funktion aktueller Baureihen besteht aus Anforderungen im dreistelligen Bereich.

Funktionsschablonen. Im Beispiel werden *Bedingungen*, die *Auslösung* und die *Beschreibung* der Funktion *Tippwischen* spezifiziert. Im Sinne der einheitlichen Vorgehensweise definieren Schablonen den Aufbau natürlichsprachlicher Spezifikationsdokumente [Poh08, S.267f]. Funktionsschablonen dienen der systematischen und strukturierten Spezifikation von Funktionen.

Funktionale und nichtfunktionale Anforderung. „Eine *funktionale Anforderung* bezieht sich auf die funktionellen Aspekte eines Systems“ [Par10, S.27]. Neben dieser spielt aber auch die *nichtfunktionale Anforderung* eine wichtige Rolle. Einige Autoren greifen zur Beschreibung des Begriffs *nichtfunktionale Anforderung* auf den Begriff *Qualitätsanforderung* zurück [Poh08, S.16]. Unabhängig vom verwendeten Begriff lautet die Definition sinngemäß: Eine *nichtfunktionale Anforderung* definiert eine qualitative Eigenschaft eines Systems [Par10, S.27] [Som12, S.119]. Im Beispiel ist die Anforderung *Anf-2: Funktioniert von -30° bis +60°* nichtfunktional, weil sie von Umwelteinflüssen abhängt [Rup13, S.24]. Häufig entsprechen nichtfunktionale Anforderungen den *-keiten (engl.: -bilities)* aus Qualitätsmodellen [SS97, S.158] wie der [ISO9126-1]. Im Beispiel sind alle Anforderungen ab *Anf-3* funktional.

Eindeutigkeit. Klaus Pohl schreibt, dass eine „Anforderung eindeutig [ist], wenn die Anforderung so formuliert ist, dass sie nur eine gültige Interpretation zulässt“ [Poh08, S.222]. Im Beispiel ist der Systemzweck *Anf-1: Der Scheibenwischer ermöglicht eine sichere Fahrt* nicht eindeutig formuliert, da er die Interpretation des weiten Begriffs *Sicherheit* zulässt. Im Vergleich dazu wäre der Zweck *Anf-1: Der Scheibenwischer befreit die Scheiben von Niederschlägen und Schmutz und sorgt somit für eine freie Sicht* wesentlich eindeutiger.

Lösungsneutralität. Die Lösungsneutralität betrifft das Wechselspiel zwischen *Problem* und *Lösung* [Poh08, S.20]. Gemäß Manfred Broy ist die „Forderung nach Lösungsneutralität nur von oben nach unten sinnvoll“ [BW13, S.66]. Er bezieht sich hierbei auf das V-Modell. Während Systemanforderungen über eine sehr niedrige Lösungsdetaillierung verfügen, besitzen verfeinernde Komponentenanforderungen und tiefere (semi)formale Anforderungen immer detaillierter werdende technische Details. Im Beispiel ist die Systemanforderung *Anf-3: Klemme 15 aktiv* nicht lösungsneutral, weil sie sich auf ein technisches Detail bezieht. Die Wiederverwendbarkeit wird gehemmt, weil bei der Änderung von Lösungsdetails auch die Systemanforderungen geändert werden müssen [BW13, S.67]. Angenommen, *Klemme 15* würde durch *Klemme 16* ersetzt. Dies hätte Auswirkung auf eine Vielzahl an Funktionen. Im Gegensatz dazu wäre die Anforderung *Anf-3: Zündung aktiv* lösungsneutral.

2.1.4. Fokus: Wiederverwendung von Systemanforderungen

Die vorangegangenen Abschnitte boten einen Überblick über grundlegende Vorgehensweisen und aktuelle Herausforderungen im Bereich der Anforderungsentwicklung. In diesem Zusammenhang wurden relevante Aspekte aus Standardwerken und aktuellen Positionspapieren zusammengefasst. Im weiteren Verlauf dieser Arbeit werden Techniken für die Verlinkung und Filterung von Systemtestfällen mit Systemanforderungen nach ihrer Wiederverwendung entwickelt. Weil sich Wiederverwendung und Verlinkung in den Bereich des *Anforderungsmanagements* eingliedern, liegt auch der Fokus dieser Arbeit im Bereich des *Anforderungsmanagements* .

2.2. Testentwicklung

Der zentrale Artefakttyp der Testentwicklung (engl.: test engineering) ist der Testfall. Mit der Hilfe von Testfällen soll aufgedeckt werden, ob ein System über geforderte Eigenschaften verfügt. Damit steht die Testentwicklung in direktem Zusammenhang mit der Anforderungsentwicklung.

2.2.1. Begriffe

Softwaretest. Andreas Spillner und Tilo Linz definieren in ihrem Standardwerk Basiswissen Softwaretest den Begriff *Softwaretest* wie folgt: „Unter dem Testen von Software wird jede Ausführung eines Testobjekts verstanden, die der Überprüfung des Testobjekts dient“. Ferner definieren sie, dass „ein Vergleich von Soll- und Ist-Verhalten des Testobjekts zur Bestimmung [dient], ob das Testobjekt die geforderten Eigenschaften erfüllt“ [SL05, S.9].

Testobjekt. Ein Testobjekt ist ein(e) „Komponente, integriertes Teilsystem oder System, das einem Test unterzogen wird“ [SL05, S.265]. Auf Seite 11 wurde beschrieben, dass sich in der Automobilindustrie im Bereich der Anforderungsentwicklung die *Funktionsorientierung* etabliert hat. Die *Fahrzeugfunktion* ist im Kontext dieser Arbeit das wesentliche Testobjekt.

Softwarefehler. Im Zusammenhang mit dem Soll-/ Ist-Verhalten kann „eine Situation nur als fehlerhaft eingestuft werden, wenn vorab festgelegt wurde, wie die erwartete, korrekte, also nicht fehlerhafte Situation aussehen soll“. Unter dieser Annahme ist „ein Fehler somit die Nichterfüllung einer festgelegten Anforderung“ [SL05, S.7]. Durch diese Aussage wird der Zusammenhang zwischen Anforderungs- und Testentwicklung deutlich: Anforderungen beschreiben ein System und Testfälle sichern dessen korrekte Umsetzung ab.

Testfall. Spillner und Linz definieren, dass zu einem „Testfall die [...] Voraussetzungen zur Ausführung, die Eingabewerte und [...] das erwartete Verhalten des Testobjekts gehören“ [SL05, S.10]. Während die Voraussetzungen den Zustand definieren, in dem sich das Testobjekt befinden muss, legt das erwartete Verhalten fest, wie sich das Testobjekt durch die Eingaben verhalten soll.

Arten von Testfällen. Die klassifizierenden Eigenschaften *Testziel*, *Teststufe* und *Testplattform* werden als Testplanungsdimensionen bezeichnet und ab Seite 20 genauer vorgestellt. An dieser Stelle soll die Aussage genügen, dass die klassifizierenden Eigenschaften häufig auch dazu verwendet werden, die Art des Testfalls zu benennen [SL05, S.11]. Neben den *Testzielen*, *Teststufen* und *Testplattformen* existieren viele weitere Eigenschaften:

- Testfälle können nach Testzielen benannt werden, beispielsweise Funktionstest, Benutzbarkeitstest oder Performanztest.
- Testmethoden können auch Namensgeber für Testfälle sein. Beispiele dafür sind erfahrungsbasierter Test oder anforderungsbasierter Test.
- Testfälle können aber auch nach den Testobjekten benannt werden, zum Beispiel GUI-Test, Datenbanktest oder Netzwerktest.
- Eine weitere Möglichkeit, Testfälle zu benennen, sind die Testplattformen. Beispiele dafür sind der HiL-Test oder der Gesamtfahrzeugtest.
- Schließlich gibt es noch die Möglichkeit, Testfälle nach ihrer Teststufe zu benennen. Modultest (engl.: unit test), Komponententest und auch der in dieser Arbeit so wichtige Systemtest sind bekannte Beispiele dafür.

Testarten hängen voneinander ab. Ein Modultest ist beispielsweise kein Gesamtfahrzeugtest, weil die Teststufe und die Testplattform inkompatibel sind.

Systemtestfall. Der Verfeinerungsgedanke spielt im Softwareentstehungsprozess eine zentrale Rolle. Im V-Modell wird festgelegt, dass lösungsneutrale Systemanforderungen über immer lösungsorientiertere Komponentenanforderungen und Modelle verfeinert werden. In modernen Entwicklungsprozessen wird der Quelltext beispielsweise sogar aus den lösungsorientierten Matlab Simulink [ML/SL] Modellen generiert. Weil Testfälle auf Testobjekten ausgeführt werden, entspricht ihr Detailgrad dem Detailgrad der Testobjekte. Anforderungsseitig stehen Systemanforderungen im Fokus dieser Arbeit. Dies hat zur Folge, dass auch testseitig Systemtestfälle im Zentrum dieser Arbeit stehen. Auf der Systemebene des V-Modells ist ein Systemtestfall als natürlichsprachliches Prüfäquivalent einer Systemanforderung aufzufassen.

2.2.2. Fundamentaler Testprozess

Analog zu Anforderungen sind auch Testfälle das Produkt eines Testentwicklungsprozesses. Dieser gliedert sich an Entwicklungsmodelle wie beispielsweise dem V-Modell an. Andreas Spillner und Tilo Linz definieren den fundamentalen Testprozess wie folgt: „[Die] Entwicklungsaufgabe *Testen* [wird] in kleine Arbeitsschritte gegliedert, nämlich in Testplanung und -steuerung, Testanalyse und -design, Testrealisierung und -durchführung, Testauswertung und Bericht sowie Abschluss der Testaktivitäten“ [SL05, S.20ff].

Testplanung und Teststeuerung. In der Testplanung werden die Umfänge des Testens festgelegt, die während der Teststeuerung verwaltet werden. In der Testplanung wird definiert, auf welcher Testplattform (Wo?) während welcher Teststufe (Wann?) welche Testobjekte (Was?) abzusichern sind und welche Testziele (Wozu?) dabei verfolgt werden. Die *Frageworte* in den Klammern werden in der Standardliteratur häufig verwendet, um in die Testplanung einzuführen. Im Rahmen dieser Arbeit werden die *Wo?*, *Wann?*, *Was?*, *Wozu?* als *Testplanungsdimensionen* bezeichnet. Die Teststufe dient beispielsweise der zeitlichen Orientierung im übergeordneten Vorgehensmodell. Im V-Modell legt die Teststufe die Beziehung vom rechten zum linken Ast fest. Als Beispiel dafür sei die Beziehung zwischen Modultests (engl.: unit test) und der Programmierung in der unteren Ebene des V-Modells genannt. Das Ergebnis der Testplanung ist das Testkonzept. Die Teststeuerung dient der korrekten Umsetzung des Testkonzepts. Dies schließt die Verwaltung des Prozesses, die Überwachung des Fortschritts, die Feststellung von Abweichungen und die Anfertigung von Berichten ein.

Testanalyse und Testdesign. In der Testanalyse wird die Testbasis untersucht. Die Testbasis enthält alle relevanten Informationen, die für die Erstellung von Testfällen benötigt werden. Zudem wird die Testbarkeit der Testobjekte untersucht. Im Rahmen des Testdesigns werden die Testfälle erstellt. Dazu werden die Testfallermittlungsverfahren angewendet, die im Testkonzept festgelegt wurden. Im Testdesign wird die Verfolgbarkeit sichergestellt, indem Testfälle mit den relevanten Testobjekten verlinkt werden.

Testrealisierung und Testdurchführung. Im Testdesign können zunächst logische Testfälle erstellt werden, die noch keine konkreten Eingabewerte enthalten. Logische Testfälle werden während der Testrealisierung konkretisiert. Die Testrealisierung schließt die Testautomatisierung ein, indem Testskripte erstellt werden, die den Ablauf von Testfällen automatisieren. In das Aufgabengebiet der Testrealisierung fällt auch das Aufsetzen der Testumgebungen. In Testumgebungen werden Testobjekte mit Hilfe der Testfälle bzw. Testskripte ausgeführt. Die Testdurchführung wird mitsamt ihren Ergebnissen im Sinne der Verfolgbarkeit kontinuierlich protokolliert.

Testauswertung und Bericht. Im Bereich des Softwaretests ist das Testende (Test-Ende) von besonderer Wichtigkeit - insbesondere, wenn entschieden werden muss, ob es erreicht ist. Das Testende ist erreicht, sobald genügend Testfälle durchgeführt wurden, um die im Testkonzept festgelegten Testendekriterien zu erfüllen. Beispiele für Testendekriterien sind die Testabdeckung pro Testobjekt, die Testabdeckung gemessen an den Testplanungsdimensionen oder die mittlere Anzahl von aufgedeckten Fehlern pro Teststunde. Wegen der Wichtigkeit der Thematik schlagen Andreas Spillner und Tilo Linz mit der Testauswertung dafür eine eigene Aktivität im fundamentalen Testprozess vor. Diese Auswertung enthält neben der Ermittlung des aktuellen Status hinsichtlich des Testendes auch die Berichterstattung bei Abweichungen vom Testende und beim Erreichen des Testendes.

Abschluss der Testaktivitäten. Die letzte Phase des fundamentalen Testprozesses dient der kontinuierlichen Verbesserung. Dazu wird das abgeschlossene Testprojekt in Hinsicht auf Leistungsparameter untersucht, um Gründe für gute oder schlechte Leistungen aufzudecken. Beispiele für Leistungsparameter sind die fristgerechte Erreichung von Meilensteinen oder die Einhaltung des geplanten Testaufwands. Das Ziel der Abschlussaktivität ist die Aufbereitung der gewonnenen Erkenntnisse (engl.: lessons learned). Die Abschlussaktivität schließt außerdem die Aufbereitung der beteiligten Artefakte ein. Dies betrifft zum einen die Aufbereitung der Testumgebungen und Testfälle für den Regressionstest des Systems. Zum anderen findet hier auch die Aufbereitung der Artefakte zur Wiederverwendung in zukünftigen Projekten statt.

2.2.3. Beispiel und Herausforderungen

Ab Seite 10 wurde auf einige Herausforderungen eingegangen, die im Rahmen der Anforderungsentwicklung auftreten. Mit Hilfe eines Beispiels wurden die *Funktionsorientierung*, die *Funktionalität und Nichtfunktionalität* sowie die *Lösungsneutralität* erläutert. Weil die Anforderungs- und Testentwicklung voneinander abhängen, wird nachfolgend untersucht, wie sich die Herausforderungen der Anforderungsentwicklung auf die Testentwicklung auswirken.

Die folgenden zwei Testfälle werden in der Teststufe *Systemtest* auf der Testplattform *Gesamtfahrzeug* ausgeführt. Sie sichern die korrekte Umsetzung des Testobjekts *Tippwischen* aus dem Beispiel von Seite 10 ab.

```
Testobjekt:    Scheibenwischen: Tippwischen
Testziel:      Robustheit
Teststufe:     Systemtest
Testplattform: Gesamtfahrzeug

Vorbedingung: 1. Klemme 15 aktiv
              2. Fahrzeug steht
Eingabe     : 1. Tippwischen 10x schnell hintereinander aktivieren (Anf-6)
Erwartetes  : 1. Tippwischen wird nicht 10x ausgeführt
Ergebnis      2. Mit letzter Aktivierung startet letzter Wischzyklus
```

```
Testobjekt:    Scheibenwischen: Tippwischen
Testziel:      Korrekte Funktionsweise
Teststufe:     Systemtest
Testplattform: Gesamtfahrzeug

Vorbedingung: 1. Klemme 15 aktiv
              2. Fahrzeug bewegt sich
              3. Intervallwischen in Wischstufe 2 aktiv
Eingabe     : 1. Tippwischen aktivieren (Anf-6)
Erwartetes  : 1. Das Intervallwischen läuft ungestört weiter,d.h.
Ergebnis        es wird vom Tippwischen nicht beeinflusst (Anf-5)
```

Der obere Testfall ist ein nichtfunktionaler Systemtestfall. Er sichert das nichtfunktionale Testziel *Robustheit* ab. Der untere Testfall ist ein funktionaler Systemtestfall. Er sichert das Testziel *Korrekte Funktionsweise* ab.

In beiden Beispieltestfällen wird das Testobjekt *Tippwischen* vom Testobjekttyp *Fahrzeugfunktion* im System *Scheibenwischen* abgesichert. Um Verfolgbarkeit zu schaffen, ist jeder Systemtestfall mit mindestens einer Systemanforderung verlinkt. Im Beispiel ist die *Eingabe 1* des oberen Testfalls mit der Anforderung *Anf-6* von Seite 10 verlinkt. Systemanforderungen verfeinern Fahrzeugfunktionen und Systemtestfälle sind mit den verfeinernden Systemanforderungen verlinkt. Somit entsteht ein Zusammenhang zwischen Testfällen und Funktionen [SZ10, S.286].

Funktionsorientierung. Die fachliche Schicht *Fahrzeugfunktion* zwischen System und Systemanforderung bietet Vorteile in der Teststeuerung. Da beispielsweise nicht alle Funktionen eines Systems zum selben Zeitpunkt fertiggestellt werden, bietet die *funktionsorientierte Teststeuerung und -durchführung* die Möglichkeit, die Zeitpläne im Systemtest mit den Fertigstellungsplänen der einzelnen Funktionen zu harmonisieren.

Die Funktionsorientierung wirkt sich auch auf die Testauswertung aus. Die Übersichtlichkeit wird deutlich gesteigert, indem Testfälle und aufgedeckte Fehler nicht nach einzelnen Anforderungen, sondern nach Fahrzeugfunktionen katalogisiert werden. Dies vereinfacht die Bestimmung der Testabdeckung und somit des Testendes. Die *funktionsorientierte Testauswertung* ermöglicht die Beherrschbarkeit der Anforderungen von über 2000 Fahrzeugfunktionen [Bro06, S.36] sowohl in der Anforderungs- als auch Testentwicklung.

Funktionalität. Funktionale Systemtestfälle sind zumeist einfach zu ermitteln. Sie werden direkt aus Systemanforderungen abgeleitet. Der zweite Beispieltestfall ist ein solcher funktionaler Systemtestfall, weil er das Testziel *Korrekte Funktionsweise* in der Teststufe *Systemtest* für die Anforderung *Anf-5* absichert. Damit der Testfall lauffähig ist, müssen die Vorbedingungen erfüllt sein: Das *Intervallwischen* muss aktiv sein während die Funktion *Tippwischen* aktiviert wird. Der Systemtestfall deckt keinen Fehler auf, wenn das *Intervallwischen* gemäß Spezifikation nicht beeinflusst wird. Da sich funktionale Testfälle direkt aus Anforderungen ableiten lassen, basiert die Definition von Testendekriterien häufig auf der Abdeckung der funktionalen Anforderungen.

Nichtfunktionalität. Die Ermittlung von nichtfunktionalen Anforderungen und die Dokumentation ihrer Beziehung zu funktionalen Anforderungen gilt als Herausforderung in der automobilen Anforderungsentwicklung [PBKS07, S.10f]. Dies wirkt sich, wie im Beispiel gezeigt, direkt auf die Testentwicklung aus: Der nichtfunktionale Robustheitstestfall soll bestätigen, dass ein mehrmaliges Auslösen des Tippwischens zu keinem eigenartigen Verhalten führt.

Obwohl in der Beschreibung der Fahrzeugfunktion auf Seite 10 nicht spezifiziert ist, wie das Scheibenwischersystem bei schnell aufeinanderfolgender zehnmaliger Auslösung des Tippwischen reagieren soll, existiert ein Robustheitstestfall, der diesen Fall abdeckt. Somit wurde eine nichtfunktionale Anforderungsspezifikationslücke aufgedeckt, da im Beispiel keine Anforderung existiert, die dem Testfall entspricht. Damit sichert der Testfall nicht spezifizierte Anforderungen ab. Gemäß der Definition aus der Standardliteratur kann ein solcher Testfall eigentlich keinen Fehler finden, weil ein Fehler die „Nichterfüllung einer festgelegten Anforderung“ ist. Eine Schwierigkeit bei der Spezifikation von nichtfunktionalen Anforderungen und den dazugehörigen Testfällen ist die Entscheidung, wann die Nichtfunktionalität eines Systems genau genug beschrieben und abgesichert ist. Lösungstechniken bietet hier die Testplanung, indem festgelegt wird, welche nichtfunktionalen Qualitätsmerkmale in welchem Umfang abzusichern sind.

Lösungsneutralität. Im Beispiel auf Seite 10 wurde die Anforderung *Anf-3: Klemme 15 aktiv* als lösungsorientiert eingestuft, weil sie über technische Details verfügt. Die Anforderung wäre lösungsneutral, wenn sie *Anf-3: Zündung an* lauten würde. Beide Beispieltestfälle sind Systemtestfälle, die während Versuchsfahrten oder Prüfklausuren im Gesamtfahrzeug durchgeführt werden. Im Gesamtfahrzeug kann während des Systemtests nicht geprüft werden, ob *Klemme 15 aktiv* tatsächlich gleichbedeutend mit *Zündung an* ist. Der Testfall ist in diesem Zusammenhang nur ausführbar, weil der Tester annimmt, dass *Klemme 15 aktiv* gleichbedeutend mit *Zündung an* ist. Tatsächlich wird im Gesamtfahrzeugtest aber kein Tester mit einem Oszilloskop den Pegel an *Klemme 15* prüfen. Am Beispiel wird deutlich, welche Auswirkungen die Lösungsneutralität sowohl auf Anforderungen als auch auf Testfälle hat.

2.2.4. Testplanungsdimensionen

Im Rahmen der Testplanung wird festgelegt, wie und in welchem Umfang ein System abzusichern ist. Das Dokument, in dem die Ergebnisse der Testplanung festgehalten werden, heißt *Testkonzept*. *Testziele*, *Teststufen* und *Testplattformen* definieren hierbei den Raum, in dem der Testumfang dimensioniert werden kann. In dieser Arbeit werden *Testziele*, *Teststufen* und *Testplattformen* daher als *Testplanungsdimensionen* bezeichnet.

Testziele. Testziele beschreiben den Zweck des Testens. Sie entsprechen den Qualitätsmerkmalen von Qualitätsmodellen, die beispielsweise in Normen wie der [ISO9126-1, S.7] beschrieben sind. Auflistungen und abstrakte Beschreibungen von Qualitätsmerkmalen existieren in der Standardliteratur [Lig09, S.8,S.426ff] [SL05, S.12f] [SBS11, S.24]. Testziele werden durch Unterziele verfeinert, bis sie quantitativ bewertbar sind. Es folgt eine beispielorientierte Beschreibung ausgewählter Qualitäts(unter)merkmale bzw. Test(unter)ziele.

- Der erste Testfall auf Seite 17 ist ein nichtfunktionaler *Robustheitstestfall*. *Robustheit* fordert als Unterziel der *Zuverlässigkeit*, dass ungewöhnliche Nutzereingaben zu keinem eigenartigen Systemverhalten führen.
- Der zweite Beispieltestfall auf Seite 17 ist ein *funktionaler* Testfall, da er das Testziel *Absicherung der korrekten Funktionalität* verfolgt.
- Das Unterziel *Übersichtlichkeit* des Ziels *Benutzbarkeit* fordert beispielsweise, dass der Fahrer mittels weniger Aktionen zu jedem Menüpunkt im Navigationssystem gelangt. Mit Hilfe von *Benutzbarkeitstestfällen* wird dieses Testziel abgesichert.
- Das Unterziel *Zeitverhalten* des Ziels *Effizienz* fordert beispielsweise, dass ein Elektrofahrzeug in angemessener Zeit aufgeladen wird. Mit Hilfe von *Effizienztestfällen* wird dieses Zeitverhalten abgesichert.

Die obigen Beispiele sind nur ein kleiner Ausschnitt von funktionalen und nichtfunktionalen Qualitätsmerkmalen bzw. Testzielen. Das Studium realer Systemanforderungen und Systemtestfälle deckt viele weitere anforderungsseitige Qualitätsmerkmale und testseitige Testziele auf.

Teststufen. Teststufen beschreiben das Verhältnis vom rechten zum linken Ast des V-Modells. Andreas Spillner und Tilo Linz beschreiben die Teststufen sehr detailliert [SL05, S.41ff]. Der Systemtest erfordert *Lösungsneutralität* (Seiten 12 und 19), die ausschließlich in den benachbarten Teststufen umsetzbar ist. Es folgt die Beschreibung der aus Systemsicht relevanten Teststufen.

- Fahrzeugsysteme bestehen nicht nur aus Softwarekomponenten. Das Scheibenwischersystem setzt sich beispielsweise aus den Komponenten Wischerarm, Wischermotor, Wassertank, -pumpe, -schlauch, Spritzdüse sowie dem Steuergerät und den einzelnen Softwarefunktionen zusammen. In der Komponententeststufe wird jede Komponente mit Hilfe von Komponententestfällen einzeln abgesichert. Demnach existieren im Komponententest nicht nur Softwaretestfälle.
- Die einzelnen Komponenten werden nach und nach zum System zusammengesetzt. Diese teilweise unvollständige Integration wird im Integrationstest abgesichert. Beispielsweise soll zunächst die Teilfunktionalität *Frontwischen ohne Scheibenwaschen* getestet werden. Dazu wird das Simulink-Modell der Softwarefunktion mit den relevanten mechanischen Bauteilen Wischerarm und Wischermotor integriert. Die unvollständige Integration ist nur ein Aspekt des Themas *Virtuelle Integration*. Es gilt bereits seit einigen Jahren als herausfordernd [Bro06, S.37].
- Im Systemtest werden einzelne aber vollständige Systeme, wie beispielsweise das vollständig integrierte Scheibenwischersystem abgesichert. Um zu prüfen, ob ein System gemäß Spezifikation funktioniert, werden Testfälle aus Anforderungen abgeleitet. Aus diesem Grund heißt der Systemtest auch *anforderungsbasierter Test* [SBS11, S.30f].
- Moderne Fahrzeuge können bis zu 100 Steuergeräte und somit mehr als 100 einzelne Systeme besitzen. Daher existieren weitere Teststufen. Beispielsweise wird der Scheibenwischer verlangsamt, wenn eine Tür geöffnet wird, um Passagiere vor Spritzwasser zu schützen. Die Integration des Scheibenwischer- und Türsystems wird im *Systemintegrationstest* oder auch in der höchsten Teststufe *Gesamtfahrzeugtest* abgesichert.

Testplattformen. Auf Testplattformen werden Testfälle durchgeführt. Während Teststufen dem zeitlichen Verlauf des Testprojekts entsprechen, realisieren Testplattformen die Testumgebung pro Teststufe. Es folgt ein kleiner Ausschnitt der Testplattformen im Automobilbereich. Ralf Nörenberg beschreibt die Testplattformen in seiner Dissertation ausführlicher [Noe12, S.49ff].

- Auf dem *Komponentenprüfstand* werden die einzelnen Softwarefunktionen eines Systems ohne die mechanischen Komponenten getestet. Beispielsweise kann auf dem Prüfstand das Testziel *Korrektheit der Funktionalität* der programmierten Fahrzeugfunktionen abgesichert werden.
- Auf dem *Integrationstestbrett* werden die Software und die mechanischen Komponenten eines Systems zusammengesetzt, um deren korrektes Zusammenspiel zu testen. Auf dem Integrationstestbrett kann beispielsweise das Testziel *Korrektheit der Schnittstellen* abgesichert werden.
- Mit dem *Erprobungs- oder Gesamtfahrzeug* werden Versuchsfahrten und Nutzungsszenarien durchgeführt, um einzelne Systeme und ihr Zusammenspiel mit anderen System abzusichern.

2.2.5. Fokus: Wiederholung von Systemtestfällen

In dieser Arbeit werden Techniken vorgestellt, um Testfälle mit Anforderungen zu verlinken, die wiederverwendet wurden. Das Ziel der Verlinkung ist es, die Nachvollziehbarkeit während der *Testplanung und Teststeuerung* zu fördern, wenn Testfälle auf der Grundlage wiederverwendeter Anforderungen wiederholt werden sollen. Im fundamentalen Testprozess lässt sich diese Arbeit somit in den Bereich der *Testplanung und Teststeuerung* verorten. Im vorherigen Unterkapitel wurden die relevanten Grundlagen im Bereich der Anforderungsentwicklung vorgestellt. Dieses Unterkapitel lieferte einen Einblick in die relevanten Aspekte der Testentwicklung. Dabei wurde der Zusammenhang zwischen Anforderungs- und Testentwicklung berücksichtigt, indem die aktuellen Herausforderungen aus der Anforderungsentwicklung wiederaufgegriffen und in den Bereich der Testentwicklung übertragen wurden.

2.3. Verfolgbarkeit

Im Verlauf dieses Kapitels zeichnete sich bereits das Thema der Verfolgbarkeit ab. Bevor auf die Verfolgbarkeit im Rahmen der Anforderungsentwicklung und Testentwicklung eingegangen wird, werden die relevanten Begriffe eingeführt. Dies stellt sich als herausfordernd dar, da in der deutschsprachigen Literatur kein einheitliches Begriffsverständnis vorherrscht.

2.3.1. Begriffe

Während Klaus Pohl den Begriff *Nachvollziehbarkeit* verwendet [Poh08, S.505], greift Helmuth Partsch auf den Begriff *Zurückführbarkeit* oder *Rückführbarkeit* zurück [Par10, S.10]. Weitere, häufig verwendete Begriffe sind *Rückverfolgbarkeit*, *Nachverfolgbarkeit*, *Durchgängigkeit* oder schlicht *Verfolgbarkeit*. Viele Autoren verwenden einfach den englischen Begriff *Traceability*. Zuletzt erschien das englischsprachige Grundlagenbuch Software and Systems Traceability, in dem die Begriffswelt rund um die *Traceability* umfassend definiert wird [CHGZ12, S.3ff]. Im Folgenden werden relevante Teile dieser Begriffswelt in das Deutsche übertragen und gegebenenfalls dem Deutschen angeglichen.

Verfolgbares Artefakt. Ein Artefakt ist eine Dateneinheit (z.B. eine einzelne Anforderung, eine Anforderungsgruppe, ein UML-Modell oder eine Java-Klasse). Ein verfolgbares Artefakt (engl.: trace artefact) steht in einer gerichteten Beziehung zu einem anderen Artefakt.

Artefakttyp. Der Artefakttyp (engl.: trace artefact type) bezeichnet die Art des Artefakts (z.B. Anforderung oder Testfall).

Verbindung/Link von Quelle zu Ziel. Eine Verbindung oder ein Link (engl.: trace link) verbindet ein Quell-Artefakt und ein Ziel-Artefakt miteinander. Ein Link startet vom Quell-Artefakt (engl.: source artefact) und endet am Ziel-Artefakt (engl.: target artefact).

Linkrichtung. Ein Link ist gerichtet (engl.: trace link direction). Er zeigt immer von der Quelle zum Ziel (z.B. vom Testfall zur Anforderung).

Linktyp. Der Verbindungstyp oder Linktyp (engl.: trace link type) beschreibt die Beziehung, in der das Quell-Artefakt und das Ziel-Artefakt stehen (z.B. Testfall sichert Anforderung ab).

Verlinkungspaar. Ein Verlinkungspaar (engl. Substantiv: the trace) besteht aus zwei verlinkten Artefakten mitsamt dem sie verbindenden Link.

Einzelpaar und Paarkette. Ein Einzelpaar (engl.: atomic trace) ist ein einzelnes Verlinkungspaar. Eine Paarkette (engl.: chained link) besteht aus mehreren Verlinkungspaaren, die eine Sequenz bilden, so dass das Ziel-Artefakt eines Einzelpaars das Quell-Artefakt des nächsten Einzelpaars ist.

Verfolgbarkeit. Die Verfolgbarkeit oder Verfolgungsmöglichkeit (engl.: traceability) bezeichnet die generelle Möglichkeit (d.h. -ability, -barkeit), Verlinkungspaare zu erzeugen und zu verwenden.

Verfolgung. Die Verfolgung (engl. Verb: to trace) ist die Aktivität des Verfolgens eines Links von der Quelle zum Ziel oder vom Ziel zur Quelle.

Rückverfolgung. Die Rückverfolgung ist die Aktivität der Verfolgung eines Links zu seinem Ursprung, d.h. vom Ziel-Artefakt zum Quell-Artefakt.

Nachverfolgung. Die Nachverfolgung ist die Aktivität der Verfolgung eines Links zu seinem Ziel, d.h. vom Quell-Artefakt zum Ziel-Artefakt.

Nachvollziehbarkeit. Die Nachvollziehbarkeit ist der Zweck der Verfolgbarkeit. Die Verfolgbarkeit ermöglicht demnach die Nachvollziehbarkeit.

Verlinkung. Die Verlinkung (engl.: traceability creation) beschreibt die Aktivität der Erstellung eines Verlinkungspaars.

Linkwartung. Die Linkwartung (engl.: traceability maintenance) beschreibt die Aktivität der Aktualisierung oder Löschung eines Verlinkungspaars.

Linknutzung. Die Linknutzung (engl.: traceability use) umfasst alle Aktivitäten, bei denen Verlinkungspaare verwendet werden.

Verlinkungsmodell. Ein Verlinkungsmodell (engl.: traceability information model) benennt alle beteiligten Artefakttypen und die möglichen Linktypen zwischen ihnen.

2.3.2. Verfolgbarkeit in der Anforderungsentwicklung

Die Verfolgbarkeit ist ein wichtiger Aspekt im Bereich der Anforderungsentwicklung. Daher existieren in der Standardliteratur ganze Buchkapitel, die sich der Thematik widmen [Poh08, S.505ff]. Im Fokus der Anforderungsverfolgbarkeit stehen Links von und zu Anforderungen.

Anforderungsverfolgbarkeit. Zuvor wurde *Verfolgbarkeit* als deutsche Übersetzung für *Traceability* vorgeschlagen. Die Verfolgbarkeit einer Anforderung ist die Fähigkeit, ein Verlinkungspaar zu erzeugen und zu verwenden, dessen Quell- und/oder Ziel-Artefakt eine Anforderung ist.

Anforderungsnachvollziehbarkeit gemäß Pohl. Klaus Pohl definiert in seinem Werk Requirements Engineering - Grundlagen, Prinzipien, Techniken den Begriff *Anforderungsnachvollziehbarkeit*: „Die Nachvollziehbarkeit einer Anforderung ist die *Fähigkeit*, den Lebenszyklus der Anforderung vom Ursprung der Anforderung über die verschiedenen Verfeinerungs- und Spezifikationsschritte bis hin zur Berücksichtigung der Anforderung in nachgelagerten Entwicklungsartefakten *verfolgen zu können*“ [Poh08, S.505]. Die Fähigkeit, verfolgen zu können bezeichnet allerdings die *Verfolgbarkeit*.

Anforderungsnachvollziehbarkeit. Die Anforderungsnachvollziehbarkeit ist der Zweck der Anforderungsverfolgbarkeit. Die obige Definition lautet umformuliert: “Die Nachvollziehbarkeit einer Anforderung dient dem Zweck, den Lebenszyklus der Anforderung vom Ursprung der Anforderung über die verschiedenen Verfeinerungs- und Spezifikationsschritte bis hin zur Berücksichtigung der Anforderung in nachgelagerten Entwicklungsartefakten zu bewerten und die Bewertung in Entscheidungen einfließen zu lassen“.

Anforderungslink. Ein Anforderungslink ist ein Link zwischen einer Anforderung und einem weiteren Softwareentwicklungsartefakt [Pin04, S.91].

Orientierung von Anforderungslinks. Anforderungslinks haben stets eine Orientierung: Der Link zeigt zu/von einer Anforderung, die das Quell- und/oder Ziel-Artefakt eines Verlinkungspaars ist. Anforderungslinks besitzen eine Vorwärts-, Rückwärts-, Inter- oder Extra-Orientierung [Pin04, S.93ff].

- Ein *Vorwärtslink* zeigt von einer Anforderung zu einem zeitlich nachgelagerten Artefakttyp. Beispiele für Vorwärtslinks sind Links von Anforderungen zu Architekturelementen, von Anforderungen zu Quelltextfragmenten oder von Anforderungen zu Testfällen.
- Ein *Rückwärtslink* dokumentiert den Ursprung einer Anforderung. Beispiele dafür sind Links von Gesetzen oder Normen.
- Ein *Inter-Link* zeigt von einer Anforderung zu einer anderen Anforderung. Diese Links können beispielsweise funktionale Abhängigkeiten zwischen Anforderungen oder Verfeinerungsbeziehungen verdeutlichen.
- Ein *Extra-Link* zeigt von einer Anforderung zu einem Artefakt eines anderen Typs. Dies können zum Beispiel Quelltext oder Testfälle sein.

Weitere Definition. Alle anderen Definitionen des Abschnitts 2.3.1 auf den Seiten 23f gelten auch im Kontext der Anforderungsentwicklung.

Anforderungsverfolgbarkeit im Kontext dieser Arbeit. Anforderungsverfolgbarkeit (engl.: requirements traceability) ist ein etabliertes Forschungsgebiet. In Kapitel 3.3 werden auf den Seiten 37ff aktuelle und relevante Arbeiten des Gebiets genauer untersucht. Diese Arbeit ordnet sich in den Bereich der Anforderungs- und Testfallverfolgbarkeit ein. Anforderungsseitig liegt der Fokus im Bereich der Anforderungswiederverwendung. In diesem Zusammenhang wird in Kapitel 3.2 auf Seite 34 ein neuer *Linktyp* für Anforderungen vorgestellt: Der *Wiederverwendet-Link*, kurz Wv-Link. Der Wv-Link verbindet eine Ziel- und eine Quell-Anforderung. Er zeigt von der Ziel-Anforderung zu der Quell-Anforderung, die durch die Ziel-Anforderung wiederverwendet wird. Die *Linkrichtung* ist somit von der Ziel-Anforderung zur Quell-Anforderung. Anfänglich erscheint es etwas ungewöhnlich, dass die Ziel-Anforderung das *Quell-Artefakt* und die Quell-Anforderung das *Ziel-Artefakt* des *Verlinkungspaars* ist. Die Besonderheit dieser Arbeit ist, dass der Wv-Link ein *Rückwärtslink* ist, der die *Rückverfolgung* zum Ursprung der wiederverwendeten Ziel-Anforderung ermöglicht. Diese Sichtweise ist bislang nicht verbreitet. In der Literatur werden häufig Gesetze und Normen, nicht jedoch die Wiederverwendung als Beispiel für den Ursprung von Anforderungen verwendet.

2.3.3. Verfolgbarkeit in der Testentwicklung

Die Testentwicklung ist eng verbunden mit der Anforderungsentwicklung: Systemanforderungen spezifizieren das Verhalten eines Systems und Testfälle prüfen, ob sich das System gemäß Spezifikation verhält. Die Verfolgbarkeit vom Testfall zur Anforderung ist daher per Definition gegeben.

Testfallverfolgbarkeit. Die Testfallverfolgbarkeit ist die Möglichkeit, ein Verlinkungspaar zu erzeugen und verwenden, dessen Quell-Artefakt und/oder Ziel-Artefakt den Artefakttyp *Testfall* hat. Der Zweck der Testfallverfolgbarkeit ist die Ermöglichung der *Testfallnachvollziehbarkeit*, um beispielsweise „festzuhalten, welche Anforderungen [...] aufgrund welcher Testbedingungen mit welchen Testfällen getestet wurden“ [SRWL14, S.30].

Testlink. Ein Testlink verbindet einen Testfall mit einem anderen Artefakt. Wie Anforderungslinks haben auch Testlinks eine Orientierung.

- Ein *Vorwärtslink* zeigt von einem Testfall zu einem im Entwicklungsprozess nachgelagerten Artefakt. Die Durchführung eines Testfalls hat das Ziel, Fehler im Testobjekt aufzudecken. Somit sind Links von Testfällen zu Fehlern Vorwärtslinks. Die *testgetriebene Entwicklung* ist in diesem Zusammenhang ein interessanter Aspekt. Hier werden erst Modultestfälle implementiert, um darauf basierend die Methoden zu programmieren. In diesem Spezialfall sind Links von Modultestfällen zu Methoden Vorwärtslinks. Testlinks sind jedoch häufig Rückwärtslinks.
- Ein *Rückwärtslink* dokumentiert den Ursprung eines Testfalls. Da Systemtestfälle aus Anforderungen gewonnen werden [SL05, S.72], sind Links von Systemtestfällen zu Anforderungen Rückwärtslinks.
- Ein *Inter-Link* zeigt von einem Testfall zu einem anderen Testfall. Zwei Testfälle können beispielsweise verlinkt werden, wenn sie den selben Fehler aufdecken. Ein weiterer Inter-Link könnte verdeutlichen, dass ein Testfall die Vorbedingungen eines anderen Testfalls herstellt.
- Ein *Extra-Link* verbindet einen Testfall mit einem Artefakt eines anderen Typs. Beispiele dafür sind Links zu Anforderungen oder Fehlern.

Testfallverfolgbarkeit im Kontext dieser Arbeit. Auch die Testfallverfolgbarkeit (engl.: test traceability) ist in der Wissenschaft und in der Praxis ein etabliertes Thema. Dies zeigt Kapitel 3.3 auf den Seiten 37ff. Da sich diese Arbeit im Bereich der Anforderungswiederverwendung und ihrer Verfolgbarkeit bewegt, stellt sich die Frage, wie sich die Testfallverfolgbarkeit in diesem Zusammenhang verhält. In Kapitel 3.2 wird auf den Seiten 34f das WvT-Problem vorgestellt, das genau diesen Sachverhalt adressiert.

2.4. Zusammenfassung

Im Zentrum dieser Arbeit stehen Systemanforderungen und -testfälle. Sie ordnet sich daher in der oberen Ebene des V-Modells ein. Aus diesem Grund wurden in diesem Kapitel für die Bereiche *Anforderungsentwicklung (engl.: requirements engineering)* und *Testentwicklung (engl.: test engineering)* die wesentlichen Begriffe definiert und allgemeine Herausforderungen dargelegt.

Da sich diese Arbeit mit Wiederverwendungsbeziehungen zwischen Anforderungen sowie Testbeziehungen zwischen Testfällen und Anforderungen auseinandersetzt, spielt die *Verfolgbarkeit (engl.: traceability)* eine wesentliche Rolle. Aus diesem Grund wurden in diesem Kapitel die Begriffe im Bereich der Verfolgbarkeit zunächst allgemein definiert. Einige der Begriffe wurden im Sinne der Anforderungs- und Testfallverfolgbarkeit spezialisiert. Außerdem wurde der Zusammenhang zwischen Anforderungs- und Testfallverfolgbarkeit hergestellt. Die Begriffe in diesem Kapitel gelten im Kontext der gesamten Arbeit.

Diese Arbeit orientiert sich an relevanten und alltäglichen Problemen aus der Praxis, um diese aus der wissenschaftlichen Perspektive auszuleuchten und methodisch zu lösen. Aufbauend auf den Grundlagen in diesem Kapitel wird im nachfolgenden Kapitel die praktische Perspektive betrachtet. Aus dieser wird schließlich die dieser Arbeit zugrunde liegende Problemstellung ausgearbeitet und mit den Möglichkeiten des Forschungsstands verglichen.

3. Problemstellung und Lösungsstrategie

Statt zu implementieren, arbeiten Entwicklungsingenieure herstellerseitig fast täglich mit Systemanforderungen und Systemtestfällen. Diese Anforderungen und Testfälle sind Bestandteil von Spezifikationsdokumenten. In diesem Kapitel wird der Aufbau von Systemlastenheften, Systemtestspezifikationen und Testkonzepten beschrieben. Anschließend wird das WvT-Problem präsentiert, das den Grundstein für alle Aspekte dieser Arbeit legt. Nachdem das WvT-Problem im Forschungsstand lokalisiert wurde, erfolgt die Vorstellung der verfolgten Ziele und der Lösungsstrategie: der 3-schichtigen Methode zur automatischen Verlinkung von Testfällen und Anforderungen.

3.1. Anforderungen und Testfälle in der Praxis

Während der Anforderungsspezifikation einer Baureihe entstehen zahlreiche Anforderungen für eine Vielzahl von softwarelastigen Systemen. Den Anforderungen werden schließlich im Rahmen der Testspezifikation Testfälle zugewiesen, die sicherstellen, dass sie im fertig implementierten System korrekt umgesetzt wurden. Um Nachvollziehbarkeit zu ermöglichen, müssen die Beziehungen zwischen Testfällen und Anforderungen verfolgbar sein. Diese Verfolgbarkeit wird technisch durch Trace-Links, i.F. Links, realisiert.

Verlinkung von Testfällen und Anforderungen. Bild 3.1 zeigt einen Ausschnitt aus Spezifikationsdokumenten. Die obere linke Box zeigt ein Quell-Systemlastenheft, das in einem früheren Baureihenprojekt erstellt wurde. Die Zeile *132* zeigt die Fahrzeugfunktion *Wischunterbrechung vorne bei Motorstart*. Die beiden Quell-Anforderungen *133* und *134* konkretisieren die Funktion. Die untere Box zeigt mehrere Testfälle, die mit den Quell-Anforderungen *133* und *134* verlinkt sind. Beispielsweise ist der Testfall *93: Waschen kurz während Motorstart* über einen Testet-Link mit der Quell-Anforderung *134* verbunden.

Wiederverwendung von Anforderungen. Quell-Systemlastenhefte werden in der Praxis wiederverwendet, indem sie zunächst in einen neuen Baureihenprojektordner kopiert und anschließend modifiziert werden. Die obere rechte Box in Bild 3.1 zeigt das Ziel-Systemlastenheft einer neuen Baureihe. Dort wurde die Ziel-Anforderung *ziel-134* besonders stark modifiziert: Die Beschreibung der *Waschunterbrechung* wurde entfernt.

Wiederverwendung der Testfallverlinkung. Bild 3.1 deutet bereits die Problemstellung dieser Arbeit an: Ist der Testfall *93: Waschen kurz während Motorstart*, der bereits die Quell-Anforderung *134* absichert, auch dazu geeignet, die Ziel-Anforderung *ziel-134* abzusichern? oder kurz: Kann der Testfall *93* mit der Ziel-Anforderung *ziel-134* verlinkt werden? Im Beispiel kann die Verlinkung zwischen dem Testfall *93* und der Ziel-Anforderung *ziel-134* nicht ungeprüft durchgeführt werden. Der Testfall würde Anforderungen absichern, die im Ziel-Systemlastenheft an dieser Stelle nicht mehr existieren.

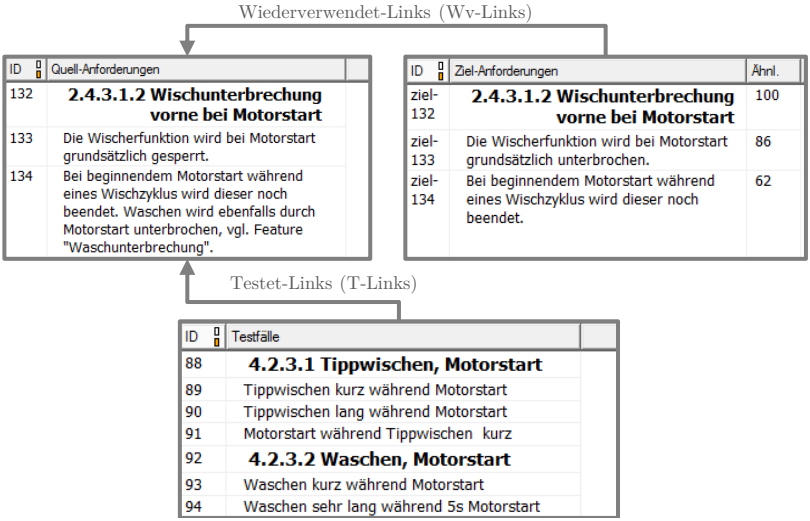


Bild 3.1.: Beispiele für Systemanforderungen und Systemtestfälle

3.1.1. Systemlastenheft (SLH)

Ein modernes Fahrzeug verfügt über viele Systeme. Jedes System wird durch ein SLH beschrieben. Ein SLH enthält neben organisatorischen Festlegungen eine Liste abstrakter Fahrzeugfunktionen, die das System anbietet. Im SLH entsprechen Fahrzeugfunktionen Unterkapiteln. Da Systemanforderungen Fahrzeugfunktionen konkretisieren, sind sie der konkrete Inhalt der Unterkapitel. Beispiele für Fahrzeugfunktionen sind *Frontwischen* und *Schiebedach schließen*. Einzelne Funktionen können sehr umfangreich sein. Dies führt bereits auf Systemebene zu einer Vielzahl an Anforderungen. Die Funktion Frontwischen soll dies beispielhaft verdeutlichen: Zunächst wird spezifiziert, wie die Funktion aktiviert wird. Dies geschieht über den *Lenkstockschalter* oder den *Regensensor*. Anschließend wird definiert, welche *Wischstufen* existieren und wie schnell diese sind. Dabei sind viele Sonderregeln zu beachten. Wenn eine *Tür geöffnet* ist, soll das *Wischen deaktiviert* werden, damit ein- und aussteigende Personen nicht mit Wischwasser bespritzt werden. Damit Fußgänger und Fahrradfahrer nicht bespritzt werden, wenn das Auto beispielsweise an einer *Ampel steht*, soll der *Wischer im Stand nur sehr langsam laufen*. Viele weitere, mitunter schon sehr technische Details verfeinern das Frontwischen. Allein die Funktion Frontwischen besteht in dem SLH eines modernen Fahrzeugs aus Systemanforderungen im dreistelligen Bereich.

Anzahl pro Baureihe. Wie viele Anforderungen beschreiben die Softwarefunktionen einer ganzen Baureihe? Die folgende Schätzung bietet eine Tendenz. Ein modernes Fahrzeug hat nahezu 100 Steuergeräte. Angenommen, auf jedem Steuergerät läuft nur ein Softwaresystem und angenommen, mittelgroße Systeme haben 5.000 einzelne Anforderungen. Für ein Fahrzeug fallen unter diesen vereinfachten Annahmen bereits 500.000 Anforderungen an.

Anforderungseigenschaften. Systemanforderungen beschreiben Fahrzeugfunktionen. Zusätzlich besitzt jede Anforderung Eigenschaften, die sie klassifizieren. Dies können Sicherheitsstufen oder Abhängigkeiten zu Anforderungen anderer SLH sein. Daraus resultieren Anforderungskategorien, z.B. besonders sicherheitskritische oder stark verteilte Anforderungen.

3.1.2. Systemtestspezifikation (STS)

Jedem SLH ist mindestens eine STS zugeordnet. STS enthalten Systemtestfälle, die die korrekte Umsetzung der SLH sicherstellen. Die Testfälle entsprechen dem allgemein bekannten Aufbau: Vor-, Nach-, Erfolgsbedingung, Testschritte usw. [SL05, S. 263]. Da die Systemtestfälle Systemanforderungen absichern, stimmen sie mit dem Abstraktionsniveau der Anforderungen überein. Sie sind das Prüfäquivalent einer natürlichsprachlichen Anforderung. Als Vorbedingung eines Testfalls für die *Wischunterbrechung bei Motorstart* soll beispielsweise der *Motor aus* sein. Die Testschritte bestehen im *Starten des Motors* bei gleichzeitiger Aktivierung des *Tippwischens mittels Lenkstockschalter*. Der Testfall ist *erfolgreich*, wenn der Wischer nicht wischt, weil die Anforderung *Wischunterbrechung* dies fordert. Neben solch einfachen Testfällen existieren auch sehr komplizierte Testfälle. Ein mittelgroßes System wie das Scheibenwischen verfügt über Testfälle im vierstelligen Bereich.

Anzahl pro Baureihe. Auch an dieser Stelle stellt sich die Frage, wie viele Systemtestfälle eine gesamte Baureihe absichern. Geht man wieder von einem modernen Fahrzeug mit etwa 100 Steuergeräten - also mit 100 Systemen und somit 100 SLH - aus, so existieren in diesem Fahrzeug auch mindestens 100 STS. Angenommen, eine einzelne STS enthält für mittelgroße Systeme 5.000 Testfälle. Daraus ergibt sich pro Baureihe eine Anzahl von 500.000 Testfällen. In der Praxis existieren in der Regel jedoch mehr Testfälle als Anforderungen.

Testfalleigenschaften. Wie Anforderungen haben auch Testfälle Eigenschaften, die sie klassifizieren. Jedem Testfall werden beispielsweise die Testziele zugewiesen, die er absichern soll. Testziele ergeben sich aus Qualitätsmodellen wie der [ISO9126-1]. Neben der funktionalen Korrektheit existieren auch nichtfunktionale Testziele wie Wartbarkeit oder Schnittstellenkorrektheit. Eine weitere Testfalleigenschaft ist die Testplattform, auf welcher der Testfall ausgeführt wird. Beispiele für automobilspezifische Plattformen sind HiL (Hardware-in-the-Loop) oder das Gesamtfahrzeug. Mit Hilfe der klassifizierenden Eigenschaften gruppieren sich Testfälle dann in Funktionstestfälle oder Schnittstellentestfälle und/oder HiL- oder Gesamtfahrzeugtestfälle.

3.1.3. Testkonzept (TK)

Das TK ist das Ergebnis der Testplanung. Dessen Vorhandensein wird von der [ISO26262-3] gefordert. Im TK werden die Testumfänge für die Artefakte jeder V-Modellebene definiert. Somit enthält das TK Aussagen dazu, wann welche Testfälle durchgeführt werden müssen, um ein definiertes Qualitätsmaß effektiv zu erreichen. Effektiv heißt hierbei die Durchführung der richtigen Testfälle zum richtigen Zeitpunkt. Dies entspricht den Testendekriterien pro V-Modellebene. Für die oberste V-Modellebene wird im TK beispielsweise für Fahrzeugfunktionen und somit auch für Systemanforderungen festgelegt, welche Art von Systemtestfällen durchgeführt werden muss. Für die nächsttiefere Ebene des V-Modells wird definiert, welche Art von Testfällen bei der Komponentenintegration durchgeführt werden muss. Die Festlegungen im TK reichen bis zur tiefsten Ebene des V-Modells, der Quelltextebene. Kurzum: Das Testkonzept klärt, wann welche Testfälle zu welchem Zweck welche Entwicklungsartefakte absichern müssen. Das TK klärt damit die Beziehung des rechten zum linken Ast des V-Modells. Im Rahmen dieser Arbeit sind jedoch lediglich die Festlegungen in der Systemebene des V-Modells relevant. Realisiert wird das TK durch Tabellen, in denen ein Eintrag den Zwang des Vorhandenseins von Testfällen verdeutlicht (vgl. Bild 5.1, S. 82).

Anforderungs- und Testfalleigenschaften. Das TK enthält weder konkrete Systemanforderungen noch konkrete Systemtestfälle. Das TK verwendet für die Festlegungen der Testplanung die klassifizierenden Eigenschaften von Anforderungen und Testfällen. Die ASIL-Sicherheitseinstufung ist eine Anforderungseigenschaft, die in einem automobilen TK eine wichtige Rolle spielt. Durch sie wird festgelegt, welche Testfälle mit welchen Eigenschaften durchgeführt werden müssen, um eine Anforderung ausreichend abzusichern. Für sicherheitskritische Anforderungen müssen beispielsweise mehr Testfälle durchgeführt werden als für unkritische Anforderungen. Dieses „Mehr“ lässt sich anhand der Testfalleigenschaft *Testziel* quantifizieren. Für die sicherheitskritischen Anforderungen wird daher festgelegt, dass mehr Testfälle mit dem Testziel *funktionale Sicherheit* durchgeführt werden müssen.

3.2. WvT-Problem

Diese Arbeit basiert auf einem spezifischen Verlinkungsmodell, das im industriellen Umfeld beobachtet wurde: dem Wiederverwendet-Testet- oder kurz WvT-Problem. Bild 3.2 zeigt, wie das WvT-Problem das Wiederverwendungsverhältnis zweier SLH und das Testverhältnis mindestens einer STS widerspiegelt. Das Wiederverwendungsverhältnis entsteht hierbei, wenn ein aktuelles SLH Teile eines früheren SLH wiederverwendet. Das Testverhältnis entsteht, wenn die Testfälle der STS die Anforderungen eines SLH absichern.

Das WvT-Problem bildet die Grundlage dieser Arbeit. Die Abkürzung WvT taucht daher sehr häufig auf: WvT-Instanz, WvT-Typ, WvT-Verlinkung, WvT-Inkonsistenz, WvT-Diagramm, WvT-Fall und WvT-Entscheidung.

Wv: Wiederverwendet. Die Wiederverwendung von Anforderungen ist gängige Praxis. Dafür wird das SLH eines Systems eines vorangegangenen Baureihenprojekts für ein neues Baureihenprojekt zunächst kopiert und nachfolgend an die Spezifika der neuen Baureihe angepasst. Damit lassen sich zwei SLH unterscheiden: das $\text{SLH}_{\text{Quell}}$ der alten Baureihe und das SLH_{Ziel} der neuen Baureihe. Die beiden SLH befinden sich in einem Wiederverwendungsverhältnis: SLH_{Ziel} wiederverwendet (Wv) $\text{SLH}_{\text{Quell}}$.

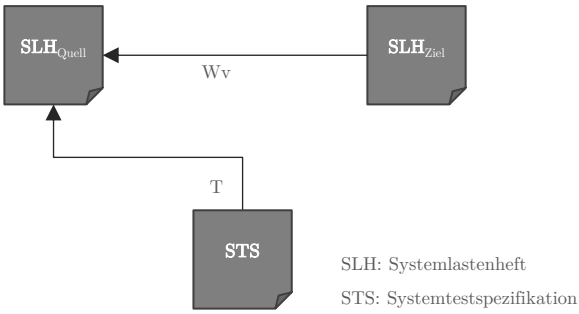


Bild 3.2.: WvT-Problem auf Dokumentenebene

T: Testet. Die STS wird im betrachteten Umfeld nicht kopiert, sondern lediglich mit dem neuen SLH_{Ziel} verlinkt. Vor dieser Verlinkung befinden sich die STS und das SLH_{Quell} in einem Testverhältnis: STS testet (T) SLH_{Quell}. Damit bezieht sich das WvT-Problem genau auf den Zeitpunkt, bevor die STS ein Testverhältnis mit dem SLH_{Ziel} eingeht.

WvT-Artefakte und WvT-Verlinkung. Bild 3.3 zeigt das WvT-Problem auf Artefaktebene. Die Dokumente SLH_{Quell}, SLH_{Ziel} und STS enthalten Artefakte des Artefakttyps *Anforderung* und *Testfall*. Im Bild zeigt ein Link mit dem Linktyp *Wiederverwendet* (kurz *Wv-Link*) von einer Ziel-Anforderung des SLH_{Ziel} zu einer Quell-Anforderung des SLH_{Quell}. Die beiden Anforderungen stehen in einem Wiederverwendungsverhältnis. Ein Link mit dem Linktyp *Testet* (kurz *T-Link*) zeigt von einem Testfall der STS zu einer Quell-Anforderung des SLH_{Quell}. Der Testfall und die Quell-Anforderung stehen in einem Testverhältnis. Das WvT-Problem beschreibt Artefakttypen, ihre Verlinkung sowie die Linktypen. Damit ist es ein *Verlinkungsmodell* (vgl. S. 24).

Da im WvT-Problem der Testfall noch nicht mit der Ziel-Anforderung verlinkt ist, existiert der im Bild gestrichelt dargestellte T-Link noch nicht. Diese Arbeit setzt sich mit der Frage auseinander, unter welchen Bedingungen dieser T-Link gesetzt werden kann und ob eine manuelle Prüfung nötig ist.

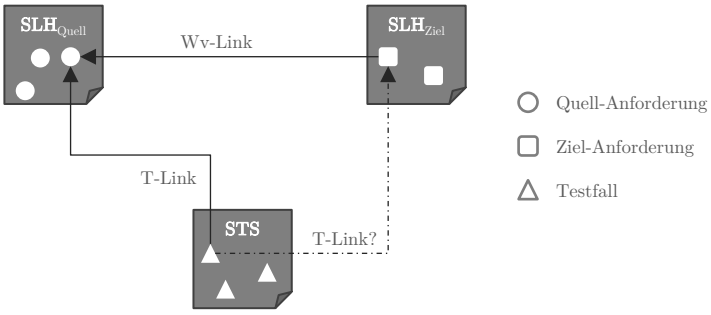


Bild 3.3.: WvT-Problem auf Artefaktebene

Stand der Technik: Manuelle Verlinkung. Das Setzen des T-Links von dem Testfall zu der Ziel-Anforderung wird als *Lösen des WvT-Problems* bezeichnet. Als zuvor die Spezifikationsdokumente vorgestellt wurden, stellte sich die Frage, wie viele Anforderungen und Testfälle im Rahmen eines Baureihenprojekts eigentlich anfallen. Ein Überschlag führte zu Anforderungen und Testfällen im sechs- bis siebenstelligen Bereich - in jedem Baureihenprojekt. Das WvT-Problem muss somit hunderttausend-, wenn nicht sogar millionenfach gelöst werden - in jedem Baureihenprojekt. In der Praxis wird momentan jedes einzelne WvT-Problem entweder manuell gelöst oder es werden einfach alle T-Links beim Kopieren des $\mathrm{SLH}_{\mathrm{Quell}}$ in das $\mathrm{SLH}_{\mathrm{Ziel}}$ übertragen.

Die erste Vorgehensweise führt ganz offensichtlich in eine Zeit- und Genauigkeitsproblematik. Bei der zweiten Vorgehensweise findet die Verlinkung zum falschen Zeitpunkt statt, da die Testfälle direkt nach dem Kopieren verlinkt werden und erst danach die Anpassung an die neuen Baureihenspezifika stattfindet. Ungeachtet der Vorgehensweise kann ein Umstand häufig beobachtet werden: Von Baureihenprojekt zu Baureihenprojekt existieren immer weniger T-Links. Das bedeutet in der Praxis zwar nicht, dass weniger getestet wird, wohl aber, dass der Test schlechter nachvollziehbar ist.

Automatische Lösung des WvT-Problems. Testfälle werden verlinkt, wenn sie aufgeschrieben werden. Diese Verlinkung ist im Vergleich zur Erstellung ein kleiner Schritt und fällt kaum ins Gewicht. Weil STS wiederverwendet werden, sammeln sich die Testfälle über die Jahre an. In neuen Baureihenprojekten müssen diese Testfälle dann immer wieder mit den wiederverwendeten Anforderungen vorwiegend manuell verlinkt werden. Es ist nachvollziehbar, dass nicht nur die Zeit, sondern auch die Motivation eine Rolle dabei spielt, dass von Projekt zu Projekt immer weniger T-Links existieren. Das Ziel dieser Arbeit ist daher, dass ein Testfall nur noch einmal manuell mit einer Anforderung verlinkt werden muss und dieser Link dann automatisch von Baureihe zu Baureihe überprüft und übernommen werden kann.

Nachfolgend wird das WvT-Problem in den verwandten Arbeiten verortet. Es wird sich zeigen, dass noch keine zufriedenstellende Lösung existiert.

3.3. Verwandte Arbeiten

Das WvT-Problem ist ein Verlinkungsmodell, das spezifische Beziehungen in den beiden Bereichen der Anforderungs- und Testentwicklung beschreibt. Der Forschungsbereich, in den sich das WvT-Problem einordnet, wird als Anforderungsverfolgbarkeit (engl.: requirements traceability) bezeichnet. Damit ist das WvT-Problem ein Anforderungsverlinkungsmodell. Noch genauer gesagt ist das WvT-Problem eine Kombination aus einem Anforderungsverlinkungsmodell und einem Testverlinkungsmodell, da auch ein Testfall als Artefakttyp und ein T-Link als Linktyp berücksichtigt werden. Nachfolgend werden allgemeine Anforderungsverlinkungsmodelle ohne explizite T-Links und spezialisierte Modelle mit expliziten T-Links vorgestellt. Es wird sich herausstellen, dass der Wv-Link zwischen Anforderungen noch nicht verbreitet ist.

Während Verlinkungsmodelle die Artefakte und ihre Links benennen, beschreiben Verlinkungstechniken den Umgang mit ihnen. Hier existieren bereits zahlreiche Ansätze, die nachfolgend vorgestellt werden. Es lassen sich zwei wesentliche Ziele von Verlinkungstechniken unterscheiden: (1) Die automatische Verlinkung und (2) die automatische Linkanalyse. Da die Techniken über einem Verlinkungsmodell operieren, stellt sich die Frage, ob die bestehenden Ansätze in der Lage sind, das WvT-Problem zu lösen. Es wird sich herausstellen, dass zwar noch keine Techniken existieren, um das WvT-Problem strukturell zu lösen. Einige der Analysetechniken lassen sich jedoch adaptieren, um die gelösten WvT-Probleme zu bewerten.

Linkorientierung im WvT-Problem. Die Einordnung in den Forschungsstand verdeutlicht die Besonderheit des WvT-Problems. Trotz seiner Einfachheit beschreibt es Vorwärts-, Rückwärts-, Inter- und Extra-Links. Das WvT-Problem enthält zwei Links: einen Wv-Link und einen T-Link. Der Wv-Link zeigt von einer Ziel-Anforderung zu einer Quell-Anforderung. Da er von einer Anforderung zu einer anderen Anforderung zeigt, ist er ein Inter-Link. Gleichzeitig ist er ein Rückwärtslink, weil er den Ursprung der Ziel-Anforderung verdeutlicht. Der T-Link zeigt von einem Testfall zu einer Quell-Anforderung. Damit ist er ein Vorwärts- und ein Extra-Link.

Forschungsstand: Verlinkungsmodelle. Verlinkungsmodelle heißen in der englischsprachigen Literatur *traceability information models*. Sie definieren verlinkbare Artefakte und den Typ der beteiligten Links [Pin04, S.106]. Auch das WvT-Problem ist ein Verlinkungsmodell. Das Konzept der Verlinkungsmodelle wurde in der Softwaretechnik bereits sehr früh aufgegriffen. Die ersten Modelle erschienen bereits in den 1980er Jahren. Die folgenden beiden Abschnitte stellen die relevanten Verlinkungsmodelle der letzten 25 Jahre vor.

Allgemeine Verlinkungsmodelle ohne explizite T-Links. Das SODOS Modell [HW86] stellt strukturierte Spezifikationsdokumente durch ein relationales Datenbankschema dar. Die Verlinkungen zwischen den Einträgen einer Datenbanktabelle sind frei definierbar. SODOS ist damit grundsätzlich in der Lage, alles mit allem zu verlinken. In den frühen 1990er Jahren galt der Trend dem Hypertext. Die Idee war es, Anforderungen und andere Softwareentwicklungsartefakte in Hypertext zu spezifizieren und durch die damals modernen Hyperlinks zu verknüpfen. Stellvertreter für Hypertext-Verlinkungsmodelle sind das IBIS-Modell [CB87], das REMAP-Modell [RD92], das RETH-Modell [Kai93], das HYDRA-Modell [PH95] und das TOORS-Modell [PG96]. Die Hyperlinks sind vorwiegend generisch, so dass auch hier grundsätzlich alles mit allem verlinkt werden kann. Einen anderen Weg schlägt das Contribution-Structures-Modell [GF95] ein. Obwohl es auch ein Hypertext-Verlinkungsmodell ist, beschränkt es sich auf die Verlinkung von Anforderungen mit Rollen, die im Softwareentwicklungsprozess auftreten können. Um die Jahrtausendwende gab es einen Wandel von Hypertext- hin zu UML-Verlinkungsmodellen, welche Inter- und Extra-Links zwischen UML-Modellen definieren [TM00] [Let02]. Das Studium des Forschungsstands erlaubt die Aussage, dass ältere Verlinkungsmodelle allgemeiner und neuere Verlinkungsmodelle spezialisierter sind. Während sich einige neuere Arbeiten auf die Beleuchtung von Inter-Links zwischen Anforderungen konzentrieren [Nar10], fokussieren sich andere auf Extra-Links zwischen Anforderung und Design [TKTW09] oder zwischen Anforderungen und Quelltextfragmenten [Sal06]. Obwohl es möglich ist, T-Links in den allgemeinen Verlinkungsmodellen zu definieren, fehlt allen bisher vorgestellten Modellen die explizite Fokussierung auf T-Links.

Verlinkungsmodelle mit T-Links. Es existieren neuere Verlinkungsmodelle, in denen T-Links explizit berücksichtigt werden. Ibrahim et al. beschreiben das totale Verlinkungsmodell, das Anforderungen (A), Testfälle (T), Design (D) und Quelltext (Q) jeweils paarweise miteinander verbindet [IMD05]. Den Anspruch der Totalität erhebt das Modell wegen der paarweisen Verlinkung und den dadurch entstehenden Extra-Links A-T, A-D, A-Q, T-D, T-Q und D-Q. Das totale Verlinkungsmodell enthält zudem die Inter-Links D-D und Q-Q. Asuncion et al. gehen mit ihrem Ende-zu-Ende-Verlinkungsmodell einen ähnlich totalen Weg [AFT07]. Ihr Verlinkungsmodell basiert auf einem Produktmodell, das Marketinganforderungen (M), Anwendungsfälle (A), funktionale Anforderungen (F) und Testfälle (T) verknüpft. Das Verlinkungsmodell enthält die Extra-Links M-A, A-F und F-T. Kirova et al. stellen ein Verlinkungsmodell vor, das die Verlinkung zwischen Anforderungen, Design und Testfällen beschreibt [KKKC08]. Ihr Verlinkungsmodell basiert auf den unterschiedlichen Anforderungsklassen Performance-Anforderung (PA), High-Level-Anforderungen (HLA), Architekturanforderungen (AA) und Systemanforderungen (SA). Zudem existiert im Modell das High-Level-Design (HLD) und das Low-Level-Design (LLD). Der Softwaretest wird schließlich durch Testfälle (T) und den Testplan (TP) repräsentiert. Das Modell von Kirova et al. ermöglicht die Extra-Links PA-AA, PA-SA, PA-T, HLA-SA, AA-SA, AA-T, AA-TP, AA-HLD, AA-LLD, SA-T, SA-TP, SA-HLD und SA-LLD. Azri und Ibrahim stellten zuletzt ein Metamodell vor, das die Verlinkung zwischen beliebigen Spezifikationsdokumenten, Quelltext, Rollen und sogar Ausgaben von entwicklungsbegleitenden Werkzeugen beschreibt [AI11].

Besonderheit des WvT-Problems. In den verwandten Arbeiten wird versucht, ein umfassendes Verlinkungsbild der Softwareentwicklung zu zeichnen, indem möglichst viele Artefakttypen und Linktypen berücksichtigt werden. Das WvT-Problem ist hingegen ein spezialisiertes Verlinkungsmodell, das sich auf einen einfachen Sachverhalt beschränkt. Mit den T-Links verfügt es über die Möglichkeit, das Verhältnis zwischen Anforderungen und Testfällen explizit auszudrücken. Gleichzeitig führt das WvT-Problem eine Neuerung ein: Die Darstellung der Anforderungswiederverwendung durch Wv-Links.

Forschungsstand: Verlinkungsmethoden und -techniken. Die Requirements Tracing Matrix (RTM) war eine der ersten Techniken, um systematisch mit der Verlinkung umzugehen [MV86]. Sie bildet eine Tabelle aus Anforderungen und den zu verlinkenden Artefakten, in der jede Zelle einen potentiellen Link darstellt. Gängige Anforderungsmanagementwerkzeuge, wie beispielsweise [DOORS], unterstützen die RTM. Die RTM ist ein sehr allgemeines Ausdrucksmittel, um die Verlinkung zu unterstützen. Modernere Arbeiten beschäftigen sich mit der automatischen Durchführung und Bewertung der Verlinkung. Im aktuellen Stand der Forschung findet die Unterstützung der Verlinkung ereignisbasiert (engl.: event-based), kostenbasiert (engl.: value-based), szenariobasiert (engl.: scenario-based), variabilitätsmodellbasiert (engl.: feature model-based), regelbasiert (engl.: rule-based), zielbasiert (engl.: goal-based) oder auf der Grundlage von Informationsrückgewinnung (engl. information retrieval) statt [RWA07, S.2f] [TGF+12, S.31].

Ereignisbasierte Verlinkung (EBV). Die EBV [CHCC03] führt einen Ereignisdienst ein, bei dem sich beliebige zu verlinkende Artefakte anmelden. Die Artefakte sind nun nicht mehr direkt über einen Link, sondern über den Ereignisdienst verbunden. Das Ziel der EBV ist es, die Linkwartung zu unterstützen, indem Ereignisse ausgelöst werden, sobald sich ein Artefakt ändert.

Regelbasierte Verlinkung (RBV). Die RBV [SZPMK04] wendet grammatikalische und lexikalische Regeln an, um in Spezifikationsdokumenten nach miteinander potentiell in Beziehung stehenden Artefakten zu suchen und diese zu verlinken. Das Ziel der RBV ist es, möglichst viele Links zwischen Textdokumenten, strukturierten Anwendungsfällen und UML-ähnlichen Klassendiagrammen automatisch herzustellen.

Variabilitätsmodellbasierte Verlinkung (VBV). Die VBV [Rie04] verwendet Feature-Modelle und setzt das Feature als verbindendes Element zwischen Anforderungen und Entwurfsmodellen sowie Anforderungen und Quelltext ein. Die VBV unterstützt die Verlinkung, indem sie die Verlinkungssituation hinsichtlich verschiedener Konsistenzkriterien prüft. Beispielsweise wird geprüft, ob jede Anforderung mindestens einem Feature zugewiesen ist.

Kostenbasierte Verlinkung (KBV). Die KBV [HB05] basiert auf der Annahme, dass die vollständige Verlinkung und Linkwartung aller miteinander in Beziehung stehenden Artefakte sehr kostenintensiv ist. Die KBV berücksichtigt daher priorisierte Anforderungen mit Hilfe von unterschiedlich präzisen Verlinkungsschemen. Das Ziel der KBV ist die Unterscheidung zwischen Links mit nachweislichem Nutzen und Links, die lediglich Kosten verursachen.

Szenariobasierte Verlinkung (SBV). Die SBV [EG05] verwendet Testszenarien, z.B. dargestellt durch Pfade in Zustandsdiagrammen, die mit Anforderungen und Quelltextfragmenten verlinkt sind. Die Verlinkung und Bewertung findet über die Analyse des Quelltextes statt: Zwei Anforderungen können verlinkt werden, wenn sie beide über ein Szenario mit sich überlappenden Quelltextfragmenten verlinkt sind. Die SBV ist damit in der Lage, unvollständige Verlinkungssituationen automatisch zu vervollständigen.

Zielbasierte Verlinkung (ZBV). Die ZBV [CHSB$^+$05] verwendet ein Qualitätsmodell, das nichtfunktionale Qualitätsziele, wie beispielsweise Benutzbarkeit und Wartbarkeit, definiert. Über eine Operationalisierung der nichtfunktionalen Ziele wird eine Verbindung zu den funktionalen Anforderungen hergestellt. Das Ziel der ZBV ist es, die Auswirkungen von funktionalen Änderungen auf die nichtfunktionalen Anforderungen nachzuvollziehen.

Information-Retrieval-basierte Verlinkung (IRV). Die IRV rückte in den letzten Jahren immer weiter in den Fokus des Forschungsgebiets der automatischen Verlinkung. Es existieren zahlreiche Herangehensweisen, um die zu verlinkenden Artefakte zu identifizieren [OGPD10]. Die generelle Funktionsweise jeder IRV-Technik ist es, miteinander in Beziehung stehende Anforderungs- bzw. Artefaktpaare auf der Grundlage von Ähnlichkeitsvergleichen zu finden und diese miteinander zu verlinken.

Automatische Verlinkung und Linkanalyse. In den verwandten Arbeiten wird entweder auf die Linkverfolgung oder die Ähnlichkeit zwischen Artefakten zurückgegriffen, um neue Links zu setzen. Während der Analyse der Verlinkungssituation werden Links verfolgt, um vorwiegend die Auswirkungen von Änderungen zu verfolgen.

Abgrenzung vom Forschungsstand. Die EBV propagiert Anforderungsänderungen zu den verlinkten Artefakten. Obwohl die automatische Verlinkung nach Anforderungswiederverwendung interessant ist, spielt die EBV für die Konzepte zur Lösung WvT-Problems zunächst keine Rolle. Die RBV verwendet grammatikalische und lexikalische Analysen, um ähnliche Anforderungen zu finden. Die RBV wird zur Lösung des WvT-Problems nicht zwingend benötigt, da davon ausgegangen wird, dass die Wv-Links im WvT-Problem bereits existieren. Wegen der Wichtigkeit von Variabilität in der Automobilbranche widmet sich Cmyrev in einer eigenständigen Arbeit der VBV [CNHR13]. Die KBV versucht, die Anzahl der zu verwaltenden Links einzuschränken, indem Anforderungen priorisiert werden. Das WvT-Problem ist ein spezialisiertes Verlinkungsmodell, das einen einfachen Sachverhalt darstellt. Diese bewusste Einfachheit des Verlinkungsmodells soll gerade ermöglichen, das vollständige Verlinkungsbild zu beleuchten. Die Einschränkung des WvT-Problems mittels Priorisierung durch eine KBV wird daher nicht angestrebt. Die SBV führt mit Szenarien ein zusätzliches Verbindungsartefakt ein, das Anforderungen miteinander verbindet. Das WvT-Problem soll gelöst werden, ohne zusätzliche Artefakte einzuführen. Daher rückt auch die SBV aus dem Kreis der näheren verwandten Arbeiten.

Erweiterung und Adaption des Forschungsstands. Das WvT-Problem ist ein spezialisiertes Verlinkungsmodell, das Wv-Links einführt. Die Wv-Links ermöglichen es, Testfälle, die mit einer wiederverwendeten Quell-Anforderung verlinkt sind, auch mit der entsprechenden Ziel-Anforderung zu verlinken. Diese als WvT-Verlinkung bezeichnete Technik erweitert den Forschungsstand um die wiederverwendungsbasierte (engl.: reuse-based) Verlinkung [NHK14].

Die Basisschicht der nachfolgend beschriebenen 3-schichtigen Methode zur automatischen Lösung des WvT-Problems setzt die T-Links auf der Grundlage der Wv-Links. Die zweite Schicht der Methode orientiert sich an der ZBV (zielbasierte Verlinkung), um während der Verlinkung die Testplanungsergebnisse aus dem Testkonzept zu berücksichtigen. Die dritte Schicht verwendet IRV (Information-Retrieval-basierte Verlinkung), um in einer Falldatenbank mittels Ähnlichkeitsvergleichen nach früheren WvT-Problemen zu suchen.

3.4. Lösungsansatz und Ziele der Arbeit

Die Methode zur automatischen Lösung des WvT-Problems verlinkt Testfälle mit Anforderungen. Während dieser Verlinkung werden Filtermechanismen eingesetzt, um Inkonsistenzen zu erkennen und somit die Genauigkeit der Verlinkung zu verbessern. Bild 3.4 zeigt den Aufbau der Methode. In der ersten Schicht werden zunächst alle Links durch die WvT-Verlinkung neu gesetzt. Zusätzlich werden strukturelle Inkonsistenzen entdeckt, die sich aus dem Vorhandensein oder Nichtvorhandensein von Wv- und T-Links ergeben. In der zweiten Schicht werden testplanungsrelevante Inkonsistenzen erkannt. Eine Form der Inkonsistenz ist gegeben, wenn das Testkonzept Testfälle fordert, die nicht mit den Ziel-Anforderungen verlinkt sind. Eine weitere Form tritt auf, wenn Testfälle mit Ziel-Anforderungen verlinkt sind, die das Testkonzept nicht fordert. Die dritte Schicht greift auf Mechanismen aus dem Bereich des fallbasierten Schließens (engl.: Case-Based Reasoning) zurück. Dabei wird in einer Falldatenbank zur Lösung aktueller WvT-Probleme nach ähnlichen bereits gelösten WvT-Problemen gesucht.

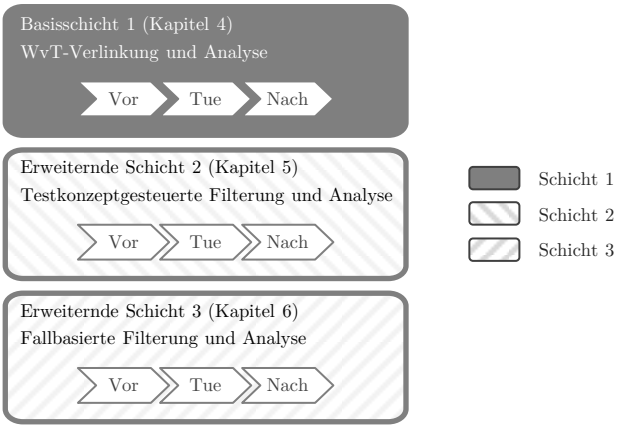


Bild 3.4.: Lösungsarchitektur (grob)

In jeder Schicht werden drei Phasen abgearbeitet: Vorbereitung (Vor), Durchführung (Tue) und Nachbereitung (Nach). In der Vorbereitungsphase werden die Daten aus den Spezifikationsdokumenten extrahiert, die für die Bearbeitung benötigt werden. In der Durchführungsphase werden Testfälle mit Ziel-Anforderungen verlinkt und/oder gefiltert. In der Nachbereitungsphase wird die Gesamtverlinkungssituation bewertet, um diejenigen Testfälle aufzudecken, die manuell geprüft werden müssen.

Bild 3.5 zeigt die Spezifikationsdokumente, die an der 3-schichtigen Methode beteiligt sind. WvT-Probleme treten auf, weil die Anforderungen und Testfälle in den SLH_{Quell}, SLH_{Ziel} und *STS* verlinkt sind. Im *Testkonzept* wird definiert, welche Eigenschaften Testfälle besitzen müssen, um Ziel-Anforderungen ausreichend abzusichern. In der *Falldatenbank* werden Regeln über den Eigenschaften von Anforderungen und Testfällen definiert, um Ähnlichkeitsvergleiche zwischen WvT-Problemen zu ermöglichen.

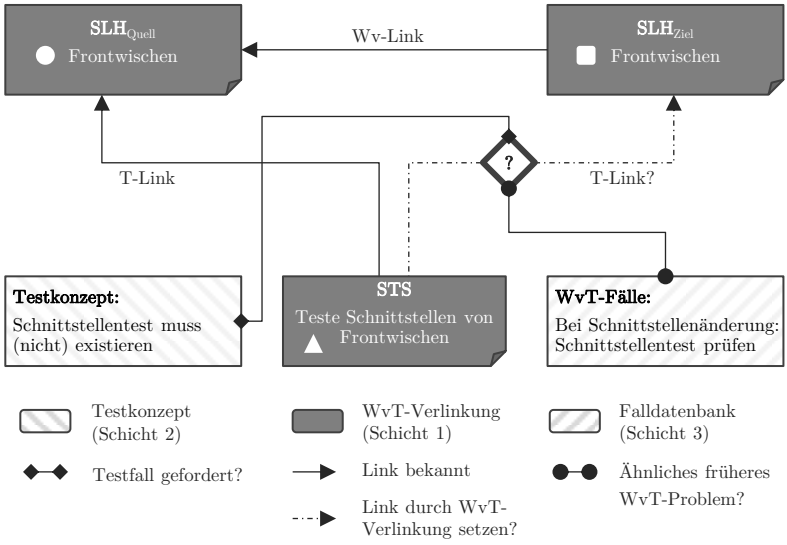


Bild 3.5.: Lösungsarchitektur (mit beteiligten Dokumenten)

Schicht 1: Grundidee (Kapitel 4). WENN eine Ziel-Anforderung und eine Quell-Anforderung durch einen Wv-Link verbunden sind UND WENN ein Testfall durch einen T-Link mit der Quell-Anforderung verbunden ist, DANN kann der Testfall auch mit der Ziel-Anforderung durch einen neuen T-Link verbunden werden. Nachdem alle Testfälle mit den Ziel-Anforderungen verlinkt worden sind, wird die Gesamtverlinkungssituation bewertet. Dabei sollen strukturelle Inkonsistenzen aufgedeckt werden. Diese Inkonsistenzen entstehen beispielsweise, wenn Testfälle zusätzliche Quell-Anforderungen - und somit Funktionen - absichern, die in der Ziel-Baureihe nicht wiederverwendet wurden. Derart verlinkte Testfälle ließen sich in der Ziel-Baureihe nicht ausführen.

Schicht 2: Grundidee (Kapitel 5). Im TK wird definiert, welche Testfälle mit welchen Anforderungen verlinkt sein müssen. Nachdem alle Verlinkungen durchgeführt wurden, wird die Gesamtverlinkungssituation des SLH_{Ziel} gemäß TK bewertet. Angenommen, der im Bild 3.5 dargestellte Testfall ist ein Schnittstellentestfall. Wenn im TK festgelegt ist, dass kein Schnittstellentest durchzuführen ist, dann sollten auch entsprechende T-Links nicht existieren. Fordert das TK wiederum den Schnittstellentest, dann müssen auch entsprechende T-Links existieren. Wenn kein entsprechender T-Link existiert, wurde ein fehlender Testfall entdeckt und somit eine Testlücke erkannt.

Schicht 3: Grundidee (Kapitel 6). Wenn WvT-Probleme fragwürdig sind, müssen sie manuell überprüft werden. Es kommt häufig vor, dass die Anforderungen und Testfälle von fragwürdigen WvT-Problemen über ähnliche Eigenschaften verfügen. Damit ähnliche WvT-Probleme durch die fallbasierte Filterung erkannt werden können, wird ein WvT-Fall in einer Falldatenbank abgespeichert. Ein WvT-Fall beschreibt die Eigenschaften eines WvT-Problems und kombiniert diese mit einem Prüfhinweis. Die Falldatenbank in Bild 3.5 enthält einen vereinfachten WvT-Fall. Der WvT-Fall besagt, dass die Verlinkung eines Schnittstellentestfalls manuell geprüft werden muss, wenn sich die Schnittstellen von der Quell-Anforderung zur Ziel-Anforderung verändert haben. Jedes WvT-Problem, bei dem sich die Schnittstellen der Anforderungen geändert haben, kann somit als fragwürdig eingestuft werden.

Ziele und Herausforderungen. In der Praxis treten WvT-Probleme millionenfach pro Baureihenprojekt auf. Dabei ergeben sich Herausforderungen, weil die beteiligten Testfälle in der Regel mit vielen verschiedenen Anforderungen verlinkt sind. Das Hauptziel dieser Arbeit ist daher die Entwicklung einer Technik, der sogenannten WvT-Verlinkung, die das WvT-Problem automatisch löst und dabei strukturelle Inkonsistenzen aufdeckt. Diese Inkonsistenzen beschreiben Situationen, in denen Testfälle zwar korrekt mit den Quell-Anforderungen, jedoch unkorrekt mit den Ziel-Anforderungen verlinkt sind. Neben diesen strukturellen Inkonsistenzen führt die Testplanung gemäß ISO 26262 zu weiteren Herausforderungen, da zusätzliche Inkonsistenzen auftreten können. Es muss geprüft werden, ob die richtigen Testfälle im Sinne der Vollständigkeit und Minimalität verlinkt sind. Sowohl die Vollständigkeit als auch die Minimalität leitet sich dabei aus den Forderungen des Testkonzepts ab. Zudem findet die Verlinkung von Testfällen und Ziel-Anforderungen nicht einmalig, sondern wiederkehrend in jedem Baureihenprojekt statt. Dabei treten auch WvT-Probleme mit ähnlichen spezifischen Eigenschaften wiederkehrend auf. Um die ähnlichen WvT-Probleme automatisch erkennen zu können, müssen Ähnlichkeitsfunktionen definiert werden.

Zusammenfassung. In diesem Kapitel wurden die Spezifikationsdokumente *Systemlastenheft*, *Systemtestspezifikation* und *Testkonzept* sowie ihr Zusammenwirken vorgestellt. Aus diesen Zusammenhängen wurde das *WvT-Problem* herausgearbeitet. Das WvT-Problem und dessen Lösung gliedern sich in den Bereich der Verlinkungsmodelle und in den Bereich der automatischen Verlinkung und Linkanalyse ein. Es konnte gezeigt werden, dass der Wv-Link zwischen Anforderungen in Verlinkungsmodellen aktuell noch nicht verbreitet ist. Daher existieren im Forschungsstand auch noch keine Techniken, die das WvT-Problem direkt lösen können.

Im weiteren Verlauf dieser Arbeit werden verschiedene Techniken vorgestellt, die das WvT-Problem lösen. Diese Techniken werden, wie in diesem Kapitel beschrieben, in einer 3-schichtige Methode angeordnet. Die nächsten drei Kapitel stellen jeweils eine Schicht im Detail vor: WvT-Verlinkung (Kap. 4), Filterung gemäß Testkonzept (Kap. 5) und fallbasierte Filterung (Kap. 6).

4. WvT-Verlinkung

Ein modernes Fahrzeug ist ein Verbund aus Systemen. Diese Systeme werden durch Anforderungen beschrieben, die entweder wiederverwendet oder neu spezifiziert werden. Dabei ist es besonders wichtig, dass die realisierten Systeme alle Anforderungen korrekt umsetzen. Um dies nachvollziehen zu können, müssen die Anforderungen mit entsprechenden Testfällen verlinkt sein.

Dieses Kapitel behandelt die Basisschicht der 3-schichtigen Methode zur automatischen Verlinkung. Es beginnt mit einem vereinfachten Beispiel, das neben WvT-Problemen auch ein Beispiel für eine WvT-Inkonsistenz zeigt. Nach diesem grundlegenden Beispiel erfolgt die Beschreibung des WvT-Problems und der WvT-Verlinkung mit Hilfe von Mengen und den Beziehungen der Elemente zwischen ihnen. Die Beziehungen zwischen dem SLH_{Quell}, dem SLH_{Ziel} und der STS führen zu dem WvT-Diagramm, mit dessen Hilfe die Spezifikationsdokumente hinsichtlich der Wv- und T-Links visualisiert werden. Danach erfolgt die Definition der WvT-Inkonsistenzen und ihre Integration in das WvT-Diagramm. Abschließend zeigen Feldstudien mit Hilfe des Diagramms, dass die automatische WvT-Verlinkung in realen SLH und STS effektiver als das bisherige manuelle Verlinkungsvorgehen ist.

4.1. Beispiel

Bild 3.1 auf Seite 30 zeigt WvT-Probleme, die sich stark an der Realität orientieren. Echte Systemanforderungen und Systemtestfälle können jedoch sehr umfangreich sein. Für den weiteren Verlauf dieser Arbeit wird daher auf ein abstraktes Beispiel zurückgegriffen, das es erlaubt, die einzelnen Konzepte in den Kapiteln einleitend und übersichtlich vorzustellen. Da die Beispiele in [DOORS] dargestellt werden, vermitteln sie zudem einen Eindruck, wie die WvT-Verlinkung praxistauglich realisiert werden kann. Die beiden Kapitel 5 und 6 bauen auf dem folgenden Beispiel auf, um in die für die jeweils beschriebene Methodenschicht relevanten Aspekte einzuführen.

Quell-Systemlastenheft (SLH_{Quell})

Quell-Anforderungen	Wv	T
1 Funktion Q		
Q 1	Z1	Ja
Q 2	Z2	Ja
Q 3	Z3	Ja
Q 4: Wird wegfallen		Ja

Systemtestspezifikation (STS)

Testfälle	T-Links: Vorher	T-Links: Nachher
1 Tests		
Test 1	Q1	Q1 , Z1
Test 2	Q2	Q2 , Z2
Test 3/4	Q3 Q4	Q3 , Z3 Q4

Ziel-Systemlastenheft (SLH_{Ziel})

Ziel-Anforderungen	Wv	T?	Ähnl.	Inkons.	WvT-Prüfhinweis
1 Funktion Z					
Z 1: Gleich	Q1	Ja	100		
Z 2: Fast gleich	Q2	Prüfen	80		- Textuelle Änderung
Z 3: Ungleich	Q3	Prüfen	60	II_Q	- Textuelle Änderung - Test 3/4 verlinkt mit Q4 (weggefallen)
Z 5: Neu		Nein			

Bild 4.1.: Einführendes Beispiel in [DOORS]

Phase 1: WvT-Probleme extrahieren. Bild 4.1 zeigt die Spezifikationsdokumente SLH_{Quell}, STS und SLH_{Ziel}. Beide SLH beschreiben Fahrzeugfunktionen, die durch Anforderungen konkretisiert werden. Das SLH_{Quell} enthält die Quell-Anforderungen Q_i. Das SLH_{Ziel} enthält die Ziel-Anforderungen Z_j. Die Spalte *Wv* zeigt in beiden SLH, auf welche Quell- bzw. Ziel-Anforderung die Wv-Links zeigen. Die Spalte *T* im SLH_{Quell} zeigt an, ob ein Testfall mit einer Quell-Anforderung verlinkt ist. Die STS enthält *Testfälle*. Die Spalte *T-Links: Vorher* zeigt, mit welchen Quell-Anforderungen die Testfälle bereits vor der WvT-Verlinkung verlinkt waren. Bild 4.1 enthält drei WvT-Probleme mit den folgenden Wv-Links ($\rightarrow_{Wv}$) und T-Links ($\rightarrow_{T}$):

$$
\begin{aligned}
Z_1 \rightarrow_{Wv} Q_1 \text{ und } Test_1 &\rightarrow_{T} Q_1 \\
Z_2 \rightarrow_{Wv} Q_2 \text{ und } Test_2 &\rightarrow_{T} Q_2 \\
Z_3 \rightarrow_{Wv} Q_3 \text{ und } Test_{3/4} &\rightarrow_{T} Q_3
\end{aligned}
$$

Phase 2: WvT-Verlinkung durchführen. Die WvT-Verlinkung unterliegt der folgenden Annahme: WENN eine Ziel-Anforderung eine Quell-Anforderung wiederverwendet UND WENN ein Testfall eine Quell-Anforderung absichert, DANN sichert der Testfall auch die Ziel-Anforderung ab. Vor der Durchführung der WvT-Verlinkung sind die Spalten *T-Links: Nachher* der STS sowie *T?* und *Prüfhinweis* des SLH_{Ziel} noch nicht ausgefüllt. Nach der WvT-Verlinkung enthält die Spalte *T-Links: Nachher* der STS die Namen der Quell- und Ziel-Anforderungen, mit denen ein Testfall durch einen T-Link verbunden ist. Die Spalte *T?* des SLH_{Ziel} zeigt an, ob eine Ziel-Anforderung mit einem Testfall verbunden ist. Sie wird auf *Prüfen* gesetzt, wenn sich die Ziel-Anforderung textuell geändert hat oder wenn WvT-Inkonsistenzen aufgedeckt wurden. Die Aufdeckung der Inkonsistenzen geschieht in der dritten Phase der ersten Methodenschicht. Wie Bild 4.1 zeigt, führt die Durchführung der WvT-Verlinkung in der zweiten Phase zu drei gelösten WvT-Problemen:

$$
\begin{aligned}
Z_1 \rightarrow_{Wv} Q_1 \text{ und } Test_1 &\rightarrow_{T} Q_1 \text{ führt zu } Test_1 \rightarrow_{T} Z_1 \\
Z_2 \rightarrow_{Wv} Q_2 \text{ und } Test_2 &\rightarrow_{T} Q_2 \text{ führt zu } Test_2 \rightarrow_{T} Z_2 \\
Z_3 \rightarrow_{Wv} Q_3 \text{ und } Test_{3/4} &\rightarrow_{T} Q_3 \text{ führt zu } Test_{3/4} \rightarrow_{T} Z_3
\end{aligned}
$$

Phase 3: Verlinkung bewerten. Das einführende Beispiel aus Bild 4.1 enthält bereits einige interessante Aspekte. Die Spalte *Ähnlichkeit (Ähnl.)* zeigt im SLH_{Ziel} in Prozent an, wie stark sich der Text einer Ziel-Anforderung vom Text der wiederverwendeten Quell-Anforderung unterscheidet. Die textuelle Ähnlichkeit kann mit Hilfe bekannter Algorithmen wie Dice, Jaro-Winkler oder Levenshtein ermittelt werden [SimMetrics]. Wenn der Ähnlichkeitswert einen konfigurierbaren Grenzwert unterschreitet, der im Beispiel bei 100% liegt, wird die Spalte *T?* des SLH_{Ziel} auf *Prüfen* gesetzt. Im Rahmen der Verlinkungsbewertung wird in der Spalte *Prüfhinweis* des SLH_{Ziel} vermerkt, warum die Spalte *T?* einer Ziel-Anforderung auf *Prüfen* gesetzt wurde.

Neben der Ähnlichkeit der Anforderungstexte werden auch fehlende Testfälle - sogenannte Testlücken - entdeckt. In der ersten Methodenschicht sind Testlücken genau die Ziel-Anforderungen, die mit keinem Testfall verlinkt sind. Im Beispiel wird in der Spalte *T?* zunächst lediglich vermerkt, dass die neue Ziel-Anforderung Z_5 noch keinen Testfall besitzt. Die weiteren Methodenschichten erlauben eine detailliertere Bewertung.

Der interessanteste Aspekt des Beispiels ist, dass die Quell-Anforderung Q_4 nicht im SLH_{Ziel} wiederverwendet wurde und dass der Testfall $Test_{3/4}$ sowohl Q_3 als auch Q_4 absichert. Während der WvT-Verlinkung wird der Testfall $Test_{3/4}$ korrekt mit der Ziel-Anforderung Z_3 verlinkt, da Z_3 mit Q_3 über einen Wv-Link und Q_3 mit $Test_{3/4}$ über einen T-Link verbunden ist. Der Testfall $Test_{3/4}$ sichert aber eine Anforderung ab, die im SLH_{Ziel} nicht mehr existiert. Dies ist eine WvT-Inkonsistenz.

Ein Beispiel verdeutlicht die WvT-Inkonsistenz II_Q (S.60f): Angenommen, die Quell-Anforderungen Q_{Front} und Q_{Heck} beschreiben das Front- und Heckwischen der M-Klasse. Beide Quell-Anforderungen werden durch $Test_{Front+Heck}$ abgesichert. Weil die S-Klasse über keinen Heckscheibenwischer verfügt, existiert zwar eine Ziel-Anforderung Z_{Front} aber kein entsprechendes Ziel-Pendant für Q_{Heck}. Die WvT-Verlinkung verbindet nun korrekterweise $Test_{Front+Heck}$ mit Z_{Front}. Die WvT-Inkonsistenz II_Q entsteht, weil der Testfall Funktionalität absichert, die in der S-Klasse gar nicht existiert: das Heckwischen.

4.2. WvT-Verlinkung

Das WvT-Problem basiert auf den folgenden Grundmengen:

$$\begin{aligned} \mathrm{SLH}_{\mathrm{Quell}} &:= \{a_q \mid a_q \text{ ist Quell-Systemanforderung}\} \\ \mathrm{SLH}_{\mathrm{Ziel}} &:= \{a_z \mid a_z \text{ ist Ziel-Systemanforderung}\} \\ \mathrm{STS} &:= \{t \mid t \text{ ist Systemtestfall}\} \end{aligned}$$

Der obere linke Kreis im Bild 4.2 symbolisiert das $\mathrm{SLH}_{\mathrm{Quell}}$, d.h. die Menge aller Quell-Systemanforderungen. Der obere rechte Kreis stellt analog dazu das $\mathrm{SLH}_{\mathrm{Ziel}}$, also die Menge aller Ziel-Systemanforderungen, dar. Der untere Kreis zeigt die STS, d.h. die Menge aller Systemtestfälle. Die drei Grundmengen sind disjunkt. Da die Elemente jedoch über Wv- und T-Links verbunden sein können, können die im Bild dargestellten Überlappungen auftreten.

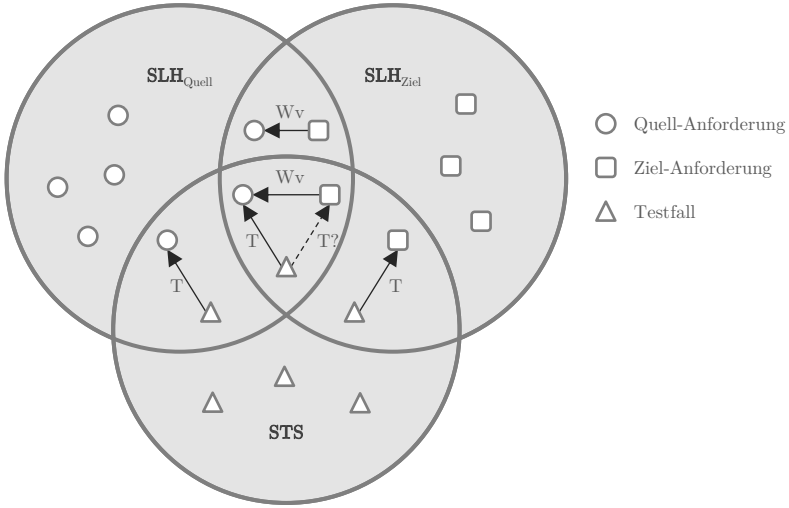


Bild 4.2.: Grundmengen des WvT-Problems

Wv-Links zwischen Anforderungen. Ein Wv-Link $a_Z \rightarrow_{Wv} a_Q$ zeigt immer von einer Ziel-Anforderung a_Z zu einer Quell-Anforderung a_Q. Eine wiederverwendende Ziel-Anforderung aus dem SLH_{Ziel} zeigt somit auf eine wiederverwendete Quell-Anforderung aus dem SLH_{Quell}. Dabei können sowohl Ziel-Anforderungen als auch Quell-Anforderungen mehrere ausgehende bzw. eingehende Wv-Links haben. Mehrere ausgehende Wv-Links treten auf, wenn mehrere Quell-Anforderungen zu einer Ziel-Anforderung zusammengefasst werden. Mehrere eingehende Wv-Links werden durch die Auftrennung einer Quell-Anforderung zu mehreren Ziel-Anforderungen verursacht.

$$
\begin{aligned}
Wv &:= \{(a_Z, a_Q) \in SLH_{Ziel} \times SLH_{Quell} \mid a_Z \rightarrow_{Wv} a_Q\} \\
Wv_Q &:= \{a_Q \in SLH_{Quell} \mid \exists a_Z : (a_Z, a_Q) \in Wv\} \\
Wv_Z &:= \{a_Z \in SLH_{Ziel} \quad \mid \exists a_Q : (a_Z, a_Q) \in Wv\}
\end{aligned}
$$

Die Menge Wv besteht aus Tupeln (a_Z, a_Q). Die beiden Anforderungen des Tupels sind durch einen Wv-Link verbunden: (a_Z, a_Q) ist somit ein Synonym für $a_Z \rightarrow_{Wv} a_Q$. Das Tupel (a_Z, a_Q) wird auch als Wiederverwendungspaar bezeichnet. Die Mengen Wv_Q und Wv_Z enthalten jeweils alle Quell- und Ziel-Anforderungen, die Teil eines Wiederverwendungspaars sind. Während Wv_Q also alle wiederverwendeten Quell-Anforderungen von SLH_{Quell} enthält, enthält Wv_Z alle wiederverwendenden Ziel-Anforderungen von SLH_{Ziel}.

T-Links von Testfällen zu Anforderungen. Ein T-Link $t \rightarrow_T a$ zeigt von einem Testfall zu einer Anforderung. Ein Testfall kann durch jeweils einen ausgehenden T-Link mit mehreren Anforderungen verlinkt sein. Eine abgesicherte Anforderung hat mindestens einen eingehenden T-Link. Je nachdem, ob ein Testfall in das SLH_{Quell} und/oder das SLH_{Ziel} zeigt, sichert er Quell-Anforderungen, Ziel-Anforderungen oder beides ab.

$$
\begin{aligned}
T(a) &:= \{t \in STS \mid t \rightarrow_T a \wedge (a \in SLH_{Quell} \vee a \in SLH_{Ziel})\} \\
T_Q &:= \{a_Q \in SLH_{Quell} \mid \exists t \in STS : t \rightarrow_T a_Q\} \\
T_Z &:= \{a_Z \in SLH_{Ziel} \quad \mid \exists t \in STS : t \rightarrow_T a_Z\}
\end{aligned}
$$

Die Menge T(a) enthält alle Testfälle, die mit der Anforderung a verlinkt sind. Die Menge T_Q enthält alle Quell-Anforderungen a_Q eines SLH_{Quell}, für die mindestens ein T-Link $t \rightarrow_T a_Q$ von einem Testfall t aus der STS zu eben diesen Anforderungen a_Q existiert. Analog dazu enthält die Menge T_Z alle Ziel-Anforderungen a_Z, die mit mindestens einem Testfall t verbunden sind.

WvT-Verlinkung. Die WvT-Verlinkung basiert auf der folgenden Annahme: WENN eine Ziel-Anforderung a_Z und eine Quell-Anforderung a_Q durch einen Wv-Link miteinander verbunden sind UND WENN ein Testfall t mit der Quell-Anforderung a_Q durch einen T-Link verbunden ist, DANN kann der Testfall t auch mit der Ziel-Anforderung a_Z durch einen neuen T-Link verbunden werden. Der folgende Ausdruck beschreibt die WvT-Verlinkung:

$$a_Z \rightarrow_{Wv} a_Q \wedge t \rightarrow_T a_Q \Rightarrow t \rightarrow_T a_Z$$

Eine konkrete WvT-Verlinkung, d.h. die Lösung eines der WvT-Probleme des einführenden Beispiels aus Bild 4.1 von Seite 48 würde wie folgt dargestellt:

$$Z_1 \rightarrow_{Wv} Q_1 \wedge Test_1 \rightarrow_T Q_1 \Rightarrow Test_1 \rightarrow_T Z_1$$

WvT-Instanzen und WvT-Typen. Eine WvT-Instanz ist eine Verlinkungssituation zwischen einem bis drei Artefakten (z.B. $\{a_Z, a_Q, t\}$). Sie wird als Menge einer Quell-Anforderung und/oder einer Ziel-Anforderung und/oder eines Testfalls, mindestens jedoch aus einem der Artefakte dargestellt. Für eine WvT-Instanz gilt, dass jedes enthaltene Artefakt über einen Link mit einem anderen Artefakt der Instanz verbunden sein muss. Es gibt auch Instanzen mit nur einem Artefakt, das somit über keinen Link zu einem anderen Artefakt verfügen kann. Anhand der Wv- und T-Links können die Instanzen zu WvT-Typen zugeordnet werden. Im Bild 4.2 von Seite 51 wird dies durch die Segmente symbolisiert. Jedes einzelne WvT-Problem ist eine WvT-Instanz des Typs *Ziel-Anforderung mit wiederverwendeter, getesteter Quell-Anforderung.* Neben diesem WvT-Typ existieren weitere Typen wie beispielsweise *nicht getestete und nicht wiederverwendete Quell-Anforderung.* Nachfolgend werden die einzelnen WvT-Typen mit Hilfe des WvT-Diagramms vorgestellt.

4.3. WvT-Diagramm

Das WvT-Diagramm in Bild 4.3 stellt die WvT-Typen dar. Jedes Diagrammsegment entspricht einem WvT-Typ. Die einzelnen Segmente des Diagramms entstehen wegen der Existenz bzw. Nichtexistenz von Wv-Links und/oder T-Links zwischen den Elementen der drei Grundmengen $\mathrm{SLH}_{\mathrm{Quell}}$, STS und $\mathrm{SLH}_{\mathrm{Ziel}}$. Die Ausdrücke in den Segmenten benennen den entsprechenden WvT-Typ. Diagramme können statt den Ausdrücken auch Zahlen enthalten, die die Anzahl WvT-Instanzen und damit die Verlinkungssituation ausdrücken.

Die Bilder 4.4a, 4.4b und 4.4c zeigen jeweils eine der drei Grundmengen $\mathrm{SLH}_{\mathrm{Quell}}$, STS und $\mathrm{SLH}_{\mathrm{Ziel}}$. Für jede der drei Mengen gilt, dass die WvT-Instanzen mindestens das Artefakt der jeweiligen Grundmenge enthalten: Jede Instanz eines Segments in Bild 4.4a enthält daher eine Quell-Anforderung. Analog dazu enthalten die Instanzen in den Segmenten der Bilder 4.4b und 4.4c mindestens Testfälle bzw. Ziel-Anforderungen.

WvT-Typen im WvT-Diagramm. Die Ausdrücke in den Segmenten entsprechen den Definition von Seite 52. Der Ausdruck Wv steht für die Menge aller Wv-Paare. Die Menge aller Quell-Anforderungen ohne Wv-Link wird im Diagramm als $\overline{Wv_Q}$ dargestellt. Der Ausdruck $\overline{Wv_Z}$ beschreibt die Menge aller Ziel-Anforderungen, die keinen Wv-Link besitzen.

Der Ausdruck T_Q spiegelt die Menge aller Quell-Anforderungen wider, die durch einen T-Link mit einem Testfall verbunden sind. Im Gegensatz dazu entspricht $\overline{T_Q}$ der Menge aller Quell-Anforderungen, die nicht mit einem Testfall verlinkt sind. Die Mengen T_Z und $\overline{T_Z}$ sind analog dazu für Ziel-Anforderungen definiert. Die Mengen $\overline{T(a_Q)}$ und $\overline{T(a_Z)}$ enthalten alle Testfälle, die nicht mit Quell- bzw. Ziel-Anforderungen verlinkt sind.

Der Ausdruck $\overline{T_{Q*Z}}$ beschreibt die Menge aller Wv-Paare, in denen weder die Quell- noch die Ziel-Anforderung mit einem Testfall verlinkt ist. Im Gegensatz dazu entspricht der Ausdruck T_{Q+Z} der Menge aller Wv-Paare, in denen mindestens eine der beiden Anforderungen mit einem Testfall verbunden ist.

Das Diagramm zeigt die Schnittmengen zwischen den Wv- und T-Mengen.

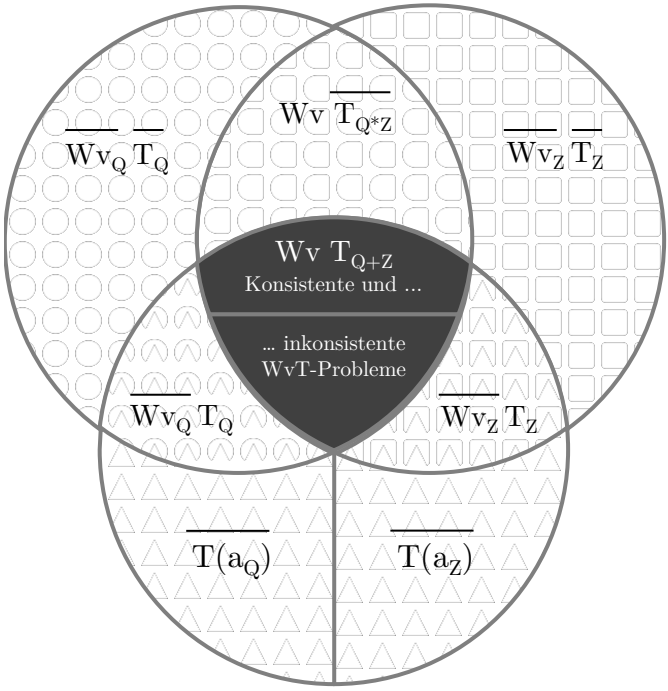
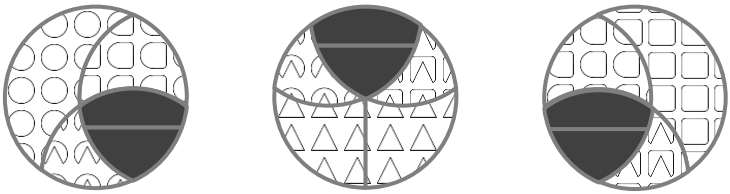


Bild 4.3.: WvT-Typen im WvT-Diagramm



(a) SLH_{Quell} **(b)** STS **(c)** SLH_{Ziel}

Bild 4.4.: Grundmengen des WvT-Diagramms

Haben keinen T-Link (○,▢,▢ : $\overline{\mathbf{T_Q}}$ oder $\overline{\mathbf{T_Z}}$). Die in Bild 4.5a dargestellten Segmente enthalten alle WvT-Instanzen, die eine Quell-Anforderung und/ oder Ziel-Anforderung und keinen Testfall enthalten. Damit enthält die Menge alle Anforderungen, die mit keinem Testfall verlinkt sind.

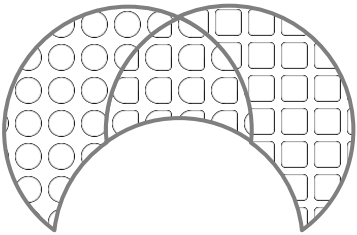
Haben weder Wv-Link noch T-Link (○,▢ : $\overline{\mathbf{Wv_Q}}\,\overline{\mathbf{T_Q}}$ oder $\overline{\mathbf{Wv_Z}}\,\overline{\mathbf{T_Z}}$). Die beiden Segmente in Bild 4.5b enthalten alle WvT-Instanzen, die entweder eine Quell- oder eine Ziel-Anforderung und keinen Testfall enthalten. Sie enthält somit alle nicht in einem Wiederverwendungsverhältnis stehenden Anforderungen, die zudem mit keinem Testfall verlinkt sind.

Haben Wv-Link und keinen T-Link (▢ : $\mathbf{Wv}\,\overline{\mathbf{T_{Q*Z}}}$). Das Segment in Bild 4.5c enthält die WvT-Instanzen, die zwar ein Wv–Paar aber keinen Testfall enthalten. Dies entspricht allen in einem Wiederverwendungsverhältnis stehenden Anforderungen, die mit keinem Testfall verlinkt sind.

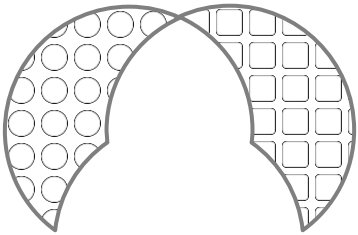
Haben T-Link (◠,⬟,◠ : $\mathbf{T_Q}$ oder $\mathbf{T_Z}$). Bild 4.5d steht für die WvT-Instanzen mit einem Testfall sowie mindestens einer Anforderung. Dies sind alle mit Testfällen verlinkten Anforderungen - unabhängig vom Wv-Link.

Haben Wv-Link und T-Link (⬟ : $\mathbf{Wv}\,\mathbf{T_{Q+Z}}$). Bild 4.5e zeigt das Diagrammzentrum und zugleich den Bereich, in dem die WvT-Verlinkung wirkt. Das Zentrum enthält alle WvT-Instanzen, die sowohl Quell- als auch Ziel-Anforderung und einen Testfall enthalten. Der Testfall besitzt hierbei einen T-Link zu mindestens einer der beiden Anforderungen. Damit werden neben den ungelösten auch die gelösten WvT-Probleme in das Zentrum eingeordnet. Das Zentrum enthält zudem all jene Wv-Paare, bei denen lediglich die Ziel-Anforderung über einen T-Link verfügt. Da WvT-Probleme wegen Abhängigkeiten zu anderen Instanzen inkonsistent sein können, ist das Segment zweigeteilt: Es enthält oben konsistente und unten inkonsistente WvT-Probleme. Die WvT-Inkonsistenzen sind Gegenstand des nächsten Unterkapitels 4.4.

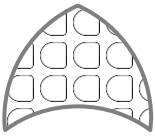
Haben keinen Wv-Link aber T-Link (◠,◠ : $\overline{\mathbf{Wv_Q}}\,\mathbf{T_Q}$ oder $\overline{\mathbf{Wv_Z}}\,\mathbf{T_Z}$). Bild 4.5f zeigt alle WvT-Instanzen, die einen Testfall sowie entweder eine Quell- oder eine Ziel-Anforderung enthalten. Dies sind alle Anforderungen, die nicht in einem Wv-Verhältnis stehen, aber mit Testfällen verlinkt sind.



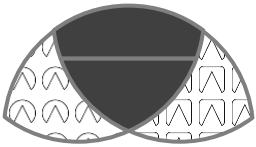
(a) $\overline{\mathrm{T_Q}}$ oder $\overline{\mathrm{T_Z}}$



(b) $\overline{\mathrm{Wv_Q}}\,\overline{\mathrm{T_Q}}$ oder $\overline{\mathrm{Wv_Z}}\,\overline{\mathrm{T_Z}}$



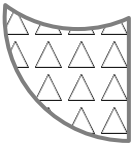
(c) $\mathrm{Wv}\,\overline{\mathrm{T_{Q*Z}}}$



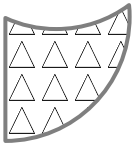
(d) $\mathrm{T_Q}$ oder $\mathrm{T_Z}$

(e) $\mathrm{Wv}\,\mathrm{T_{Q+Z}}$

(f) $\overline{\mathrm{Wv_Q}}\,\mathrm{T_Q}$ oder $\overline{\mathrm{Wv_Z}}\,\mathrm{T_Z}$



(g) $\overline{\mathrm{T(a_Q)}}$



(h) $\overline{\mathrm{T(a_Z)}}$

Bild 4.5.: Segmente im WvT-Diagramm

Haben keinen T-Link ins SLH (Δ : $\overline{\mathbf{T(a_Q)}}$ oder $\overline{\mathbf{T(a_Z)}}$). Die Bilder 4.5g und 4.5h zeigen schließlich noch die WvT-Instanzen, die lediglich Testfälle enthalten. Die Testfälle haben keine T-Links zu Quell- und/oder Ziel-Anforderungen. Im Kontext des entsprechenden SLH sind sie nicht verlinkt.

Anzahl der WvT-Instanzen. Jedes Diagrammsegment enthält alle WvT-Instanzen eines WvT-Typs. In Bild 4.6 wird deutlich, dass die WvT-Instanzen auf unterschiedliche Art gezählt werden müssen, weil sie je nach WvT-Typ Quell- und/oder Ziel-Anforderungen und/oder Testfällen enthalten.

Anzahl$_{\mathbf{QZT}}$. Die Anzahl$_{\mathrm{QZT}}$:= $((\#\mathrm{Wv_Q} : \#\mathrm{Wv_Z}), \#\mathrm{T_Q}, \#\mathrm{T_Z})$ wird im Diagrammzentrum dargestellt. Der erste Wert $(\#\mathrm{Wv_Q} : \#\mathrm{Wv_Z})$ entspricht der Anzahl der Quell- und Ziel-Anforderungen, die über einen Wv-Link miteinander verbunden sind. Die beiden Werte $\#\mathrm{Wv_Q}$ und $\#\mathrm{Wv_Z}$ können sich unterscheiden, weil Quell- und Ziel-Anforderungen aufgetrennt oder zusammengefasst werden können. Der zweite Wert $\#\mathrm{T_Q}$ ist die Anzahl aller T-Links, die von Testfällen zu den Quell-Anforderungen zeigen. Analog dazu ist der dritte Wert $\#\mathrm{T_Z}$ die Anzahl aller T-Links, die zu den Ziel-Anforderungen zeigen.

Anzahl$_{\mathbf{Q\overline{Z}T}}$, Anzahl$_{\mathbf{\overline{Q}T}}$ und Anzahl$_{\mathbf{\overline{Z}T}}$. Das Segment direkt über dem Zentrum enthält alle Wv-Paare, auf die keine Testfälle zeigen. Daher ist Anzahl$_{\mathrm{Q\overline{Z}T}}$:= $(\#\mathrm{Wv_Q} : \#\mathrm{Wv_Z})$. Die beiden Segment links und rechts über dem Zentrum enthalten völlig unverlinkte Anforderungen. Die Anzahl$_{\mathrm{\overline{Q}T}}$:= $\#\mathrm{Wv_Q}$ besteht daher nur aus einem einzelnem Wert, der die Anzahl der Quell-Anforderungen ohne Wv- und T-Links anzeigt. Die Anzahl$_{\mathrm{\overline{Z}T}}$ ist analog dazu definiert.

Anzahl$_{\mathbf{\overline{Q}T}}$ und Anzahl$_{\mathbf{\overline{Z}T}}$. Die Segmente links und rechts neben dem Zentrum enthalten Quell- bzw. Ziel-Anforderungen ohne Wv-Link aber mit T-Link. Die Quell-Anforderungen werden mit Anzahl$_{\mathrm{\overline{Q}T}}$:= $(\#\overline{\mathrm{Wv_Q}}, \#\mathrm{T_Q})$ quantifiziert. Die Werte $\#\overline{\mathrm{Wv_Q}}$ und $\#\mathrm{T_Q}$ entsprechen der Anzahl der nicht wiederverwendeten Quell-Anforderungen und der Anzahl der T-Links zu diesen Anforderungen. Die Anzahl$_{\mathrm{\overline{Z}T}}$ ist analog für Ziel-Anforderungen definiert.

Anzahl$_{\mathbf{\overline{T}}}$. Die beiden unteren Segmente enthalten die Testfälle, die nicht in das $\mathrm{SLH_{Quell}}$ bzw. $\mathrm{SLH_{Ziel}}$ zeigen. In diesen Segmenten wird die Anzahl der unverlinkten Testfälle mit jeweils einem Wert repräsentiert.

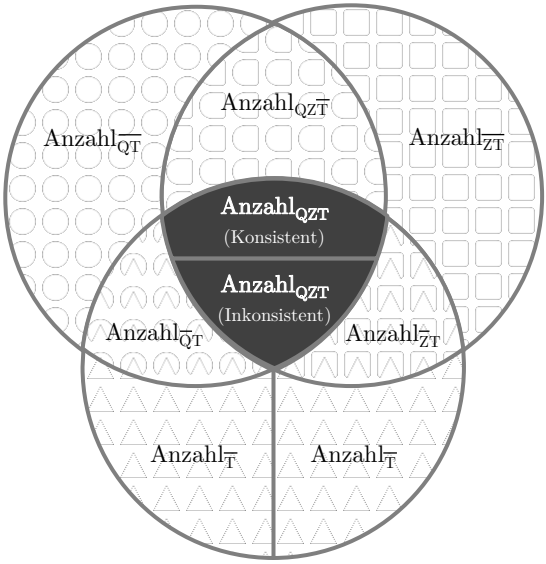


Bild 4.6.: Anzahlen im WvT-Diagramm

Nutzen des WvT-Diagramms. Mittelgroße SLH_{Quell} und SLH_{Ziel} enthalten bereits Anforderungen im vierstelligen Bereich. Zusätzliche enthalten STS Testfälle im vier- bis fünfstelligen Bereich. Die Quell- und Ziel-Anforderungen sind eventuell untereinander und eventuell mit Testfällen verlinkt. Das WvT-Diagramm stellt diese Vielzahl verschiedener Verlinkungssituationen übersichtlich dar. In diesem Kapitel wird das WvT-Diagramm noch eine wichtige Rolle spielen. Im Rahmen der Feldstudie visualisiert es, dass die WvT-Verlinkung effektiver als das bisherige manuelle Vorgehen ist (S. 69).

WvT-Inkonsistenzen im Diagrammzentrum. Das Diagrammzentrum enthält die (un)gelösten WvT-Probleme, die konsistent oder inkonsistent sein können. Im nächsten Unterkapitel werden verschiedene fragwürdige Verlinkungssituation definiert, die als WvT-Inkonsistenzen bezeichnet werden. Im Anschluss daran folgen die Feldstudien, in denen das WvT-Diagramm Einsatz findet.

4.4. WvT-Inkonsistenzen

WvT-Inkonsistenzen beschreiben fragwürdige Verlinkungssituationen. Die Ursache von WvT-Inkonsistenzen sind fehlende T-Links oder Beziehungen zwischen WvT-Problemen und anderen WvT-Instanzen. Weitere Inkonsistenzen ergeben sich, wenn Anforderungen im Rahmen der Wiederverwendung aufgetrennt oder zusammengefasst werden. Die folgenden Regeln I_Q, I_Z, II_Q, II_Z, III_Q und III_Z beschreiben konsistente Verlinkungssituationen. Eine Inkonsistenz tritt auf, wenn eine Konsistenzregel nicht erfüllt ist. Die Konsistenzregeln sind auf Wv-Paare (a_Z, a_Q) anwendbar, wobei mindestens eine der beiden Anforderungen mit einem Testfall verlinkt sein muss:

$$
\begin{aligned}
I_Q(a_Z, a_Q) &:= T(a_Q) \subseteq T(a_Z) \\
I_Z(a_Z, a_Q) &:= T(a_Z) \subseteq T(a_Q) \\
II_Q(a_Z, a_Q) &:= \forall t \in T(a_Z) \cup T(a_Q) : \forall a'_Q \in SLH_{Quell} : a'_Q \neq a_Q \Rightarrow \\
&\qquad (t \rightarrow_T a'_Q \Rightarrow \exists a'_Z \in SLH_{Ziel} : t \rightarrow_T a'_Z \wedge (a'_Z, a'_Q) \in Wv) \\
II_Z(a_Z, a_Q) &:= \forall t \in T(a_Z) \cup T(a_Q) : \forall a'_Z \in SLH_{Ziel} : a'_Z \neq a_Z \Rightarrow \\
&\qquad (t \rightarrow_T a'_Z \Rightarrow \exists a'_Q \in SLH_{Quell} : t \rightarrow_T a'_Q \wedge (a'_Z, a'_Q) \in Wv) \\
III_Q(a_Z, a_Q) &:= \nexists a'_Z \in SLH_{Ziel} : a_Z \neq a'_Z : (a'_Z, a_Q) \in Wv \\
III_Z(a_Z, a_Q) &:= \nexists a'_Q \in SLH_{Quell} : a_Q \neq a'_Q : (a_Z, a'_Q) \in Wv
\end{aligned}
$$

Konsistenzregel I (fehlender T-Link). Es existieren zwei Ausprägungen der ersten Konsistenzregel: I_Q und I_Z. Die erste Ausprägung $I_Q(a_Z, a_Q)$ fordert, dass die Menge aller Testfälle, die auf a_Q zeigen, eine Teilmenge der Testfälle ist, die auch auf a_Z zeigen. Neben der Teilmengenaussage $T(a_Q) \subseteq T(a_Z)$ gibt es eine weitere Möglichkeit, die Konsistenzregel I_Q auszudrücken: Jeder Testfall t, der mit einer Quell-Anforderung a_Q eines Wiederverwendungspaars (a_Z, a_Q) verbunden ist, muss auch mit der Ziel-Anforderung a_Z verbunden sein. Die zweite Regelausprägung I_Z ist analog zu I_Q definiert: Alle Testfälle, die auf die Ziel-Anforderung a_Z zeigen, müssen auch auf die wiederverwendete Quell-Anforderung a_Q zeigen. Ein Wiederverwendungspaar (a_Z, a_Q) ist konsistent I, wenn beide Regelausprägungen I_Q und I_Z erfüllt sind.

Konsistenzregel II (kein Wv-Paar). Testfälle sind meistens nicht nur mit einer, sondern mit mehreren Anforderungen verlinkt. Ein Wv-Paar (a_Z, a_Q) ist konsistent II, wenn jeder Testfall, der neben (a_Z, a_Q) auch auf andere Anforderungen a'_Z und a'_Q zeigt, immer auf beide Partner eines Wv-Paars (a'_Z, a'_Q) zeigt. Damit ist ein Wv-Paar konsistent II, wenn ein Testfall neben dem Wv-Paar (a_Z, a_Q) nicht auch verworfene oder neue Anforderungen absichert.

Die zweite Konsistenzregel hat die beiden Ausprägungen II_Q und II_Z. Angenommen, ein Testfall zeigt nicht nur auf das a_Q des Wiederverwendungspaars (a_Z, a_Q), sondern auch zusätzlich auf andere Quell-Anforderungen a'_Q des SLH_{Quell}. Das Wiederverwendungspaar (a_Z, a_Q) ist konsistent II_Q, wenn jeder Testfall, der auf eine Quell-Anforderung a'_Q zeigt, auch auf die entsprechende Ziel-Anforderung a'_Z zeigt, so dass gilt, dass (a'_Z, a'_Q) ein Wiederverwendungspaar bildet. Die Regelausprägung II_Z ist wieder analog zu II_Q definiert. Ein Wv-Paar ist konsistent II, wenn sowohl II_Q als auch II_Z gilt.

Konsistenzregel III (Auftrennung/Zusammenfassung). Anforderungen können im Rahmen der Wiederverwendung aufgetrennt oder zusammengefasst werden. Dies ist anhand der Anzahl der ein- und ausgehenden Wv-Links erkennbar. Wenn eine Quell-Anforderung in mehrere Ziel-Anforderungen aufgetrennt wird, dann zeigen mehrere eingehende Wv-Links auf die Quell-Anforderung. Wenn mehrere Quell-Anforderungen in einer Ziel-Anforderung zusammengefasst werden, dann hat die Ziel-Anforderung mehrere ausgehende Wv-Links. Ein Wv-Paar (a_Z, a_Q) ist konsistent III, wenn beide Anforderungen nur einen eingehenden bzw. ausgehenden Link haben.

Regel III hat die beiden Ausprägungen III_Q und III_Z. Wenn zwei Wv-Paare (a_{Z1}, a_Q) und (a_{Z2}, a_Q) existieren, dann hat die Quell-Anforderung a_Q zwei eingehende Wv-Links von a_{Z1} und a_{Z2}. Daher wurde a_Q aufgetrennt und beide Paare sind inkonsistent III_Q. Regel III_Z ist analog definiert.

Inkonsistenz und vollständige Konsistenz. Die Negation einer Konsistenzregel wird als Inkonsistenz bezeichnet. Ein Wiederverwendungspaar ist vollständig konsistent, wenn $I_Q \wedge I_Z \wedge II_Q \wedge II_Z \wedge III_Q \wedge III_Z$ gilt.

Inkonsistenz I_Q und I_Z. Das Wiederverwendungspaar (a_{Z1}, a_{Q1}) aus Bild 4.7a ist inkonsistent I_Q, da $t \in T(a_{Q1})$ und $t \notin T(a_{Z1})$ und somit $T(a_{Q1}) \not\subseteq T(a_{Z1})$. Im Bild wird dies anhand des gestrichelten T-Links von dem Testfall t zu der Quell-Anforderung a_{Q1} veranschaulicht. Dieser T-Link verursacht die Inkonsistenz I_Q, weil er exklusiv auf a_{Q1} zeigt. Diese Inkonsistenz tritt im Rahmen der WvT-Verlinkung nicht auf. Sie kommt aber im Rahmen des manuellen Vorgehens vor, wenn T-Links übersehen werden.

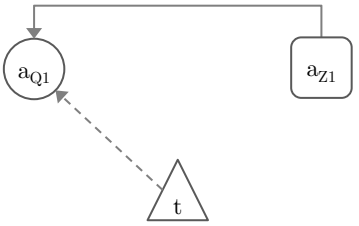
In Bild 4.7a wäre das Paar (a_{Z1}, a_{Q1}) inkonsistent I_Z, wenn der gestrichelt dargestellte T-Link zu a_{Z1} und nicht zu a_{Q1} zeigen würde. Die Inkonsistenz I_Z deckt Testfälle auf, die neu mit Ziel-Anforderungen verlinkt worden sind.

Inkonsistenz II_Q. Das Wiederverwendungspaar (a_{Z1}, a_{Q1}) in Bild 4.7b ist zunächst konsistent I, da beide Anforderungen T-Links zum selben Testfall t besitzen. Der Testfall t zeigt nicht nur auf die Quell-Anforderung a_{Q1}, sondern zusätzlich auf a_{Q2}. Das Wv-Paar (a_{Z1}, a_{Q1}) ist inkonsistent II_Q, da $t \rightarrow_T a_{Q2} \wedge \nexists a_{Z2} \in SLH_{Ziel}$, so dass $t \rightarrow_T a_{Z2} \wedge (a_{Z2}, a_{Q2}) \in Wv$. Im Bild wird dies durch den gestrichelten T-Link dargestellt, der auf eine zusätzliche Quell-Anforderung, aber auf keinen entsprechenden Partner im SLH_{Ziel} zeigt. Das Beispiel mit dem *verworfenen Heckwischer* von Seite 50 verdeutlicht II_Q.

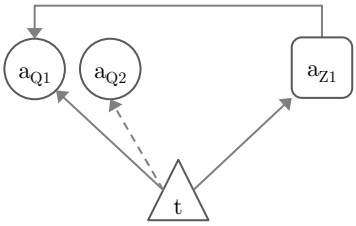
Bild 4.7b würde Inkonsistenz II_Z zeigen, wenn der gestrichelte T-Link nicht zu a_{Q2}, sondern zu a_{Z2} zeigen würde. Die Inkonsistenz II_Z ermittelt Testfälle, die neben wiederverwendeten auch neue Ziel-Anforderungen absichern.

Inkonsistenz $II_Q \wedge II_Z$. Das Paar (a_{Z1}, a_{Q1}) aus Bild 4.7c ist konsistent I. Die Anforderungen a_{Z2} und a_{Q2} bilden kein Wv-Paar. Wegen des gestrichelten T-Links $t \rightarrow_T a_{Q2}$ ist (a_{Z1}, a_{Q1}) inkonsistent II_Q. Der gestrichelte T-Link $t \rightarrow_T a_{Z2}$ verursacht für (a_{Z1}, a_{Q1}) die zusätzliche Inkonsistenz II_Z.

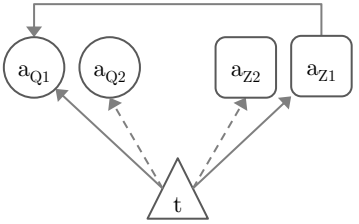
Inkonsistenz III_Q. Die beiden Paare (a_{Z1}, a_{Q1}) und (a_{Z2}, a_{Q1}) aus Bild 4.7d sind konsistent I und II. Weil die beiden Ziel-Anforderungen a_{Z1} und a_{Z2} die Quell-Anforderung a_{Q1} auftrennen, sind die beiden Paare inkonsistent III_Q. Das DOORS Beispiel auf Seite 30 deutet die Inkonsistenz III_Q an. Dort wurde die Beschreibung der *Waschunterbrechung* aus Anforderung *134* entfernt und an eine andere Stelle des SLH_{Ziel} kopiert.



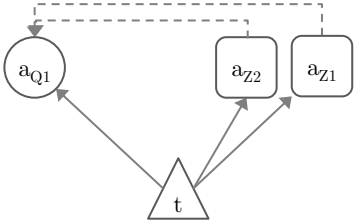
(a) Inkonsistenz I_Q



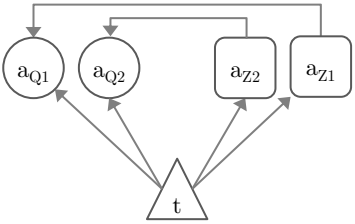
(b) Inkonsistenz II_Q



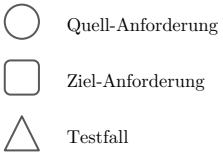
(c) Inkonsistenz $II_Q \wedge II_Z$



(d) Inkonsistenz III_Q



(e) Konsistenz I $\wedge$ II $\wedge$ III



(f) Legende

Bild 4.7.: Beispiele für Inkonsistenzen und Konsistenz

Konsistenz. Bild 4.7e zeigt zwei Wv-Paare (a_{Z1}, a_{Q1}) und (a_{Z2}, a_{Q2}). Die Partner beider Wv-Paare sind mit demselben Testfall verlinkt. Der Testfall zeigt auch auf keine Anforderung, die nicht Partner eines Wv-Paars ist. Zudem wurde keine der Anforderungen zusammengefasst oder aufgetrennt. Damit sind beide Wv-Paare konsistent.

Die Bilder 4.7c und 4.7e zeigen einen interessanten Aspekt. Der fehlende Wv-Link zwischen a_{Z2} und a_{Q2} in Bild 4.7c führt zur Inkonsistenz $II_Q \wedge II_Z$ von (a_{Z1}, a_{Q1}). In Bild 4.7e sind a_{Z2} und a_{Q2} ein Wv-Paar und beide Paare (a_{Z1}, a_{Q1}) und (a_{Z2}, a_{Q2}) sind konsistent. Das Auftreten einer Inkonsistenz $II_Q \wedge II_Z$ kann demnach auf potentiell fehlende Wv-Links hinweisen.

Verknüpfung der Konsistenzregeln. Die Inkonsistenz $II_Q \wedge II_Z$ verknüpft zwei negierte Konsistenzregeln. Jede einzelne Konsistenzregel kann für ein Wv-Paar erfüllt oder nicht erfüllt sein. Zudem kann jede Konsistenzregel unabhängig von den anderen Regeln für ein Wv-Paar auftreten. Dies ermöglicht die konjunktive Verknüpfung aller Konsistenzregeln. Tabelle 4.1 zeigt dies auf Seite 70 im Feldstudienabschnitt für die Regeln I_Q, I_Z, II_Q und II_Z.

Quell- und Ziel-Inkonsistenz. Quell-Inkonsistenzen decken fragwürdige Verlinkungssituationen auf. Die Inkonsistenz I_Q erfasst Testfälle, die exklusiv mit der Quell-Anforderung eines Wv-Paars verlinkt sind. Die Inkonsistenz II_Q erfasst Testfälle, die neben einem Wv-Paar zusätzlich nicht wiederverwendete Quell-Anforderungen absichern. Die Inkonsistenz III_Q erfasst Testfälle, die die Quell-Anforderung womöglich nur teilweise absichern und somit nur die Ziel-Anforderungen absichern können, die diese Teile wiederverwenden.

Die Ziel-Inkonsistenzen I_Z und II_Z decken Situationen auf, in denen im SLH_{Ziel} neue Links hinzugekommen sind. Die Inkonsistenz I_Z erfasst Testfälle, die neu mit der Ziel-Anforderung eines Wv-Paars verlinkt wurden. Die Inkonsistenz II_Z erfasst Testfälle, die neben einem Wv-Paar zusätzlich neue Ziel-Anforderungen absichern. Die Inkonsistenz III_Z erfasst Testfälle, die mit einer Ziel-Anforderung verlinkt sind, die aus mehreren Quell-Anforderungen zusammengefasst wurde. Testfälle, die mit einer der Quell-Anforderungen verlinkt sind, sichern auch die entsprechenden Teile der Ziel-Anforderung ab.

Obwohl I_Z, II_Z und III_Z Ziel-Inkonsistenzen genannt werden, führt ihr Vorkommen - vorausgesetzt sie treten nicht in Kombination mit I_Q, II_Q und/oder III_Q auf - nicht zu einer fragwürdigen Verlinkungssituation. Tritt beispielsweise eine Inkonsistenz I_Q auf, kann dies darauf hindeuten, dass ein T-Link übersehen wurde. Im Gegensatz dazu verdeutlicht eine pure Inkonsistenz I_Z lediglich, dass neue T-Links im SLH_{Ziel} hinzugekommen sind.

Darstellung im Zentrum des WvT-Diagramms. Die WvT-Konsistenzregeln sind auf Wv-Paare anwendbar, wobei mindestens ein Testfall mit mindestens einer der beiden Anforderungen verlinkt sein muss. Dies entspricht den gelösten und ungelösten WvT-Problemen. Das Zentrum des WvT-Diagramms enthält diese WvT-Probleme. Damit treten die WvT-Inkonsistenzen lediglich im Zentrum des WvT-Diagramms auf.

Bislang wurde im Diagrammzentrum zwischen konsistenten (oben) und inkonsistenten (unten) WvT-Problemen unterschieden. Die konsistent gelösten WvT-Probleme werden weiterhin oben dargestellt. Bild 4.8 zeigt die Unterscheidung zwischen den WvT-Problemen mit Quell- und Ziel-Inkonsistenzen. Der linke untere Bereich enthält alle WvT-Probleme, die mindestens eine fragwürdige Quell-Inkonsistenz haben. Der rechte untere Bereich enthält alle WvT-Probleme, die keine Quell-Inkonsistenz haben. Dies ermöglicht die Unterscheidung zwischen den „schlechten“ und „guten“ Inkonsistenzen.

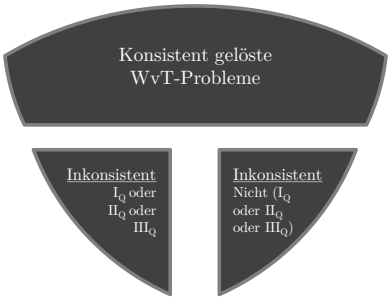


Bild 4.8.: Erweiterung des WvT-Diagrammzentrums

4.5. Feldstudie

Die automatische WvT-Verlinkung verlinkt Testfälle im vierstelligen Bereich inklusive der Linkanalyse in wenigen Minuten. Wenn diese Tätigkeit manuell durchgeführt werden muss, kann dies mehrere Tage dauern. Diese offensichtliche Effizienzsteigerung soll nicht Gegenstand der Feldstudie sein. Vielmehr soll evaluiert werden, ob die automatische Verlinkung zu einer vergleichbaren Verlinkungssituation wie das vorwiegend manuelle Vorgehen führt.

4.5.1. Ziel und Vorgehensweise

Die Automatisierung von kleinteiligen und oft wiederholten Arbeitsschritten kann sehr zuverlässige Ergebnisse liefern. Deswegen soll die Feldstudie zeigen, ob die WvT-Verlinkung nicht nur vergleichbar, sondern sogar genauer als das bisherige Vorgehen ist. Daher wird zunächst untersucht, ob alle Testfälle, die bereits vorher mit Ziel-Anforderungen verlinkt waren, auch durch die WvT-Verlinkung verlinkt werden. Zusätzlich wird untersucht, ob mehr T-Links existieren und weniger WvT-Inkonsistenzen auftreten.

Im Rahmen der Feldstudie wird auf eine DOORS-Produktivdatenbank zugegriffen. In dieser Datenbank existieren Spezifikationsdokumente, die gelöste WvT-Probleme enthalten. Die WvT-Verlinkung von Ziel-Anforderungen und Testfällen wurde teilweise manuell (Abk.: mnl) durchgeführt. Daher werden die Dokumente $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Quell}}$, $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Ziel}}$ und $\mathrm{STS}^{\mathrm{mnl}}$ genannt.

Die existierenden Dokumente werden mit allen Wv- und T-Links kopiert und umbenannt zu $\mathrm{SLH}^{\mathrm{auto}}_{\mathrm{Quell}}$, $\mathrm{SLH}^{\mathrm{auto}}_{\mathrm{Ziel}}$ und $\mathrm{STS}^{\mathrm{auto}}$. Im Anschluss daran werden alle T-Links gelöscht, die von der $\mathrm{STS}^{\mathrm{auto}}$ in das $\mathrm{SLH}^{\mathrm{auto}}_{\mathrm{Ziel}}$ zeigen. Schließlich wird die automatische WvT-Verlinkung durchgeführt, um die gerade gelöschten T-Links erneut zu setzen.

Wenn die Feldstudie zeigt, dass die existierenden T-Links auch nach der WvT-Verlinkung existieren, dann ist die automatische Verlinkung so genau wie das manuelle Vorgehen. Wenn die Feldstudie zeigt, dass mehr T-Links zu weniger WvT-Inkonsistenzen führen, ist die WvT-Verlinkung sogar genauer.

4.5.2. System A von Baureihe 1 nach Baureihe 2

In der Feldstudie soll die vorgefundene Verlinkungssituation mit der Situation nach der automatischen WvT-Verlinkung verglichen werden. Dazu wurde in der Produktivdatenbank nach Testspezifikationen gesucht, die T-Links in zwei SLH haben. Diese zwei SLH müssen zudem in einem Wiederverwendungsverhältnis stehen. Um die detaillierte Untersuchung handhabbar und übersichtlich zu gestalten, wurden in der DOORS-Produktivdatenbank SLH und STS mit wenigen Artefakten identifiziert. Die beiden $\mathrm{SLH}_{\mathrm{Quell}}$ (Baureihe 1) und $\mathrm{SLH}_{\mathrm{Ziel}}$ (Baureihe 2) sowie die STS des Systems A sind mit 292 Quell- und 296 Ziel-Anforderungen sowie 191 Testfällen ausreichend klein.

Vorbereitung der Spezifikationsdokumente. Die Verfolgbarkeit der Wiederverwendung mittels Wv-Links ist ein neues Konzept. Der Linktyp *Wv-Link* existiert im betrachteten Umfeld daher noch nicht. Aus diesem Grund müssen die Wv-Links nachträglich gesetzt werden. Dies wird realisiert, indem die internen IDs der Anforderungen von $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Quell}}$ und $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Ziel}}$ miteinander verglichen werden. Wenn zwei Anforderung die gleiche ID haben, werden sie mit einem Wv-Link verbunden.

Die vorgefundenen $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Quell}}$, $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Ziel}}$ und $\mathrm{STS}^{\mathrm{mnl}}$ wurden mit den nachträglich gesetzten Wv-Links und den vorhandenen T-Links kopiert und in $\mathrm{SLH}^{\mathrm{auto}}_{\mathrm{Quell}}$, $\mathrm{SLH}^{\mathrm{auto}}_{\mathrm{Ziel}}$ und $\mathrm{STS}^{\mathrm{auto}}$ umbenannt. Im Anschluss daran wurden alle T-Links in das $\mathrm{SLH}^{\mathrm{auto}}_{\mathrm{Ziel}}$ gelöscht.

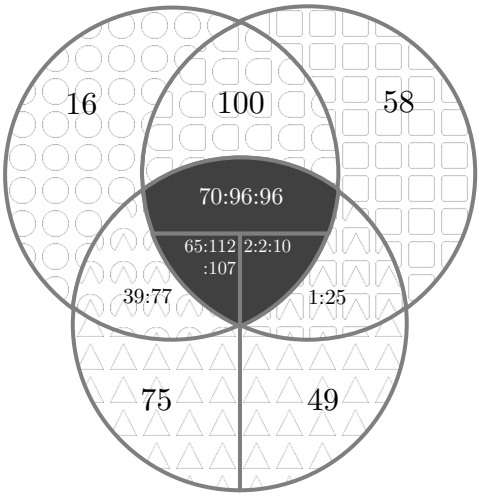
Sonderfall: Exklusiver T-Link zu Ziel-Anforderung. Die Löschung aller T-Links ins $\mathrm{SLH}^{\mathrm{auto}}_{\mathrm{Ziel}}$ führt zu einer Schwierigkeit: In der vorgefundenen Verlinkungssituation wurden Testfälle mit dem $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Ziel}}$ neu verlinkt, die zuvor nicht mit dem $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Quell}}$ verlinkt waren. Damit hat das $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Ziel}}$ nachträglich verlinkte Testfälle, die auch im $\mathrm{SLH}^{\mathrm{auto}}_{\mathrm{Ziel}}$ nachträglich verlinkt werden würden. Dies sind beispielsweise neue Testfälle oder Testfälle, die im Quell-Baureihenprojekt nicht benötigt werden. Um die Vergleichbarkeit hinsichtlich der Inkonsistenzen $\mathrm{I_Z}$ und $\mathrm{II_Z}$ von $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Ziel}}$ und $\mathrm{SLH}^{\mathrm{auto}}_{\mathrm{Ziel}}$ zu wahren, werden alle T-Links, die exklusiv in das $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Ziel}}$ zeigen, in das $\mathrm{SLH}^{\mathrm{auto}}_{\mathrm{Ziel}}$ übertragen.

WvT-Diagramme. Die WvT-Diagramme in den Bildern 4.9a und 4.9b visualisieren die vorgefundene Verlinkungssituation und die Situation nach der WvT-Verlinkung. Die Zahlen in den einzelnen Segmenten entsprechen der Anzahl der WvT-Instanzen eines WvT-Typs. Im Verlauf der Feldstudie werden die beiden Verlinkungssituationen anhand dieser Zahlen verglichen.

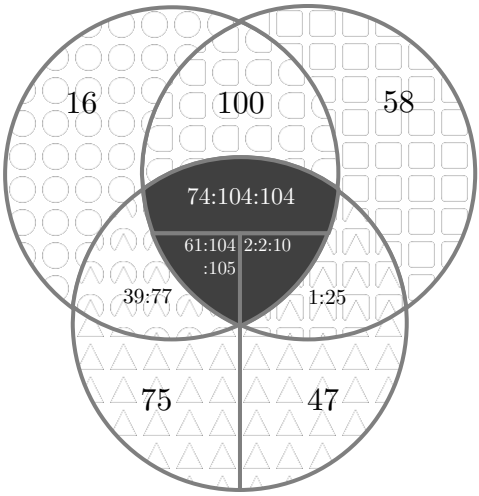
Besonderes Interesse gilt in der Feldstudie den Instanzen des Typs WvT-Problem. Das Diagrammzentrum enthält die WvT-Probleme. Die drei Zahlen (a:b:c) im Diagrammzentrum entsprechen (a) der Anzahl der Quell- und Ziel-Anforderungen mit Wv-Link und mit mindestens einem verlinkten Testfall, (b) der Anzahl der T-Links zu den Quell-Anforderungen und (c) der Anzahl der T-Links zu den Ziel-Anforderungen. Die Beschreibung der WvT-Typen und der Zahlendarstellungen im Diagramm erfolgte auf den Seiten 55ff.

Untersuchung der peripheren Bereiche. Bevor der detaillierte Vergleich der Diagrammzentren stattfindet, erfolgt die Untersuchung der peripheren Bereiche der beiden WvT-Diagramme. Da die WvT-Verlinkung die Verlinkungssituation zwischen einer STS und einem $\text{SLH}_{\text{Quell}}$ nicht ändert, bleiben die Anzahlen in allen peripheren Bereichen - bis auf eine Ausnahme - gleich. Die vorgefundene Verlinkungssituation ist dadurch gekennzeichnet, dass 49 Testfälle nicht mit dem $\text{SLH}^{\text{mnl}}_{\text{Ziel}}$ verlinkt sind. Im Gegensatz dazu zeigen nach der WvT-Verlinkung weniger Testfälle nicht in das $\text{SLH}^{\text{auto}}_{\text{Ziel}}$: Während also zuvor 49 Testfälle nicht in das $\text{SLH}^{\text{mnl}}_{\text{Ziel}}$ zeigten ($\Delta^{\text{mnl}}_{\text{Ziel}}$), zeigen gemäß Bild 4.9b nach der WvT-Verlinkung zwei Testfälle weniger, nämlich nun nur noch 47 Testfälle nicht in das $\text{SLH}^{\text{auto}}_{\text{Ziel}}$ ($\Delta^{\text{auto}}_{\text{Ziel}}$). Dies ist bereits ein erster Anhaltspunkt dafür, dass die WvT-Verlinkung effektiver als das bisherige Vorgehen ist.

Die anderen peripheren Bereichen sind nicht durch die WvT-Verlinkung betroffen. Beide Verlinkungssituationen enthalten 16 nicht wiederverwendete Quell-Anforderungen ohne Wv-Links und ohne T-Links (○), 100 Wv-Paare ohne T-Links (⊐) sowie 58 neue Ziel-Anforderungen ohne Wv-Links und ohne T-Links (□). Beide Diagramme zeigen 39 Quell-Anforderungen (⌒) und eine Ziel-Anforderung (⊓) ohne Wv-Link aber mit T-Links. 75 Testfälle sind nicht mit dem $\text{SLH}_{\text{Quell}}$ T-verlinkt (Δ_{Quelle}).



(a) Vorgefundene Verlinkungssituation



(b) Situation nach WvT-Verlinkung

Bild 4.9.: WvT-Diagramme für System A

(In)konsistenz	**(a)Vorgefunden**	**(b) Nach WvT**
0: Konsistent	70:96:96	74:104:104
1: I_Q	1:5:4	-
2: I_Z	2:2:10	2:2:10
3: $I_Q \wedge I_Z$	-	-
4: II_Q	56:92:92	57:94:94
5: $I_Q \wedge II_Q$	4:5:0	-
6: $I_Z \wedge II_Q$	1:2:3	1:2:3
7: $I_Q \wedge I_Z \wedge II_Q$	-	-
8: II_Z	-	-
9: $I_Q \wedge II_Z$	-	-
10: $I_Z \wedge II_Z$	-	-
11: $I_Q \wedge I_Z \wedge II_Z$	-	-
12: $II_Q \wedge II_Z$	3:8:8	3:8:8
13: $I_Q \wedge II_Q \wedge II_Z$	-	-
14: $I_Z \wedge II_Q \wedge II_Z$	-	-
15: $I_Q \wedge I_Z \wedge II_Q \wedge II_Z$	-	-

Tabelle 4.1.: (In)konsistenzen in den Diagrammzentren

Untersuchung der WvT-Inkonsistenzen. Tabelle 4.1 stellt die Zentren der beiden WvT-Diagramme 4.9a und 4.9b detaillierter dar. Sie zeigt analog zu den Diagrammzentren die Anzahl der Anforderungen und T-Links pro (In)konsistenz 0 bis 15. Die Inkonsistenzen ergeben sich aus der Verknüpfung der Konsistenzregeln I_Q, I_Z, II_Q und II_Z. Die Wv-Links wurden nachträglich auf der Grundlage identischer IDs automatisch gesetzt. Deswegen ist immer eine Quell-Anforderung mit einer Ziel-Anforderung verbunden (1-zu-1). Die Inkonsistenz III (Auftrennung/Zusammenfassung) tritt daher nie auf.

WvT-Diagramme und Tabelle 4.1. Um die Belastbarkeit der Feldstudie zu erhöhen, wurden für die Berechnung der Zahlen in den Diagrammzentren und der Zahlen in den Feldern der Tabelle 4.1 voneinander unabhängige DOORS DXL-Skripte implementiert. Um die Korrektheit der Feldstudiendaten mit Hilfe der doppelten Zählung zu plausibilisieren, müssen die folgenden Regeln erfüllt sein: Die Zahlen (a:b:c) im oberen Teil der Diagrammzentren entsprechen den Zahlen (a:b:c) der Zeile 0 der Tabelle. Das untere rechte Segment der Diagrammzentren enthält die Anzahl der exklusiven Ziel-Inkonsistenzen I_Z, II_Z sowie $I_Z \wedge II_Z$. Dies entspricht der Summe der Zahlen (a:b:c) der Zeilen 2, 8 und 10. Die Summe der Zahlen (a:b:c) aller übrigen Tabellenzeilen, d.h. aller Zeilen, in denen mindestens Inkonsistenz I_Q oder II_Q gilt, entspricht den Zahlen (a:b:c) des unteren linken Segments der Diagrammzentren.

Mit Hilfe dieser Daten wird nun gezeigt, dass die automatische Verlinkung nicht nur vergleichbar, sondern sogar besser als das bisherige Vorgehen ist.

Die gleichen Testfälle sind verlinkt. Zunächst wird die Effektivität der WvT-Verlinkung gezeigt. Sie ist effektiv, wenn mindestens die Testfälle mit Ziel-Anforderungen verlinkt wurden, die auch bereits vorher verlinkt waren.

Der Nachweis der Effektivität wird argumentativ über die nicht T-verlinkten Testfälle geführt. Das untere rechte Segment ($\Delta^{\text{auto}}_{\text{Ziel}}$) des WvT-Diagramms 4.9b enthält 47 Testfälle, die nach der WvT-Verlinkung nicht mit dem $\text{SLH}^{\text{auto}}_{\text{Ziel}}$ verlinkt sind. Das Segment ($\Delta^{\text{mnl}}_{\text{Ziel}}$) des WvT-Diagramms 4.9a enthält 49 Testfälle, die nicht mit dem $\text{SLH}^{\text{mnl}}_{\text{Ziel}}$ verlinkt sind. Eine Für-alle Schleife zeigte, dass alle unverlinkten Testfälle von $\Delta^{\text{auto}}_{\text{Ziel}}$ auch in $\Delta^{\text{mnl}}_{\text{Ziel}}$ enthalten sind.

Alle Testfälle, die in der vorgefundenen Verlinkungssituation mit dem $\text{SLH}^{\text{mnl}}_{\text{Ziel}}$ verlinkt waren, sind nach der automatischen WvT-Verlinkung auch mit dem $\text{SLH}^{\text{auto}}_{\text{Ziel}}$ verlinkt. Wäre dies nicht der Fall, dann müsste die Anzahl nicht verlinkter Testfälle in $\Delta^{\text{auto}}_{\text{Ziel}}$ größer sein als in $\Delta^{\text{mnl}}_{\text{Ziel}}$. Da dies wegen $47 < 49$ nicht gegeben ist, ist die automatische WvT-Verlinkung in System A effektiv.

Die Zählung in den beiden Diagrammzentren ergab 137 übereinstimmende Wv-Paare. Nachfolgend wird untersucht, ob die Ziel-Anforderungen in $\bullet^{\text{auto}}$ mindestens mit den gleichen Testfällen wie in $\bullet^{\text{mnl}}$ verlinkt sind.

Mehr Testfälle sind verlinkt, weniger Inkonsistenzen treten auf. Wenn die WvT-Verlinkung mehr Testfälle verlinkt und dabei zu weniger Inkonsistenzen führt, ist sie nicht nur effektiv, sondern sogar effektiver als das Vorgehen, das zu der vorgefundenen Verlinkungssituation führte.

Ein Blick in die Segmente ($\Delta_{\text{Ziel}}^{\text{mnl}}$) und ($\Delta_{\text{Ziel}}^{\text{auto}}$) der beiden WvT-Diagramme 4.9a und 4.9b belegt die bessere Effektivität hinsichtlich der Anzahl der verlinkten Testfälle. Zuvor wurde argumentiert, dass die gleichen Testfälle verlinkt sind. Wenn nach der WvT-Verlinkung die gleichen Testfälle verlinkt sind aber insgesamt weniger Testfälle nicht mit Ziel-Anforderungen verlinkt sind, dann müssen mehr Testfälle mit dem $\text{SLH}_{\text{Ziel}}^{\text{auto}}$ verlinkt sein.

Ein Blick in Zeile 0 von Tabelle 4.1 belegt, dass nach der automatischen Verlinkung mehr konsistent gelöste WvT-Probleme existieren. In der vorgefundenen Situation waren 70 Wv-Paare konsistent mit jeweils 96 T-Links zu beiden Anforderungen verlinkt. Nach der WvT-Verlinkung werden 74 Wv-Paare gezählt, die über 104 T-Links mit Testfällen verbunden sind. In System A ist die automatische Verlinkung damit effektiver als das bisherige Vorgehen.

Die Effektivitätssteigerung lässt sich auch in den beiden WvT-Diagrammen 4.9a und 4.9b anhand der Zentren $\bullet^{\text{mnl}}$ und $\bullet^{\text{auto}}$ zeigen. Der obere Teil des Zentrums enthält in $\bullet^{\text{auto}}$ nach der automatischen Verlinkung mehr konsistente WvT-Probleme als in $\bullet^{\text{mnl}}$. Gleichzeitig verringert sich durch die WvT-Verlinkung die Anzahl der inkonsistenten WvT-Instanzen im unteren linken Segment des Zentrums.

Herkunft der neuen Konsistenzen. Während $\bullet^{\text{mnl}}$ 70 konsistente und 65 inkonsistente WvT-Probleme enthält, enthält $\bullet^{\text{auto}}$ 74 konsistente und 61 inkonsistente WvT-Probleme. Der Übergang der vier Inkonsistenzen ist mit den WvT-Diagrammen oder mit Tabelle 4.1 jedoch nicht erklärbar. Tabelle 4.2 zeigt die Übergänge der Inkonsistenzen aus Tabelle 4.1. In den folgenden Abschnitten wird geklärt, welche vorgefundenen Inkonsistenzen nach der WvT-Verlinkung in System A konsistent wurden. Im weiteren Verlauf der Feldstudie wird sich zeigen, dass Übergangsregeln existieren. Diese werden durch anonymisierte Beispiele veranschaulicht.

Übergänge der Inkonsistenzen. Jedes Wv-Paar wird in der Feldstudie hinsichtlich der Inkonsistenzen zweimalig betrachtet: einmal in der vorgefundenen Verlinkungssituation und einmal nach der automatischen WvT-Verlinkung. Diese zweimalige Betrachtung ermöglicht es, die Inkonsistenz eines vorgefundenen Wv-Paars mit der Inkonsistenz des gleichen automatisch verlinkten Wv-Paars zu vergleichen.

Tabelle 4.2 enthält in der ersten Spalte *Vorgefunden* $\Rightarrow$ *WvT-Verlinkung* alle Übergänge, die im Rahmen der Feldstudie auftraten. Die zweite Spalte enthält die gezählte Anzahl der *Übergänge*. Der Übergang $0 \Rightarrow 0$ besagt, dass ein Wv-Paar, das bereits vorher konsistent war, auch nach der WvT-Verlinkung konsistent ist. Dies war für 70 - und damit vorweg greifend für alle vorgefundenen konsistenten Wv-Paare der Fall. Der Übergang $1 \Rightarrow 0$ zeigt an, dass genau ein Wv-Paar, das bereits vorher inkonsistent $\mathrm{I_Q}$ war, nach der WvT-Verlinkung konsistent ist. Die Untersuchung von Tabelle 4.2 ermöglicht es, Übergangsregeln aufzudecken, die im Rahmen der Untersuchung des Systems A auftraten. Es folgt die Auseinandersetzung mit diesen Übergangsregeln.

Vorgefunden $\Rightarrow$ WvT-Verlinkung		**Übergänge**
$0 \Rightarrow 0$	(Konsistent $\Rightarrow$ konsistent)	70
$1 \Rightarrow 0$	($\mathrm{I_Q} \Rightarrow$ konsistent)	1
$2 \Rightarrow 2$	($\mathrm{I_Z} \Rightarrow \mathrm{I_Z}$)	2
$4 \Rightarrow 0$	($\mathrm{II_Q} \Rightarrow$ konsistent)	1
$4 \Rightarrow 4$	($\mathrm{II_Q} \Rightarrow \mathrm{II_Q}$)	55
$5 \Rightarrow 0$	($\mathrm{I_Q} \wedge \mathrm{II_Q} \Rightarrow$ konsistent)	2
$5 \Rightarrow 4$	($\mathrm{I_Q} \wedge \mathrm{II_Q} \Rightarrow \mathrm{II_Q}$)	2
$6 \Rightarrow 6$	($\mathrm{I_Z} \wedge \mathrm{II_Q} \Rightarrow \mathrm{I_Z} \wedge \mathrm{II_Q}$)	1
$12 \Rightarrow 12$	($\mathrm{II_Q} \wedge \mathrm{II_Z} \Rightarrow \mathrm{II_Q} \wedge \mathrm{II_Z}$)	3

Tabelle 4.2.: Übergänge der Inkonsistenzen von $\mathrm{SLH}^{\mathrm{mnl}}_{\mathrm{Ziel}}$ zu $\mathrm{SLH}^{\mathrm{auto}}_{\mathrm{Ziel}}$

Konsistent bleibt konsistent. Die Tabelle 4.2 legt offen, dass im untersuchten System jedes Wv-Paar, das bereits vorher konsistent war auch nach der WvT-Verlinkung konsistent ist: Jedes der 70 vorgefundenen konsistenten Wv-Paare ist auch nach der WvT-Verlinkung konsistent ($0 \Rightarrow 0$).

Inkonsistenz I_Q wird eliminiert. Die Inkonsistenz I_Q besagt, dass ein Testfall mit einer wiederverwendeten Quell-Anforderung, nicht jedoch mit der Ziel-Anforderung verlinkt ist. Die WvT-Verlinkung eliminiert die Inkonsistenz I_Q per Definition, weil T-Links eben genau auf der Grundlage der Wv-Links zwischen Quell- und Ziel-Anforderung gesetzt werden. Die Tabellen 4.1 und 4.2 zeigen die Eliminierung von I_Q, indem wegen der konjunktiven Verknüpfung der Konsistenzregeln und der binären Zählweise nach der WvT-Verlinkung keine ungeraden Inkonsistenzen existieren.

II_Q bleibt II_Q oder wird eliminiert. Die Inkonsistenz II_Q besagt, dass ein Testfall eine andere Quell-Anforderung absichert, die im SLH_{Ziel} entweder weggefallen ist oder zwar wiederverwendet, aber nicht mit dem Testfall verlinkt wurde. Wenn eine Inkonsistenz II_Q verursacht wird, weil an einer anderen Stelle eine Inkonsistenz I_Q auftritt, kann ein Wv-Paar mit einer Inkonsistenz II_Q konsistent werden. Wird die Inkonsistenz II_Q verursacht, weil eine Ziel-Anforderung tatsächlich weggefallen ist, bleibt sie erhalten. In Tabelle 4.2 zeigt sich dies, indem eine vorgefundene Inkonsistenz II_Q erhalten bleibt ($4 \Rightarrow 4, 5 \Rightarrow 4, 6 \Rightarrow 6, 12 \Rightarrow 12$) oder eliminiert wird ($4 \Rightarrow 0, 5 \Rightarrow 0$).

I_Z und/oder II_Z bleibt erhalten. Um die Vergleichbarkeit hinsichtlich der Ziel-Inkonsistenzen zu wahren, wurden nach der WvT-Verlinkung alle Testfälle, die exklusiv in das SLH_{Ziel}^{mnl} zeigten, auch mit den entsprechenden Ziel-Anforderungen des SLH_{Ziel}^{auto} verlinkt. Da die Ziel-Inkonsistenzen genau diese nachträglich verlinkten Testfälle adressieren, bleiben sie in der Feldstudie auch erhalten. Ein Blick in Tabelle 4.2 bestätigt dies ($2 \Rightarrow 2, 6 \Rightarrow 6, 12 \Rightarrow 12$).

Beispiele für Übergänge. Es folgt pro Zeile von Tabelle 4.2 ein realer, aber anonymisierter Repräsentant aus den Spezifikationsdokumenten des Systems A. Dies verbessert das Verständnis der beobachteten Übergänge von der vorgefundenen zur automatisch hergestellten Verlinkungssituation.

Beispiel: Konsistent (0) bleibt konsistent (0). Die linke Seite von Bild 4.10 zeigt die beiden vorgefundenen, konsistent mit dem Testfall t verlinkten Wv-Paare (1,a) und (2,b). Die beiden T-Links von t nach 2 und 1 wurden gelöscht und durch die WvT-Verlinkung erneut gesetzt. Dadurch entstand auf der rechten Bildseite eine im Vergleich zur linken Bildseite identische Verlinkungssituation. Die Wv-Paare (1,a) und (2,b) sind auf beiden Seiten konsistent.

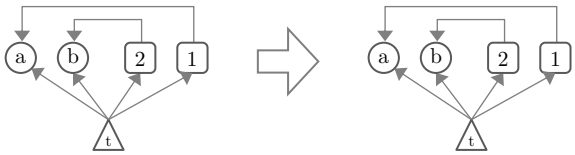


Bild 4.10.: $0 \Rightarrow 0$ (Links: Vorgefunden, Rechts: Automatisch)

Beispiel: Inkonsistenz I_Q (1) wird eliminiert (0). Die Inkonsistenz I_Q wird per Definition durch die WvT-Verlinkung eliminiert - auch in Kombination mit anderen Inkonsistenzen. Bild 4.11 zeigt, wie ein Wv-Paar, das zuvor pur inkonsistent I_Q war, nach der WvT-Verlinkung konsistent ist. Im Beispiel äußerst sich die vorgefundene Inkonsistenz I_Q, indem zwar der T-Link von dem Testfall t zur Quell-Anforderung a existierte, nicht jedoch vom Testfall zur Ziel-Anforderung 1. Dies ändert sich durch die automatische Verlinkung, so dass der T-Link vom Testfall zur Ziel-Anforderung 1 nach der WvT-Verlinkung existiert. Damit ist das Wv-Paar $(1, \mathrm{a})$ nun konsistent.

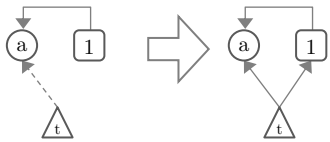


Bild 4.11.: $1 \Rightarrow 0$

Beispiel: Inkonsistenz I_Z (2) bleibt erhalten. Bild 4.12 zeigt einen offensichtlichen Aspekt: Die nachträglich gesetzten T-Links in das SLH_{Ziel} bleiben von der WvT-Verlinkung unberührt. Die Inkonsistenz I_Z bleibt erhalten.

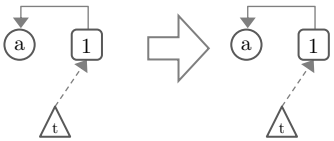


Bild 4.12.: $2 \Rightarrow 2$ (Links: Vorgefunden, Rechts: Automatisch)

Beispiel: Inkonsistenz II_Q (4) bleibt Inkonsistenz II_Q (4). Die Inkonsistenz II_Q besagt, dass ein Testfall Anforderungen im SLH_{Quell} absichert, die im SLH_{Ziel} gar nicht existieren. Bild 4.13 zeigt dies. Auf der linken Bildseite werden beide Partner des vorgefundenen Wv-Paars (2,b) durch den Testfall t abgesichert. Zusätzlich ist t mit der Quell-Anforderung a verlinkt, die ihrerseits nicht im SLH_{Ziel} wiederverwendet wurde: Der T-Link von t zu a verursacht für das Wv-Paar (2,b) die Inkonsistenz II_Q. Da die WvT-Verlinkung die Verlinkungssituation in das SLH_{Quell} nicht berührt und den T-Link von dem Testfall t zu der Ziel-Anforderung 2 setzt, führt die WvT-Verlinkung für (2,b) zu einer identischen Verlinkungssituation, wie dies bereits vorher gegeben war: Die Inkonsistenz II_Q bleibt in diesem Beispiel erhalten. Dass die Inkonsistenz II_Q aber auch konsistent werden kann, zeigt das nächste Beispiel.

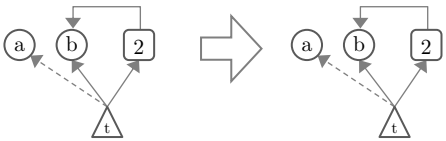


Bild 4.13.: $4 \Rightarrow 4$

Beispiel: II_Q (4) wird konsistent (0). Bild 4.14 zeigt mit Hilfe mehrerer Wv-Paare zwei verschiedene vorgefundene Inkonsistenzen, die durch die WvT-Verlinkung konsistent wurden. Dieser Paragraf behandelt zunächst das Wv-Paar (3,c), das vorher inkonsistent II_Q war. Der Testfall t zeigt auf beide Partner des Wv-Paares (3,c). Damit ist es nicht inkonsistent I_Q. Da der Testfall t jedoch auf die Quell- und nicht auf die Ziel-Anforderungen der beiden Wv-Paare (1,a) und (2,b) zeigt, ist das Paar (3,c) inkonsistent II_Q. Zuvor wurde bereits erkannt, dass die WvT-Verlinkung jegliches Vorkommen der Inkonsistenz I_Q per Definition eliminiert. Dies führt dazu, dass die nicht vorhandenen T-Links von dem Testfall t zu den Ziel-Anforderungen der Paare (1,a) und (2,b) gesetzt werden. Die Ursache für die Inkonsistenz II_Q des Wv-Paares (3,c) ist eliminiert und (3,c) ist damit konsistent.

Beispiel: $I_Q \wedge II_Q$ (5) wird konsistent (0). Bild 4.14 zeigt eine zweite vorgefundene Inkonsistenz: die Inkonsistenz $I_Q \wedge II_Q$ von (1,a) und (2,b). Die beiden Wv-Paare sind inkonsistent I_Q, weil der Testfall t zwar jeweils die Quell-, nicht jedoch die Ziel-Anforderung absichert. Diese Inkonsistenzen I_Q führen für (1,a) und (2,b) jeweils wechselseitig zur Inkonsistenz II_Q: Da der Testfall t die Quell-Anforderung des Wv-Paares (1,a) und gleichzeitig die Quell-Anforderung von (2,b) absichert, muss t gemäß der Definition von Konsistenz II_Q auch die Ziel-Anforderung 2 absichern. Weil dies jeweils nicht gegeben ist, sind (1,a) und (2,b) nicht nur inkonsistent I_Q, sondern $I_Q \wedge II_Q$. Die rechte Seite von Bild 4.14 zeigt, dass Paare, die zuvor inkonsistent $I_Q \wedge II_Q$ waren, wegen der Eliminierung der Inkonsistenz I_Q konsistent werden können.

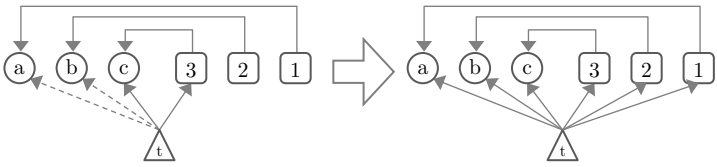


Bild 4.14.: $4 \Rightarrow 0$ und $5 \Rightarrow 0$

Beispiel: Inkonsistenz $I_Q \wedge II_Q$ (5) wird II_Q (4). Soeben wurde gezeigt, wie Wv-Paare, die zuvor inkonsistent $I_Q \wedge II_Q$ waren, wegen der Eliminierung der Inkonsistenz I_Q durch die WvT-Verlinkung konsistent werden können. Dass dies keine generelle Regel ist, zeigt Bild 4.15. Auf der linken Bildseite ist die vorgefundene Inkonsistenz $I_Q \wedge II_Q$ dargestellt. Das Wv-Paar (2,b) ist inkonsistent I_Q, weil der Testfall t zwar auf die Quell-Anforderung b, nicht jedoch auf die Ziel-Anforderung 2 zeigt. Gleichzeitig existiert aus der Perspektive von (2,b) eine zweite Quell-Anforderung a, die zwar über einen T-Link von dem Testfall, nicht jedoch über einen Wv-Link aus dem SLH^{mnl}_{Ziel} verfügt. Damit sichert der Testfall, wie in einigen der vorherigen Beispiele, Quell-Anforderungen ab, die im SLH^{mnl}_{Ziel} nicht existieren. Die rechte Bildseite zeigt die Verlinkungssituation nach der WvT-Verlinkung: Die Inkonsistenz I_Q von (2,b) wurde erwartungsgemäß eliminiert. Da die Quell-Anforderung a nicht Bestandteil eines Wv-Paars ist, kann auch kein neuer T-Link gesetzt werden. Damit zeigt das Bild 4.15 ein Beispiel dafür, dass die Inkonsistenz $I_Q \wedge II_Q$ wegen der Eliminierung von I_Q nicht zwangsläufig konsistent werden muss, sondern auch inkonsistent II_Q werden kann.

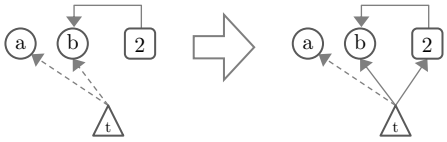


Bild 4.15.: $5 \Rightarrow 4$ (Links: Vorgefunden, Rechts: Automatisch)

Beispiel: Inkonsistenz I_Z und II_Z bleiben erhalten. Bild 4.16 zeigt die beiden letzten Beispiele. Es enthält wieder mehrere vorgefundene Inkonsistenzen. Zunächst wird jedoch wieder der offensichtliche Aspekt deutlich, dass alle Ziel-Inkonsistenzen wegen der Vorbereitung der Feldstudie erhalten bleiben: Nachträglich gesetzte T-Links in das SLH^{mnl}_{Ziel} werden auch im SLH^{auto}_{Ziel} nachträglich gesetzt. In der Anwendung bedeutet dies, dass die Inkonsistenzen jederzeit im Baureihenprojekt erneut bewertet werden können.

Beispiel: Inkonsistenz $I_Z \wedge II_Q$ (6) bleibt Inkonsistenz $I_Z \wedge II_Q$ (6). Tabelle 4.2 von Seite 73 enthält alle Übergänge der Inkonsistenzen des Systems A. Bild 4.16 zeigt die beiden letzten Übergangsarten. Zunächst wird das Wv-Paar (2,b) betrachtet. Während die Ziel-Anforderung 2 des Paars durch den Testfall t abgesichert wird, besitzt die Quell-Anforderung b keinen T-Link von t. Damit ist (2,b) inkonsistent I_Z. Gleichzeitig ist das Paar wegen des T-Links von t zur nicht wiederverwendeten Quell-Anforderung a inkonsistent II_Q. Die Inkonsistenz I_Z bleibt, wie zuvor erkannt, erhalten. Zugleich kann der T-Link wegen des Wegfalls der Quell-Anforderung a im SLH_{Ziel} nicht gesetzt werden und II_Q bleibt erhalten. Damit bleibt das Wv-Paar (2,b) auch nach der WvT-Verlinkung inkonsistent $I_Z \wedge II_Q$.

Beispiel: Inkonsistenz $II_Q \wedge II_Z$ (12) bleibt Inkonsistenz $II_Q \wedge II_Z$ (12). Das Paar (3,c) ist auf beiden Seite von Bild 4.16 inkonsistent $II_Q \wedge II_Z$: Da der Testfall die Quell-Anforderung a absichert, die im SLH_{Ziel} nicht existiert, ist das Paar inkonsistent II_Q. Gleichzeitig zeigt t auf die Ziel-Anforderung 2 und nicht auf die Quell-Anforderung b. Damit ist (3,c) inkonsistent II_Z. Da kein T-Link von a ins SLH_{Ziel} gesetzt werden kann und Ziel-Inkonsistenzen generell unberührt bleiben, bleibt die Inkonsistenz $II_Q \wedge II_Z$ im Beispiel erhalten.

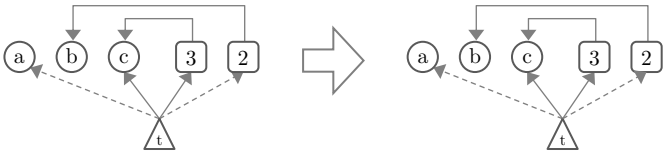


Bild 4.16.: $6 \Rightarrow 6$ und $12 \Rightarrow 12$

Zweite, bestätigende Feldstudie. Die Feldstudie zeigt die Effektivitätssteigerung der WvT-Verlinkung im Vergleich zum bisherigen Vorgehen. Durch die automatische Verlinkung werden übersehene T-Links vermieden, die an anderen Stellen Inkonsistenzen verursachen. Dabei wurden Übergangsregeln entdeckt, die anhand einer zweiten Feldstudie in Anhang C bestätigt werden.

4.6. Fazit

In diesem Kapitel wurde die *WvT-Verlinkung* vorgestellt, um Testfälle mit wiederverwendeten Anforderungen zu verlinken: WENN eine Ziel-Anforderung mit einer Quell-Anforderung über einen Wv-Link verbunden ist UND WENN eine Quell-Anforderung mit einem Testfall über einen T-Link verbunden ist, DANN wird der Testfall auch AUTOMATISCH mit der Ziel-Anforderung verlinkt.

In diesem Kapitel wurde zudem das *WvT-Diagramm* vorgestellt, das die Abhängigkeiten zwischen wiederverwendeten Systemlastenheft und Systemtestspezifikationen visualisiert. Die Visualisierung stellt die Gesamtverlinkungssituation zwischen den Spezifikationsdokumenten übersichtlich dar.

In einer *Feldstudie* wurde in diesem Kapitel nachgewiesen, dass die WvT-Verlinkung zu einer vergleichbaren Verlinkungssituation wie das manuelle Vorgehen führt. Es konnte sogar gezeigt werden, dass die WvT-Verlinkung zu weniger *Inkonsistenzen* als das bisherige Vorgehen führt.

Forschungsstand und Stand der Technik. In Kapitel 3 wurden Konzepte vorgestellt, um Anforderungen mit anderen Artefakten automatisch zu verlinken. Keines dieser Konzepte greift jedoch explizit auf die Wiederverwendungsbeziehungen zwischen Artefakten zurück. Die WvT-Verlinkung erweitert somit den Forschungsstand im Bereich der Anforderungsverfolgbarkeit um die *wiederverwendungsbasierte Verfolgbarkeit (engl.: reuse-based traceability)*.

Die WvT-Verlinkung findet im industriellen Umfeld wegen ihrer Relevanz und Effektivität Anwendung. Ein Erfolgsfaktor des entwickelten DOORS-Plugins ist die Wiederholbarkeit der Konsistenzanalysen während der Spezifikation. Die Anhänge A und B zeigen das Plugin und die erweiterten DOORS-Module.

WvT-Verlinkung und Testplanung. Die WvT-Verlinkung verlinkt alle Testfälle, die auf der Grundlage der Wiederverwendung auch verlinkbar sind. Diese ungefilterte Verlinkung aller Testfälle verlinkt auch diejenigen Testfälle, die gemäß Testplanung gar nicht im Ziel-Baureihenprojekt benötigt werden. Im nächsten Kapitel wird gezeigt, wie diese nicht benötigten Testfälle im Ziel-Baureihenprojekt hinsichtlich der Testplanung gefiltert werden.

5. Filterung gemäß Testkonzept

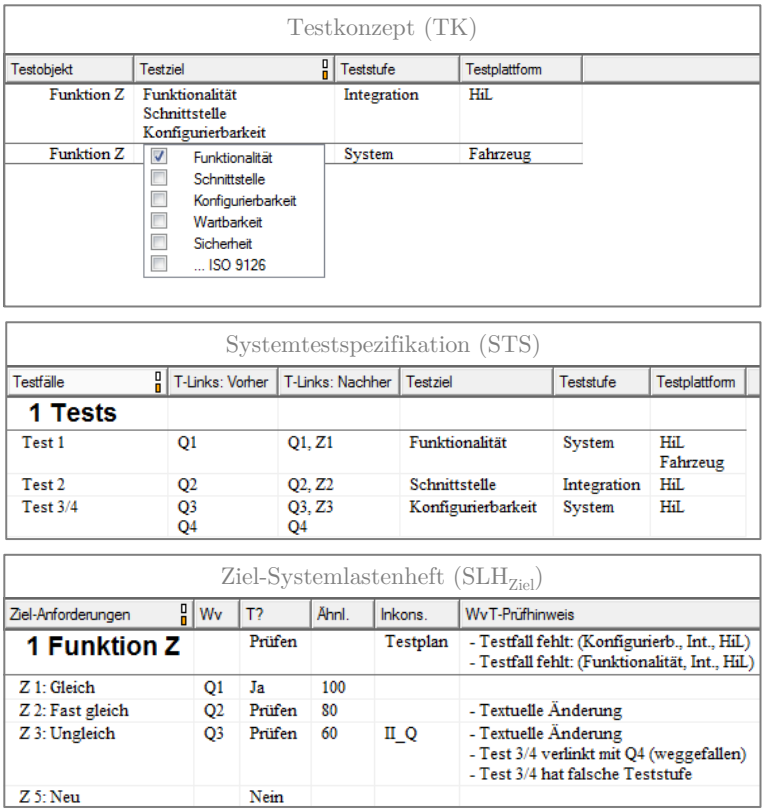
Die manuelle Durchführung von Testfällen beansprucht viel Zeit. Im anforderungsbasierten Test findet eine manuelle Testdurchführung statt, wenn Entwickler in (teil)integrierten Fahrzeugen Bedienelemente händisch betätigen, um das Verhalten des Fahrzeugs zu bewerten. Dabei werden diejenigen Testfälle durchgeführt, die mit den zu prüfenden Anforderungen verlinkt sind.

Die automatische WvT-Verlinkung verbindet alle verlinkbaren Testfälle. Im Rahmen der Testplanung wird jedoch im Testkonzept pro Baureihe definiert, welche Testfälle zur Absicherung welcher Anforderungen existieren bzw. nicht existieren müssen. In diesem Kapitel wird eine Technik vorgestellt, die die Testfälle filtert, die nicht vom Testkonzept gefordert werden. Gleichzeitig kann ermittelt werden, ob alle geforderten Testfälle existieren. Damit kann die Testabdeckung von Anforderungen bewertet werden.

Dieses Kapitel beginnt mit einem Beispiel. Anschließend folgt die Beschreibung der Filterung gemäß Testkonzept sowie der Vollständigkeit und Minimalität. Um den Aufwand der zusätzlichen Testplanung zu rechtfertigen, schließt das Kapitel mit einem Exkurs in die rechtlichen Grundlagen der [ISO26262-1].

5.1. Beispiel

Das folgende Beispiel in Bild 5.1 erweitert das einführende Beispiel aus Bild 4.1 von Seite 48. Die zweite Methodenschicht führt gemäß Seite 43f neben dem Testkonzept (TK) die Klassifikation der Testfälle hinsichtlich der Testplanungsdimensionen *Testziel*, *Teststufe* und *Testplattform* ein. Testziele entstammen aus Qualitätsmodellen wie beispielsweise der [ISO9126-1]. Teststufen orientieren sich am V-Modell. Sie definieren den zeitlichen Ablauf des Testprojekts [SL05, S. 41f]. Auf den automobilspezifischen Testplattformen, wie HiL oder Gesamtfahrzeug, werden Testfälle ausgeführt. Das TK beantwortet eine wesentliche Frage im Baureihenprojekt: Welche Testziele sind in welcher Teststufe auf welcher Testplattform pro Fahrzeugfunktion abzusichern?



Testkonzept (TK)

Testobjekt	Testziel	Teststufe	Testplattform
Funktion Z	Funktionalität Schnittstelle Konfigurierbarkeit	Integration	HiL
Funktion Z	☑ Funktionalität ☐ Schnittstelle ☐ Konfigurierbarkeit ☐ Wartbarkeit ☐ Sicherheit ☐ ... ISO 9126	System	Fahrzeug

Systemtestspezifikation (STS)

Testfälle	T-Links: Vorher	T-Links: Nachher	Testziel	Teststufe	Testplattform
1 Tests					
Test 1	Q1	Q1, Z1	Funktionalität	System	HiL Fahrzeug
Test 2	Q2	Q2, Z2	Schnittstelle	Integration	HiL
Test 3/4	Q3 Q4	Q3, Z3 Q4	Konfigurierbarkeit	System	HiL

Ziel-Systemlastenheft (SLH_{Ziel})

Ziel-Anforderungen	Wv	T?	Ähnl.	Inkons.	WvT-Prüfhinweis
1 Funktion Z		Prüfen		Testplan	- Testfall fehlt: (Konfigurierb., Int., HiL) - Testfall fehlt: (Funktionalität, Int., HiL)
Z 1: Gleich	Q1	Ja	100		
Z 2: Fast gleich	Q2	Prüfen	80		- Textuelle Änderung
Z 3: Ungleich	Q3	Prüfen	60	II_Q	- Textuelle Änderung - Test 3/4 verlinkt mit Q4 (weggefallen) - Test 3/4 hat falsche Teststufe
Z 5: Neu		Nein			

Bild 5.1.: Beispiel in [DOORS]

Phase 1.1: Forderungen der Testplanung extrahieren. Bild 5.1 zeigt ein vereinfachtes Testkonzept (TK). In dem TK wird pro Fahrzeugfunktion festgelegt, welche Testziele in welcher Teststufe auf welcher Testplattform abzusichern sind. Im TK wird somit definiert, in welchem Umfang die einzelnen Fahrzeugfunktionen im Baureihenprojekt abgesichert werden müssen. Damit dient das TK der Planung und Steuerung des Testprojekts. Im TK wird festgelegt, welche Testfälle spezifiziert, verlinkt, durchgeführt und dokumentiert werden müssen. Im Beispiel aus Bild 5.1 existieren vier Forderungen an die Absicherung von Funktion_Z, die zunächst aus dem TK extrahiert werden:

$$\begin{aligned}
\text{Funktion}_\text{Z} &: (\text{Funktionalität}, \text{Integration}, \text{HiL}) \\
\text{Funktion}_\text{Z} &: (\text{Schnittstelle}, \text{Integration}, \text{HiL}) \\
\text{Funktion}_\text{Z} &: (\text{Konfigurierbarkeit}, \text{Integration}, \text{HiL}) \\
\text{Funktion}_\text{Z} &: (\text{Funktionalität}, \text{System}, \text{Fahrzeug})
\end{aligned}$$

Im Beispiel fordert das TK, dass für Funktion_Z die Funktionalität während der Integrationsteststufe auf der HiL-Plattform abzusichern ist. Dies impliziert, dass entsprechende Testfälle mit den Anforderungen von Funktion_Z über T-Links verbunden sein müssen.

Phase 1.2: Angereichertes WvT-Problem extrahieren. Um einen Bezug zwischen dem TK und der Systemtestspezifikation (STS) herzustellen, werden die Testfälle der STS hinsichtlich der Testplanungsdimensionen des TK klassifiziert. Jedem Testfall werden somit Testziele, Teststufen und Testplattformen zugewiesen. Dies führt zu dem *angereicherten* WvT-Problem, das nicht nur die Verlinkungssituation des Testfalls zu Quell- und Ziel-Anforderungen beschreibt, sondern zudem die klassifizierenden Eigenschaften des Testfalls enthält. Die drei Testfälle in der STS aus Bild 5.1 sind wie folgt klassifiziert:

$$\begin{aligned}
\text{Test}_1 &: (\text{Funktionalität}, \text{System}, \text{HiL}) \\
\text{Test}_1 &: (\text{Funktionalität}, \text{System}, \text{Fahrzeug}) \\
\text{Test}_2 &: (\text{Schnittstelle}, \text{Integration}, \text{HiL}) \\
\text{Test}_{3/4} &: (\text{Konfigurierbarkeit}, \text{System}, \text{HiL})
\end{aligned}$$

Phase 2: Gemäß Testkonzept filtern. Die Filterung gemäß Testkonzept wird durch einen Vergleich der angereicherten WvT-Probleme mit den Forderungen des TK realisiert: Während ein Testfall mit einer Ziel-Anforderung verlinkt wird, wird geprüft, ob seine Eigenschaften mit den vom TK geforderten Eigenschaften übereinstimmen.

Der Testfall Test_1 sichert im Beispiel u.a. die Funktionalität während der Systemteststufe im Fahrzeug ab. Dies fordert auch das TK für Funktion_Z. Daher kann die Ziel-Anforderung Z_1 von Funktion_Z mit dem Testfall Test_1 verlinkt werden. Analog dazu wird auch der Testfall Test_2 mit Z_2 verlinkt.

Eine interessante Situation ergibt sich bei der WvT-Verlinkung von $\text{Test}_{3/4}$ und Z_3: Der Testfall sichert die Konfigurierbarkeit in der Systemteststufe auf der HiL-Plattform ab. Das TK fordert jedoch, dass die Konfigurierbarkeit bereits in der Integrationsteststufe abgesichert werden muss. Damit stimmen die Eigenschaften von $\text{Test}_{3/4}$ nicht mit den Forderungen des TK überein. Im Beispiel wird $\text{Test}_{3/4}$ mit Z_3 zwar verlinkt aber markiert, indem im SLH_Ziel ein Prüfhinweis hinterlegt wird. Dieser weist bei Anforderung Z_3 darauf hin, dass die Klassifikation von $\text{Test}_{3/4}$ nicht mit dem TK übereinstimmt.

Phase 3: Verlinkung bewerten. In der dritten Phase jeder Methodenschicht wird die Gesamtverlinkungssituation des SLH_Ziel bewertet. Nachdem die Filterung gemäß TK während der WvT-Verlinkung durchgeführt wurde, wird wieder die Verlinkungssituation bewertet - erweitert um den Aspekt der Testplanung: Es werden Testlücken hinsichtlich des TK, d.h. fehlende Testfälle aufgedeckt. Dazu werden die Konsistenzregeln der ersten Methodenschicht um *Vollständigkeit* und *Minimalität* erweitert. Dies bewertet die Effektivität und Effizienz der Umsetzung des TK: Das TK ist effektiv (vollständig) umgesetzt, wenn alle Testfälle existieren, die es fordert. Das TK ist effizient (minimal) umgesetzt, wenn keine Testfälle existieren, die es nicht fordert.

Im Beispiel ist Funktion_Z unvollständig abgesichert, da sie mit keinem Testfall mit der Klassifikation $(\text{Konfigurierbarkeit}, \text{Integration}, \text{HiL})$ verlinkt ist. Funktion_Z ist zudem nicht minimal abgesichert, weil sie mit $\text{Test}_{3/4}$ verlinkt ist, dessen Teststufe nicht mit den Forderungen des TK übereinstimmt.

5.2. Klassifikation des Testfalls des WvT-Problems

Das WvT-Problem ist ein Verlinkungsmodell. Es beschreibt den Zusammenhang zwischen den Artefakttypen Quell-Anforderung, Ziel-Anforderung und Testfall. Dabei verbinden die Links mit den Linktypen Wv-Link und T-Link die Artefakte miteinander. Das WvT-Problem beschreibt somit eine Verlinkungssituation strukturell.

Artefakte besitzen aber nicht lediglich Links, sondern auch klassifizierende Eigenschaften. Solche Eigenschaften ermöglichen die Unterscheidung von Artefakten. In diesem Zusammenhang hat jeder Testfall die klassifizierenden Eigenschaften *Testziel*, *Teststufe* und *Testplattform*. Im TK werden diese Eigenschaften auch zur Testplanung verwendet. Daher werden sie auch als Testplanungsdimensionen bezeichnet.

Bild 5.2 veranschaulicht, wie die Testplanungsdimensionen in diesem Kapitel den Testfall des WvT-Problems um nützliche Informationen anreichern: Der Testfall t hat Testziele, Teststufen und Testplattformen. Die Idee zur Klassifikation der Testfälle hinsichtlich der Testplanungsdimensionen fand ihren Ursprung in der Klassifikationsbaummethode [Gri95].

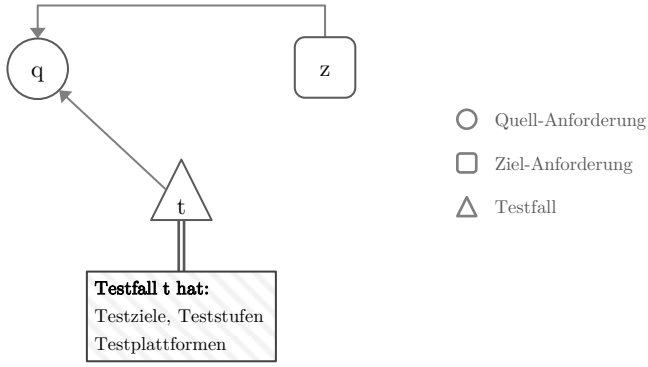


Bild 5.2.: Anreicherung des WvT-Problems

Testplanungsdimensionen. Die Testplanungsdimensionen dienen der Definition des Testumfangs. Mit ihrer Hilfe wird im Testkonzept festgelegt, für welche Fahrzeugfunktionen (Was?) welche Testziele (Wozu?) in welcher Teststufe (Wann?) auf welcher Testplattform (Worauf?) durch Testfälle abzusichern sind. Während die Testplanungsdimensionen im Testkonzept tatsächlich der Testplanung dienen, dienen sie in der Systemtestspezifikation dazu, die Testfälle zu klassifizieren. Die ISO 26262 fordert u.a. die Testplanungsdimensionen Testziel und Teststufe [ISO26262-4] sowie Testplattform [ISO26262-6]. Diese drei Dimensionen werden wie folgt als Mengen definiert:

$$\begin{aligned} \mathrm{Dim_{ziel}} &= \mathrm{D_z} = \{\mathrm{z} \mid \mathrm{z\ ist\ Testziel}\} \\ \mathrm{Dim_{stufe}} &= \mathrm{D_s} = \{\mathrm{s} \mid \mathrm{s\ ist\ Teststufe}\} \\ \mathrm{Dim_{plattform}} &= \mathrm{D_p} = \{\mathrm{p} \mid \mathrm{p\ ist\ Testplattform}\} \end{aligned}$$

Die einzelnen Elemente der Mengen werden als Ausprägungen der jeweiligen Testplanungsdimension bezeichnet. Beispiele für konkrete Ausprägungen der Testplanungsdimensionen sind

- Testziel: Korrekte Funktionalität, Korrekte Schnittstellen, Robustheit,
- Teststufe: Systemtest, Komponenten-Integrationstest und
- Testplattform: Gesamtfahrzeug, System-HiL, Komponenten-HiL.

Testfallklassifikation gemäß Testplanungsdimensionen. In Kapitel 4 wurde die Systemtestspezifikation (STS) auf Seite 51 als Menge aus Testfällen definiert. Im Folgenden wird die $\mathrm{STS_K}$ als Menge aus klassifizierten Testfällen definiert. Jedem klassifizierten Testfall wird mindestens ein Testziel, mindestens eine Teststufe und mindestens eine Testplattform zugewiesen. Jeder Testfall der $\mathrm{STS_K}$ ist gemäß den Testplanungsdimensionen klassifiziert:

$$\mathrm{STS_K} = \{\mathrm{t} : (\mathrm{T_z}, \mathrm{T_s}, \mathrm{T_p}) \mid \mathrm{t} \in \mathrm{STS}, \mathrm{T_z} \subseteq \mathrm{D_z}, \mathrm{T_s} \subseteq \mathrm{D_s}, \mathrm{T_p} \subseteq \mathrm{D_p}, \mathrm{T_{z/s/p}} \neq \emptyset\}$$

Der Testfall t wird um das Tripel $(\mathrm{T_z}, \mathrm{T_s}, \mathrm{T_p})$ erweitert, dessen Elemente die Mengen sind, die die Ausprägungen der jeweiligen Dimension enthalten. Für klassifizierte Testfälle gilt, dass keine der Mengen $\mathrm{T_z}$, $\mathrm{T_s}$, $\mathrm{T_p}$ leer sein darf.

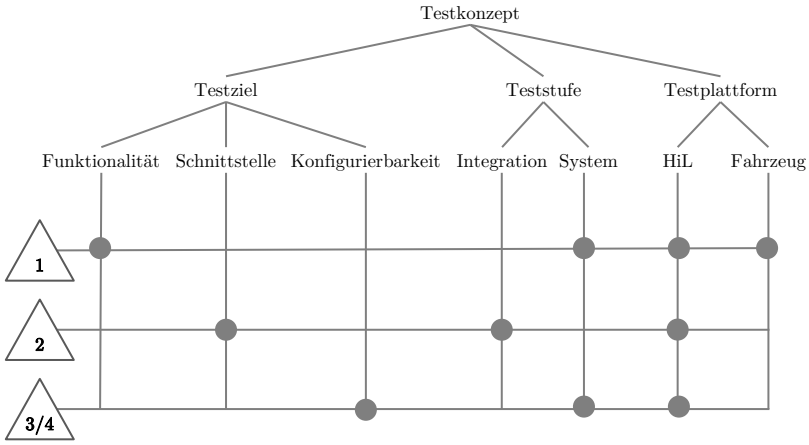


Bild 5.3.: Angereicherter Testfall im Klassifikationsbaum

Testfallklassifikation und Klassifikationsbaum. Die Testplanungsdimensionen können mitsamt ihren Ausprägungen in der Kopfreihe einer Tabelle angeordnet werden. Die Auflistung der Testfälle erfolgt in der Kopfspalte der Tabelle. Bild 5.3 zeigt den resultierenden Klassifikationsbaum [Gri95, S.52f]. Jeder markierte Schnittpunkt der Tabelle zeigt an, dass ein Testfall die Ausprägung einer klassifizierenden Eigenschaft besitzt. Die Markierungen in Bild 5.3 entsprechen den klassifizierten Testfällen aus dem aufbauenden Beispiel 5.1:

$$
\begin{aligned}
\mathrm{Test}_1 &: (\{\text{Funktionalität}\}_\mathrm{z}, \{\text{System}\}_\mathrm{s}, \{\text{HiL}, \text{Fahrzeug}\}_\mathrm{p}) \\
\mathrm{Test}_2 &: (\{\text{Schnittstelle}\}_\mathrm{z}, \{\text{Integration}\}_\mathrm{s}, \{\text{HiL}\}_\mathrm{p}) \\
\mathrm{Test}_{3/4} &: (\{\text{Konfigurierbarkeit}\}_\mathrm{z}, \{\text{System}\}_\mathrm{s}, \{\text{HiL}\}_\mathrm{p})
\end{aligned}
$$

Die Mechanismen der Klassifikationsbaummethode lassen sich, obwohl an dieser Stelle nicht weiter vertieft, anwenden, um systematisch einen Satz Testfälle zu ermitteln, der dann der WvT-Verlinkung zugeführt wird. Um diese Testfälle während der WvT-Verlinkung gemäß TK filtern zu können, bedarf es einer Verbindung zwischen den klassifizierten Testfällen der $\mathrm{STS_K}$ und dem TK.

5.3. Testfallklassifikation gemäß Testkonzept filtern

Während der Spezifikation, d.h. dem Aufschreiben von Testfällen, wird jeder Testfall hinsichtlich der Testplanungsdimensionen Testziel, Teststufe und Testplattform klassifiziert. In der Testkonzeptionierung, oder synonym der Testplanung, werden diese Testplanungsdimensionen auch verwendet. Für jede Fahrzeugfunktion wird im Testkonzept definiert, in welchem Umfang sie abgesichert werden muss. Im Sinne der WvT-Verlinkung wird während der Testkonzeptionierung festgelegt, welche T-Links existieren müssen, um die Anforderungen einer Fahrzeugfunktion ausreichend abzusichern.

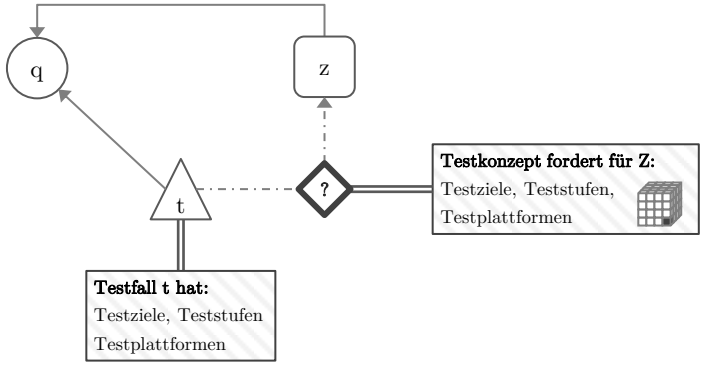


Bild 5.4.: Filterung gemäß Testkonzept

Bild 5.4 zeigt die Funktionsweise der Filterung gemäß TK. Der Testfall t aus der STS_K hat bestimmte Testziele, Teststufen und Testplattformen. Gleichzeitig fordert das TK, dass für die Fahrzeugfunktion Z des SLH_{Ziel} bestimmte Testziele in bestimmten Teststufen auf bestimmten Testplattformen abzusichern sind. Fahrzeugfunktionen bestehen aus Anforderungen. Der Testfall t wird mit der Ziel-Anforderung z von Z verlinkt, wenn die Klassifikation von t zu den Forderungen des TK für Z passt. Ist dies nicht gegeben, wird t zwar auch mit z verlinkt aber entsprechend markiert, um t filtern zu können.

Zugriff auf die Klassifikation eines Testfalls. Einem Testfall können mehrere Ausprägungen der jeweiligen Testplanungsdimension zugewiesen werden. Er kann beispielsweise gleichzeitig mehrere Testziele absichern oder potentiell während mehrerer Teststufen und/oder auf mehreren Testplattformen ablaufen. Im Beispiel aus Bild 5.1 von Seite 82 ist dies für den Testfall Test_1 gegeben, da die Ausprägungsmenge der Testplanungsdimension *Plattformen* zwei Elemente enthält. Der Zugriff auf die Ausprägungsmengen der Testplanungsdimensionen eines Testfalls t aus STS_K erfolgt folgendermaßen:

$$\begin{aligned}\text{ziel}(\text{t} : (\text{T}_\text{z}, \text{T}_\text{s}, \text{T}_\text{p})) &= \text{T}_\text{z} \text{ mit } \text{T}_\text{z} \subseteq \text{Dim}_\text{ziel} \\ \text{stufe}(\text{t} : (\text{T}_\text{z}, \text{T}_\text{s}, \text{T}_\text{p})) &= \text{T}_\text{s} \text{ mit } \text{T}_\text{s} \subseteq \text{Dim}_\text{stufe} \\ \text{plattform}(\text{t} : (\text{T}_\text{z}, \text{T}_\text{s}, \text{T}_\text{p})) &= \text{T}_\text{p} \text{ mit } \text{T}_\text{p} \subseteq \text{Dim}_\text{plattform}\end{aligned}$$

Der Testfall Test_1 aus dem Beispiel in Bild 5.1 hat die Ausprägungsmengen:

$$\begin{aligned}\text{ziel}(\text{Test}_1) &= \{\text{Funktionalität}\} \\ \text{stufe}(\text{Test}_1) &= \{\text{System}\} \\ \text{plattform}(\text{Test}_1) &= \{\text{HiL}, \text{Fahrzeug}\}\end{aligned}$$

Das mögliche Auftreten mehrerer Ausprägungen pro Testplanungsdimension führt dazu, dass für den Filtermechanismus eine Produktfunktion nötig wird. Diese Funktion ermittelt das kartesische Produkt der einzelnen Ausprägungen. Hierfür wird jedes Element der Ausprägungsmenge einer Dimension mit allen Elementen der Ausprägungsmengen der anderen Dimensionen verbunden. Die Produktfunktion ist definiert als:

$$\text{produkt}_\text{zsp}(\text{t}) = \text{ziel}(\text{t}) \times \text{stufe}(\text{t}) \times \text{plattform}(\text{t}) \text{ mit } \text{t} \in \text{STS}_\text{K}$$

Der Beispieltestfall Test_1 von Seite 82 sichert genau ein Testziel während einer Teststufe ab. Allerdings kann der Testfall auf zwei Testplattformen ausgeführt werden. Daher ergibt sich sich für Test_1 das kartesische Produkt wie folgt:

$$\begin{aligned}\text{produkt}_\text{zsp}(\text{Test}_1) = \{&(\text{Funktionalität}, \text{System}, \text{HiL}), \\ &(\text{Funktionalität}, \text{System}, \text{Fahrzeug})\}\end{aligned}$$

Testkonzept in Würfelform. Die Testplanung besitzt einen multidimensionalen Charakter. Sie ist nicht auf die drei Dimensionen Testziel, Teststufe und Testplattform beschränkt. Beispiele für weitere Dimensionen sind Testfallermittlungsmethoden, Testpersonen und natürlich auch der Testaufwand [SL05, S.11]. In dem Umfeld, in dem die WvT-Verlinkung Anwendung findet, liegt der Fokus der Testplanung auf den drei bislang betrachteten Testplanungsdimensionen. Der Übersichtlichkeit halber erfolgt die Definition der Filtertechnik auf der Grundlage der drei Dimensionen. Die Technik ist jedoch nicht auf diese Dimensionen beschränkt. Bild 5.5 zeigt den Testkonzeptwürfel.

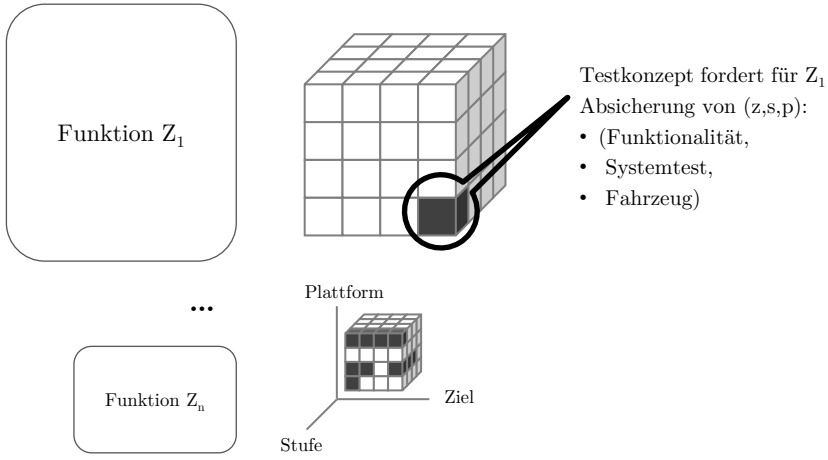


Bild 5.5.: Testkonzeptwürfel pro Fahrzeugfunktion

Für jede Fahrzeugfunktion des SLH_{Ziel} wird im Testkonzept festgelegt, welche Forderungen bezüglich der Absicherung an sie gestellt werden. Für jede Fahrzeugfunktion existiert demnach ein Würfel. Die Markierung eines Würfelfelds impliziert für die zugewiesene Fahrzeugfunktion, dass ein entsprechend klassifizierter Testfall existieren muss, um dem Testkonzept gerecht zu werden.

Testkonzeptwürfel und Testkonzept. Ein Testkonzeptwürfel wird durch den Raum definiert, den die Testplanungsdimensionen aufspannen. Der Begriff Würfelbelegung bezeichnet die Ausprägungsmengen der Testplanungsdimensionen. Die Würfelbelegung entspricht somit der Menge schwarz markierter Felder eines Würfels aus Bild 5.5. Das Testkonzept enthält mehrere Würfel, da im Rahmen der Testplanung für jede Fahrzeugfunktion eine spezifische Würfelbelegung festgelegt wird.

$$\begin{aligned} \text{Testkonzeptwürfel} &= \text{Dim}_{\text{ziel}} \times \text{Dim}_{\text{stufe}} \times \text{Dim}_{\text{plattform}} \\ \text{Testkonzept(Funktion)} &\subseteq \text{Testkonzeptwürfel} \end{aligned}$$

Filterung gemäß Testkonzept (TK). Der Testkonzeptwürfel steuert mit Hilfe der Würfelbelegung die WvT-Verlinkung. Ein klassifizierter Testfall t wird nur kommentarlos mit der Ziel-Anforderung a_Z einer Funktion f_Z verlinkt, wenn die Würfelbelegung einem Testfall entspricht, der wie t klassifiziert ist. Der Filtermechanismus $\text{filtere}_{\text{gemäßTK}}$ wird der WvT-Verlinkung vorangestellt:

$$\text{filtere}_{\text{gemäßTK}}(a_Z, t) \Rightarrow a_Z \rightarrow_{\text{Wv}} a_Q \wedge t \rightarrow_{\text{T}} a_Q \Rightarrow t \rightarrow_{\text{T}} a_Z$$

Wenn die Klassifikation des Testfalls t zur Würfelbelegung für a_Z - bzw. der Fahrzeugfunktion f_Z, die durch a_Z spezifiziert wird - passt, dann evaluiert $\text{filtere}_{\text{gemäßTK}}$ zu wahr und die WvT-Verlinkung wird regulär durchgeführt. Ansonsten wird der Testfall zwar verlinkt aber entsprechend markiert.

$$\begin{aligned} \text{holeFunktion}(a_Z) &= \text{Funktion } f_Z, \text{ die durch } a_Z \text{ beschrieben wird} \\ \text{filtere}_{\text{gemäßTK}}(a_Z, t) &= \exists \text{zsp} \in \text{produkt}_{\text{zsp}}(t) : \\ &\qquad \text{zsp} \in \text{Testkonzept}(\,\text{holeFunktion}(a_Z)\,) \end{aligned}$$

Die Funktion $\text{filtere}_{\text{gemäßTK}}$ evaluiert zu wahr, wenn mindestens ein Tripel (z, s, p) der Produktmenge $\text{produkt}_{\text{zsp}}$ des Testfalls t auch in der Würfelbelegung von a_Z enthalten ist. Genau dann stimmt die Klassifikation des Testfalls mit den Forderungen des Testkonzepts für die Anforderung a_Z und somit für die Fahrzeugfunktion f_Z überein.

5.4. WvT-Inkonsistenzen: Vollständig und minimal

Mit dem Testkonzept wird es möglich, die Testabdeckung von Fahrzeugfunktionen zu bestimmen. Da Fahrzeugfunktionen durch Anforderungen beschrieben werden, lässt sich mit Hilfe des Testkonzepts die Testabdeckung von Anforderungen bestimmen. Dabei können zwei Abdeckungskriterien unterschieden werden: Die Vollständigkeit und die Minimalität.

Beispiel. Das Testkonzept fordert für eine Fahrzeugfunktion Z_1 die Existenz von Testfällen zur Absicherung der Testziele *Funktionalität* und *Schnittstellenkorrektheit* während der Teststufe *Systemtest* auf der Testplattform *HiL*. Dann wären im Testkonzeptwürfel von Funktion Z_1 zwei Felder markiert:

$$\begin{aligned} \text{Feld}_\text{A} &= Z_1 \ : \ (\text{Funktionalität}, \text{System}, \text{HiL}) \\ \text{Feld}_\text{B} &= Z_1 \ : \ (\text{Schnittstellen}, \text{System}, \text{HiL}) \end{aligned}$$

Der Testfall Test_1 aus Beispiel 5.1 von Seite 82 sichert das Testziel *Funktionalität* während der Teststufe *Systemtest* ab - sowohl auf der Testplattform *HiL* als auch im *Gesamtfahrzeug*. Damit besitzt er die zwei Klassifikationen:

$$\begin{aligned} \text{Klassifikation}_\text{A} &= \text{Test}_1 \ : \ (\text{Funktionalität}, \text{System}, \text{HiL}) \\ \text{Klassifikation}_\text{B} &= \text{Test}_1 \ : \ (\text{Funktionalität}, \text{System}, \text{Fahrzeug}) \end{aligned}$$

Der Testfall Test_1 wird mit einer Anforderung von Z_1 unmarkiert verlinkt, da das Feld_A des Würfels und die $\text{Klassifikation}_\text{A}$ von Test_1 übereinstimmen. Im Beispiel existieren keine weiteren Testfälle, die mit Z_1 verlinkt sind. Der T-Link von Test_1 zu Z_1 stellt die Gesamtverlinkungssituation von Z_1 dar.

Mit Hilfe der folgenden Definitionen wird bestimmt, ob die Gesamtverlinkungssituation einer Funktion vollständig und/oder minimal ist:

$$\begin{aligned} \text{vollständig}(\text{f}_\text{Z}) = \forall \text{zsp}_\text{Feld} \in \text{Testkonzept}(\text{f}_\text{Z}) : \exists \text{t} \in \text{T}_\text{alle}(\text{f}_\text{Z}) : \\ \exists \text{zsp}_\text{Klass.} \in \text{produkt}_\text{zsp}(\text{t}) : \text{zsp}_\text{Feld} = \text{zsp}_\text{Klass.} \\ \text{minimal}(\text{f}_\text{Z}) = \forall \text{t} \in \text{T}_\text{alle}(\text{f}_\text{Z}) : \forall \text{zsp}_\text{Klass.} \in \text{produkt}_\text{zsp}(\text{t}) : \\ \exists \text{zsp}_\text{Feld} \in \text{Testkonzept}(\text{f}_\text{Z}) : \text{zsp}_\text{Klass.} = \text{zsp}_\text{Feld} \end{aligned}$$

Im Testkonzept wird pro Fahrzeugfunktion festgelegt, welche Testfälle existieren müssen. Testfälle sind aber mit den Anforderungen einer Funktion verlinkt. Die Vollständigkeit und Minimalität einer Funktion wird daher anhand ihrer Anforderungen bewertet. Vollständigkeit wird erreicht, wenn mindestens eine Anforderung einer Funktion vollständig abgesichert ist. Minimalität wird erreicht, wenn keine ihrer Anforderung nicht minimal abgesichert ist.

Der Ausdruck $T_{alle}(f_Z)$ iteriert durch alle Anforderungen einer Funktion f_Z. Er führt den Ausdruck $T(a_Z)$ von Seite 52 für alle Anforderungen a_Z der Funktion f_Z aus. $T_{alle}(f_Z)$ gibt somit die Menge aller Testfälle der STS_K zurück, die über einen T-Link zu einer Anforderung von f_Z zeigen.

Vollständigkeit. Eine Funktion f_Z ist vollständig abgesichert, wenn für jedes markierte Feld des Testkonzeptwürfels ein Testfall t existiert, der mit einem T-Link auf eine Anforderung a_Z von f_Z zeigt und der ebenso klassifiziert ist, wie das markierte Würfelfeld. Vollständig bedeutet somit, dass keine Testfälle fehlen dürfen. Eine Funktion, die dieser Vollständigkeit nicht genügt, ist unvollständig abgesichert.

Die Funktion Z_1 ist beispielsweise nicht vollständig abgesichert, da der Testfall $Test_1$ als einziger Testfall von Z_1 keine Klassifikation besitzt, die dem $Feld_B$ des Würfels entspricht. Das bedeutet, dass für Z_1 ein Testfall fehlt, der die Schnittstellen während der Systemteststufe auf der HiL-Plattform absichert.

Minimalität. Eine Funktion f_Z ist minimal abgesichert, wenn keine Anforderung von f_Z mit einem Testfall t verlinkt ist, der eine Klassifikation besitzt, die im Testkonzeptwürfel nicht belegt ist. Minimal bedeutet, dass nicht benötigte Testfälle nicht mit Anforderungen verlinkt sein sollten. Eine Funktion, die diesen Ansprüchen nicht genügt, ist nicht minimal abgesichert.

Die Funktion Z_1 ist im Beispiel nicht minimal abgesichert, weil der Testfall $Test_1$ neben der $Klassifikation_A$ auch die $Klassifikation_B$ besitzt. Diese zweite $Klassifikation_B$ entspricht keinem Feld des Testkonzepts. Sie verletzt somit die Minimalitätsregel für die Funktion Z_1.

Konjunktive Verknüpfung der Regeln. Eine Funktion f_Z ist konsistent hinsichtlich des Testkonzepts, wenn für f_Z *vollständig* $\wedge$ *minimal* gegeben ist.

5.5. Falluntersuchung

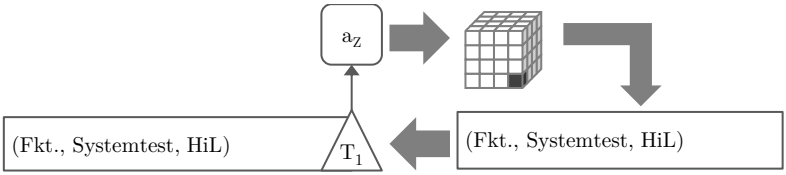
Wegen der konjunktiven Verknüpfung der zuvor definierten Regeln ergeben sich verschiedene Fälle hinsichtlich der Abdeckung: Abdeckung, Überabdeckung, Unterabdeckung, Fehlabdeckung. Es folgen Beispiele für jeden Abdeckungsfall. Für jedes Beispiel in Bild 5.6 gilt, dass a_Z die einzige Anforderung einer nicht dargestellten Funktion f_Z ist. Damit gelten die Aussagen zur Vollständigkeit und Minimalität gleichermaßen für a_Z und für f_Z.

Vollständige und minimale Abdeckung. Bild 5.6a zeigt die Anforderung a_Z und den mit ihr verlinkten Testfall T_1. T_1 ist ebenso klassifiziert, wie die einzige Feldmarkierung (Funktionalität, Systemtest, HiL) des Testkonzeptwürfels von f_Z. Damit ist jede Feldmarkierung abgedeckt und es existiert auch keine weitere Testfallklassifikation. Da a_Z die einzige Anforderung von Funktion f_Z ist, ist f_Z vollständig und minimal abgesichert.

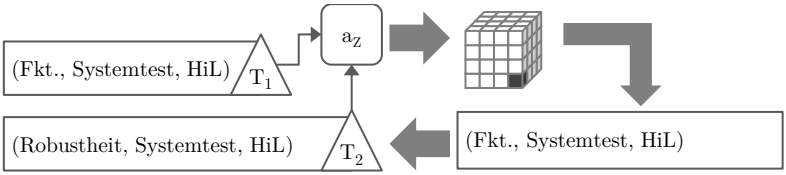
Überabdeckung: Vollständig und nicht minimal. Bild 5.6b entspricht Bild 5.6a - erweitert um den Testfall T_2, der zusätzlich mit der Anforderung a_Z verlinkt ist. Funktion f_Z bleibt wie in 5.6a wegen T_1 vollständig. Der Testfall T_2 hat die Klassifikation (Robustheit, Systemtest, HiL). Da das Testkonzept keinen Robustheitstest für f_Z fordert, T_2 aber dennoch mit a_Z verlinkt ist, ist f_Z hinsichtlich des Testkonzepts nicht minimal.

Unterabdeckung: Unvollständig und minimal. In Bild 5.6c ist die Anforderung a_Z wieder exklusiv mit dem Testfall T_1 verlinkt. In diesem Beispiel verlangt der Würfel für f_Z jedoch die zusätzliche Existenz eines Robustheitstestfalls. Da die einzige Klassifikation von T_1 eine Entsprechung in der Würfelbelegung hat, ist a_Z und somit f_Z zunächst minimal abgesichert. Das Fehlen eines Robustheitstestfalls führt jedoch zu einer unvollständigen Absicherung von f_Z. Damit ist f_Z zwar minimal, nicht jedoch vollständig abgesichert.

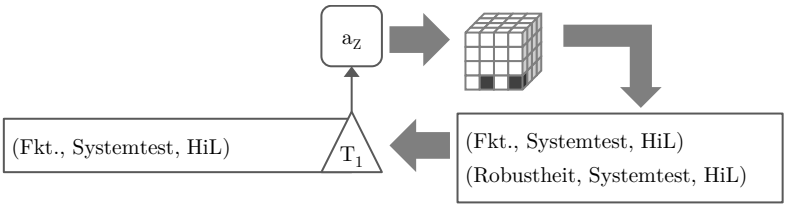
Fehlabdeckung: Unvollständig und nicht minimal. Im letzten Bild 5.6d ist der Robustheitstest T_2 exklusiv mit der Anforderung a_Z verlinkt. Es existiert kein Testfall, dessen Klassifikation einem Feld der Würfelbelegung entspricht. Gleichzeitig ist im Würfel kein Feld markiert, das der Klassifikation von T_2 entspricht. Damit ist f_Z weder vollständig, noch minimal abgesichert.



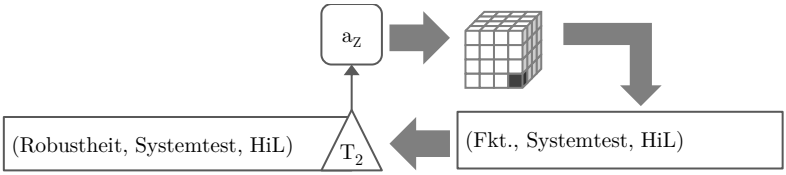
(a) Vollständige und minimale Abdeckung



(b) Überabdeckung: Vollständig und nicht minimal



(c) Unterabdeckung: Unvollständig und minimal



(d) Fehlabdeckung: Unvollständig und nicht minimal

Bild 5.6.: Beispiele für Testabdeckungen gemäß Testkonzept

5.6. Rechtliche Grundlagen und ISO 26262

Besonders Hersteller von sicherheitskritischen Systemen müssen die Produkte testen, die sie auf den Markt bringen. Falls trotz Test ein Fehler beim Verbraucher auftritt, muss jederzeit verfolgbar sein, welche Testfälle welche Funktionen absichern und dass die Testfälle tatsächlich durchgeführt wurden. Die WvT-Verlinkung unterstützt diese Verfolgbarkeit direkt, da sie Anforderungen mit Testfällen verlinkt. Die Existenz dieser Links ermöglicht die Nachvollziehbarkeit, wenn auch die Testfalldurchführungs- und Fehlerdokumentation mit den Testfällen in Verbindung gesetzt werden. Es reicht jedoch nicht, dass Testfälle Anforderungen absichern - es müssen auch die richtigen Testfälle mit den richtigen Anforderungen verlinkt sein. Der Stand der Technik liefert durch Normen branchenübliche Hinweise darauf, welche Testziele in welchen Teststufen auf welchen Testplattformen abzusichern sind. Die Filterung gemäß Testkonzept erweitert die WvT-Verlinkung um die Möglichkeit, die Verlinkungssituation hinsichtlich der Forderungen der ISO 26262 zu bewerten - doch Erweiterungen, insbesondere wenn sie neue Dokumente in den Entwicklungsprozess einführen, können auf Widerstand treffen.

Verpflichtende Erweiterung des Fahrzeugtests. Wenn gerechtfertigt werden muss, dass etablierte Prozesse durch neue Methoden erweitert werden müssen, gibt es häufig schwer überwindbare Gegenargumente. Neben potenziellen Kostensteigerungen durch zusätzliche Aufgaben wird das Risiko aufgeführt, dass Methoden in der Praxis nicht die beabsichtigte Wirkung erzielen. Auf der anderen Seite existiert ein externer Zwang, der Prozesserweiterungen notwendig macht. Ein aus Haftungsgründen besonders wichtiger Aspekt ist hierbei der Stand der Technik aus Normen. Die aus Normensicht relevanten Haftungsgrundlagen entspringen dabei aus den beiden Rechtskreisen Produkt- und Produzentenhaftung [SRDD11, S.38].

Produzentenhaftung. „Wer vorsätzlich oder fahrlässig das Leben, den Körper, die Gesundheit, die Freiheit, das Eigentum oder ein sonstiges Recht eines anderen widerrechtlich verletzt, ist dem anderen zum Ersatz des daraus entstehenden Schadens verpflichtet“ [BGB, § 823 Schadensersatzpflicht, Abs. 1].

Delikthaftigkeit. Das „Wer“ im Gesetzestext der Produzentenhaftung verdeutlicht die Delikthaftigkeit der Schadenersatzpflicht. Der Produzent in Form einer Person muss vorsätzlich oder fahrlässig gehandelt haben und dieses Handeln ist im Rahmen eines Produzentenhaftungsfalls durch den Geschädigten nachzuweisen. „Fahrlässig handelt, wer die im Verkehr erforderliche Sorgfalt außer Acht lässt“ [BGB, §276 Verantwortlichkeit des Schuldners, Abs. 2]. Das Testen von Produkten, die in den Verkehr gebracht werden, ist hierbei als erforderliche Sorgfalt aufzufassen [Koc03, S.400]. Dass ein Hersteller seiner erforderlichen Sorgfalt nicht nachkam, muss durch den Geschädigten nachgewiesen werden. Die Beweislast liegt damit beim Verbraucher.

Nachweis durch Verbraucher. Um einen Hersteller beweisbar zu belasten, müsste ein außenstehender Verbraucher die internen Abläufe eines Unternehmens detailliert kennen. Würde ein Softwarefehler in einem Steuergerät einen Schaden verursachen, müsste der Verbraucher im Sinne der Produzentenhaftung nachweisen, dass der Fahrzeughersteller die Software nicht ausreichend getestet hat. Bis zum Jahr 1990 gab es für Ansprüche aus fehlerhaften Produkten ausschließlich die Produzentenhaftung - zum Nachteil des Verbrauchers. Um diesen Nachteil zu eliminieren, trat am 1. Januar 1990 das Produkthaftungsgesetz in Kraft [ProdHG, § 19 Inkrafttreten].

Produkthaftung. „Wird durch den Fehler eines Produkts jemand getötet, sein Körper oder seine Gesundheit verletzt oder eine Sache beschädigt, so ist der Hersteller des Produkts verpflichtet, dem Geschädigten den daraus entstehenden Schaden zu ersetzen. Im Falle der Sachbeschädigung gilt dies nur, wenn eine andere Sache als das fehlerhafte Produkt beschädigt wird und diese andere Sache ihrer Art nach gewöhnlich für den privaten Ge- oder Verbrauch bestimmt und hierzu von dem Geschädigten hauptsächlich verwendet worden ist“ [ProdHG, §1 Haftung, Absatz 1].

Gefährdungshaftung. Das „Wird“ verdeutlicht, dass die Produkthaftung eine Gefährdungshaftung ist. Die Beweislast liegt beim Hersteller. Ein Produkt ist eine bewegliche Sache [ProdHG, §2 Produkt] und Software ist eine bewegliche Sache, wenn sie mit Hardware kombiniert wird [SKL11, S. 41f].

Abwenden des Haftungsfalls durch Hersteller. Das Produkthaftungsgesetz beschreibt, wie ein Haftungsfall abgewendet werden kann [ProdHG, §1 Haftung, Abs. 2, Satz 1 bis 5]. Satz 2 besagt, dass „das Produkt den Fehler, der den Schaden verursacht hat noch nicht hatte, als der Hersteller es in den Verkehr brachte“. Der Beweispflicht des Herstellers wird nachgekommen, wenn er durch die Dokumentation der Testdurchführung nachweisen kann, dass die Software gemäß Anforderungsspezifikation funktionierte. Eine interessante, hier jedoch nicht weiter verfolgte Frage ist, wie Softwarefehler, die weder das ursprüngliche Produkt, noch Aktualisierungen zur Inverkehrbringung hatten, nachträglich in das Produkt gelangen können. Satz 5 besagt, dass „die Ersatzpflicht des Herstellers ausgeschlossen ist, wenn der Fehler nach dem Stand der Wissenschaft und Technik in dem Zeitpunkt, in dem der Hersteller das Produkt in den Verkehr brachte, nicht erkannt werden konnte“. Der Satz erzwingt die Umsetzung von Normen. Er rechtfertigt somit die Einführung des Testkonzepts und die damit einhergehende Erweiterung der WvT-Verlinkung um die Filterung gemäß Testkonzept. Ein Automobilhersteller kann durch die erweiterte WvT-Verlinkung nachvollziehbar belegen, dass er die Anforderungen in der Art abgesichert hat, wie es die ISO 26262, die den Stand der Technik zur Absicherung sicherheitskritischer Systeme beschreibt, fordert.

Stand der Technik aus Normen. Definition aus [DIN45020]: „Ein normatives Dokument zu einem technischen Gegenstand wird zum Zeitpunkt seiner Annahme als der Ausdruck einer anerkannten Regel der Technik anzusehen sein, wenn es in Zusammenarbeit der betroffenen Interessen durch Umfrage- und Konsensverfahren erzielt wurde“. In der Literatur zum Thema Technikrecht wird darauf geschlossen, dass „trotz ihres Empfehlungscharakters [...] internationale technische Normen [...] eine hohe Akzeptanz [haben], die in der Praxis zu einem faktischen Befolgungszwang führt“ [BHD10, S.176]. Daraus folgt für Automobilhersteller aus dem ProdHG direkt, dass sie Normen, die den Stand der Technik widerspiegeln, sowohl in der Entwicklung als auch in der Fertigung von Fahrzeugen zu berücksichtigen haben. Daraus folgt eine Motivation der WvT-Verlinkung: Die ISO 26262 fordert Verfolgbarkeit - also müssen Testfälle und Anforderungen verlinkt sein.

Rückrufe wegen fehlerhafter Software. Dass fehlerhafte Software jeden Automobilhersteller betreffen kann, zeigen Rückrufaktionen. Diese sind gemäß [BGB, §823] als Produktbeobachtungspflicht im Rahmen der Verkehrssicherungspflicht geschuldet. In der Rückrufdatenbank des ADAC gibt es hierfür eine große Anzahl an Beispielen [ADAC]. Es folgen chronologisch sortierte Einträge aus der Datenbank. Opel rief im Juli 2013 den Insignia (Baujahr 2009 bis 2012) zurück, weil Softwarefehler im Motorsteuergerät zu Leistungsabfall und im ungünstigsten Fall zu Motorschäden führen können. Volkswagen rief im Mai 2013 den VW Up (Baujahr 2013) wegen einer fehlerhaften Parametrierung der Diagnosesoftware für das Airbag-Steuergerät zurück. Daimler rief den Mercedes-Benz CLS (Baujahr 2010 bis 2011) im Oktober 2011 zurück, da sich wegen eines fehlerhaften Signals in einem Positionssensor die Lenkkraftunterstützung deaktivieren kann. BMW rief im Oktober 2010 alle Modelle mit dem 3.0 Twinturbo Benzinmotor (Baujahr 2007 bis 2010) zurück, da Softwarefehler in der Motorsteuerung mindestens zu Startproblemen, in Extremfällen aber auch zu Motorschäden führen können. Diese Beispiele sind ein kleiner Ausschnitt der ADAC Datenbank. An ihnen wird deutlich, dass Hersteller ihrer Produktbeobachtungspflicht pflichtgemäß nachkommen.

Das Airbag-Urteil. Glücklicherweise gab es bislang selten Autounfälle durch Softwarefehler, in denen Menschen zu Schaden kamen. Ein Fall ergab sich am 24. April 2003. Er führte über mehrere Instanzen bis zum Bundesgerichtshof [BGH-Urteil]. Das Urteil wurde in den Medien als Airbag-Urteil bekannt. Der Fahrer eines BMW 330 D, der im Jahr 2000 erstmals zugelassen wurde, fuhr durch ein Schlagloch. Daraufhin öffnete sich der Seiten-Airbag und verletzte den Kläger an der Halsschlagader. Dies hatte einen Hirninfarkt zur Folge. Sowohl das Landesgericht Erfurt, als auch das Oberlandesgericht (OLG) Jena wiesen die Schadenersatzforderung des Klägers ab. Die Begründung war, dass die „Fehlauslösung nicht durch den Stand der Technik zu verhindern gewesen sei“. Weiterhin „komme es nicht auf technische Möglichkeiten der Optimierung des Airbag-Systems an, weil diese aufwändig, kostenintensiv und nicht Stand der Technik seien“. Der Bundesgerichtshof (BGH) urteilte überraschend, dass der Fall durch das Revisionsgericht neu verhandelt werden muss.

Branchenüblichkeit definiert nicht den Stand der Technik. Der BGH kam nämlich zu dem Schluss, dass das OLG Jena den Stand der Technik mit Branchenüblichkeit verwechselte. Nach Aussage eines Gutachters wäre es möglich gewesen, die Fehlauslösung des Airbags durch zusätzliche Sensoren zu vermeiden. Der BGH erkannte, dass „Gefahrenvermeidungsmaßnahmen, die technisch möglich sind, [...] begrifflich nicht über den Stand von Wissenschaft und Technik hinaus [gehen]“ [BGH-Urteil, S. 11]. Der Hersteller muss nun beweisen, dass das technisch Mögliche nicht zumutbar gewesen wäre oder dass er der Instruktionspflicht nachkam. Aktuell gibt es keine weiteren Verfahren. Ein Vergleich zwischen Hersteller und Kläger ist nicht auszuschließen.

Rechtliche Folgen des Urteils. Aus dem Airbag-Urteil folgt, dass sowohl das Produkt als auch die Prozesse dem Stand der Technik genügen müssen. Um Haftungsfälle abwenden zu können, muss ein Hersteller in jeder Phase der Produktentstehung sicherstellen, dass Risiken erkannt und, wenn zumutbar, konstruktiv behandelt werden. Die ISO 26262 bündelt als Sicherheitsnorm den aktuellen Stand der Technik. Sie entstand im Konsensverfahren unter Beteiligung der Automobilindustrie. Die Norm erhebt den Anspruch, „gesetzliche und behördliche Anforderungen zu berücksichtigen [... und ...] bereits in der Konzeptphase, in der Entwicklung und in der Produktion rechtliche Anforderungen [zu] erfüllen“ [SRDD11]. Da sich die Norm am V-Modell orientiert, fordert sie funktionale Sicherheit für den gesamten Entwicklungsprozess - also auch für wiederverwendete Anforderungen, Testfälle und deren Verlinkung.

Möglichkeiten der Testkonzeption. Gemäß [ProdHG, §2 Produkt] ist ein „Produkt eine bewegliche Sache, auch wenn sie einen Teil einer anderen beweglichen Sache [...] bildet“. In einem Produkthaftungsfall ist demnach nicht das gesamte Fahrzeug Gegenstand einer Schadenersatzforderung, sondern die Fehler verursachende Funktion. Die ISO 26262 beschreibt mit ASIL die Möglichkeit, Funktionen hinsichtlich ihrer Kritikalität einzustufen. Diese Kritikalität treibt dann den Testaufwand. Die testkonzeptgesteuterte Filterung berücksichtigt diesen Aspekt direkt, indem jeder Funktion ein spezifischer Testkonzeptwürfel zugewiesen wird und die Filterung und Bewertung auf der Grundlage des funktionsspezifischen Würfels durchgeführt wird.

Airbag-Urteil und WvT-Problem. Das Airbag-Urteil enthält eine interessante Passage zum Thema Wiederverwendung. Der Hersteller hatte das Airbag-Steuergerät MRS2 (Mehrfach-Rückhalte-System) der Vorgängerbaureihe weiterentwickelt. Das Steuergerät der Nachfolgebaureihe enthielt die neue Softwareversion MRS3. Der Hersteller hatte wegen eines Fehlers im MRS2, der zu Fehlauslösungen des Airbags führen konnte, eine Rückrufaktion durchgeführt. Laut dem BGH-Urteil hätte der Hersteller wissen müssen, dass es auch in der Nachfolgebaureihe zu Fehlauslösungen kommen konnte.

Airbag-Urteil und WvT-Verlinkung. Das BGH urteilte, dass „die unzutreffende Annahme des Herstellers, eine bekannte Gefahr durch konstruktive Verbesserungen des bestehenden Systems behoben zu haben, [...] nicht aus[reicht] [...]“ [BGH-Urteil, S. 18]. Die ISO 26262 sagt diesbezüglich aus, dass Wiederverwendung in allen Ebenen des V-Modells dokumentiert werden kann. Dies betrifft sowohl die Markierung neuer als auch wiederverwendeter modifizierter und nichtmodifizierter Entwicklungsartefakte [ISO26262-6, S. 12]. Zudem kann eine Analyse der Auswirkungen (engl.: impact analysis) aufdecken, welchen Einfluss die Modifikationen auf andere Entwicklungsartefakte besitzen [ISO26262-6, S. 13]. Hier zeigt sich erneut der direkte Nutzen der WvT-Verlinkung und der WvT-Inkonsistenzen. Zuvor wurde aufgedeckt, dass die Airbag-Steuergeräte MRS2 und MRS3 in einem Wiederverwendungsverhältnis stehen. Angenommen, dieses Verhältnis spiegelt sich auch in den Spezifikationsdokumenten wider, so dass Wv-Links zwischen den Quell- und Ziel-Anforderungen existieren können. Im Urteil des BGH wird darauf aufgebaut, dass das MRS2 bereits zurückgerufen wurde, weil es zu einer Fehlauslösung kommen konnte. Nun könnte es sein, dass T-Links von Testfällen zu den Anforderungen des MRS2 existieren, die relevant für den Rückruf waren. Diese Verfolgbarkeit führt zu neuen Argumentationsmöglichkeiten. Beispielsweise sind die für den Rückruf relevanten Anforderungen möglicherweise nicht dieselben, wie die Anforderungen, die im Airbag-Urteil relevant sind. Ungeachtet dieser Spekulationen zeigt sich, dass die Verfolgbarkeit von Fehlern über Testfälle bis hin zu wiederverwendeten Anforderungen grundsätzlich juristisch verwertbar ist, um „zutreffende Annahmen“ zu tätigen.

5.7. Fazit

In diesem Kapitel wurde eine Technik vorgestellt, die die WvT-Verlinkung um die *Filterung gemäß Testkonzept* erweitert: WENN eine Ziel-Anforderung mit einer Quell-Anforderung über einen Wv-Link verbunden ist UND WENN eine Quell-Anforderung mit einem Testfall über einen T-Link verbunden ist, DANN wird der Testfall GEMÄSS TESTKONZEPT mit der Ziel-Anforderung T-verlinkt.

Das Testkonzept wird von der ISO 26262 gefordert. Im Testkonzept wird im Rahmen der Testplanung für Testobjekte - und darunter zählen auch Systemanforderungen - festgelegt, welche Testziele in welchen Teststufen auf welchen Testplattformen abzusichern sind. Diese Ziele, Stufen und Plattformen finden hierbei nicht nur im Testkonzept Anwendung - sie klassifizieren auch die Testfälle, die durch die WvT-Verlinkung verlinkt werden.

In diesem Kapitel wurden die *klassifizierenden Eigenschaften von Testfällen* eingeführt: Jeder Testfall hat die Eigenschaften Testziel, Teststufe und Testplattform. Es wurde eine Filtertechnik vorgestellt, die die klassifizierenden Eigenschaften der Testfälle mit den Festlegungen des Testkonzepts vergleicht. Darauf basierend wurden die *Vollständigkeit* und *Minimalität* definiert, mit deren Hilfe die *Testabdeckung von Anforderungen* bewertet werden kann.

Mit Hilfe eines BGH-Urteils konnte gezeigt werden, dass Normen im Rahmen der Produkthaftung einem „ faktischen Befolgungszwang“ unterliegen. Dabei zeigte sich, dass das Testkonzept kein Aspekt ist, der professionelle Testplanung charakterisiert. Im Gegenteil - es ist geboten, weil es nicht über den Stand der Technik hinausgeht. Mit der Filterung gemäß Testkonzept gelang es, die Ergebnisse der Testplanung in die WvT-Verlinkung einfließen zu lassen.

Potential durch vollständige Klassifikation. Das WvT-Problem enthält eine Quell-Anforderung, eine Ziel-Anforderung und einen Testfall. In diesem Kapitel wurde jedoch lediglich die Klassifikation des Testfalls eingeführt. Im nächsten Kapitel wird bislang ungenutztes Potential offengelegt, indem alle Artefakte des WvT-Problems klassifiziert werden. Dies ermöglicht eine detailliertere Filterung, indem Regeln mit Hilfe von WvT-Fällen definiert werden.

6. Fallbasierte Filterung

Wenn Anforderungen wiederverwendet werden, werden sie im Rahmen des neuen Baureihenprojekts häufig auch angepasst. Diese Änderungen können sich direkt auf die Testfälle auswirken, die mit den entsprechenden Anforderungen verlinkt sind. Dann stellt sich für jede geänderte Anforderung die Frage, ob die verlinkten Testfälle noch für ihre Absicherung geeignet sind.

Diese Frage kann meistens intuitiv durch die Entwickler beantwortet werden, da sie wegen der Wiederverwendung häufig bereits viele Jahre mit den Anforderungen und Testfällen arbeiten. Die Schwierigkeit ist vielmehr, dass die relevanten Änderungen in der Vielzahl der Anforderungen und Testfälle untergehen. Mit der fallbasierten Filterung wird daher eine Technik entwickelt, die die relevanten Änderungen erkennt und im Systemlastenheft markiert.

Das Kapitel beginnt mit einem Beispiel. Im Anschluss daran wird der Aufbau des WvT-Falls beschrieben. Danach erfolgt die Beschreibung der fallbasierten Filterung, die Vorstellung einiger interessanter Fälle und die Anwendung der Grundprinzipien des fallbasierten Schließens (engl.: Case-Based Reasoning, CBR) auf die WvT-Verlinkung. Das Kapitel schließt mit der Beschreibung der aktuellen Einsatzmöglichkeiten von CBR.

6.1. Beispiel

Das Beispiel aus Bild 6.1 baut auf dem Beispiel aus Bild 5.1 von Seite 82 auf, das seinerseits bereits Beispiel 4.1 von Seite 48 erweiterte. Die Testfälle der STS sind wie im Beispiel 5.1 von Seite 82 klassifiziert. In der dritten Methodenschicht wird berücksichtigt, dass neben Testfällen auch die Quell- und Ziel-Anforderungen klassifiziert sein können. Beispiele für Anforderungsklassifikationen sind ihre ASIL-Einstufung oder ihre Schnittstellen zu anderen Systemen. Mit der dritten Methodenschicht wird ein neues Dokument eingeführt: Die Falldatenbank. Mit ihrer Hilfe können Filterregeln definiert werden.

Quell-Systemlastenheft (SLH_{Quell})

Quell-Anforderungen	Wv	T	Schnittstelle
1 Funktion Q			
Q 1	Z1	Ja	
Q 2	Z2	Ja	System 1
Q 3	Z3	Ja	
Q 4: Wird wegfallen		Ja	

Systemtestspezifikation (STS)

Testfälle	T-Links: Vorher	T-Links: Nachher	Testziel	Teststufe	Testplattform
1 Tests					
Test 1	Q1	Q1, Z1	Funktionalität	System	HiL Fahrzeug
Test 2	Q2	Q2, Z2	Schnittstelle	Integration	HiL
Test 3/4	Q3 Q4	Q3, Z3 Q4	Konfigurierbarkeit	System	HiL

Ziel-Systemlastenheft (SLH_{Ziel})

Ziel-Anforderungen	Wv	T?	Schnittstelle	Inkons.	WvT-Prüfhinweis
1 Funktion Z		Prüfen		Testplan	- Testfall fehlt: (Konfigurierb., Int., HiL) - Testfall fehlt: (Funktionalität, Int., HiL)
Z 1: Gleich	Q1	Ja			
Z 2: Fast gleich	Q2	Prüfen	System 1 System 2	WvT-Fall	- Textuelle Änderung - Schnittstelle geändert. Schnittstellentestfälle prüfen: Test 2
Z 3: Ungleich	Q3	Prüfen		II_Q	- Textuelle Änderung - Test 3/4 verlinkt mit Q4 (weggefallen) - Test 3/4 hat falsche Teststufe
Z 5: Neu		Nein			

Falldatenbank

WvT-Fälle (engl.: WvT-Cases)	WvT-Entscheidung	Quell-Anf.: Schnittstelle	Ziel-Anf.: Schnittstelle	Test: Testziel
1 Falldatenbank				
Schnittstelle geändert. Schnittstellentestfälle prüfen	Prüfen	System 1	System 1 System 2	Schnittstelle

Bild 6.1.: Beispiel in [DOORS]

Phase 1: Klassifiziertes WvT-Problem extrahieren. Bild 6.1 zeigt ein SLH_{Quell}, eine STS und ein SLH_{Ziel}. Diese Spezifikationsdokumente enthalten klassifizierte Artefakte. Die drei Artefakte Q_2, Z_2 und $Test_2$ bilden im Beispiel ein klassifiziertes WvT-Problem, da sowohl die Quell- und Ziel-Anforderung als auch der Testfall klassifizierende Eigenschaften besitzen.

In Bild 6.1 verfügen das SLH_{Quell} und das SLH_{Ziel} über eine neue Spalte *Schnittstelle*. Damit besitzen die Quell- und Ziel-Anforderung Q_2 und Z_2 eine gleiche klassifizierende Eigenschaft. Dies ermöglicht es, Änderungen der klassifizierenden Eigenschaft *Schnittstelle* von Quelle zu Ziel zu erkennen. Die STS besitzt, wie bereits im Beispiel 5.1 von Kapitel 5 auf Seite 82, die drei Spalten *Testziel*, *Teststufe* und *Testplattform*. Dies sind die klassifizierenden Eigenschaften der Testfälle der STS. In der ersten Phase der dritten Methodenschicht werden die klassifizierten WvT-Probleme aus dem SLH_{Quell}, dem SLH_{Ziel} und der STS extrahiert und wie folgt repräsentiert:

```
WvT-Problem(Q2, Z2, Test2) =
   (
      [
          ('Schnittstelle',  {'System 1'},{'System 1', 'System 2'})
      ],
      [
          ('Testziel',      {'Schnittstelle'}),
          ('Teststufe',     {'Integration'}),
          ('Testplattform', {'HiL'})
      ]
  )
```

Ein WvT-Problem besteht immer aus einer Quell-Anforderung, einer Ziel-Anforderung und einem Testfall. Im Beispiel sind dies Q_2, Z_2 und $Test_2$. Der erste [..] Block enthält die klassifizierenden Eigenschaften des Wiederverwendungspaars (Z_2, Q_2). Die Quell-Anforderung Q_2 hat eine Schnittstelle zu $System_1$. Z_2 hat neben $System_1$ eine neue Schnittstelle zu $System_2$. Anforderungseigenschaften werden durch zwei Mengen repräsentiert: Die Menge der Quell-Eigenschaften und die Menge der Ziel-Eigenschaften. Der zweite [..] Block enthält die Eigenschaften von $Test_2$. Da das WvT-Problem nur einen Testfall enthält, klassifizieren Testfalleigenschaften nur ein Artefakt. Daher werden die Testfalleigenschaften nur durch eine Menge repräsentiert.

Phase 2: Gemäß Falldatenbank filtern. Nachdem ein klassifiziertes WvT-Problem extrahiert wurde, wird die Falldatenbank nach einem ähnlichen WvT-Fall durchsucht. Jeder Fall enthält eine Verlinkungsentscheidung und einen Prüfhinweis, die auf ein ähnliches WvT-Problem angewendet werden können. Die Falldatenbank im Bild 6.1 enthält einen einfachen WvT-Fall: Wenn sich die Schnittstellen von der Quell-Anforderung zur Ziel-Anforderung geändert haben, müssen die Schnittstellentestfälle geprüft werden. WvT-Fälle sind, mit Ausnahme der Verlinkungsentscheidung und dem Prüfhinweis, exakt wie WvT-Probleme aufgebaut:

```
WvT-Fall =
   (
      [
          ('Schnittstelle', {'System 1'},{'System 1', 'System 2'})
      ],
      [
          ('Testziel', {'Schnittstelle'})
      ],
      ('Prüfen','Schnittstelle geändert. Schnittstellentestfälle prüfen')
  )
```

An dieser einführenden Stelle soll es genügen, dass sich im einfachen Beispiel die klassifizierenden Eigenschaften des WvT-Problems $(Q_2, Z_2, Test_2)$ direkt im oben dargestellten WvT-Fall wiederfinden lassen. Das WvT-Problem und der WvT-Fall sind damit sehr ähnlich. Das Beispiel in Bild 6.1 zeigt, dass die Verlinkungsentscheidung 'Prüfen' auf die Ziel-Anforderung Z_2 angewendet und der Prüfhinweis in die Kommentarspalte des SLH_{Ziel} kopiert wurde.

Phase 3: Verlinkung bewerten. Mit Hilfe der Falldatenbank können auch bereits verlinkte SLH_{Ziel} bewertet werden. Selektierte Fälle können verwendet werden, um Suchanfragen zu stellen. Eine Beispielsuchanfrage in den Spezifikationsdokumenten aus Bild 6.1 wäre: „Suche alle Ziel-Anforderungen mit geänderten Schnittstellen und wiederverwendeten Schnittstellentests“.

Ähnlichkeitsbestimmung. Die Falldatenbank enthält WvT-Fälle. Diese definieren Regeln, mit deren Hilfe während der WvT-Verlinkung suspekte WvT-Probleme entdeckt und markiert werden können. Die fallbasierte Filterung basiert auf den nachfolgend vorgestellten Ähnlichkeitsfunktionen.

6.2. Fallbasierte Filterung

Das Ziel der fallbasierten Filterung ist es, suspekte WvT-Probleme zu erkennen und mittels „Prüfen“ zu markieren. In einer Falldatenbank werden dazu WvT-Fälle definiert. Ein WvT-Fall ist als eine Regelsammlung zu verstehen. Die Regeln definieren, wie ein WvT-Problem klassifiziert sein muss, um als suspekt eingestuft zu werden. Damit stellen WvT-Fälle Verlinkungssituationen dar, in denen die WvT-Verlinkung fragwürdig ist.

6.2.1. Grundidee

Mit Hilfe von Ähnlichkeitsfunktionen wird während der Verlinkung für jedes zu lösende WvT-Problem ermittelt, ob die Falldatenbank einen ähnlichen WvT-Fall enthält. Die Ähnlichkeitsfunktionen greifen dabei auf die klassifizierenden Eigenschaften von den Quell- und Ziel-Anforderungen sowie den Testfällen von WvT-Fällen und -Problemen zurück. Die Funktionen vergleichen die Ausprägungsmengen der Eigenschaften hinsichtlich der enthaltenen Elemente und bilden diese Informationen auf einen Prozentwert ab.

Fallrepräsentation. Ein WvT-Problem enthält zwei Anforderungen (Quelle und Ziel). Beide Anforderungen haben gleiche klassifizierende Eigenschaften (KE) wie beispielsweise *Schnittstellen zu anderen Systemen*. Der Testfall tritt als einzelnes Artefakt im WvT-Problem auf. Auch er hat KE wie beispielsweise *Testziele*. Anforderungs-KE und Testfall-KE reichern das WvT-Problem an.

Analog dazu werden die KE auch im WvT-Fall durch Ausprägungsmengen repräsentiert. Die Ähnlichkeit von WvT-Fällen und WvT-Problemen wird mit Hilfe des Vergleichs der Elemente in diesen Ausprägungsmengen bestimmt.

Ähnlichkeitsfunktionen. Die Ähnlichkeitsfunktionen verwenden bekannte binäre Ähnlichkeitsmaße, um Ähnlichkeiten für jede einzelne Anforderungs- bzw. Testfall-KE zu berechnen. Die Berechnungsgrundlage sind hierbei die KE, die sowohl im WvT-Fall als auch im WvT-Problem auftreten. Die Ähnlichkeit einer einzelnen KE wird als lokale Ähnlichkeit bezeichnet. Die globale Ähnlichkeit ergibt sich aus dem Produkt der lokalen Ähnlichkeiten.

6.2.2. Klassifikation von WvT-Problem und WvT-Fall

Das Konzept der Klassifikation wurde im Rahmen der testkonzeptgesteuerten Filterung bereits in Kapitel 5.2 auf den Seiten 85ff vorgestellt. Eine klassifizierende Eigenschaft (KE) ermöglicht die Unterscheidung zweier Artefakte. Eine KE ist eine Menge, deren Elemente als Ausprägungen der KE bezeichnet werden. Ein Beispiel für eine KE ist *Farbe*. Die Menge *{rot, orange, grün}* enthält die Ausprägungen der KE *Farbe*.

Die Idee im vorangegangenen Kapitel 5 war, neben den Wv-Links und T-Links im WvT-Problem auch zusätzliche Informationen zu speichern. Testfälle wurden mit Hilfe der KE *Testziel*, *Teststufe* und *Testplattform* klassifiziert. Bild 6.2 zeigt, wie nicht nur Testfälle, sondern alle Artefakte des WvT-Problems klassifiziert werden können.

Eine Besonderheit ergibt sich aus dem Wiederverwendungsverhältnis einer Quell-Anforderung q und einer Ziel-Anforderung z. Dieses Verhältnis ermöglicht es, Änderungen von KE in z zu erkennen, die nach der Wiederverwendung entstanden sind. Beispielsweise lässt sich erkennen, ob sich die KE *Schnittstelle* von Quelle-zu-Ziel geändert hat.

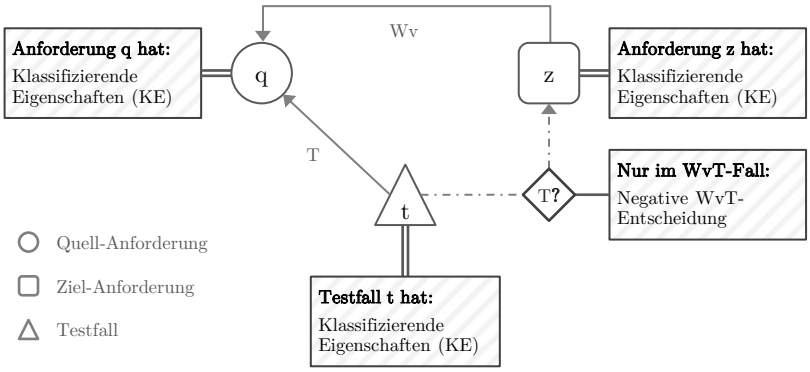


Bild 6.2.: Klassifiziertes WvT-Problem, klassifizierter WvT-Fall

Klassifiziertes WvT-Problem. Alle drei Artefakte des WvT-Problems im Bild 6.2 besitzen KE. Die Anforderungen können über die gleichen Anforderungs-KE mit jeweils (un)ähnlichen Ausprägungsmengen verfügen. Die Testfall-KE können den KE aus Kapitel 5 entsprechen, sind aber nicht auf diese beschränkt. Generell gilt, dass beliebige KE verwendet werden können.

Im Bild 6.3 ist auf der linken Seite der Aufbau des klassifizierten WvT-Problems dargestellt. Die beiden Abschnitte [Anforderungs-KE] und [Testfall-KE] enthalten jeweils die KE 1 bis n und 1 bis m. Eine Anforderungs-KE besteht aus zwei Mengen: Die Menge der Quell-Ausprägungen $\{...\}_{\text{Quelle}}$ und die Menge der Ziel-Ausprägungen $\{...\}_{\text{Ziel}}$. Da WvT-Probleme nur einen Testfall enthalten, enthält eine Testfall-KE auch nur eine Ausprägungsmenge $\{...\}_{\text{Test}}$.

Klassifizierter WvT-Fall. Bild 6.3 zeigt auf der rechten Seite den Aufbau des WvT-Falls. Die [Anforderungs-KE] und [Testfall-KE] werden identisch zum WvT-Problem dargestellt. Dies ermöglicht den Ähnlichkeitsvergleich der KE zwischen Fall und Problem. Neben den KE enthält ein Fall zusätzlich die WvT-Verlinkungsentscheidung und einen Prüfhinweis.

Ein Beispiel verdeutlicht, warum im Fall nur die negativen Entscheidungen *Prüfen* und *Nein* gespeichert werden: Wenn sich die Schnittstellen von Quelle-zu-Ziel geändert haben, kann davon ausgegangen werden, dass ein Schnittstellentest geprüft werden muss. Es kann jedoch nicht davon ausgegangen werden, dass keine Überprüfung nötig ist, wenn sich die Schnittstellen nicht geändert haben. Auf den Seiten 129f wird auf diese Einschränkung näher eingegangen.

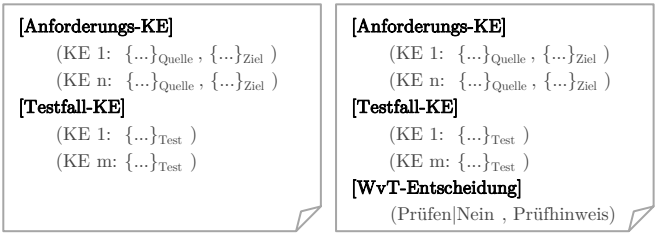


Bild 6.3.: Aufbau von WvT-Problem (links) und WvT-Fall (rechts)

Datenstruktur. Im einführenden Beispiel wurde je ein Beispiel für ein WvT-Problem auf Seite 105 und einen WvT-Fall auf Seite 106 vorgestellt. Die allgemeine Repräsentation von WvT-Problemen und WvT-Fällen ist:

```
Anforderungs-KE  = (Name,Quell-Ausprägungsmenge,Ziel-Ausprägungsmenge)
Testfall-KE      = (Name,Test-Ausprägungsmenge)
WvT-Entscheidung = (Prüfen|Nein,Prüfhinweis)

WvT-Problem      = ([Anforderungs-KE],[Testfall-KE])
WvT-Fall         = ([Anforderungs-KE],[Testfall-KE],WvT-Entscheidung)
```

Jede KE hat einen *Namen*, der sie eindeutig identifiziert. Pro *Anforderungs-KE* werden die Mengen der *Quell-Ausprägungen* und *Ziel-Ausprägungen* gespeichert. Eine *Testfall-KE* besteht aus einer Menge der *Test-Ausprägungen*.

Sowohl ein *WvT-Problem* als auch ein *WvT-Fall* kann mehrere *[Anforderungs-KE]* und *[Testfall-KE]* enthalten. Der *WvT-Fall* enthält zusätzlich eine *WvT-Entscheidung*. Diese kann den Wert *Prüfen* oder *Nein* annehmen. Die *WvT-Entscheidung* enthält außerdem einen *Prüfhinweis*.

Zum besseren Verständnis der Datenstruktur wird der Beispielfall von Seite 106 wiederholt:

```
WvT-Fall =
   (
      [
          ('Schnittstelle', {'System 1'},{'System 1', 'System 2'})
      ],
      [
          ('Testziel', {'Schnittstelle'})
      ],
      ('Prüfen','Schnittstelle geändert. Schnittstellentestfälle prüfen')
  )
```

Zusammenhang von Problem und Fall. Der Aufbau von Fällen und Problemen ist gleichartig, weil ein WvT-Fall nicht nur als Regelsammlung, sondern auch als früher gelöstes WvT-Problem aufgefasst werden kann. Die Gleichartigkeit des Aufbaus ermöglicht es, nachfolgend Funktionen zu definieren, die die Ähnlichkeit zwischen Problemen und Fällen berechnen. Wenn ein neues Problem einem früheren Fall ähnelt, kann die WvT-Entscheidung des Falls direkt für die Lösung des neuen Problems wiederverwendet werden.

6.2.3. Ähnlichkeit zwischen Problem- und Fallklassifikation

Ein WvT-Problem und ein WvT-Fall sollen auf Ähnlichkeit untersucht werden. Genauer gesagt werden die KE untersucht, die sowohl im Problem als auch im Fall vorkommen. Dies wird mit Ähnlichkeitsfunktionen realisiert.

Aus dem Aufbau von WvT-Problemen und -Fällen folgen zwei Möglichkeiten, die Ähnlichkeit von KE zu berechnen: der lokale Wiederverwendungstrend von Anforderungs-KE und die lokale direkte Ähnlichkeit von Anforderungs-KE und Testfall-KE. Der Trend resultiert aus dem Quelle-zu-Ziel-Vergleich der Anforderungs-KE eines Wv-Paars, d.h. der Quell- und Ziel-Anforderung innerhalb von Problem oder Fall. Dies ermöglicht es, Änderungen zu erkennen, die nach der Wiederverwendung aufgetreten sind. Der direkte Vergleich wird über die KE von Quelle-zu-Quelle, Ziel-zu-Ziel und Test-zu-Test geführt. Dies ermöglicht den direkten Vergleich der KE zwischen Problem und Fall.

Lokaler Wiederverwendungstrend. Bild 6.4 zeigt, wie der lokale Wiederverwendungstrend ermittelt wird. Für den WvT-Fall und das WvT-Problem wird zunächst ermittelt, wie stark sich die Ausprägungen einer Anforderungs-KE intern von Quelle-zu-Ziel verändert haben. Aus den beiden internen trendbasierten Ähnlichkeiten von Fall und Problem wird schließlich der Abstand ermittelt. Der Abstand zeigt an, ob sich die Anforderungs-KE im WvT-Fall und -Problem nach der Wiederverwendung ähnlich stark geändert hat.

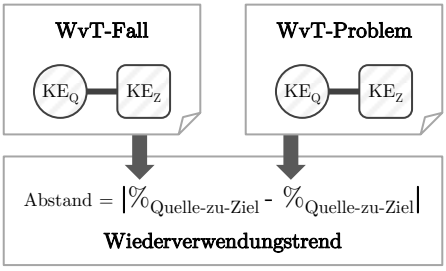


Bild 6.4.: Lokaler Wiederverwendungstrend von Anforderungs-KE

Lokale direkte Ähnlichkeit. Bild 6.5 zeigt, wie die lokale Ähnlichkeit direkt berechnet wird. Jede KE eines Artefakts des WvT-Falls wird mit der entsprechenden KE des Artefakts des WvT-Problems verglichen:

- Der Quelle-zu-Quelle-Vergleich ermittelt die lokale Ähnlichkeit einer Anforderungs-KE zwischen den beiden Quell-Anforderungen.
- Der Ziel-zu-Ziel-Vergleich ermittelt die lokale Ähnlichkeit einer Anforderungs-KE zwischen den beiden Ziel-Anforderungen.
- Der Test-zu-Test-Vergleich ermittelt schließlich die lokale Ähnlichkeit einer Testfall-KE zwischen den beiden Testfällen.

Im Gegensatz zum Wiederverwendungstrend kann die direkte lokale Ähnlichkeit für alle KE, d.h. sowohl für Anforderungs-KE als auch für Testfall-KE berechnet werden. Die einzige Bedingung ist, dass die beiden lokalen Ähnlichkeiten der Quell-KE und der Ziel-KE gemittelt werden. Würden die beiden direkten Ähnlichkeiten einer Anforderungs-KE nicht gemittelt werden, flössen die Testfall-KE nur mit dem halben Gewicht in die globale Ähnlichkeit ein.

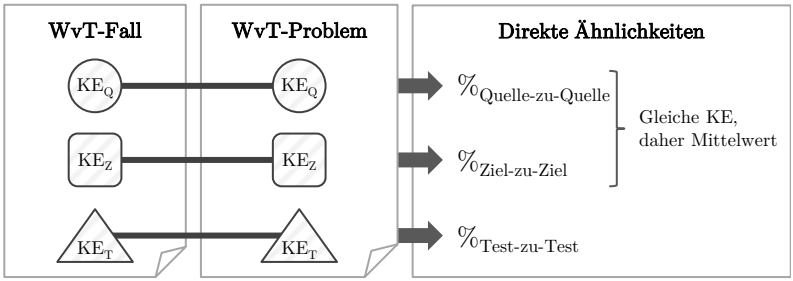


Bild 6.5.: Lokale direkte Ähnlichkeit

Globale Ähnlichkeit. Ob die lokale Ähnlichkeit einer Anforderungs-KE trendbasiert oder direkt ermittelt wird, wird bei der Falldefinition pro KE individuell festgelegt. Der Anwendungszweck bestimmt hierbei, welche Ähnlichkeitsfunktion zweckmäßig ist. Die globale Ähnlichkeit von Fall und Problem berechnet sich aus dem Produkt der lokalen Ähnlichkeiten.

Berechnungsbausteine

Der lokale Wiederverwendungstrend und die lokale direkte Ähnlichkeit greifen auf die gleichen Berechnungsbausteine zurück: *Binäres Enthaltensein analysieren* und *Ähnlichkeitsmaß anwenden*. Der Wiederverwendungstrend und der direkte Vergleich unterscheiden sich lediglich dadurch voneinander, welche *Ausprägungsmengen* A_1 und A_2 einer KE miteinander verglichen werden.

$$\begin{aligned} \text{BinE} &= (\text{a}, \text{b}, \text{c}, \text{d}) \\ \text{analysiereBinE} &:: \text{Menge A}_1 \rightarrow \text{Menge A}_2 \rightarrow \#\text{KE} \rightarrow \text{BinE} \\ \text{anwendeMaß} &:: \text{BinE} \rightarrow \text{Prozent} \end{aligned}$$

Binäres Enthaltensein analysieren. Das binäre Enthaltensein basiert auf dem zählenden Vergleich der Elemente zweier *Ausprägungsmengen* A_1 und A_2 einer KE. Es wird gezählt, wie viele Elemente in beiden Ausprägungsmengen und wie viele Elemente exklusiv in jeweils einer der Mengen enthalten sind. Der Datentyp *BinE* speichert das Ergebnis dieser Zählung. Er ist ein Vier-Tupel (a,b,c,d), welches die Anzahl der Ausprägungen enthält, die

(a) in beiden Ausprägungsmengen enthalten sind (binär: 11),

(b) exklusiv in der ersten Menge enthalten sind (binär: 10),

(c) exklusiv in der zweiten Menge enthalten sind (binär: 01) oder

(d) in keiner Menge enthalten sind (binär: 00) [Bro13, S.678].

Der binäre Charakter der Enthaltenseinsanalyse wird daran deutlich, dass ein Element in einer Ausprägungsmenge enthalten ist (1) oder nicht enthalten ist (0). Die Funktion *analysiereBinE* führt den Vergleich der beiden *Ausprägungsmengen* A_1 und A_2 durch und gibt *BinE* zurück. Die Funktion erhält als Eingabeparameter zudem die Anzahl aller bekannten Ausprägungen einer KE (*#KE*). Dies ist nötig, um die Anzahl der Ausprägungen zu berechnen, die in keiner der beiden Ausprägungsmengen enthalten sind. Das vierte Element *d* des Vier-Tupels *binE* ergibt sich aus $\text{d} = \#\text{KE} - \text{a} - \text{b} - \text{c}$. Das *d* ermöglicht die Verwendung von Maßen, die die Ähnlichkeit auf eine Prozentskala abbilden.

Ähnlichkeitsmaß anwenden. Die Funktion *anwendeMaß* führt ein binäres Ähnlichkeits- bzw. Distanzmaß aus. Ein solches Maß wandelt die Ergebnisse der Enthaltenseinsanalyse auf der Grundlage des Vier-Tupels (a,b,c,d) in einen Zahlenwert um [Bro13, S.678]. Dieser Zahlenwert zeigt die Ähnlichkeit bzw. Distanz der beiden Ausprägungsmengen einer KE an, aus denen das Vier-Tupel *BinE* ermittelt wurde. Ähnlichkeiten und Distanzen sind ineinander umwandelbar, wenn diese auf eine Prozentskala abbilden: Ähnlichkeit = 100% – Distanz. Es existieren viele verschiedene Ähnlichkeits- und Distanzmaße für binäre Daten [Bro13, S.703ff].

Ähnlichkeitsmaße unterscheiden sich darin, ob sie die Anzahl der Ausprägungen berücksichtigen, die in keiner der zu vergleichenden Mengen vorkommen. Diese Information werden in der Variable *d* von *BinE* gespeichert. Die Maße, die die Variable *d* von *BinE* in die Berechnung der Ähnlichkeit einbeziehen, sind in der Regel skaliert. Sie resultieren in einem prozentualen Ähnlichkeitswert. Die Ähnlichkeitsmaße, die die Variable *d* von *BinE* nicht einbeziehen, fordern keine Kenntnis über die Gesamtanzahl der Ausprägungen einer KE. Sie bilden die Ähnlichkeit somit immer auf eine offene Skala ab.

Im Anwendungskontext der fallbasierten Filterung soll der Vergleich eines WvT-Falls mit einem WvT-Problem unbedingt zu einer Zahl zwischen 0% und 100% führen. Es kommen somit nur Maße in Frage, die die Variable *d* von *BinE* berücksichtigen. Das Ähnlichkeitsmaß Rogers Tanimoto setzt sich beispielsweise wie folgt zusammen [Bro13, S.684]: $\frac{a+d}{a+2b+2c+d}$. Die Variablen der Formel entsprechen dem Vier-Tupel *BinE*. Die Maße, die auf eine offene Skala abbilden, enthalten keine Variable *d*.

Berechne Ähnlichkeit. Die Funktion *berechneÄ* führt die beiden Funktionen *analysiereBinE* und *anwendeMaß* nacheinander aus. Sie ermittelt aus den *Ausprägungsmengen* A_1 und A_2 sowie der Ausprägungsanzahl *#KE* das Vier-Tupel *BinE*. Die Funktion *berechneÄ* resultiert in einer Prozentzahl. Die Funktion genügt, um alle lokalen Ähnlichkeitsfunktionen zu definieren.

```
berechneÄ :: Menge A1 → Menge A2 → #KE → Prozent
```

Lokale Ähnlichkeitsfunktionen

Die folgenden Notationen erleichtern den Zugriff auf die Ausprägungsmengen in WvT-Fällen und WvT-Problemen. Da KE in WvT-Fällen ebenso gespeichert werden, wie in WvT-Problemen, kann auf sie mit derselben Notation zugegriffen werden. Der Parameter x steht für einen WvT-Fall oder ein WvT-Problem. Ein Zugriff gibt die Menge mit den Ausprägungen einer KE i zurück:

$$Q(x, i) = \text{Quell-Ausprägungsmenge der Anforderungs-KE i von x}$$
$$Z(x, i) = \text{Ziel-Ausprägungsmenge der Anforderungs-KE i von x}$$
$$T(x, i) = \text{Test-Ausprägungsmenge der Testfall-KE i von x}$$

Anzahl der Ausprägungen einer KE. Für jede KE i ist die Gesamtanzahl ihrer Ausprägungen $\#KE_i$ bekannt. Sie wird ermittelt, indem die Ausprägungen einer KE in der Falldatenbank und in den SLH bzw. der STS gezählt werden.

$$\#KE_i : \text{Anzahl aller Ausprägungen der KE i}$$

Lokaler Wiederverwendungstrend von Anforderungen. Der Wiederverwendungstrend $trendQZ(x, i)$ erhält einen WvT-Fall oder ein WvT-Problem x als Eingabe. Die Funktion führt mit den Quell- und Ziel-Ausprägungen einer Anforderungs-KE i einen Quelle-zu-Ziel-Vergleich durch. Unter Berücksichtigung von $\#KE_i$ wird die prozentuale lokale Ähnlichkeit berechnet.

$$trendQZ(x, i) = \text{berechneÄ}(\ Q(x, i)\ ,\ Z(x, i)\ ,\ \#KE_i)$$

Die Funktion $trendAnf(f, p, i)$ führt den internen Quelle-zu-Ziel-Vergleich der KE i sowohl für den WvT-Fall f als auch für das WvT-Problem p durch. Der Abstand der beiden internen Trends $|trendQZ(f, i) - trendQZ(p, i)|$ zeigt an, wie stark sich die Quelle-zu-Ziel-Trends von f und p ähneln.

$$trendAnf(f, p, i) = 100\% - |trendQZ(f, i) - trendQZ(p, i)|$$

Lokale direkte Anforderungsähnlichkeit. Die Funktion direktQQ(f, p, i) erhält einen WvT-Fall f und ein WvT-Problem p als Eingabe. Die Funktion verwendet die Quell-Anforderungen von f und p, um mit einer Anforderungs-KE i einen Quelle-zu-Quelle-Vergleich durchzuführen. Analog dazu führt die Funktion direktZZ(f, p, i) einen Ziel-zu-Ziel-Vergleich durch. Beide Funktionen stellen die Ähnlichkeit mit Hilfe der Gesamtanzahl $\#KE_i$ prozentual dar.

$$\begin{aligned} \text{direktQQ}(f, p, i) &= \text{berechneÄ}(\ Q(f, i)\ ,\ Q(p, i)\ ,\ \#KE_i) \\ \text{direktZZ}(f, p, i) &= \text{berechneÄ}(\ Z(f, i)\ ,\ Z(p, i)\ ,\ \#KE_i) \end{aligned}$$

Die beiden Funktionen direktQQ(f, p, i) und direktZZ(f, p, i) berechnen für eine Anforderungs-KE i zweimalig die lokale direkte Ähnlichkeit: Einmal von Quelle-zu-Quelle und einmal von Ziel-zu-Ziel. Die beiden lokalen direkten Ähnlichkeitswerte werden in der Funktion direktAnf(f, p, i) gemittelt. Diese Mittelwertbildung ist für die korrekte Ermittlung der globalen Ähnlichkeit nötig. Flössen beide lokalen Werte ein, würden in der globalen Produktfunktion Anforderungs-KE doppelt bzw. Testfall-KE nur halb gewertet werden.

$$\text{direktAnf}(f, p, i) = \frac{\text{direktQQ}(f, p, i) + \text{direktZZ}(f, p, i)}{2}$$

Lokale direkte Testfallähnlichkeit. Die Testfallähnlichkeit ist die letzte und einfachste Funktion zur Berechnung der lokalen Ähnlichkeit. Sie ist einfach, weil jede Testfall-KE im Gegensatz zu Anforderungs-KE sowohl in WvT-Fällen als auch in WvT-Problemen nur über eine einzige Ausprägungsmenge verfügt. Die Funktion direktTT(f, p, i) erhält einen WvT-Fall f, ein WvT-Problem p und eine Testfall-KE i als Eingabe. Die Funktion führt mit der Testfall-KE i einen Testfall-zu-Testfall-Vergleich zwischen f und p durch. Da für Testfall-KE kein Abstand oder Mittelwert gebildet werden muss, gibt direktTest(f, p, i) lediglich das Ergebnis von direktTT(f, p, i) zurück.

$$\begin{aligned} \text{direktTT}(f, p, i) &= \text{berechneÄ}(\ T(f, i)\ ,\ T(p, i)\ ,\ \#KE_i) \\ \text{direktTest}(f, p, i) &= \text{direktTT}(f, p, i) \end{aligned}$$

Globale Ähnlichkeit. Die lokalen prozentualen Ähnlichkeiten werden zu einer globalen Ähnlichkeit zusammengefasst. Der WvT-Fall f bestimmt hierbei die relevanten KE. Das heißt, dass genau die KE einfließen, die f besitzt. Hat ein WvT-Problem zusätzliche KE, so spielen diese während der Berechnung keine Rolle. Die fehlenden KE des WvT-Problems fließen in die lokalen Ähnlichkeitsfunktionen als leere Ausprägungsmengen ein. Die Mengen X_f, Y_f und Z_f enthalten die trendbasiert bzw. direkt zu ermittelnden KE von f.

$$
\begin{aligned}
X_f &= \text{Menge trendbasierter Anforderungs-KE von WvT-Fall f} \\
Y_f &= \text{Menge direkter Anforderungs-KE von f} \\
Z_f &= \text{Menge direkter Testfall-KE von f}
\end{aligned}
$$

Die Funktion globaleÄ(f, p) multipliziert die lokalen Ähnlichkeiten aller KE des Falls f und des Problems p. Die Funktion ist aus drei Produktfunktionen zusammengesetzt. Ein WvT-Fall hat zunächst die trendbasiert zu vergleichenden Anforderungs-KE X_f. Die erste Produktfunktion $\prod_{i_x \in X_f} \text{trendAnf}(f, p, i_x)$ multipliziert die lokalen Ähnlichkeiten aller KE aus X_f. Analog dazu verfahren die beiden anderen Produktfunktionen für Y_f und Z_f.

$$
\begin{aligned}
\text{globaleÄ}(f, p) = & \prod_{i_x \in X_f} \text{trendAnf}(f, p, i_x) * \\
& \prod_{i_y \in Y_f} \text{direktAnf}(f, p, i_y) * \\
& \prod_{i_z \in Z_f} \text{direktTest}(f, p, i_z)
\end{aligned}
$$

Beispielrechnung. In Anhang D werden ab Seite 151 verschiedene Konfigurationen von lokalen Ähnlichkeiten X_f, Y_f und Z_f beispielhaft durchgerechnet. Die Beschreibung der einzelnen Rechenschritte zeigt, wie die verschiedenen lokalen Ähnlichkeitsfunktionen in konkreten lokalen und globalen prozentualen Ähnlichkeitswerten resultieren. Zudem wird in Anhang D dargestellt, dass unterschiedliche Einsatzszenarien für lokale Wiederverwendungstrends und direkte Ähnlichkeiten von Anforderungs-KE existieren.

6.3. Falluntersuchung

In Anhang D wird gezeigt, dass es rechnerisch möglich ist, WvT-Fälle und WvT-Probleme hinsichtlich der Ähnlichkeit ihrer KE zu vergleichen. Dies ermöglicht es, WvT-Fälle in der Falldatenbank zu definieren und ähnliche WvT-Probleme in den Spezifikationsdokumenten automatisch zu erkennen. Dieses Vorgehen entspricht einem regelbasierten Vorgehen: Mit Hilfe der Fälle werden Regeln definiert, deren Eintreten durch die Anwendung der vorgestellten Ähnlichkeitsfunktionen überprüft wird.

Die fallbasierte Filterung baut auf Mechanismen aus dem Bereich des fallbasierten Schließens (engl.: case-based reasoning, CBR) auf. CBR ist vielseitig einsetzbar. Im betrachteten Umfeld sind jedoch noch einige Herausforderungen zu meistern, bevor die fallbasierte Filterung im Sinne von CBR eingesetzt werden kann. Neben diesen Herausforderungen wird im nachfolgenden Unterkapitel 6.4 auch auf die vielseitige Einsetzbarkeit von CBR eingegangen.

Im Rahmen der Falluntersuchung sollen in diesem Unterkapitel interessante WvT-Fälle aufgezeigt werden. Dies soll die weitere Forschung zur Integration der WvT-Verlinkung und CBR motivieren. Dazu werden existierende klassifizierende Eigenschaften in realen $\mathrm{SLH}_{\mathrm{Quell}}$, $\mathrm{SLH}_{\mathrm{Ziel}}$ und STS betrachtet. Mit den vorgefundenen Eigenschaften werden WvT-Fälle formuliert und jeweils der Nutzen für die WvT-Verlinkung beschrieben. Aus der Anzahl der betrachteten klassifizierenden Eigenschaften ergeben sich zwei WvT-Fallgruppen:

1. Einfache WvT-Fälle verwenden eine Anforderungs-KE und eine Testfall-KE, um die Artefakte des WvT-Problems zu klassifizieren.
2. Komplexe WvT-Fälle verwenden mehrere Anforderungs-KE und/oder mehrere Testfall-KE, um kompliziertere Sachverhalte darzustellen.

Die WvT-Verlinkung ist implementiert, ausgerollt und wird wegen ihrer Effizienz in der Praxis angewendet. Die fallbasierte Filterung ist zwar auch implementiert, sie ist aber wegen der Heterogenität in der Verwendung der KE noch nicht anwendbar. Die Falluntersuchung soll belegen, dass die fallbasierte Filterung dazu in der Lage ist, die WvT-Verlinkung sinnvoll zu erweitern.

Schnittstellen und Schnittstellentest

Dieser WvT-Fall wurde bereits mehrfach in diesem Kapitel aufgegriffen. Er verwendet die Anforderungs-KE *Schnittstelle* und die Testfall-KE *Testziel*. Die Schnittstellen einer Systemanforderung definieren, von welchen anderen Systemen die Anforderung funktional abhängt. Mit Hilfe der Testfall-KE *Testziel* wird geprüft, ob der Testfall die *Schnittstellen absichert*. Der WvT-Fall ähnelt allen WvT-Problemen, in denen sich die Schnittstellen der Ziel-Anforderung geändert haben und der Testfall die Schnittstellen absichert.

Nutzen für die WvT-Verlinkung. Bild 6.6 zeigt einen WvT-Fall, dessen Quell-Anforderung eine Schnittstelle zum Zündschloss hat, um zu prüfen, ob sich der Zündschlüssel in Zündstellung befindet. In der Ziel-Anforderung wurde eine neue Schnittstelle zum Motorsteuergerät geschaffen, um zusätzlich zu prüfen, mit welcher Drehzahl der Motor aktuell läuft. Dieser WvT-Fall nutzt der WvT-Verlinkung, indem er Testfälle ermittelt, die geänderte Schnittstellen absichern. Im direkten Vergleich kann ermittelt werden, welche Schnittstellen sich konkret geändert haben. Daher könnte auch der Prüfhinweis wesentlich detaillierter sein. Der WvT-Fall kann jedoch auch trendbasiert, d.h. unabhängig von den konkreten Ausprägungen, genau die Ziel-Anforderungen finden, in denen eine Schnittstelle identisch blieb und eine neue Schnittstelle hinzukam.

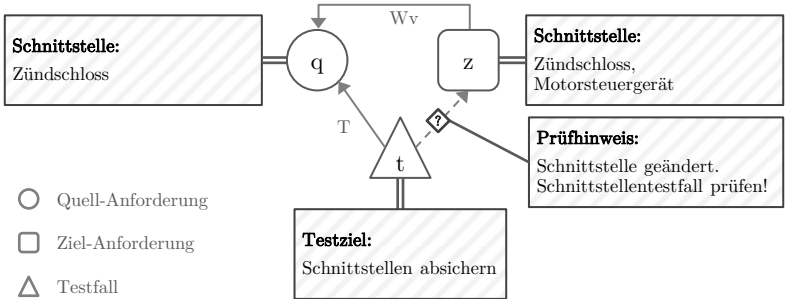


Bild 6.6.: Schnittstellen und Schnittstellentest

ASIL-Einstufung und Sicherheitstest

Dieser WvT-Fall ähnelt allen WvT-Problemen, in denen sich die Sicherheitseinstufung der Ziel-Anforderung geändert hat und in denen die funktionale Sicherheit abgesichert wird. Dafür verwendet er die Anforderungs-KE *Sicherheitskritikalität*. Zudem verwendet der Fall die Testfall-KE *Testziel*, um zu prüfen, ob der Testfall die *funktionale Sicherheit* (FuSi) absichert. Wenn ein WvT-Problem gefunden wurde, das diesem WvT-Fall entspricht, wird der Entwickler auf die veränderte Sicherheitseinstufung hingewiesen. Eine manuelle Überprüfung der verlinkten FuSi-Testfälle ist dann nötig.

Nutzen für die WvT-Verlinkung. Funktionale Sicherheit spielt wegen der ISO 26262 eine immer bedeutendere Rolle in der Automobilbranche. Kapitel 5 setzte sich damit technisch und rechtlich auseinander. Eine aktuelle Promotion widmet sich dem Thema der Wiederverwendung von FuSi-Analysen. Das Ziel ist es, wiederverwendete Ziel-Anforderungen mit wiederverwendeten FuSi-Ergebnisse in Beziehung zu setzen. Bild 6.7 zeigt einen WvT-Fall, in dem sich die Sicherheitskritikalität der Ziel-Anforderung erhöht hat. Diese konkrete Erhöhung von ASIL C auf ASIL D kann mit Hilfe des direkten Vergleichs ermittelt werden. Der Wiederverwendungstrend ermöglicht es, Änderungen beliebigen Abstands in beiden Richtungen zu ermitteln.

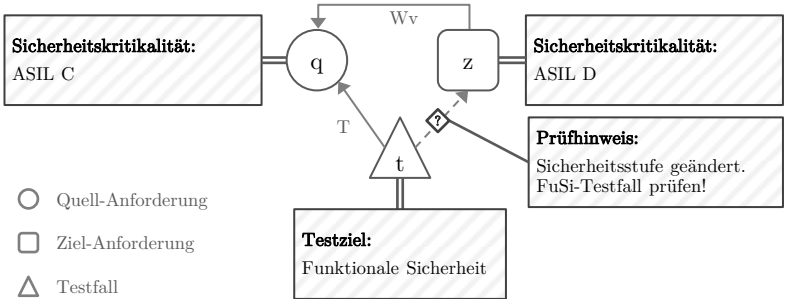


Bild 6.7.: ASIL-Einstufung und Sicherheitstest

Anforderungs- und Testfalleigentümer

Es existieren viele klassifizierende Eigenschaften in den betrachteten Systemlastenheften und Testspezifikationen. Dieser WvT-Fall soll jedoch der letzte einfache Fall sein, der im Rahmen der Falluntersuchung betrachtet wird. Jede Anforderung und jeder Testfall verfügt über einen Eigentümer, dessen Aufgaben, Kompetenzen und Verantwortlichkeiten um diese Artefakte kreisen. Der WvT-Fall aus Bild 6.8 verwendet die Anforderungs-KE und die Testfall-KE *Eigentümer*, um anforderungsseitig einen Eigentümerwechsel zu erkennen.

Nutzen für die WvT-Verlinkung. Bild 6.8 zeigt, dass der Eigentümer der Quell-Anforderung nicht derselbe ist, wie der Eigentümer der Ziel-Anforderung. Die Übergabe eines Systemlastenhefts findet beispielsweise bei einem Generationenwechsel statt, wenn junge Mitarbeiter die Aufgaben von alten Mitarbeitern übernehmen. In diesem Zusammenhang muss auch die Verantwortung über die Testfälle abgegeben werden. Der dargestellte WvT-Fall erkennt im direkten Vergleich alle WvT-Probleme, deren Quell-Anforderung und Testfall im Verantwortungsbereich von *Thomas* lagen und deren Ziel-Anforderung nun durch *Jonathan* verantwortet wird. Die fallbasierte Filterung erweitert die WvT-Verlinkung, indem sie *Jonathan* darauf hinweist, dass er fortan in der Pflicht ist, sich auch um die Testfälle des „Alten“ zu kümmern.

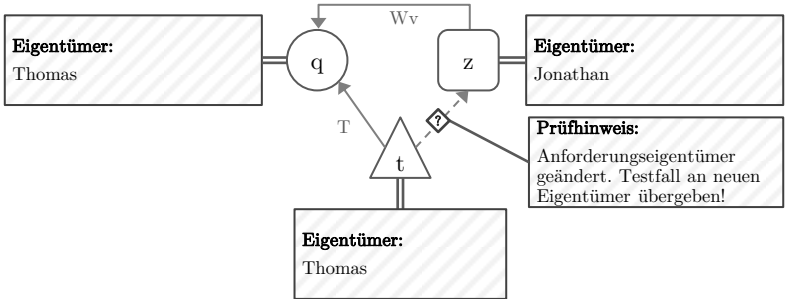


Bild 6.8.: Anforderungs- und Testfalleigentümer

Viele weitere WvT-Fälle

Große Konzerne blicken häufig auf eine lange Geschichte zurück. Während dieser Zeit entwickelten sich viele individuelle Herangehensweisen. Im betrachteten Umfeld gibt es in diesem Zusammenhang viele KE, die in den Systemlastenheften und Systemtestspezifikationen des einen Bereichs verwendet werden, während ein anderer Bereich ganz andere KE benutzt. In internen Arbeitskreisen kann diese Vielfalt in bereichsspezifischen Falldatenbanken zu unterschiedlichsten WvT-Fällen führen. In der Falluntersuchung wurde nur ein kleiner Einblick in die Möglichkeiten der fallbasierten Filterung geboten.

Komplexere WvT-Fälle. In der Falluntersuchung wurden zudem nur einfache WvT-Fälle betrachtet. Im komplexeren WvT-Fall aus Bild 6.9 hat die Ziel-Anforderung zunächst eine neue *Schnittstelle* zum Motorsteuergerät. Gleichzeitig verwendet der WvT-Fall die Anforderungs-KE *Lieferantenstatus*, um auszudrücken, dass der Lieferant die Ziel-Anforderung nicht mehr akzeptiert. Im WvT-Fall akzeptiert der Lieferant die Ziel-Anforderung nicht, weil er Schnittstellentestfälle nicht ausführen kann, um die Ziel-Anforderung abzusichern. Dieses abschließende Beispiel soll verdeutlichen, dass weitere große Herausforderungen entstehen, wenn WvT-Fälle mehrere KE enthalten.

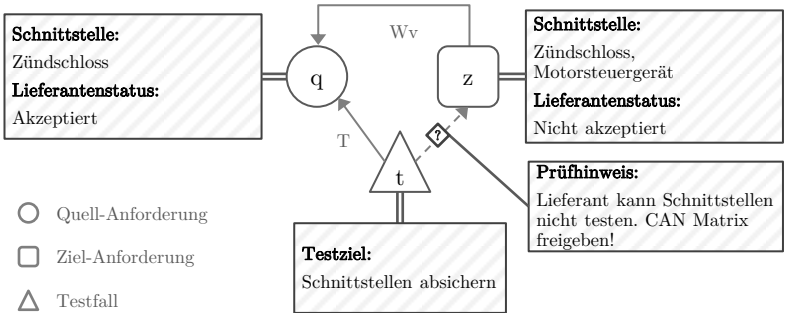


Bild 6.9.: Komplexerer WvT-Fall

6.4. Fallbasierte Filterung und Case-Based Reasoning (CBR)

Das fallbasierte Schließen (engl.: case-based reasoning, CBR) basiert auf zwei einfachen Prinzipien: (1) Ähnliche Probleme besitzen ähnliche Lösungen und (2) ähnliche Probleme treten immer wieder auf [Lea96, S.1]. Diese Prinzipien lassen sich sehr einfach und sogar direkt auf die WvT-Verlinkung übertragen: (1) Ähnliche WvT-Probleme besitzen ähnliche Lösungen und (2) ähnliche WvT-Probleme treten immer wieder auf.

6.4.1. Allgemeine Funktionsweise

Ein Fall ist die Repräsentation eines früher aufgetretenen Problems und seiner Lösung. Jeder Fall war demnach zu einem früheren Zeitpunkt ein Problem, das gelöst wurde. Tritt ein neues Problem auf, wird es in die Repräsentation konvertiert, in der auch die Fälle in der Falldatenbank gespeichert sind. Dies ermöglicht es, mit Ähnlichkeitsfunktionen für ein aktuelles Problem ähnliche Fälle in der Falldatenbank zu finden. Die gespeicherte Lösung in einem ähnlichen Fall wird schließlich auch zur Lösung des neuen Problems verwendet.

Methodenschritte. Bild 6.10 zeigt links die vier allgemeinen Schritte der CBR-Methode und rechts die Realisierung dieser Schritte in der WvT-Verlinkung. Im ersten Schritt werden die ähnlichen Fälle für ein aktuelles Problem ermittelt (engl.: retrieve). Wurde ein ähnlicher Fall gefunden, wird der gespeicherte Lösungsvorschlag zur Lösung des aktuellen Problems wiederverwendet (engl.: reuse). Im dritten Schritt wird die Lösung des neuen Problems geprüft und angepasst (engl.: revise). Im letzten Schritt wird das gelöste Problem als neuer Fall in die Falldatenbank aufgenommen (engl.: retain). Der neue Fall steht dann für zukünftige Suchen bzw. CBR-Zyklen zu Verfügung.

Repräsentationsformen. Die Ablageform bestimmt, wie Fälle später wieder aufgefunden werden. Daher ist sie ein wesentlicher Bestandteil von CBR. Es existieren verschiedene Herangehensweisen hinsichtlich der Ablage [BKP06, S.209f]: Repräsentation als Merkmalsvektor sowie strukturierte und textuelle Repräsentation. WvT-Fälle entsprechen der strukturierten Repräsentation.

Retrieve: Allgemein. Die CBR-Methode beginnt mit der Transformation eines aktuellen Problems in eine, der Ablageform der Fälle entsprechende Problemrepräsentation. Diese Problemrepräsentation dient als Eingabe einer Ähnlichkeitsfunktion, um in der Falldatenbank mittels paarweiser Vergleiche oder Entscheidungsbäumen nach ähnlichen Fällen zu suchen [Wat99].

Reuse: Allgemein. Anschließend werden die Lösungsvorschläge der ähnlichen Fälle zur Lösung des aktuellen Problems wiederverwendet. Hierbei können die Ähnlichkeiten und Unterschiede von Fällen und Problemen berücksichtigt werden, um die Zuverlässigkeit der Lösungsvorschläge zu bewerten.

Revise: Allgemein. Es ist nicht davon auszugehen, dass CBR ausschließlich korrekte Lösungsvorschläge für neue Probleme unterbreitet. Um einen Lerneffekt zu erzielen, werden falsche Anwendungen früherer Lösungen auf aktuelle Probleme in diesem Schritt nicht nur entdeckt, sondern auch korrigiert.

Retain: Allgemein. CBR schließt mit dem Erfahrungsaufbau ab. Es wird ermittelt, ob das gelöste Problem zu einem Erkenntnisgewinn beiträgt und somit als neuer Fall in die Falldatenbank aufgenommen werden kann.

Retrieve: WvT. Die klassifizierten WvT-Fälle entsprechen strukturell den klassifizierten WvT-Problemen. Mittels der Analyse der Klassifikationsübereinstimmung werden für WvT-Probleme die ähnlichsten WvT-Fälle ermittelt.

Reuse: WvT. Das WvT-Problem beschreibt eine Situation, in der entschieden werden muss, ob eine WvT-Verlinkung durchführbar ist. Diese Entscheidung wird als WvT-(Verlinkungs)Entscheidung bezeichnet. Jeder WvT-Fall enthält eine solche Entscheidung. Der zweite CBR-Schritt wendet die Verlinkungsentscheidung eines WvT-Falls auf ein ähnliches WvT-Problem an.

Revise: WvT. Im dritten CBR-Schritt wird die Korrektheit der mittels Entscheidungsübernahme gelösten WvT-Probleme manuell überprüft.

Retain: WvT. Die durch Entscheidungsübernahme gelösten WvT-Probleme könnten nach der Überprüfung automatisch in die Falldatenbank aufgenommen werden. Für die praktikable Erweiterung der WvT-Verlinkung gelten jedoch einige Einschränkungen, auf die auf den Seiten 129f eingegangen wird.

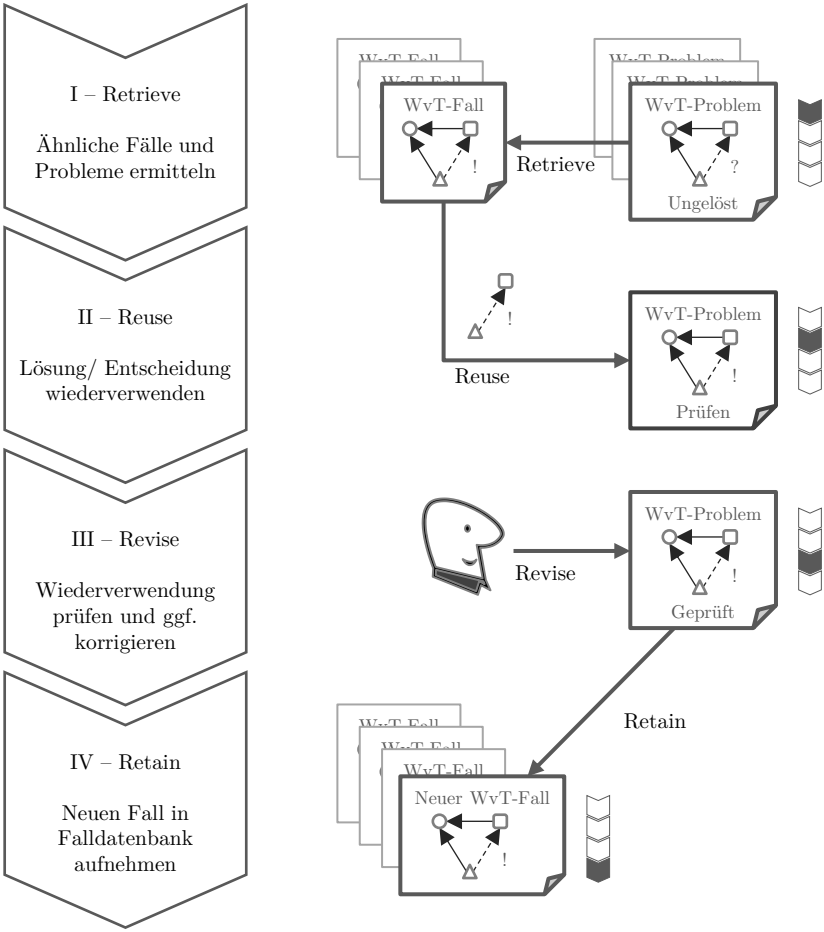


Bild 6.10.: CBR-Methode gemäß [AP94, S.8]

6.4.2. Verbreitung von CBR

Das fallbasierte Schließen (CBR) ist eine vielseitig einsetzbare Methode, die auch tatsächlich vielseitig eingesetzt wird. Ihr Einsatz lässt sich in zahlreichen Bereichen seit vielen Jahren beobachten. Mediziner, Maschinenbauer und auch Softwaretechniker treten häufig als CBR-Anwender auf.

Anwendungen in der Medizin. In der Medizin wird CBR für die Diagnose verwendet, um von Symptomen auf Krankheiten und Behandlungsmethoden zu schließen [BR09, S.327]. Dies deckt sich direkt mit den beiden CBR-Grundsätzen: (1) Ähnliche Krankheiten werden ähnlich behandelt und (2) ähnliche Krankheiten treten immer wieder auf. CBR eignet sich im Medizinbereich besonders, da Ärzte in derselben Art auf Krankheiten schließen, wie es die CBR-Methode vorsieht [BSLH99, S.79]. Die Suche nach Behandlungsmethoden für vorstellige Patienten findet in einer Falldatenbank statt, in der Anamnesedaten früherer Patienten gespeichert sind. Neuere Arbeiten geben Beispiele für Domänen, in welchen CBR in der Medizin angewendet wird. Diese beschäftigen sich mit der Entwicklung von Werkzeugen für die Domänen Leukämie [CD08], Stressmanagement [ABO$^+$10] und Diabetes [MWC$^+$11].

Anwendungen im Ingenieurwesen. CBR findet im Ingenieurwesen in jeder Phase des Produktlebenszyklus Anwendung, von dem Produktentwurf über die Produktionsprozesse bis hin zur Produktwartung. Im Produktentwurf wird CBR eingesetzt, um Entwurfsfehler zu vermeiden und gutes Produktdesign zu fördern. Konkrete Einsatzgebiete sind hier die Berücksichtigung der Umweltverträglichkeit im allgemeinen Produktentwurf [JL11] oder die Sicherstellung der Kompatibilität von Komponenten im Kraftwerksentwurf in Hinblick auf spezifizierte Leistungsparameter [MBdS11]. In Produktionsprozessen wird CBR eingesetzt, um Parameter, beispielsweise in der Textilproduktion [VSM12] oder in der Pirelli Reifenproduktion [BCSV04], zu optimieren. Schließlich findet CBR auch in der Produktwartung Einsatz, unter anderem, um von Sensordaten und mechanischen Symptomen von Boeing Flugzeugtriebwerken [Hei96] oder General Electric Gasturbinen [YC12] auf potentielle Probleme und auch Reparaturanweisungen zu schließen.

Weitere verblüffende Anwendungen. Es ist durchaus überraschend, in welchen weiteren Domänen sich Anwendungen von CBR finden lassen. In der Massentierhaltung wird es verwendet, um auf Gründe erhöhter Sterblichkeit in Fischfarmen zu schließen [TBA12]. In der Forensik wird CBR eingesetzt, um Fehler in der Interpretation von Beweisen zu vermeiden [HLV12]. Auch im Bereich der Sozioökonomie findet CBR Anwendung, um schon frühzeitig abschätzen zu können, welche Schadenssummen durch Naturkatastrophen zu erwarten sind [ZWZX12]. Die Liste interessanter Anwendungen ließe sich, wenn gewollt, beliebig erweitern. Dies verdeutlicht die Anerkennung und domänenübergreifende Verbreitung der CBR-Mechanismen.

Anwendungen in der Softwaretechnik. Die internationale Konferenz für Case-Based Reasoning (ICCBR) und der seit 2008 integrierte Europäische Workshop für CBR (EWCBR) spiegeln die Trends in CBR und dessen Anwendung wider. Die Sichtung der Titel der Veröffentlichungen von 2014 bis zurück nach 1993 lässt Tendenzen in der CBR-Anwendung erkennen [ICCBR][EWCBR]. Zur Anwendung in der Softwaretechnik gab es vermehrt Papiere von den ersten Konferenzen bis zum Jahr 2003. In diesem Zeitraum gab es Veröffentlichungen, die thematisch das gesamte V-Modell abdecken. Der Einsatzzweck von CBR war zumeist die Unterstützung der Wiederverwendung von Softwareartefakten. Bereits 1988 erkannte Will Tracz, ein früher Wegbereiter der Wiederverwendungsthematik im Bereich der Softwaretechnik, dass Wiederverwendung von den Mechanismen des maschinellen Lernens profitieren kann [Tra88, S.18]. Im Umfeld der ICCBR gab es zuletzt weniger Veröffentlichungen zur Anwendung von CBR in der Softwaretechnik.

Wiederverwendung von Rollenmodellen. Im Softwaredesign werden Modelle verwendet, die den Aufbau und die Funktion eines Softwaresystems beschreiben. CBR wird eingesetzt, um Muster aus diesen Modellen zu extrahieren und nach ähnlichen Mustern in der Falldatenbank zu suchen. Rollenmodelle werden beispielsweise verwendet, um Akteure eines Systems und die Interaktionen zwischen ihnen zu beschreiben. Die Knoten und Kanten eines entstehenden Rollenmodells werden analysiert, um in der Falldatenbank nach ähnlichen Rollenmodellen zu suchen und diese wiederzuverwenden [GPP+03].

Wiederverwendung durch Klassen- und Datenflussdiagramme. Ein weiterer modellbasierter Ansatz verwendet UML-Klassendiagramme [Bjo03]. Während der Erstellung neuer Diagramme werden die Beziehungen zwischen den entstehenden Klassen und ihren Variablen und Methoden als Ausgangssituation verwendet, um in einer Falldatenbank nach ähnlichen Designs zu suchen und für die Wiederverwendung vorzuschlagen. Ein weiterer Ansatz verwendet Datenflussdiagramme, um in einer Falldatenbank nach C-Blöcken zu suchen, die diese Diagramme umsetzen [FM93]. Die Quelltextfragmente werden anschließend automatisch zusammengesetzt.

Wiederverwendung von Quelltext. Auch bei der Wiederverwendung von Quelltext gibt es interessante Anwendungen von CBR. Das generelle Vorgehen ist es, die Eigenschaften gerade entstehender Code-Bestandteile zu extrahieren und in der Falldatenbank nach ähnlichen Code-Abschnitten zu suchen. Die Idee ist es, CBR zur entwicklungsbegleitenden Unterstützung zu verwenden. Ein Ansatz extrahiert lexikalische und semantische Informationen aus Smalltalk-Klassen [FCGCGAHY96], um nach ähnlichen Klassen in der Falldatenbank zu suchen. Ein anderer Ansatz extrahiert Methodensignaturen, Member-Variablen, Vererbungsbeziehungen sowie die Namen der Variablen, Methoden und Klassen von entstehendem Java-Quelltext, um die Auto-Vervollständigung ganzer Methoden und Klassen zu ermöglichen [TWP98].

Wiederverwendung von Testfällen. Die Sichtung der Papiertitel im Umfeld der Fachkonferenzen ICCBR und EWCBR bis zum Jahr 2014 ergab eine Arbeit, die sich mit der Anwendung von CBR auf die Testfallwiederverwendung auseinandersetzt [MH00]. Testfälle werden in Sektionen spezifiziert. Mögliche Sektionen sind neben Informationen zur Funktionsweise des Testobjekts auch Testschritte und das erwartete Ergebnis. Jeder Testfall wird in der Falldatenbank abgespeichert. Die Besonderheit der Arbeit von Frau Minor ist, dass die verschiedenen Reifegrade, die ein Testfall während seiner Erstellung durchläuft, berücksichtigt werden. Die Herausforderung besteht darin, dass die Ähnlichkeitsfunktion Testfälle unterschiedlicher Reifegrade miteinander vergleicht und auch für neue Testfälle mit unvollständigen oder leeren Sektionen ähnliche, vollständig spezifizierte Testfälle in der Falldatenbank auffindet.

6.4.3. Besonderheiten der Integration von CBR und WvT

Die Erweiterung der WvT-Verlinkung um CBR basiert auf den klassifizierenden Eigenschaften der Artefakte des WvT-Problems. Zuvor wurde gezeigt, dass CBR vielseitig einsetzbar ist und auch tatsächlich vielseitig eingesetzt wird. Trotzdem gelten einige Einschränkungen im Zusammenhang mit dem Umfeld, in dem die WvT-Verlinkung durchgeführt wird.

Während einige Systemverantwortliche die Klassifikation von Anforderungen und Testfällen fordern, fordern andere Verantwortliche dies nicht. Selbst wenn in den SLH zweier Systeme Anforderungen klassifiziert sind, sind die Ausprägungen von gleichen klassifizierenden Eigenschaften oftmals nicht einheitlich benannt. Die bedingte Anwendbarkeit ergibt sich aus der heterogenen und unvollständigen Artefaktklassifikation. Ein weiterer Aspekt, der die Integration von CBR als Vollautomatismus erschwert, ist die Nachvollziehbarkeit von Entscheidungen. Selbstlernende Systeme sind hierbei als kritisch zu erachten, weil getroffene Entscheidungen nicht zwingend nachvollziehbar sind.

Fallbasiertes Ausschließen statt Schließen. In Konzernen mit vielen verteilt arbeitenden Teams existieren unterschiedliche Vorgehensweisen. Weil die Artefakte des WvT-Problems heterogen und unvollständig klassifiziert sind, ist das fallbasierte Schließen im gegebenen Umfeld momentan nicht umsetzbar.

Zur Verdeutlichung wird der Beispielfall von Seite 119 erneut aufgegriffen: Die Schnittstellen haben sich von der Quell- zur Ziel-Anforderung geändert und ein Schnittstellentestfall soll mit der Ziel-Anforderung verlinkt werden. Mit Hilfe dieses WvT-Falls kann explizit ausgeschlossen werden, dass der Testfall ohne Überprüfung verlinkbar ist. Auf der anderen Seite kann bei identischen Schnittstellen aber nicht explizit darauf geschlossen werden, dass der Testfall ohne Überprüfung übernommen werden kann. Der Grund dafür ist, dass viele weitere klassifizierende Eigenschaften existieren, die die Verlinkungsentscheidung direkt oder indirekt beeinflussen können. Somit lassen sich lediglich bestimmte Muster in den WvT-Problemen erkennen, anhand derer eine Verlinkung als fragwürdig eingestuft werden kann. Dieses Vorgehen dient nicht dem Schluss, sondern dem systematischen Ausschluss von Verlinkungen.

Beide Suchrichtungen. Im CBR-Standardvorgehen wird ein aktuelles Problem als Eingabe der Suchfunktion verwendet, um in der Falldatenbank nach einer Menge ähnlicher Fälle zu suchen. Die fallbasierte Filterung greift während der WvT-Verlinkung auch auf diese Suchrichtung zurück. Im Gegensatz dazu wird die Suchrichtung umgekehrt, wenn die Gesamtverlinkungssituation hinsichtlich der Falldatenbank bewertet wird. Ein WvT-Fall aus der Falldatenbank dient dann als Eingabe der Suchfunktion, um nach der Menge ähnlicher WvT-Probleme zu suchen. In der Anwendung resultiert dies in der Anweisung „Suche im SLH_{Ziel} nach allen Ziel-Anforderungen, die sich in einem ähnlichen WvT-Problem befinden, wie dieser WvT-Fall“. Unabhängig von der Suchrichtung gilt, dass der WvT-Fall stets die Menge der klassifizierenden Eigenschaften definiert, die auch das WvT-Problem besitzen muss.

Keine volle Automatisierung. Gespräche mit Entwicklern bestätigen die folgende Aussage: Jede Form eines maschinellen Eingriffs in Entwicklungsprozesse muss nachvollziehbar und vor allem auch korrekt sein. Bereits in der ersten Methodenschicht war aus diesem Grund ein Kompromiss unumgänglich: Die Entwickler forderten, dass die textuelle Ähnlichkeit bei der WvT-Verlinkung zwar berücksichtigt werden muss, der Grenzwert aber bei 100% liegen sollte. Da ein manuelles Review der SLH gefordert ist - und dies betrifft auch die Verlinkung - hilft es bereits deutlich, wenn die nicht geänderten Anforderungen markiert werden, so dass sich hier der Prüfaufwand verringert. Entwickler sicherheitskritischer Systeme führen nun fort, dass dies auch für selbstlernende Systeme gilt: Im Vergleich zu einem selbstlernenden System ist es nachvollziehbarer, wenn Experten in einem Arbeitskreis regelmäßig zusammentreffen, um WvT-Fälle im Sinne von „Best Practices“ zu definieren. Neben dem Vorteil der gesteigerten Nachvollziehbarkeit führt dies zu einer Reduzierung des Aufwands während der Werkzeugqualifizierung.

Teilautomatischer fallbasierter Ausschluss. CBR ist im betrachteten Umfeld momentan nicht vollautomatisch umsetzbar. Es konnte jedoch gezeigt werden, dass die WvT-Verlinkung zumindest um die ersten beiden CBR-Schritte erweitert werden kann. Zukünftige empirische Untersuchungen könnten offenlegen, inwiefern die verbleibenden CBR-Schritte umsetzbar sind.

6.4.4. Aktuelle Einsatzmöglichkeiten

Die ersten beiden CBR-Schritte aus Bild 6.10 von Seite 125 wurden als Vollautomatismus umgesetzt. Es ist technisch möglich, klassifizierte WvT-Fälle und WvT-Probleme miteinander zu vergleichen (engl.: retrieve) und die WvT-Entscheidung des Falls auf das Problem anzuwenden (engl.: reuse). Offen blieb bislang, wie die verbleibenden CBR-Schritte methodisch umsetzbar sind.

Revise: Manuell durch Entwickler. In Qualitätsprozessen wird definiert, wann und in welchem Umfang Spezifikationsdokumente vor der Freigabe geprüft werden müssen (engl.: quality gate, QG). Die Prüfung schließt auch die Verlinkung zwischen Testfällen und Anforderungen ein. Die fallbasierte Filterung erleichtert den Vorgang: Sie markiert zu prüfende Ziel-Anforderungen mit einem Prüfvermerk und einer natürlichsprachlichen Prüfanweisung. Das Durchschreiten des QG entspricht exakt der Definition des CBR-Schritts *revise*.

Retain: Manuell im Arbeitskreis. In einem über 100 Jahre alten Großkonzern existieren viele und methodisch sehr weit voneinander entfernte Fachbereiche. Ein kleinster gemeinsamer Nenner ist das allgemeine WvT-Problem: Testfälle müssen mit Anforderungen verlinkt sein - unabhängig vom Fachbereich. Ein weiterer gemeinsamer Nenner ist, dass klassifizierende Eigenschaften grundsätzlich existieren. Diese sind vereinzelt einheitlich, überwiegend aber fachbereichsspezifisch. Diese Heterogenität erfordert fachbereichsspezifische Arbeitskreise, die die relevanten WvT-Fälle für den Fachbereich definieren. Gemäß Standardvorgehen in großen Unternehmen müsste dann ein übergeordneter Steuerkreis alle fachbereichsspezischen Falldatenbanken nach allgemeingültigen Fällen durchsuchen und diese bereichsübergreifend ausrollen.

Aktuell nur regelbasierte Filterung. Der Begriff *fallbasierte Filterung* suggeriert zunächst, dass ein selbstlernendes System die WvT-Verlinkung erweitert. Aus den zuletzt erarbeiteten Gründen gleicht die fallbasierte Filterung derzeit vielmehr einem einfachen Mechanismus, in dem ein WvT-Fall lediglich als Regelsammlung aufgefasst wird. Die Automatisierung der beiden letzten CBR-Schritte *revise* und *retain* erscheint aus der aktuellen Perspektive als äußerst herausfordernd, so dass an dieser Stelle weitere Forschung nötig ist.

6.5. Fazit

In diesem Kapitel wurde die WvT-Verlinkung um die Technik der *fallbasierten Filterung* erweitert: WENN eine Ziel-Anforderung mit einer Quell-Anforderung verlinkt ist UND WENN ein Testfall mit der Quell-Anforderung verlinkt ist, DANN wird der Testfall GEMÄSS FALL-/REGELDATENBANK mit der Ziel-Anforderung verlinkt. Die Falldatenbank enthält WvT-Fälle. Ein *WvT-Fall* ist ein WvT-Problem inklusive seiner *WvT-Verlinkungsentscheidung*.

Die fallbasierte Filterung ist grundsätzlich dazu in der Lage, Verlinkungsregeln in die WvT-Verlinkung einfließen zu lassen. Dazu wurden *Ähnlichkeitsfunktionen* definiert, die die *globale Ähnlichkeit* auf der Grundlage von *trendbasierten* und/oder *direkten lokalen Ähnlichkeiten* ermitteln. Mit Hilfe eines WvT-Falls und zweier WvT-Probleme werden im Anhang D ab Seite 151 die Berechnungen zur Ermittlung von globalen Ähnlichkeiten mit verschiedenen lokalen Szenarien beispielhaft Schritt für Schritt durchgeführt. Die Beispielrechnung zeigt, dass die fallbasierte Filterung flexibel genug ist, um je nach Anwendungszweck verschiedene Trennschärfen zu unterstützen.

Zukünftige Arbeiten. Die WvT-Verlinkung aus Kapitel 4 und die Filterung gemäß Testkonzept aus Kapitel 5 fanden tatsächlich Eingang in die Verlinkung einer aktuellen Baureihe. Die fallbasierte Filterung ist bislang nicht derart einsatzfähig. Die Gründe dafür wurden in diesem Kapitel diskutiert. Weil die fallbasierte Filterung momentan nicht praktikabel ist, wurde in einer Falluntersuchung zumindest gezeigt, dass einige interessante WvT-Fälle existieren, die die WvT-Verlinkung potenziell erweitern könnten. In diesem Zusammenhang muss das Interesse zukünftiger Forschung einer empirischen Untersuchung gelten, die den gesamten CBR-Lernzyklus evaluiert. Für reale SLH und STS muss untersucht werden, inwiefern sich einfache und komplexe WvT-Fälle tatsächlich dazu eignen, die richtigen WvT-Probleme zu entdecken. Zudem muss in den SLH und STS evaluiert werden, unter welchen Bedingungen gelöste WvT-Probleme Eingang in die Falldatenbank finden können. Als schwierigster Aspekt wird sich hierbei die Notwendigkeit der einheitlichen Klassifikation der Artefakte in $\text{SLH}_{\text{Quell}}$, SLH_{Ziel} und STS darstellen.

7. Zusammenfassung und Ausblick

In dieser Arbeit wurde eine 3-schichtige Methode vorgestellt, die sich der Lösung eines spezifischen Problems, dem WvT-Problem, widmet. Das WvT-Problem beschreibt die Situation, in der sich zwei Systemlastenhefte (SLH) und eine Systemtestspezifikation (STS) befinden: Die Ziel-Anforderungen des SLH_{Ziel} werden aus dem $\text{SLH}_{\text{Quell}}$ wiederverwendet und die Testfälle der STS sind mit den Quell-Anforderungen des $\text{SLH}_{\text{Quell}}$ aber noch nicht mit den Ziel-Anforderungen des SLH_{Ziel} verlinkt. Die WvT-Verlinkung löst das WvT-Problem, indem die fehlenden Links in das SLH_{Ziel} gesetzt und Konsistenz- sowie Ähnlichkeitsanalysen durchgeführt werden. Zudem wurde die Verlinkung um Filtertechniken erweitert, die die Verlinkung detaillieren.

Das zentrale Ziel der vorgestellten Verlinkungs-, Analyse- und Filtertechniken ist die Schaffung von Nachvollziehbarkeit hinsichtlich

1. der Herkunft, Änderung und Testverlinkung nach Wiederverwendung,
2. der korrekten Umsetzung der Testplanung sowie
3. suspekter Verlinkungssituationen basierend auf Erfahrungen.

Nachvollziehbarkeit nach Wiederverwendung. Obwohl die Wiederverwendung von SLH und STS etablierte Praxis ist, ist sie bislang stark manuell ausgeprägt. Weil Entwickler immer wenig Zeit haben, kann beobachtet werden, wie die Anzahl der Links zwischen Anforderungen und Testfällen nach der Wiederverwendung von Baureihe zu Baureihe abnimmt. Weil Links die Nachvollziehbarkeit überhaupt erst ermöglichen, sinkt mit der Linkanzahl auch die Nachvollziehbarkeit von Baureihe zu Baureihe. Die WvT-Verlinkung adressiert dieses Problem, das WvT-Problem, direkt: Testfälle müssen nur noch einmalig mit Anforderungen verlinkt werden, da Links nun automatisch in das SLH_{Ziel} übertragen werden können. Das Werkzeug, das die WvT-Verlinkung umsetzt, wurde erfolgreich in die Praxis überführt. Die Auseinandersetzung mit dem Forschungsstand zeigte, dass die WvT-Verlinkung diesen um die wiederverwendungsbasierte Verlinkung (engl.: reuse-based traceability) erweitert.

Nachvollziehbarkeit der Testplanung. Die Domänen Avionik und Bahn verfügen seit Längerem über jeweils eine eigene Sicherheitsnorm. Sie hatten somit bereits Zeit, ihre Entwicklungsprozesse umzustellen. In der Automobilbranche findet die Schärfung der Prozesse hinsichtlich der funktionalen Sicherheit gemäß ISO 26262 jetzt statt. Ein Teilprozess ist hierbei der Testplanungsprozess, in dem der Testumfang einer Baureihe festgelegt wird. In diesem Zusammenhang etabliert sich im betrachteten Umfeld gerade ein neues Dokument: das Testkonzept (TK). Im TK wird definiert, welche Testfälle existieren müssen, um eine Baureihe ausreichend abzusichern. Um nachvollziehen zu können, ob der Zustand der ausreichenden Absicherung erreicht ist, spielt wieder die Verlinkung zwischen Anforderungen und Testfällen eine wesentliche Rolle. Mit der testkonzeptbasierten Filterung wurde in dieser Arbeit eine pragmatische Technik vorgestellt, die die WvT-Verlinkung erweitert und gleichzeitig eine aktuelle Herausforderung der Automobilbranche betrifft.

Nachvollziehbarkeit von Entscheidungen. Da sich WvT-Probleme wiederholen können, wiederholen sich auch die entsprechenden Verlinkungsentscheidungen. Hieraus ergeben sich wiederkehrende Situationen, in denen entschieden werden muss, ob ein Testfall noch dazu in der Lage ist, eine Anforderung nach ihrer Wiederverwendung abzusichern. Eine auf dieser Grundlage getroffene Entscheidung ist zunächst nicht zwingend nachvollziehbar, da sie oftmals auf den Erfahrungen des einzelnen Entwicklers aus früheren Wiederverwendungen basiert. In dieser Arbeit wurde die WvT-Verlinkung um diesen Aspekt erweitert: In einer Falldatenbank können gelöste WvT-Probleme manuell abgelegt werden, um sie in neuen Projekten automatisch wiederzufinden. Obwohl die Machbarkeit der fallbasierten Filterung mit Hilfe von Beispielen und einer Implementierung belegt wurde, scheitert die vollautomatische Umsetzung noch an den heterogenen Vorgehensweisen in Konzernen.

Weiterführende Arbeiten. In dieser Arbeit wurde eine Methode vorgestellt, die ein praktisches Problem mit Hilfe eines grundlegenden Fundaments löst. Die Implementierung der WvT-Verlinkung findet bereits Anwendung. Aus dieser Anwendung heraus wurde jedoch immer deutlicher, dass die WvT-Verlinkung zu mehr fähig ist, als in dieser Arbeit dargestellt.

Entwickler sind Tüftler. Die Auseinandersetzung mit Problemen im Rahmen ihrer täglichen Arbeit führen in Kombination mit mangelnder Zeit oftmals zu kreativen Lösungen mit bestehenden Mitteln. Das Werkzeug, das die WvT-Verlinkung umsetzt, war Gegenstand solcher pragmatischen Tüfteleien. Die Tüftler erkannten, dass in dem Werkzeug nicht nur SLH_{Quell}, SLH_{Ziel} und STS, sondern beliebige miteinander verlinkte Dokumente ausgewählt werden konnten. Diese vermeintlich schlechte Ausprägung des Qualitätsziels *Robustheit* führte letztlich zu der Möglichkeit, die WvT-Verlinkung auf vielfältige Art und Weise einzusetzen:

1. Es wird nicht das gesamte SLH an die Lieferanten verteilt, sondern nur die für den jeweiligen Lieferanten relevanten Ausschnitte. Hier unterstützt die WvT-Verlinkung, die als Eingabe das vollständige SLH, den Lieferantenausschnitt und eine STS erhält.
2. Ein weiterer Aspekt, der hier nur angedeutet wird, betrifft das Variantenmanagement. Es werden 150% SLH gepflegt, die alle Anforderungen enthalten. Aus den 150% SLH werden pro Baureihe baureihenspezifische SLH ausgeleitet. Die WvT-Verlinkung erhält als Eingabe ein 150% SLH, ein spezifisches SLH und die 150% STS oder die spezifische STS.
3. Aus beiden obigen Anwendungen ergibt sich ein neuer Aspekt. Wenn im ausgeleiteten SLH bzw. dem Lieferantenausschnitt neue Testfälle verlinkt werden, können die neuen Links mit Hilfe der WvT-Verlinkung zurück in das 150% SLH bzw. das gesamte SLH übertragen werden.

Zum Ende dieser Arbeit wurde aus der kreativen Anwendung heraus ein interessanter Gedankengang geboren, dessen Verfolgung vielversprechend scheint: Die WvT-Verlinkung ist genereller Natur und die in dieser Arbeit beschriebene Verlinkung von Anforderungen und Testfällen ist nur ein spezifischer Anwendungsfall. Weiterführende Arbeiten könnten sich mit der Frage beschäftigen, welche Möglichkeiten ein generelles WvT-Problem bietet. Es wäre denkbar, dass durch *Drehen* und *Kippen* des Dreiergeflechts, d.h. des WvT-Problems, beliebige Beziehungsgeflechte unterstützt werden können, indem zwischen jedem *Drehen* und *Kippen* eine WvT-Verlinkung durchgeführt wird.

Anhänge

Die folgenden vier Anhänge enthalten DOORS-Bildschirmabbilder sowie zusätzliche Aspekte zur Vertiefung der Kapitel 4 und 6:

- Anhang A zeigt Bildschirmabbilder zur Umsetzung der WvT-Verlinkung
- Anhang B zeigt mit Hilfe von Bildschirmabbildern, wie die fallbasierte Filterung aus Kapitel 6 in DOORS realisiert werden kann
- Anhang C enthält eine zweite Feldstudie, um die Ergebnisse der ersten Feldstudie aus Kapitel 4.5 von Seite 66ff zu bestätigen
- Anhang D zeigt die Anwendung der Ähnlichkeitsfunktionen aus Kapitel 6.2.3 von Seite 111ff mittels eines WvT-Falls und zweier WvT-Probleme

A. Auto-Linker: DOORS Plugin

Das Werkzeug DOORS aus der Rational Suite von IBM ist eine der am weitesten verbreiteten Lösungen im Umfeld der Anforderungs- und Testentwicklung [DOORS]. Neue Nutzer vergleichen es manchmal mit Excel, da die Hauptansicht von DOORS einer Excel-Tabelle ähnelt. Die Zeilen in dieser Tabelle bezeichnen Objekte und die Spalten ihre Attribute. Ein Objekt des Typs *Anforderung* hat beispielsweise die Spalten Anforderungstext, Eigentümer, ASIL und Schnittstellen. Ein Objekt des Typs *Testfall* kann wiederum die Spalten Vorbedingung, Nachbedingung und erwartetes Ergebnis besitzen. DOORS ist höchst flexibel, da die Objekttypen und Spalten frei definierbar sind. Dies führt dazu, dass DOORS vielseitig, unter anderem auch im Änderungs- und Variantenmanagement, eingesetzt wird.

Umsetzung in DOORS DXL. DOORS ist noch vielseitiger, als oben dargestellt, weil es über die Erweiterungssprache DXL (engl.: DOORS eXtension Language) verfügt. Die C-artige Skriptsprache erlaubt es, DOORS beliebig anzupassen. Im betrachteten Umfeld existieren tausende Skripte, die spezifische Aufgaben automatisieren, die sonst manuell erledigt werden müssten. Auch der Auto-Linker, also die Implementierung der WvT-Verlinkung, der Filterung gemäß Testkonzept und der fallbasierten Filterung, wurde in DXL umgesetzt. Obwohl DOORS einen Verlinkungsmechanismus zur Verfügung stellt, unterstützt es die WvT-Verlinkung von Hause aus nicht. Der Grund dafür ist, dass zum falschen Zeitpunkt verlinkt wird: Direkt nach dem Kopieren des $\mathrm{SLH}_{\mathrm{Quell}}$ und somit vor dem Ändern des $\mathrm{SLH}_{\mathrm{Ziel}}$. Der Auto-Linker unterstützt die bereichsübergreifenden SLH- und STS-Vorlagen, ist jedoch nicht an diese gebunden. Grundsätzlich können der Verlinkung alle Spezifikationsdokumente zugeführt werden, sofern sich WvT-Probleme in ihnen befinden. Für die Umsetzung wurde das Modell-Präsentation-Steuerung Entwurfsmuster (engl.: MVC, Model View Controller) verwendet. Da Modell und Präsentation von DOORS geboten werden, beschränkt sich die Implementierung fast ausschließlich auf die Steuerung und einige wenige Dialogelemente.

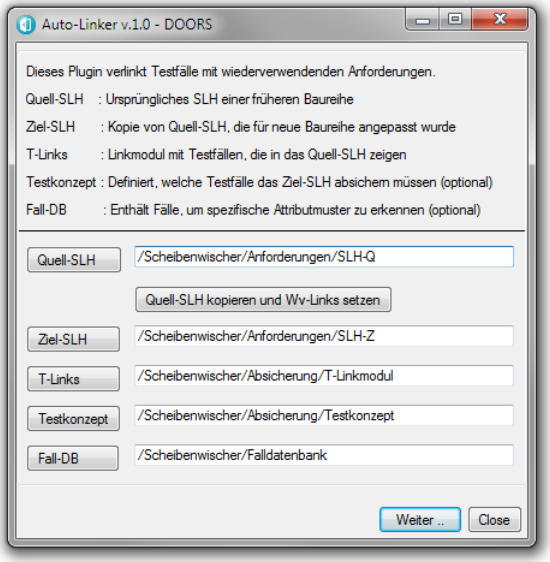


Bild A.1.: Auswahl der DOORS Module

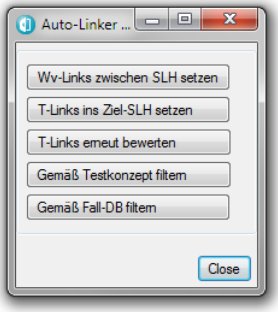


Bild A.2.: Umsetzung der 3-schichtigen Methode

Modulauswahldialog. Bild A.1 zeigt den ersten Dialog der Umsetzung, in dem die beteiligten DOORS-Module ausgewählt werden. Der Dialog bietet die Möglichkeit, das SLH_{Quell} zu kopieren und die Wv-Links in das neue SLH_{Ziel} zu setzen. Diese Vorgehensweise ist vorteilhaft und empfehlenswert: Das Setzen der Wv-Links zwischen dem SLH_{Quell} und dem SLH_{Ziel} ist dann trivial, weil die beiden SLH nach dem Kopieren noch identisch sind. Um Links zu setzen, müssen lediglich die IDs in beiden SLH abgeglichen und Links zwischen den Anforderungen mit gleichen IDs gesetzt werden. Die Pfade zum Testkonzept und zur Falldatenbank sind optional. Wenn sie nicht angegeben werden, findet die WvT-Verlinkung ohne die jeweilige Filterung statt.

Hauptdialog. Der Hauptdialog des Auto-Linkers ist bewusst einfach gehalten: Er verfügt lediglich über fünf Knöpfe. Mit Hilfe des ersten Knopfs können Wv-Links zwischen zwei SLH gesetzt werden. Hier wird nach identischen IDs gesucht, um Wiederverwendungspaare aufzufinden. Dieses Vorgehen führt jedoch selten zum Erfolg, da DXL-Skripte eingesetzt werden, um die Einzigartigkeit von IDs zu wahren, indem sie nach dem Kopieren im SLH_{Ziel} verändert werden. Es empfiehlt sich daher, das SLH_{Quell} mit Hilfe des Modulauswahldialogs zu kopieren und erst danach die DXL-Skripte zur Wahrung der Einzigartigkeit auszuführen. Der zweite Knopf führt die WvT-Verlinkung durch. Je nachdem, ob im Modulauswahldialog Pfade zum Testkonzept und/oder zur Falldatenbank angegeben wurden, wird die Filterung gemäß Testkonzept und/oder die fallbasierte Filterung durchgeführt. Der zweite Knopf wird in der Regel nur einmalig verwendet, um nach dem Kopieren des SLH_{Quell} die T-Links in das SLH_{Ziel} zu übertragen. In der Anwendung zeigte sich, dass der dritte Knopf der wertvollste Knopf des Auto-Linkers ist. Mit dessen Hilfe kann die Gesamtverlinkungssituation jederzeit erneut bewertet werden. Die beiden letzten Knöpfe führen die Filtermechanismen einzeln aus. Mit Hilfe des vierten Knopfs werden all diejenigen Ziel-Anforderungen angezeigt, die nicht gemäß Testkonzept abgesichert sind. Dies betrifft sowohl Ziel-Anforderungen, für die Testfälle fehlen als auch Ziel-Anforderungen, die mit zu vielen Testfällen verlinkt sind. Der fünfte Knopf erlaubt schließlich, nach WvT-Problemen zu suchen, die den in der Falldatenbank aktivierten WvT-Fällen ähneln.

Stapelverarbeitung. Einige Bereiche spezifizieren Fahrzeugfunktionen nicht im SLH, sondern jeweils in eigenen Modulen. In neuen Baureihenprojekten werden diese Funktionsmodule kopiert, angepasst und mittels DXL-Plugins zu einem SLH zusammengestellt. Wenn hunderte solcher Module verlinkt werden sollen, müsste die Verlinkung hundertfach durchgeführt werden. Da dies nicht praktikabel ist, zeigt Bild A.3 die Umsetzung einer Stapelverarbeitung.

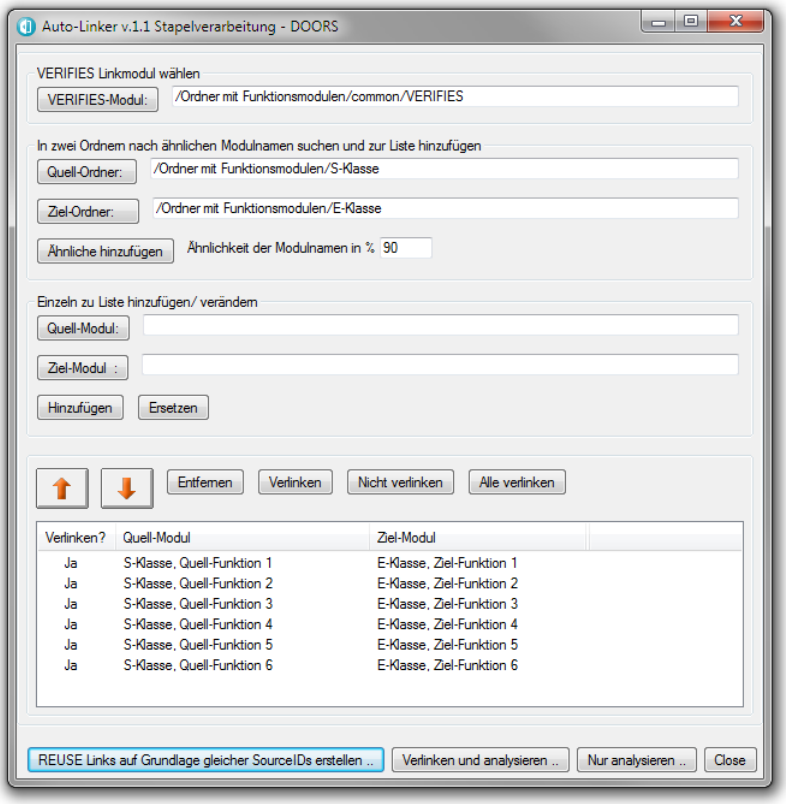


Bild A.3.: Stapelverarbeitung

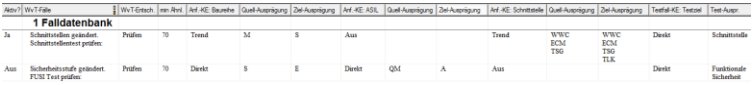
B. Auto-Linker: SLH und Falldatenbank

Darstellung im SLH_{Ziel}. Bild B.1 soll einen Eindruck vermitteln, wie der Auto-Linker das SLH_{Ziel} um wertvolle Informationen anreichert, die DOORS von Hause aus nicht liefert. Das Bild zeigt einen realen Ausschnitt eines SLH_{Ziel} nach der WvT-Verlinkung. Zur Anonymisierung wurden lediglich die Anforderungstexte entfernt. Der Auto-Linker erweitert das SLH_{Ziel} um die Spalten *Wv*, *T?*, *Ähnl. (textuelle Ähnlichkeit)*, *Inkons. (Inkonsistenz)* und *WvT-Prüfhinweis*. Mit Hilfe der DOORS-eigenen Filtermechanismen können individuelle Filter auf den Spalten *Wv*, *T*, *Ähnl.* und *Inkons.* definiert werden. Dies erlaubt es beispielsweise, alle geänderten, nicht getesteten und/oder zu prüfenden Ziel-Anforderung anzuzeigen. Der *Prüfhinweis* wird während der WvT-Verlinkung gemäß den gefundenen Inkonsistenzen im SLH_{Ziel} zusammengesetzt. Er enthält IDs, die die zu prüfenden Artefakte identifizieren. Im vorangegangenen Anhang A wurde bereits erwähnt, dass der Knopf *T-Links erneut bewerten* am Wertvollsten ist. Wenn er betätigt wird, werden die Auto-Linker Spalten aktualisiert. Somit begleitet der Auto-Linker die Spezifikation der Ziel-Baureihe unterstützend.

ID	Ziel-Anforderungen	Wv	T?	Ähnl.	Inkons.	WvT-Prüfhinweis
2923	**1 Funktion**	Ja	Prüfen	100	Testplan	- Testfall fehlt: (Konfigurierbarkeit, Integration, HiL)
2928	Anforderung	Ja	Ja	100		- 3 Tests wiederverwendet von Q-Anf [2894]
2938	Anforderung	Ja	Prüfen	92	II_Q	- Textuelle Änderung - 1 Test wiederverwendet von Q-Anf [2903] - Quell-SLH: 1 inkonsistenter Test - Test [420] verlinkt exklusiv nach Q-Anf [2930]
2991	Anforderung	Ja	Prüfen	65	II_Q	- Textuelle Änderung - 2 Tests wiederverwendet von [2751] - Quell-SLH: 2 inkonsistente Tests - Test [40] verlinkt exklusiv nach Q-Anf [2832, 2994] - Test [36] verlinkt exklusiv nach Q-Anf [2832, 2836, 2994]
3057	Anforderung	Ja	Prüfen	80	Fall-DB	- Textuelle Änderung - 1 Test wiederverwendet von Q-Anf [3018] - Schnittstellen geändert. Schnittstellentest prüfen: [43]
3095	Anforderung	Nein	Nein			
3096	Anforderung	Nein	Nein			

Bild B.1.: Ausschnitt von SLH_{Ziel} nach WvT-Verlinkung

Darstellung der Falldatenbank. Um den Umgang mit der Falldatenbank zu erleichtern, werden WvT-Fälle nicht durch Textstrukturen wie auf Seite 106 repräsentiert. Sie können nativ in der DOORS-üblichen Oberfläche eingegeben werden. Bild B.2 verdeutlicht, dass dieses Vorgehen zu sehr langen Zeilen führt, weil für jeden WvT-Fall viele Angaben nötig sind. Nachfolgend wird an einem WvT-Fall gezeigt, wie die Falldatenbank in DOORS realisiert wurde.



Aktiv?	WvT-Fälle	WvT-Entsch	min. Ähnl.	Anf.-KE: Baureihe	Quell-Ausprägung	Ziel-Ausprägung	Anf.-KE: ASIL	Quell-Ausprägung	Ziel-Ausprägung	Anf.-KE: Schnittstelle	Quell-Ausprägung	Ziel-Ausprägung	Testfall-KE: Testziel	Test-Auspr.
	1 Falldatenbank													
Ja	Schnittstellen geändert. Schnittstellentest prüfen.	Prüfen	70	Trend	M	S	Aus			Trend	WWC ECM TSG	WWC ECM TSG TLK	Direkt	Schnittstelle
Aus	Sicherheitsstufe geändert. FUSI Test prüfen.	Prüfen	70	Direkt	S	E	Direkt	QM	A	Aus			Direkt	Funktionale Sicherheit

Bild B.2.: Fälle mit drei Anforderungs- und einer Testfalleigenschaft(en)

Repräsentation von WvT-Fällen. Bild B.3 zeigt einen WvT-Fall in drei Teilen. Teilbild B.3a zeigt zunächst die Fallbeschreibung. Diese besteht aus dem Prüfhinweis, der in der Spalte *WvT-Fälle* hinterlegt wird. Der WvT-Fall wird mit Hilfe der Spalte *Aktiv?* aktiviert bzw. deaktiviert. Wenn der WvT-Fall aktiviert ist und ein WvT-Problem entdeckt wird, das dem Fall ähnelt, dann wird der Prüfhinweis an die entsprechende Stelle in das SLH_{Ziel} kopiert. Wie hoch die globale Ähnlichkeit zwischen WvT-Fall und WvT-Problem sein muss, wird in der Spalte *min. Ähnl. (minimale globale Ähnlichkeit)* festgelegt. Teilbild B.3b zeigt die beiden klassifizierenden Anforderungseigenschaften (Anforderungs-KE) des WvT-Falls. Jede Anforderungs-KE besteht aus drei Spalten. Die erste Spalte *Anf.-KE: ...* bietet die Möglichkeit, für die KE die Suchart *Trend*, *Direkt* oder *Aus* festzulegen. Die zweite und dritte Spalte enthalten jeweils die Quell- bzw. Ziel-Ausprägungen der Anforderungs-KE. Teilbild B.3c zeigt schließlich die klassifizierende Testfalleigenschaft (Testfall-KE) des WvT-Falls. Für eine Testfall-KE existieren nur zwei Spalten. Die Spalte *Testfall-KE: ...* legt wieder die Suchart der KE fest. Da Testfall-KE lediglich von Test-zu-Test miteinander verglichen werden, können Ausprägungen einer Testfall-KE nur direkt miteinander verglichen werden. Die Suchart ist damit auf *Direkt* und *Aus* beschränkt. Die Spalte *Test-Ausprägung* enthält die Ausprägungen der Testfall-KE.

Aktiv?	WvT-Fälle	WvT-Entsch.	min Ähnl.
Ja	Schnittstellen geändert. Schnittstellentest prüfen:	Prüfen	70

(a) WvT-Fallbeschreibung mit ..

WvT-Fälle	Anf.-KE: Baureihe	Quell-Ausprägung	Ziel-Ausprägung	Anf.-KE: Schnittstelle	Quell-Ausprägung	Ziel-Ausprägung
Schnittstellen geändert. Schnittstellentest prüfen:	Trend	M	S	Trend	WWC ECM TSG	WWC ECM TSG TLK

(b) .. zwei Anforderungseigenschaften (Anf.-KE) und ..

WvT-Fälle	Testfall-KE: Testziel	Test-Auspr.
Schnittstellen geändert. Schnittstellentest prüfen:	Direkt	Schnittstelle

(c) .. einer Testfalleigenschaft (Testfall-KE).

Bild B.3.: WvT-Fall in DOORS

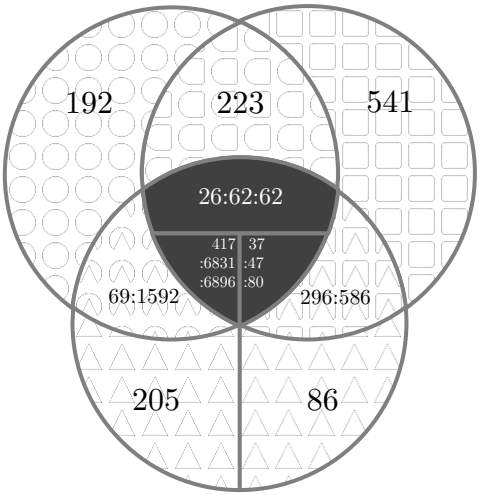
Erweiterbarkeit. Die bereichsspezifische Definition und Pflege der WvT-Fälle findet in Arbeitskreisen statt. Anforderungs-KE und Testfall-KE können in der Falldatenbank beliebig hinzugefügt werden. Da die Namen der KE in den Falldatenbanken, Systemlastenheften und Testspezifikationen von Bereich zu Bereich sehr unterschiedlich sein können, müssen sie aufeinander abgebildet werden. Dies geschieht in DOORS, indem pro Spalte *Anf.-KE: ...* bzw. pro Spalte *Testfall.-KE: ...* in einem Attribut festgelegt wird, wie die jeweilige KE in den bereichsspezifischen Systemlastenheften und Testspezifikationen benannt ist. Dies ermöglicht einen sehr flexiblen Einsatz verschiedener bereichsspezifischer Falldatenbanken. Im vorangegangenen Anhang A wurde der Knopf *Gemäß Fall-DB filtern* des Hauptdialogs vorgestellt. Der Knopf ermöglicht es, die fallbasierte Filterung unabhängig von der WvT-Verlinkung im Sinne einer Suche durchzuführen. Es ist nun möglich, gezielt nach WvT-Problemen zu suchen, beispielsweise: „Suche alle Ziel-Anforderungen, bei denen sich die Schnittstellen geändert haben und auf die ein Schnittstellentestfall zeigt“. Dies erweitert die DOORS-Suche um die fallbasierte Filterung.

C. Feldstudie: System B von Baureihe 3 nach Baureihe 1

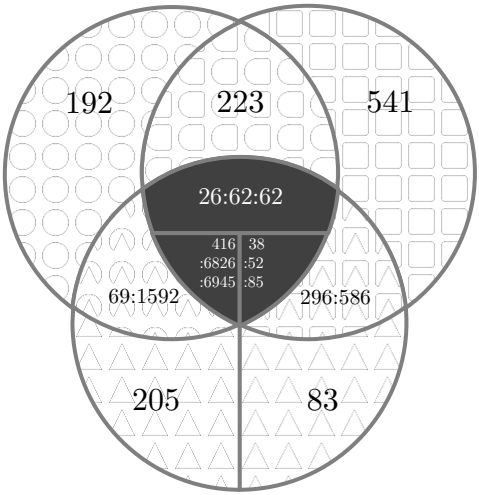
Die Feldstudie in Kapitel 4 wurde mit Hilfe der Spezifikationsdokumente eines kleinen Systems A durchgeführt. Es konnte gezeigt werden, dass die WvT-Verlinkung dieselben Ziel-Anforderungen mit Testfällen verbindet, die auch in der vorgefundenen Situation schon verbunden waren. In diesem Zusammenhang wurden Übergangsregeln beobachtet. Diese Beobachtungen werden im Folgenden mit Hilfe des größeren Systems B bestätigt - unter Befolgung derselben Vorbereitungsprozedur wie im kleinen System A. Das System B besteht aus 964 Quell-Anforderungen, 1540 Ziel-Anforderungen und 1030 Testfällen.

Untersuchung der peripheren Bereiche. Die Bilder C.1a und C.1b zeigen die Diagramme, die die vorgefundene Situation und die Situation nach der WvT-Verlinkung widerspiegeln. Wie im System A blieben die Zahlen in den peripheren Bereichen - exklusive der nicht ziel-verlinkten Testfälle - gleich. Damit bestätigt auch die zweite Feldstudie, dass nach der WvT-Verlinkung mehr Testfälle in das $\mathrm{SLH}_{\mathrm{Ziel}}$ zeigen, als dies zuvor gegeben war: Die WvT-Verlinkung ist erneut effektiver als das bisherige Vorgehen.

Untersuchung der Diagrammzentren. Auch für System B zeigen die beiden Diagrammzentren, dass die WvT-Verlinkung hinsichtlich der Testfallverlinkung konsistenzerhaltend ist. Dies wird daran ersichtlich, dass sich die Anzahlen im oberen Bereich der Diagrammzentren gleichen. Ein Blick auf den unteren linken Bereich der Zentren bestätigt die Beobachtung, dass die WvT-Verlinkung zu weniger Quell-Inkonsistenzen führt. Die Feldstudie verdeutlicht durch den unteren rechten Bereich der Diagrammzentren auch, dass neue pure Ziel-Inkonsistenzen entstehen können, wenn Quell-Inkonsistenzen, die in Kombination mit Ziel-Inkonsistenzen auftraten, eliminiert werden. Mit Hilfe der folgenden Tabelle C.1 werden die bereits in Kapitel 4 im System A beobachteten Übergangsregeln der Inkonsistenzen bestätigt.



(a) Vorgefundene Verlinkungssituation



(b) Situation nach WvT-Verlinkung

Bild C.1.: WvT-Diagramme für System B

Konsistent bleibt konsistent. Anhand der ersten Zeile $0 \Rightarrow 0$ in Tabelle C.1 wird bestätigt, dass auch in System B alle Wv-Paare, die zuvor konsistent mit Testfällen verlinkt waren auch nach der WvT-Verlinkung konsistent mit Testfällen verlinkt sind.

Inkonsistenz I_Q wird eliminiert. Wie auch in der detaillierten Feldstudie ist in dieser bestätigenden Feldstudie klar, dass die WvT-Verlinkung die Inkonsistenz I_Q per Definition eliminiert. In Tabelle C.1 wird dies wieder daran ersichtlich, dass alle Inkonsistenzen nach der automatischen WvT-Verlinkung wegen der binären Zählweise eine gerade Nummerierung besitzen. Die Eliminierung von I_Q - wie zuvor auch in Kombination mit anderen Inkonsistenzen - wird an den Übergangen $5 \Rightarrow 4$, $7 \Rightarrow 6$, $13 \Rightarrow 12$ und $15 \Rightarrow 14$ deutlich.

Vorgefunden ⇒ WvT-Verlinkung		**Übergänge**
$0 \Rightarrow 0$	(Konsistent ⇒ konsistent)	26
$2 \Rightarrow 2$	$(I_Z \Rightarrow I_Z)$	6
$4 \Rightarrow 4$	$(II_Q \Rightarrow II_Q)$	116
$5 \Rightarrow 4$	$(I_Q \wedge II_Q \Rightarrow II_Q)$	3
$6 \Rightarrow 6$	$(I_Z \wedge II_Q \Rightarrow I_Z \wedge II_Q)$	18
$7 \Rightarrow 6$	$(I_Q \wedge I_Z \wedge II_Q \Rightarrow I_Z \wedge II_Q)$	1
$8 \Rightarrow 8$	$(II_Z \Rightarrow II_Z)$	12
$10 \Rightarrow 10$	$(I_Z \wedge II_Z \Rightarrow I_Z \wedge II_Z)$	19
$12 \Rightarrow 8$	$(II_Q \wedge II_Z \Rightarrow II_Z)$	1
$12 \Rightarrow 12$	$(II_Q \wedge II_Z \Rightarrow II_Q \wedge II_Z)$	194
$13 \Rightarrow 12$	$(I_Q \wedge II_Q \wedge II_Z \Rightarrow II_Q \wedge II_Z)$	27
$14 \Rightarrow 14$	$(I_Z \wedge II_Q \wedge II_Z \Rightarrow I_Z \wedge II_Q \wedge II_Z)$	51
$15 \Rightarrow 14$	$(I_Q \wedge I_Z \wedge II_Q \wedge II_Z \Rightarrow I_Z \wedge II_Q \wedge II_Z)$	6

Tabelle C.1.: Übergänge der Inkonsistenzen von SLH_{Ziel}^{mnl} zu SLH_{Ziel}^{auto}

II_Q bleibt II_Q oder wird eliminiert. Dass die Inkonsistenz II_Q erhalten bleiben kann, wurde bereits in der ersten Feldstudie beobachtet. Ein Blick in Tabelle C.1 bestätigt diese Beobachtung wieder: II_Q bleibt in den Übergängen $4 \Rightarrow 4$, $5 \Rightarrow 4$, $6 \Rightarrow 6$, $7 \Rightarrow 6$, $12 \Rightarrow 12$, $13 \Rightarrow 12$, $14 \Rightarrow 14$ sowie $15 \Rightarrow 14$ erhalten. In der ersten Feldstudie wurde auch beobachtet, dass die Inkonsistenz II_Q nicht zwangsläufig erhalten bleiben muss, sondern auch konsistent werden kann. Diese Beobachtung wird durch Tabelle C.1 bestätigt: $12 \Rightarrow 8$.

I_Z und/oder II_Z bleibt erhalten. Für die Feldstudie wurden alle T-Links, die zuvor exklusiv auf Ziel-Anforderungen, nicht jedoch auf Quell-Anforderungen zeigten auch nach der WvT-Verlinkung nachträglich gesetzt. Dies simuliert die nachträgliche Verlinkung von Ziel-Anforderungen mit Testfällen. Daher muss gelten, dass jede Ziel-Inkonsistenz I_Z und/oder II_Z erhalten bleibt. Ein Blick in Tabelle C.1 bestätigt dies: $2 \Rightarrow 2$, $6 \Rightarrow 6$, $7 \Rightarrow 6$, $8 \Rightarrow 8$, $10 \Rightarrow 10$, $12 \Rightarrow 8$, $12 \Rightarrow 12$, $13 \Rightarrow 12$, $14 \Rightarrow 14$ und $15 \Rightarrow 14$.

Bestätigung der vier Regeln. In der ersten Feldstudie wurden vier Übergangsregeln beobachtet und mit Hilfe von anonymisierten Beispielen detailliert vorgestellt: Konsistent bleibt konsistent, I_Q wird eliminiert, II_Q bleibt II_Q oder wird eliminiert, I_Z und/oder II_Z bleibt erhalten. In der zweiten Feldstudie bestätigt Tabelle C.1 diese Regeln nicht nur - sie verdeutlicht auch, dass die vier Regeln genügen, um alle beobachteten Übergänge zu erklären.

Bestätigung der Effektivität. Das Ziel der Feldstudie war es zu zeigen, dass die WvT-Verlinkung die Testfälle mit den Ziel-Anforderungen verlinkt, die durch das bisherige Vorgehen miteinander verlinkt wurden. Das Feldstudienziel wurde erreicht, indem eben dies für zwei reale Systeme gezeigt wurde: Sowohl in System A als auch in System B waren nach der WvT-Verlinkung ausnahmslos alle Ziel-Anforderungen mit den Testfällen verlinkt, mit denen sie auch zuvor verlinkt waren. Es zeigte sich zudem, dass die WvT-Verlinkung nicht nur effektiv, sondern effektiver als das bisherige Vorgehen ist, weil sie sogar mehr Testfälle mit Ziel-Anforderungen verlinkt und dabei weniger Inkonsistenzen erzeugt.

D. Beispiel: Ähnlichkeit von WvT-Problemen

Bild D.1 zeigt einen Beispielfall und zwei Beispielprobleme. Nachfolgend werden die Ähnlichkeitsfunktionen aus Kapitel 6.2 beispielhaft durchgerechnet.

Gesamtanzahl der Ausprägungen der KE. Angenommen, neben a und b existieren keine weiteren WvT-Probleme. Weiterhin wird in diesem kleinen Rechenbeispiel vorausgesetzt, dass f der einzige Fall in der Falldatenbank ist. Dann ist die Gesamtanzahl aller möglichen Schnittstellen $\#\mathrm{KE}_{\mathrm{Schnittstelle}} = 4$ (WWC: Wiper Control, TSG: Türsteuergerät, TLK: Trunk Lock, IC: Instrument Cluster). Die Anzahl aller Baureihen ist $\#\mathrm{KE}_{\mathrm{Baureihe}} = 3$ (M-Klasse, S-Klasse, E-Klasse). Die Anzahl aller Testziele ist $\#\mathrm{KE}_{\mathrm{Testziel}} = 1$ (Korrekte Funktionalität an Schnittstellen).

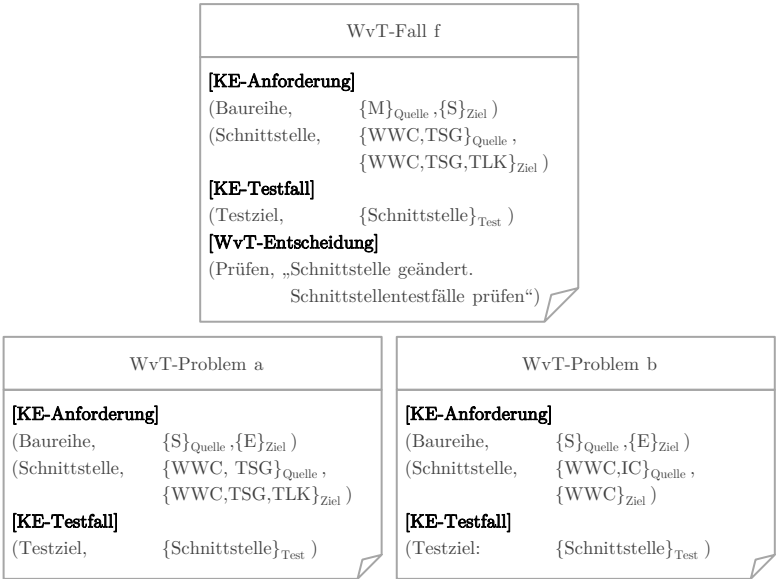


Bild D.1.: Beispielfall und -probleme

D.1. Pure trendbasierte Ähnlichkeit von Anforderungen

Lokale Trends. Die pure trendbasierte Ähnlichkeit vergleicht alle KE von Anforderungen trendbasiert, d.h. fall- und problemintern von Quelle-zu-Ziel. Die Testfall-KE werden direkt, d.h. von Falltestfall-zu-Problemtestfall, bewertet. Für die Anforderungs-KE *Schnittstelle* des Falls f endet die binäre Enthaltenseinsanalyse beispielsweise mit dem Resultat *BinE (a=2,b=0,c=1,d=1)*:

(a) Anzahl der gemeinsamen Quell- und Ziel-Schnittstellen: 2 (WWC,TSG)

(b) Anzahl der exklusiven Quell-Schnittstellen: 0

(c) Anzahl der exklusiven Ziel-Schnittstellen: 1 (TLK)

(d) Anzahl der Schnittstellen, welche keine der Anforderungen hat: 1 (IC)

Die Variablen des Maßes Rogers Tanimoto nehmen die Werte von BinE an: $\frac{a+d}{a+2b+2c+d} \Rightarrow \frac{2+1}{2+2*0+2*1+1}$. Die Ähnlichkeit der *Schnittstellen* der Quell- und Ziel-Anforderung des WvT-Falls f beträgt 60%. Der Wiederverwendungstrend sagt somit für f aus, dass sich 60% der Schnittstellen nicht geändert haben. Es zeigt sich, dass für die WvT-Probleme a und b die internen Quelle-zu-Ziel-Vergleiche der KE *Schnittstelle* wie bereits für f zu einem Wert von 60% führen. Tabelle D.1 zeigt zunächst die Ergebnisse des Zwischenschritts, in dem die Quelle-zu-Ziel-Vergleiche für alle Anforderungs-KE des Falls f und der beiden Probleme a und b durchgeführt werden. In jeder Tabellenzelle ist links das Vier-Tupel *BinE* und rechts die Rogers Tanimoto Ähnlichkeit dargestellt. Die Ähnlichkeit der Testfall-KE *Testziel* zwischen f und a sowie f und b wird, der Begründung auf Seite 116 folgend, ausschließlich direkt ermittelt.

	Baureihe (trendQZ)	**Schnittstelle (trendQZ)**	**Testziel (direktTT)**
Fall f	(0,1,1,1) ⇒ 20%	(2,0,1,1) ⇒ 60%	-
Problem a	(0,1,1,1) ⇒ 20%	(2,0,1,1) ⇒ 60%	(1,0,0,0) ⇒ 100%
Problem b	(0,1,1,1) ⇒ 20%	(1,1,0,2) ⇒ 60%	(1,0,0,0) ⇒ 100%

Tabelle D.1.: Interner Quelle-zu-Ziel-Wiederverwendungstrend

Gleiche Ähnlichkeit trotz ungleicher BinE. Tabelle D.1 zeigt für die KE *Schnittstelle* einen interessanten Aspekt. Sie zeigt, dass Ähnlichkeitsmaße aus ungleichen *BinE* gleiche prozentuale Ähnlichkeiten ermitteln können: Die Änderungen der *Schnittstellen* von der Quell- zur Ziel-Anforderung in f, a und b sind nicht identisch. Die Gemeinsamkeit ist, dass sich die Schnittstellen jeweils von Quelle-zu-Ziel um genau eine Schnittstelle unterscheiden. Für die KE *Schnittstelle* haben f, a und b daher dieselbe Ähnlichkeit von 60%.

Globaler Trend. Für jede Anforderungs-KE i wird der Abstand der Quelle-zu-Ziel-Vergleiche zwischen WvT-Fall und WvT-Problem ermittelt, z.B. für f und a: $\text{abstand}(f, a, i) = |\text{trendQZ}(f, i) - \text{trendQZ}(a, i)|$. Die internen Quelle-zu-Ziel-Vergleiche der Anforderungs-KE *Schnittstelle* resultieren für f und a jeweils in einem Wert von 60%. Der Abstand der beiden Werte liegt wegen $|60\%_f - 60\%_a|$ bei 0%. Der lokale Wiederverwendungstrend von f und a für die Anforderungs-KE *Schnittstelle* (i) ergibt sich damit zu $\text{trendAnf}(f, a, i) = 100\% = 100\% - 0\%$. Für Testfall-KE kann nur die direkte Ähnlichkeit mit Hilfe eines Test-zu-Test-Vergleichs in die globale Ähnlichkeit einfließen. Der globale Trendvergleich eines WvT-Falls mit einem WvT-Problem berechnet sich somit aus dem Produkt der Abstände der internen Wiederverwendungstrends aller Anforderungs-KE, multipliziert mit dem Produkt der direkten Vergleiche aller Testfall-KE: $(100\% - \text{abstand}_{\text{Baureihe}}) * (100\% - \text{abstand}_{\text{Schnittstelle}}) * \text{direktTest}_{\text{Testziel}}$. Der globale Trend von f und a ergibt sich zu $100\% = 1.0 = 1.0_{\text{Baureihe}} * 1.0_{\text{Schnittstelle}} * 1.0_{\text{Testziel}}$. Tabelle D.2 zeigt, dass f und a sowie f und b hinsichtlich des Wiederverwendungstrends zu 100% ähnlich sind, weil sich die Anforderuns-KE von Quelle-zu-Ziel gleich stark geändert haben.

Globale Ähnl.	**trendAnf**$_{\text{Baureihe}}$	**trendAnf**$_{\text{Schnittst.}}$	**direktTest**$_{\text{Testziel}}$
$f \circ a = 100\%$	100%	100%	100%
$1.0 * 1.0 * 1.0$	$1.0 - (0.2 - 0.2)$	$1.0 - (0.6 - 0.6)$	1.0
$f \circ b = 100\%$	100%	100%	100%
$1.0 * 1.0 * 1.0$	$1.0 - (0.2 - 0.2)$	$1.0 - (0.6 - 0.6)$	1.0

Tabelle D.2.: Globale Wiederverwendungstrends

D.2. Pure direkte Ähnlichkeit

Lokale Ähnlichkeiten. Die pure direkte Ähnlichkeit vergleicht alle KE lokal direkt, d.h. von Fallquelle-zu-Problemquelle (direktQQ), von Fallziel-zu-Problemziel (direktZZ) und von Falltestfall-zu-Problemtestfall (direktTT). Auch die Berechnung der direkten Ähnlichkeit basiert auf dem zählenden Vergleich von den Ausprägungsmengen einer KE und der Anwendung eines Ähnlichkeitsmaßes. Im Beispiel resultiert der Quelle-zu-Quelle-Vergleich von f und a für die Anforderungs-KE *Schnittstelle* in dem *BinE (a=2,b=0,c=0,d=2)*:

(a) Anzahl der gemeinsamen Quell- und Ziel-Schnittstellen: 2 (WWC,TSG)

(b) Anzahl der exklusiven Quell-Schnittstellen: 0

(c) Anzahl der exklusiven Ziel-Schnittstellen: 0

(d) Anzahl der Schnittstellen, welche keine der Anford. hat: 2 (TLK,IC)

Das Einsetzen des BinE in das Maß Rogers Tanimoto ergibt $\frac{a+d}{a+2b+2c+d} \Rightarrow \frac{2+2}{2+2*0+2*0+2}$ und führt zu einer Ähnlichkeit von 100%. Tabelle D.3 zeigt die Ergebnisse der Enthaltenseinsanalyse für die verglichenen KE von f und a sowie f und b. Im Beispiel hat die KE *Baureihe* in allen Vergleichen eine geringe Ähnlichkeit von 20%. Da WvT-Fälle frühere WvT-Probleme sind, besitzen sie häufig andere Baureihen, als das WvT-Problem. Nachfolgend zeigt sich, dass sich dies negativ auf die globale Ähnlichkeit auswirkt.

Lokale Ähnl.	**Baureihe**	**Schnittstelle**	**Testziel**
$f_Q \circ a_Q$ (direktQQ/TT)	(0,1,1,1) ⇒ 20%	(2,0,0,2) ⇒ 100%	(1,0,0,0) ⇒ 100%
$f_Z \circ a_Z$ (direktZZ)	(0,1,1,1) ⇒ 20%	(3,0,0,1) ⇒ 100%	-
$f_Q \circ b_Q$ (direktQQ/TT)	(0,1,1,1) ⇒ 20%	(1,1,1,1) ⇒ 33%	(1,0,0,0) ⇒ 100%
$f_Z \circ b_Z$ (direktZZ)	(0,1,1,1) ⇒ 20%	(1,2,0,1) ⇒ 50%	-

Tabelle D.3.: Quelle-zu-Quelle-, Ziel-zu-Ziel-, Test-zu-Test-Vergleiche

Globale Ähnlichkeit. Die globale direkte Ähnlichkeit setzt sich ausschließlich aus direkten lokalen Ähnlichkeiten zusammen. Sie greift anforderungsseitig zweifach auf lokale Vergleiche von nur einer Anforderungs-KE zurück: von Fallquelle-zu-Problemquelle und von Fallziel-zu-Problemziel. Testseitig fließt eine Testfall-KE durch den Falltestfall-zu-Problemtestfall-Vergleich nur einfach in die globale Ähnlichkeit ein. Die globale Ähnlichkeit ist das Produkt der lokalen Ähnlichkeiten. Das doppelte Einfließen einer Anforderungs-KE würde somit dazu führen, dass eine Testfall-KE im Vergleich zu einer Anforderungs-KE nur halb gewichtet wäre. Diese Ungleichheit wird ausgeglichen, indem die beiden Quelle-zu-Quelle- und Ziel-zu-Ziel-Ähnlichkeiten einer Anforderungs-KE gemittelt werden: $\frac{\text{direktQQ}+\text{direktZZ}}{2}$. Der Mittelwert für die KE *Schnittstelle* für den Vergleich von Fall f und Problem b ergibt sich damit wegen $\frac{0.33+0.5}{2}$ gerundet als 42%. Die globale direkte Ähnlichkeit zwischen f und b ist damit $8{,}4\% = 0.084 = 0.2 * 0.42 * 1.0 = \frac{0.2+0.2}{2}_{\text{Baureihe}} * \frac{0.33+0.5}{2}_{\text{Schnittstelle}} * 1.0_{\text{Testziel}}$. Die Tabelle D.4 stellt die lokalen direkten Ähnlichkeiten für jede KE ab der zweiten Spalte dar. Die erste Spalte enthält die multiplikativ ermittelten, globalen direkten Ähnlichkeiten. Beide WvT-Probleme a und b sind nicht ähnlich zum WvT-Fall f. Trotzdem ist erkennbar, dass das WvT-Problem a mit einer Ähnlichkeit von 20% näher an f ist, als das WvT-Problem b mit 8,4% Ähnlichkeit. Das Beispiel zeigt deutlich, dass die pure direkte Ähnlichkeit dafür geeignet ist, übereinstimmende Probleme für einen Fall zu finden. Um dies zu entschärfen, kann eine KE, wie beispielsweise die KE *Baureihe*, auch als Trend in die gemischte globale Ähnlichkeit einfließen.

Globale Ähnl.	**direktAnf**$_{\text{Baureihe}}$	**direktAnf**$_{\text{Schnittst.}}$	**direktTest**$_{\text{Testziel}}$
f ∘ a = 20%	20%	100%	100%
$0.2 * 1.0 * 1.0$	$\frac{0.2+0.2}{2}$	$\frac{1.0+1.0}{2}$	1.0
f ∘ b = 8,4%	20%	42%	100%
$0.2 * 0.42 * 1.0$	$\frac{0.2+0.2}{2}$	$\frac{0.33+0.5}{2}$	1.0

Tabelle D.4.: Direkte globale Ähnlichkeiten

D.3. Trendbasierte und direkte (gemischte) Ähnlichkeit

Gemischte lokale Ähnlichkeiten. Während die pure trendbasierte globale Ähnlichkeit viele ähnliche WvT-Probleme findet, in denen sich die Ausprägungen der KE im Vergleich zum WvT-Fall stark unterscheiden, findet die pure direkte Ähnlichkeit wenige WvT-Probleme, in denen sich die Ausprägungen der KE im Vergleich zum WvT-Fall weniger stark unterscheiden. In den Beispielberechnungen wurde dies daran deutlich, dass die Probleme a und b im Trend identisch und im direkten Vergleich sehr unähnlich zu f waren. Obwohl beide pure Ähnlichkeiten einen Anwendungszweck haben, bietet die gemischte Ähnlichkeit weitere Möglichkeiten bei der Suche nach ähnlichen Problemen.

Globale Ähnlichkeit. In einer Suche genügt es beispielsweise, den Trend der Veränderungen der *Baureihe* zu erkennen. Gleichzeitig kann in dieser Suche relevant sein, welche *Schnittstellen* sich genau geändert haben und dass dasselbe *Testziel* verfolgt wird. Diese drei Forderungen werden im WvT-Fall definiert, indem die Anforderungs-KE *Baureihe* auf den Vergleichstyp *Trend* und die beiden verbleibenden KE auf *Direkt* gesetzt werden. In die globale Ähnlichkeitsfunktion fließen dann sowohl trendbasierte, als auch direkte lokale Ähnlichkeiten ein. Tabelle D.5 veranschaulicht dies. Die globale Ähnlichkeit von f und b errechnet sich als $42\% = 0.42 = 1.0 * 0.42 * 1.0 = (1.0–(0.2–0.2))_{\text{Baureihe}} * \frac{0.33+0.5}{2}_{\text{Schnittstelle}} * 1.0_{\text{Testziel}}$. Anhand des Beispiels wird klar, dass sich die WvT-Probleme a und b wegen ihrer unterschiedlichen globalen Ähnlichkeit nun deutlicher voneinander abgrenzen lassen.

Globale Ähnl.	**Baureihe**$_{\text{Trend}}$	**Schnittst.**$_{\text{Direkt}}$	**Testziel**$_{\text{Direkt}}$
f ∘ a = 100%	100%	100%	100%
$1.0 * 1.0 * 1.0$	$1.0 – (0.2 – 0.2)$	$\frac{1.0+1.0}{2}$	1.0
f ∘ b = 42%	100%	42%	100%
$1.0 * 0.42 * 1.0$	$1.0 – (0.2 – 0.2)$	$\frac{0.33+0.5}{2}$	1.0

Tabelle D.5.: Gemischte globale Ähnlichkeiten

E. Literaturverzeichnis

[ABO+10] Mobyen Uddin Ahmed, Shahina Begum, Erik Olsson, Ning Xiong, Peter Funk. Case-Based Reasoning for Medical and Industrial Decision Support Systems. In *Successful Case-based Reasoning Applications*, Stefania Montani, Lakhmi C. Jain (Herausgeber), Kapitel 2, Seiten 7–52. Springer, 1. Auflage, 2010. Referenziert auf Seite 126.

[ADAC] ADAC. *http://www.adac.de/infotestrat/reparatur-pflege-und-wartung/rueckrufe/default.aspx , Rückrufdatenbank*, 24.02.2015. Referenziert auf Seite 99.

[AFT07] Hazeline U. Asuncion, Frédéric Francois, Richard N. Taylor. An End-To-End Industrial Software Traceability Tool. In *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, Seiten 115–124, Dubrovnik, Kroatien, 2007. ACM. Referenziert auf Seite 39.

[AI11] Azri Azmi, Suhaimi Ibrahim. Implementing Test Management Traceability Model to Support Test Documents. *International Journal of Digital Information and Wireless Communications (IJDIWC)*, 1, Nr. 1, Seite 109–125, 2011. Referenziert auf Seite 39.

[AP94] Agnar Aamodt, Enric Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI communications*, 7, Nr. 1, Seite 39–59, 1994. Referenziert auf den Seiten xvi und 125.

[BCSV04] Stefania Bandini, Ettore Colombo, Fabio Sartori, Giuseppe Vizzari. Case-Based Reasoning and Production Process Design: The Case of P-Truck Curing. In *Proceedings of the 7th European Conference on Case-Based Reasoning (ECC-*

BR), Peter Funk, Pedro A. González-Calero (Herausgeber), Seiten 504–517, Madrid, Spanien, 2004. Springer. Referenziert auf Seite 126.

[BGB] *Bürgerliches Gesetzbuch (BGB)*, 2011. Referenziert auf den Seiten 96, 97 und 99.

[BGH-Urteil] Bundesgerichtshof. *VI ZR 107/08*, 2009. Referenziert auf den Seiten 99, 100 und 101.

[BHD10] Petra Buck-Heeb, Andreas Dieckmann. *Selbstregulierung im Privatrecht*. Mohr Siebeck, 1. Auflage, 2010. Referenziert auf Seite 98.

[Bjo03] Solveig Bjornestad. Analogical Reasoning for Reuse of Object-Oriented Specifications. *Case-Based Reasoning Research and Development*, 2689, Seite 50–64, 2003. Referenziert auf Seite 128.

[BKP06] Ralph Bergmann, Janet Kolodner, Enric Plaza. Representation in Case-Based Reasoning. *The Knowledge Engineering Review*, 20, Nr. 3, Seite 209–213, 2006. Referenziert auf Seite 123.

[BR09] Isabelle Bichindaritz, John C. Reed. Methodology for Classifying and Indexing Case-Based Reasoning Systems in the Health Sciences. In *Proceedings of the 22th International FLAIRS Conference*, H. Chad Lane, Hans W. Güsgen (Herausgeber), Seiten 325–330, Sanibel Island, FL, USA, 2009. AAAI Press. Referenziert auf Seite 126.

[Bro06] Manfred Broy. Challenges in Automotive Software Engineering. *Proceedings of the 28th International Conference on Software Engineering - ICSE '06*, Seite 33, 2006. Referenziert auf den Seiten 6, 11, 18 und 21.

[Bro13] Felix Brosius. Distanz- und Ähnlichkeitsmaße. In *SPSS 21*, Kapitel 31, Seiten 693–709. mitp Professional, 1. Auflage,

2013. Referenziert auf den Seiten 113 und 114.

[BSLH99] Brigitte Bartsch-Spörl, Mario Lenz, André Hübner. Case-Based Reasoning - Survey and Future Directions. *XPS-99: Knowledge-Based Systems - Survey and Future Directions*, Seiten 67–89, 1999. Referenziert auf Seite 126.

[BW13] Manfred Broy, Klaus Webers. Anforderungsmanagement in der Softwareentwicklung. *ATZelektronik*, 8, Nr. 1, Seite 64–69, 2013. Referenziert auf Seite 12.

[CB87] Jeff Conklin, Michael L. Begeman. gIBIS : A Hypertext Tool for Team Design Deliberation. In *Proceedings of the ACM Conference on Hypertext*, Seiten 247–251, Chapel Hill, NC, USA, 1987. ACM Press. Referenziert auf Seite 38.

[CD08] Juan M. Corchado, Juan F. De Paz. Using CBR Systems for Leukemia Classification. *Hybrid Artificial Intelligence Systems - Lecture Notes in Computer Science*, 5271, Seite 688–695, 2008. Referenziert auf Seite 126.

[Cha11] Fadi Chabarek. *Interaktive Vervollständigung von Szenariospezifikationen*. Doktorarbeit, Technische Universität Berlin, 2011. Referenziert auf Seite 8.

[CHCC03] Jane Cleland-Huang, Carl K. Chang, Mark Christensen. Event-Based Traceability for Managing Evolutionary Change. *IEEE Transactions on Software Engineering*, 29, Nr. 9, Seite 796–810, 2003. Referenziert auf Seite 40.

[CHGZ12] Jane Cleland-Huang, Orlena Gotel, Andrea Zisman. *Software and Systems Traceability*. Springer, 1. Auflage, 2012. Referenziert auf Seite 23.

[CHSB+05] Jane Cleland-Huang, Raffaella Settimi, Oussama Ben-Khadra, Eugenia Berezhanskaya, Selvia Christina. Goal-Centric Traceability for Managing Non-Functional Require-

ments. In *Proceedings of the 27th International Conference on Software Engineering (ICSE)*, Seiten 362–371, St. Louis, MO, USA, 2005. ACM Press. Referenziert auf Seite 41.

[CNHR13] Anastasia Cmyrev, Ralf Nörenberg, Daniel Hopp, Ralf Reissing. Consistency Checking of Feature Mapping between Requirements and Test Artefacts. *Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment*, 1, Nr. 1, Seite 121–132, 2013. Referenziert auf Seite 42.

[DIN45020] Deutsches Institut für Normung. *Normung und damit zusammenhängende Tätigkeiten - Allgemeine Begriffe (DIN EN 45020:2007-03)*, 2007. Referenziert auf Seite 98.

[DOORS] IBM. *http://www.ibm.com/software/products/de/ratidoor, DOORS (Dynamic Object Oriented Requirements System)*, 24.02.2015. Referenziert auf den Seiten xv, xvi, 40, 47, 48, 82, 104 und 139.

[EG05] Alexander Egyed, Paul Grünbacher. Supporting Software Understanding with Automated Requirements Traceability. *International Journal of Software Engineering and Knowledge Engineering*, 15, Nr. 5, Seite 783–810, 2005. Referenziert auf Seite 41.

[EWCBR] Proceedings of European Workshop on Case-Based Reasoning. *http://dblp.uni-trier.de/db/conf/ewcbr/index.html*, 24.02.2015. Referenziert auf Seite 127.

[FC07] Syed Ahsan Fahmi, Ho-Jin Choi. Software Reverse Engineering to Requirements. In *Proceedings of 2007 International Conference on Convergence Information Technology (ICCIT 2007)*, Seiten 2199–2204, Gyeongju, Korea, November 2007. Referenziert auf Seite 10.

[FCGCGAHY96] Carmen Fernández-Chamizo, Pedro Antonio González-

Calero, Mercedes Gómez-Albarrán, Luis Hernández-Yánez. Supporting Object Reuse Through Case-Based Reasoning. In *Proceedings of 3rd European Workshop on Case-Based Reasoning (EWCBR)*, Ian Smith, Boi Faltings (Herausgeber), Volume 1168, Seiten 135–149, Lausanne, Schweiz, 1996. Springer. Referenziert auf Seite 128.

[FM93] Gilles Fouqué, Stan Matwin. Compositional Software Reuse with Case-Based Reasoning. In *Proceedings of 9th Conference on Artificial Intelligence for Applications*, Seiten 128–134, Orlando, FL, USA, 1993. IEEE Computer Society Press. Referenziert auf Seite 128.

[GF95] Orlena Gotel, Anthony Finkelstein. Contribution Structures. In *Proceedings of the 2nd IEEE International Symposium on Requirements Engineering*, Seiten 1–27, York, England, 1995. IEEE Computer Society Press. Referenziert auf Seite 38.

[GPP+03] Paulo Gomes, Francisco C. Pereira, Paulo Paiva, Nuno Seco, Paulo Carreiro, José L. Ferreira, Carlos Bento. Case-Based Reuse of UML Diagrams. In *Proceedings of the 15th International Conference on Software Engineering & Knowledge Engineering*, Seiten 335–339, San Francisco, CA, USA, 2003. Research Gate. Referenziert auf Seite 127.

[Gri95] Klaus Grimm. *Systematisches Testen von Software - Eine neue Methode und eine effektive Teststrategie*. Dissertation, TU Berlin, 1995. Referenziert auf den Seiten 85 und 87.

[HB05] Matthias Heindl, Stefan Biffl. A Case Study on Value-Based Requirements Tracing. In *Proceedings of the 10th European Software Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Seiten 60–69, Lissabon,

Portugal, 2005. ACM Press. Referenziert auf Seite 41.

[Hei96] Richard Heider. Troubleshooting CFM 56-3 Engines for the Boeing 737 Using CBR and Data-Mining. In *Proceedings of 3rd European Workshop on Case-Based Reasoning (EWCBR)*, Ian Smith, Boi Faltings (Herausgeber), Seiten 512–518, Lausanne, Schweiz, 1996. Springer. Referenziert auf Seite 126.

[HLV12] Graeme Horsman, Christopher Laing, Paul Vickers. A Method for Reducing the Risk of Errors in Digital Forensic Investigations. *Communications and Multimedia Security, Lecture Notes in Computer Science*, 7394, Seite 99–106, 2012. Referenziert auf Seite 127.

[HW86] Ellis Horowitz, Ronald C. Williamson. SODOS : A Software Documentation Support Environment - Its Definition. *IEEE Transactions on Software Engineering*, 12, Nr. 8, Seite 849–859, 1986. Referenziert auf Seite 38.

[ICCBR] Proceedings of International Conference on Case-Based Reasoning. *http://dblp.uni-trier.de/db/conf/iccbr/index.html*, 24.02.2015. Referenziert auf Seite 127.

[IEEE610.12] Institute of Electrical and Electronics Engineers. *IEEE Std 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology*, 1990. Referenziert auf Seite 5.

[IMD05] Suhaimi Ibrahim, Malcolm Munro, Aziz Deraman. Implementing a Document-Based Requirements Traceability: A Case Study. In *Proceedings of International Conference on Software Engineering*, Peter Kokol (Herausgeber), Seiten 124–131, Innsbruck, Österreich, 2005. ACTA Press. Referenziert auf Seite 39.

[ISO26262-1] International Organization for Standardization. *Road Vehicles - Functional Safety, Part 1: Vocabulary (ISO 26262-*

1:2011), 2011. Referenziert auf Seite 81.

[ISO26262-3] International Organization for Standardization. *Road Vehicles - Functional Safety, Part 3: Concept Phase (ISO 26262-3:2011)*, 2011. Referenziert auf Seite 33.

[ISO26262-4] International Organization for Standardization. *Road Vehicles - Functional Safety, Part 4: Product Development: System Level (ISO 26262-4:2011)*, 2011. Referenziert auf Seite 86.

[ISO26262-6] International Organization for Standardization. *Road Vehicles - Functional Safety, Part 6: Product Development: Software Level (ISO 26262-6:2011)*, 2011. Referenziert auf den Seiten 86 und 101.

[ISO9126-1] European Committee for Standardization and International Organization for Standardization. *Software Engineering - Product Quality, Part 1: Quality Model (ISO 9126-1:2001)*, 2001. Referenziert auf den Seiten 11, 20, 32 und 81.

[JL11] Cheng Jung, Jahau Lewis. Accelerating Preliminary Eco-Innovation Design for Products that Integrates Case-Based Reasoning and TRIZ Method. *Journal of Cleaner Production*, 19, Nr. 9-10, Seite 998–1006, 2011. Referenziert auf Seite 126.

[Kai93] Herman Kaindl. The Missing Link in Requirements Engineering. *ACM SIGSOFT Software Engineering Notes*, 18, Nr. 2, Seite 30–39, 1993. Referenziert auf Seite 38.

[KKKC08] Vassilka Kirova, Neil Kirby, Darshak Kothari, Glenda Childress. Effective Requirements Traceability : Models , Tools , and Practices. *BELL LABS Technical Journal*, 12, Nr. 4, Seite 143–157, 2008. Referenziert auf Seite 39.

[Koc03] Frank Koch. *Handbuch Software-und Datenbank-Recht*. Springer, 1. Auflage, 2003. Referenziert auf Seite 97.

[Lea96] David B. Leake. CBR in Context : The Present and Future. In *Case Based Reasoning: Experiences, Lessons, and Future Directions*, David B. Leake (Herausgeber), Kapitel 1, Seiten 3–35. AAAI Press, MIT Press, 1996. Referenziert auf Seite 123.

[Let02] Patricio Letelier. A Framework for Requirements Traceability in UML-Based Projects. In *Proceedings of 1st International Workshop on Traceability in Emerging Forms of Software Engineering*, Seiten 30–41, Edinburgh, Schottland, 2002. Referenziert auf Seite 38.

[Lig09] Peter Liggesmeyer. *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. Spektrum Akademischer Verlag, 2. Auflage, 2009. Referenziert auf Seite 20.

[MBdS11] José Alexandre Matelli, Edson Bazzo, Jonny Carlos da Silva. Development of a Case-Based Reasoning Prototype for Cogeneration Plant Design. *Applied Energy*, 88, Nr. 9, Seite 3030–3041, September 2011. Referenziert auf Seite 126.

[MH00] Mirjam Minor, Alexandre Hanft. The Life Cycle of Test Cases in a CBR System. *Advances in Case-Based Reasoning, Lecture Notes in Computer Science*, 1898, Seite 455–466, 2000. Referenziert auf Seite 128.

[ML/SL] Mathworks. *http://www.mathworks.de/products/simulink, Matlab Simulink*, 24.02.2015. Referenziert auf Seite 14.

[Moo03] Christopher W. Moore. *The Mediation Process: Practical Strategies for Resolving Conflicts*. Wiley, 3. Auflage, 2003. Referenziert auf Seite 7.

[MV86] Jean MacMillan, John R. Vosburgh. Software Quality Indicators. Technischer Report, Scientific System Inc., Cambridge MA., 1986. Referenziert auf Seite 40.

[MWC+11] Cindy Marling, Matthew Wiley, Tessa Cooper, Razvan Bu-

nescu, Jay Shubrook, Frank Schwartz. The 4 Diabetes Support System: A Case Study in CBR Research and Development. *Case-Based Reasoning Research and Development, Lecture Notes in Computer Science*, 6880, Seite 137–150, 2011. Referenziert auf Seite 126.

[Nar10] Murat Narmanli. *A Business Rule Approach to Requirements Traceability.* Masterarbeit, Middle East Technical University, 2010. Referenziert auf Seite 38.

[NHK14] Thomas Noack, Steffen Helke, Thomas Karbe. Reuse-Based Test Traceability: Automatic Linking of Test Cases and Reusing Requirements. *International Journal On Advances in Software*, 7, Nr. 3/4, 2014. Referenziert auf Seite 42.

[Noe12] Ralf Noerenberg. *Effizienter Regressionstest von E/E-Systemen nach ISO 26262.* Doktorarbeit, Karlsruher Institut für Technologie, 2012. Referenziert auf Seite 22.

[OGPD10] Rocco Oliveto, Malcom Gethers, Denys Poshyvanyk, Andrea De Lucia. On the Equivalence of Information Retrieval Methods for Automated Traceability Link Recovery. In *18th IEEE International Conference on Program Comprehension (ICPC)*, Seiten 68–71, Minho, Portugal, June 2010. IEEE Computer Society Press. Referenziert auf Seite 41.

[Par10] Helmuth Partsch. *Requirements-Engineering systematisch - Modellbildung für softwaregestützte Systeme.* Springer, 2. Auflage, 2010. Referenziert auf den Seiten 11 und 23.

[PBKS07] Alexander Pretschner, Manfred Broy, Ingolf H. Kruger, Thomas Stauner. Software Engineering for Automotive Systems: A Roadmap. In *Proceedings of 7th International Conference on Future of Software Engineering*, Seiten 55–71, Washington, DC, USA, 2007. Referenziert auf Seite 19.

[PG96] Francisco A. C. Pinheiro, Joseph A. Goguen. An Object-Oriented Tool for Tracing Requirements. In *Proceedings of the 2nd International Conference on Requirements Engineering*, Seiten 52–64, Colorado Springs, CO, USA, 1996. IEEE Computer Society Press. Referenziert auf Seite 38.

[PH95] Klaus Pohl, Peter Haumer. HYDRA: A Hypertext Model for Structuring Informal Requirements Representations. In *Proceedings of the 2nd International Workshop on Requirements Engineering (REFSQ)*, Klaus Pohl, Peter Peters (Herausgeber), Seiten 118–134, Jyväskylä, Finnland, 1995. Verlag der Augustinus-Buchhandlung. Referenziert auf Seite 38.

[Pin04] Francisco A. C. Pinheiro. Requirements Traceability. In *Perspectives on Software Requirements*, Julio Cesar Smapaio do Prado Leite, Jorge Horacio Doorn (Herausgeber), Kapitel 5, Seiten 91–113. Kluwer Academic Publishers, 1. Auflage, 2004. Referenziert auf den Seiten 25 und 38.

[Poh08] Klaus Pohl. *Requirements Engineering - Grundlagen, Prinzipien, Techniken.* dpunkt.verlag, 2. Auflage, 2008. Referenziert auf den Seiten 6, 7, 8, 9, 11, 12, 23 und 25.

[ProdHG] *Gesetz über die Haftung für fehlerhafte Produkte (Produkthaftungsgesetz - ProdHaftG)*, 2002. Referenziert auf den Seiten 97, 98 und 100.

[RD92] Balasubramaniam Ramesh, Vasant Dhar. Supporting Systems Development by Capturing Deliberations During Requirements Engineering. *IEEE Transactions on Software Engineering*, 18, Nr. 6, Seite 498–510, 1992. Referenziert auf Seite 38.

[Rie04] Matthias Riebisch. Supporting Evolutionary Development by Feature Models and Traceability Links. In *11th IEEE International Conference and Workshop on the Engineer-*

ing of Computer-Based Systems (ECBS), Seiten 370–377, Brno, Tschechien, 2004. IEEE Computer Society Press. Referenziert auf Seite 40.

[Rup12] Chris Rupp. *Requirements Engineering - Ein Überblick*. dpunkt.verlag, 3. Auflage, 2012. Referenziert auf Seite 5.

[Rup13] Chris Rupp. Schablonen für alle Fälle. Technischer Report, Die SOPHISTen, 2013. Referenziert auf den Seiten 8 und 11.

[RWA07] Siti Rochimah, Wan M. N. Wan Kadir, Abdul H. Abdullah. An Evaluation of Traceability Approaches to Support Software Evolution. In *International Conference on Software Engineering Advances (ICSEA 2007)*, Seiten 19–38, Cap Esterel, Frankreich, August 2007. IEEE Computer Society. Referenziert auf Seite 40.

[Sal06] Ahmed M. Salem. Improving Software Quality through Requirements Traceability Models. In *Proceedings of IEEE / ACS International Conference on Computer Systems and Applications (AICCSA)*, Seiten 1159–1162, Dubai, VAE, 2006. IEEE Computer Society Press. Referenziert auf Seite 38.

[SBS11] Harry M. Sneed, Manfred Baumgartner, Richard Seidl. *Der Systemtest: Von den Anforderungen zum Qualitätsnachweis*. Carl Hanser Verlag, 3. Auflage, 2011. Referenziert auf den Seiten 20 und 21.

[SimMetrics] Source Forge. *SimMetrics (Open Source Bibliothek mit zahlreichen Algorithmen zur Bewertung der Ähnlichkeit zweier Texte)*, 2014. Referenziert auf Seite 50.

[SKL11] Gerald Spindler, Robert Koch, Egon Lorenz. *Karlsruher Forum 2010: Haftung und Versicherung im IT-Bereich*. Verlag Versicherungswirtschaft, Karlsruhe, 1. Auf-

lage, 2011. Referenziert auf Seite 97.

[SL05] Andreas Spillner, Tilo Linz. *Basiswissen Softwaretest.* dpunkt.verlag, Heidelberg, 4. Auflage, 2005. Referenziert auf den Seiten 13, 14, 15, 20, 21, 27, 32, 81 und 90.

[Som12] Ian Sommerville. *Software Engineering.* Pearson, 9. Auflage, 2012. Referenziert auf Seite 11.

[SRDD11] Martin Schmidt, Marcus Rau, Bernhard Dr. Bauer, Ekkehard Dr. Helmig. Rechtliche Folgen der ISO 26262. *HANSER Automotive*, 11, Seite 38–42, 2011. Referenziert auf den Seiten 96 und 100.

[SRWL14] Andreas Spillner, Thomas Rossner, Mario Winter, Tilo Linz. *Praxiswissen Softwaretest: Testmanagement.* dpunkt.verlag, 4. Auflage, 2014. Referenziert auf Seite 27.

[SS97] Ian Sommerville, Pete Sawyer. *Requirements Engineering.* Wiley, 1. Auflage, 1997. Referenziert auf Seite 11.

[Sto13] Nadya Stoyanova. *Verbesserung der Qualität natürlichsprachlicher Spezifikationen.* Doktorarbeit, Universität Stuttgart, 2013. Referenziert auf Seite 8.

[SZ10] Jörg Schäuffele, Thomas Zurawka. *Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen.* Springer Vieweg, 4. Auflage, 2010. Referenziert auf den Seiten 6, 11 und 18.

[SZPMK04] George Spanoudakis, Andrea Zisman, Elena Pérez-Minana, Paul Krause. Rule-Based Generation of Requirements Traceability Relations. *Journal of Systems and Software*, 72, Nr. 2, Seite 105–127, July 2004. Referenziert auf Seite 40.

[TBA12] Axel Tidemann, Finn Olav Bjornson, Agnar Aamodt. Operational Support in Fish Farming through Case-Based Reasoning. In *Proceedings of 25th International Conference*

on Industrial Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE), Randy Goebel, Yuzuru Tanaka, Wolfgang Wahlster, Jörg Siekmann (Herausgeber), Seiten 104–113, Dalina, China, 2012. Springer. Referenziert auf Seite 127.

[TGF+12] Richard Torkar, Tony Gorschek, Robert Feldt, Mikael Svahnberg, Uzair Akbar Raja, Kashif Kamran. Requirements Traceability: A Systematic Review and Industry Case Study. *International Journal of Software Engineering and Knowledge Engineering*, 22, Nr. 3, Seite 385–433, 2012. Referenziert auf Seite 40.

[TKTW09] Bernhard Turban, Markus Kucera, Athanassios Tsakpinis, Christian Wolff. Bridging the Requirements to Design Traceability Gap. *Intelligent Technical Systems, Lecture Notes in Electrical Engineering*, 38, Seite 275–288, 2009. Referenziert auf Seite 38.

[TM00] Toshihiko Tsumaki, Yoshitomi Morisawa. A Framework of Requirements Tracing using UML. In *Proceedings of 7th Asia-Pacific Software Engineering Conference (APSEC)*, Seiten 206–213, Singapur, Singapur, 2000. IEEE Computer Society Press. Referenziert auf Seite 38.

[Tra88] Will Tracz. Software Reuse Myths. *ACM SIGSOFT Software Engineering Notes*, 13, Nr. 1, Seite 17–21, 1988. Referenziert auf Seite 127.

[TWP98] Bjornar Tessem, R. Alan Whitehurst, Christopher L. Powell. Retrieval of Java Classes for Case-Based Reuse. In *4th European Workshop on Advances in Case-Based Reasoning (EWCBR)*, Barry Smith, Páraig Cunningham (Herausgeber), Seiten 148–159, Dublin, Irland, 1998. Springer. Referenziert auf Seite 128.

[VSM12] Beatriz Sevilla Villanueva, Miquel Sánchez-Marré. Case-

Based Reasoning Applied to Textile Industry Processes. In *20th International Conference on Case-Based Reasoning (ICCBR)*, Belén Diaz Agudo, Ian Watson (Herausgeber), Seiten 428–442, Lyon, Frankreich, 2012. Springer. Referenziert auf Seite 126.

[Wat99] Ian Watson. Case-Based Reasoning is a Methodology not a Technology. *Knowledge-Based Systems*, 12, Nr. 5, Seite 303–308, 1999. Referenziert auf Seite 124.

[YC12] Aisha Yousuf, William Cheetham. Case-Based Reasoning for Turbine Trip Diagnostics. In *20th International Conference on Case-Based Reasoning (ICCBR)*, Belén Diaz Agudo, Ian Watson (Herausgeber), Seiten 458–468, Lyon, Frankreich, 2012. Springer. Referenziert auf Seite 126.

[ZWZX12] Jie Zhang, Rujing Wang, Linli Zhou, Guo Xu. The Application of Socio-Economic Data in the Loss Assessment of Disasters Based on Case-Based Reasoning. In *Proceedings of IEEE Symposium on Robotics and Applications (ISRA)*, Seiten 498–501, Kuala Lumpur, Malaysia, 2012. IEEE Computer Society Press. Referenziert auf Seite 127.