

OPTIMIERTE VARIANTENSELEKTION FÜR EINEN EFFIZIENTEN E/E-PRODUKTLINIENTEST

vorgelegt von
Anastasia Cmyrev, M.Sc.
geb. in Kustanai, Kasachstan

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktorin der Naturwissenschaften
— Dr. rer. nat. —

genehmigte Dissertation

PROMOTIONSAUSSCHUSS:

Prof. Dr. rer. nat. Rolf Niedermeier (Vorsitzender)
Prof. Dr.-Ing. Stefan Jähnichen (Gutachter)
Prof. Dr. rer. nat. Ralf Reißing (Gutachter)
Prof. Dr.-Ing. Steffen Helke (Gutachter)

TAG DER WISSENSCHAFTLICHEN AUSSPRACHE:

11. Juli 2014

Berlin 2014
D 83

Technische Universität Berlin

Fakultät Elektrotechnik und Informatik
Fachgebiet Softwaretechnik
Ernst-Reuter-Platz 7
D-10587 Berlin

Hochschule Coburg

Fakultät Maschinenbau und Automobiltechnik
Friedrich-Streib-Straße 2
D-96450 Coburg

Brandenburgische Technische Universität Cottbus-Senftenberg

Fakultät Mathematik, Naturwissenschaften, Informatik
Lehrstuhl Sichere Softwaresysteme
Platz der Deutschen Einheit 1
D-03046 Cottbus

Anastasia Cmyrev: *Optimierte Variantenselektion für einen effizienten E/E-Produktlinientest,*

© 2014

Für meine Eltern.

DANKSAGUNG

Die Anfertigung dieser Arbeit wäre ohne die Unterstützung und das Vertrauen zahlreicher Personen nicht möglich gewesen.

Mein besonderer Dank gilt Herrn Prof. Dr. Stefan Jähnichen für die Ermöglichung und die umfassende Betreuung der Arbeit, die vielen hilfreichen Anmerkungen und seine Motivation zum Schreiben der Dissertation. Herrn Prof. Dr. Ralf Reißing danke ich herzlich für die Übernahme des Zweitgutachtens, die intensive Betreuung über den gesamten Zeitraum sowie seine stete Diskussionsbereitschaft und Hilfestellung bei zahlreichen fachlichen Fragen. Bei Herrn Prof. Dr. Steffen Helke bedanke ich mich sehr für die Übernahme und kurzfristige Anfertigung des Drittgutachtens.

Ebenfalls bedanke ich mich bei Herrn Michael Weber für die Anstellung als Doktorandin im Rahmen des Doktorandenprogramms der Daimler AG und Unterstützung bspw. durch die Etablierung der regelmäßigen Doktorandenseminare. Ein ganz besonderer Dank geht an den ehemaligen Leiter des Teams Testmethodik, Herrn Dr. Michael Stotz, der mir das Thema anvertraut und mich während der Promotionszeit und darüber hinaus (strategisch) unterstützt hat. Zudem bedanke ich mich bei Herrn Daniel Hopp für die Einführung in das Thema Variantenmanagement sowie die konstruktive Zusammenarbeit bei teamübergreifenden Themen. Auch danke ich Herrn Dr. Jacques Kamga für die initiiierende Übernahme der fachlichen Betreuung.

Einen großen Beitrag zum Gelingen der Arbeit haben Kollegen und Freunde geleistet. Insbesondere bedanke ich mich bei: Andreas Dereani, Kerstin Hartig, Mustafa Jaber, Dr. Thomas Karbe, Tobias Morciniec, Thomas Noack, Dr. Ralf Nörenberg, Dr. Alexander Rein-Jury, Christian Reuter und Dr. Nadya Stoyanova für das Korrekturlesen zahlreicher promotionsrelevanter Dokumente, die fachlichen Diskussionen sowie Motivation zur Vollendung der Dissertation.

Besonders viel Dank gilt Matthias Kühnel, der bei vielfältigsten (inhaltlichen) Fragestellungen unterstützte, mir häufig ein moralischer Ruhepol war und damit erheblich zum erfolgreichen Abschluss der Arbeit beigetragen hat. Schließlich danke ich meinen Eltern, die mich immer darin bestärkt haben den Bildungsweg voranzuschreiten und mir mit Rückhalt und Liebe zur Seite stehen.

ZUSAMMENFASSUNG

Um der Individualität möglichst vieler Kundenwünsche sowie der Volatilität der Märkte begegnen zu können, bieten Automobilhersteller immer mehr Modellvarianten mit einer Vielzahl an Ausstattungsmöglichkeiten an. Dies führt zu einer schnell wachsenden Variantenvielfalt sowie einer steigenden Produktkomplexität, welche während des Entwicklungsprozesses von sogenannten Produktlinien bewältigt werden muss. Insbesondere während der Phase des Testens stellt sich die Frage nach den „richtigen“ zu testenden Varianten, da aufgrund der hohen Anzahl das Testen jeder einzelnen Variante nicht praktikabel bzw. wirtschaftlich wäre.

In der vorliegenden Arbeit wird eine Methode zur Identifikation der repräsentativ zu testenden Varianten vorgestellt. Durch das Testen dieser Varianten wird die Absicherung der gesamten Produktlinie gewährleistet. Hierfür wird zunächst eine einheitliche Dokumentationsbasis der Variabilität über die verschiedenen Entwicklungsphasen hinweg definiert. Dabei erfolgt die Beschreibung der Gemeinsamkeiten und Unterschiede zwischen den Varianten innerhalb *eines* universellen Merkmalmodells. Ausgehend davon werden Merkmale auf die Entwicklungsartefakte, wie Anforderungen und Testfälle, abgebildet. Dadurch liegen Anforderungs- und Testspezifikationen vor, welche für die gesamte Produktlinie gelten. Mittels einer Konsistenzprüfung wird die Durchgängigkeit der abgebildeten Variabilität zwischen Artefakten unterschiedlicher Entwicklungsphasen sichergestellt. Mithilfe von zwei verschiedenen Optimierungsalgorithmen werden die zu testenden Varianten selektiert, welche jede im Lastenheft spezifizierte Anforderung sowie jedes im entsprechenden Merkmalmodell definierte Merkmal mindestens einmal abdecken.

Die Praxistauglichkeit sowie Effektivität der Konsistenzprüfung und Selektionsmethodik konnten anhand von zwei realen E/E-Produktlinien der Daimler AG nachgewiesen werden. Die Beseitigung der identifizierten Inkonsistenzen zwischen Entwicklungsartefakten führte zu einer Steigerung der Qualität in Spezifikationsdokumenten. Darüber hinaus bewirkte die anforderungs- und merkmalsbasierte Selektion der zu testenden Varianten eine signifikante und gleichzeitig systematische Reduktion des Testumfangs.

ABSTRACT

Within the automotive industry, the customers' high demand for individual products as well as the markets' volatility are major reasons for the large number of possible configurations. This results in a rapidly growing space of vehicle variants and an increasing complexity which the manufacturer has to cope with during the development process of so-called product lines. Especially during the testing phase the „right“ variants have to be determined since the testing of each variant would be infeasible and economically unjustifiable.

In this thesis, a new test approach based on requirements coverage and feature coverage is developed. For this purpose, first a concept for variability documentation across the different development phases is defined. To represent the commonality and variability of variants a single universal feature model is used. Subsequently, the features are mapped to the development artifacts such as requirements and test cases. As a result, the according specification documents contain artifacts which are valid for the whole product line. To ensure a consistent traceability of feature mapping between the different development artifacts the concept of consistency checking is developed. Two different optimization algorithms are used to select a small subset of variants which cover every requirement of the requirements specification and every feature of the feature model at least once. This small set of variants is then to be tested in detail. Because of the coverage achieved by the variant set, the quality of all other variants can be inferred from the test results.

The usability and effectiveness of the consistency checking and the selection method has been demonstrated using two real E/E product lines of Daimler AG. The elimination of identified inconsistencies between development artifacts has led to a quality intensification within specification documents. Additionally, the systematic requirements and feature based selection of variants resulted in a significant reduction of testing effort.

VERÖFFENTLICHUNGEN

Die in dieser Dissertation vorgestellte Arbeit entstand zwischen **November 2010** und **Oktober 2013** an der Technischen Universität Berlin, Fakultät für Elektrotechnik und Informatik. Einige Ideen und Darstellungen sind bereits in den im Folgenden aufgelisteten vorherigen Veröffentlichungen erschienen:

Veröffentlichungen im Rahmen der Dissertation:

- **A. Cmyrev**, R. Reissing: *Efficient and Effective Testing of Automotive Software Product Lines*, 1st International Conference on Engineering Science and Innovative Technology (ESIT), April 2014, Krabi, Thailand
- **A. Cmyrev**, R. Reißing: *Optimierte Varianten- und Anforderungsabdeckung im Test*, 43. GI Jahrestagung, 11. Workshop Automotive Software Engineering, September 2013, Koblenz
- **A. Cmyrev**, R. Nörenberg, D. Hopp, R. Reißing: *Durchgängiges Variantenmanagement auf Basis eines Merkmalmodells für effizientes Testen hoch variabler Systeme*, 4. AutoTest Fachkonferenz „Test von Hard- und Software in der Automobilentwicklung“, Oktober 2012, Stuttgart
- **A. Cmyrev**, R. Noerenberg, D. Hopp, R. Reissing: *Consistency Checking of Feature Mapping between Requirements and Test Artefacts*, 19th ISPE International Conference on Concurrent Engineering (CE), September 2012, Trier
→ Auszeichnung des Beitrags mit dem **Best Paper Award** durch die International Society for Productivity Enhancement (ISPE)
- R. Nörenberg, **A. Cmyrev**, R. Reißing und K. D. Müller-Glaser: *An Efficient Specification-Based Regression Test Selection Technique for E/E-Systems*, 41. GI Jahrestagung, 9. Workshop Automotive Software Engineering, Oktober 2011, Berlin
- R. Nörenberg, **A. Cmyrev**, R. Reißing und K. D. Müller-Glaser: *Efficient verification planning for ISO-conformant functional testing of automotive applications*, Technische Akademie Esslingen, 4. Symposium Testen im System- und Software-Life-Cycle, November 2011, Esslingen

Invited Talks im Rahmen der Dissertation:

- **A. Cmyrev**: *Merkmalbasiertes Variantenmanagement für effizienten Test von Produktlinien im E/E-Bereich*, ATAMI Workshop 2013 - Advances in Testing - Academia meets Industry, Fraunhofer FOKUS, Mai 2013, Berlin

Vorherige Veröffentlichungen:

- V. Solovyeva, **A. Cmyrev**, R. Sachser, H. Reith und M. Huth: *Influence of irradiation-induced disorder on the Peierls transition in TTF-TCNQ microdomains*, Journal of Physics D: Applied Physics 44, 385301, September 2011

INHALTSVERZEICHNIS

I	EINLEITUNG	1
1	Einleitung	3
1.1	Problembeschreibung	4
1.2	Ziele und Beiträge	6
1.3	Struktur der Arbeit	7
II	GRUNDLAGEN UND VERWANDTE ARBEITEN	9
2	Absicherung von eingebetteten Systemen	11
2.1	Eingebettetes System	11
2.2	Entwicklungsprozess	12
2.3	Anforderungsspezifikation	14
2.4	Softwaretest	16
2.4.1	Statischer Test	16
2.4.2	Dynamischer Test	17
2.5	Methodische Testfallableitung	18
2.5.1	Testkonzeption	19
2.5.2	Testspezifikation	20
3	Produktlinienentwicklung	23
3.1	Einführung in die Produktlinienentwicklung	23
3.2	Domänen- und Applikationsentwicklung	24
3.3	Variabilität in Produktlinien	27
3.3.1	Variante vs. Version	28
3.3.2	Merkmalmmodellierung	28
3.3.3	Orthogonale Variabilitätsmodellierung	31
3.3.4	Entscheidungsmodellierung	32
4	Testen von Produktlinien	35
4.1	Herausforderungen des Produktlinientests	35
4.2	Methoden des Produktlinientests	36
4.2.1	Aufteilung nach Verantwortlichkeiten	36
4.2.2	Product by Product Testen	38
4.2.3	Inkrementelles Testen	39
4.2.4	Modellbasierter Produktlinientest	41
4.2.5	Priorisierung	43
4.2.6	Kombinatorisches Testen	47
4.2.7	Anforderungsbasierte Selektion	50
4.3	Evaluierung der Methoden	52
III	OPTIMIERTE VARIANTENSELEKTION FÜR EFFIZIENTEN E/E-PRODUKTLINIEN-TEST	55
5	Variabilität in Entwicklungsartefakten	57
5.1	Überblick über die Produktlinieninfrastruktur	57
5.2	Merkmalmmodellierung im Variantenmodul	58

5.3	Merkmalabbildung in Entwicklungsartefakten	61
5.4	Konsistenzmanagement	63
5.4.1	Konsistenzprüfung	65
5.4.2	Ursachen und Folgen von Inkonsistenzen	77
5.4.3	Behebung von Inkonsistenzen	79
6	Selektionsmethodik	85
6.1	Konzept der Optimierung	85
6.2	Optimierungsvorgehen	87
6.2.1	Basis des Optimierungsvorgehens	88
6.2.2	Optimierung mittels Greedy-Algorithmus	90
6.2.3	Optimierung mittels Simulierter Abkühlung	97
6.3	Selektion von Testfällen	106
7	Umsetzung und Validierung	109
7.1	Werkzeugunterstützung	109
7.2	Aufbau der Fallstudien	110
7.3	Fallstudien zur Konsistenzprüfung	112
7.4	Fallstudien zur Selektionsmethodik	117
7.4.1	Fallstudien zur Selektion mittels Greedy-Algorithmus	117
7.4.2	Fallstudien zur Selektion mittels Simulierter Abkühlung	121
7.4.3	Ausleitung von Testfällen	128
7.5	Evaluierung und Zusammenfassung der Ergebnisse	128
IV	ABSCHLUSS	133
8	Zusammenfassung und Ausblick	135
8.1	Fazit	135
8.2	Ausblick	137
V	ANHANG	139
A	Konsistenzmanagement	141
A.1	Disjunktive Assoziationssemantik	141
A.2	Konsistenzprüfung	141
B	Algorithmen Abkürzungsverzeichnis	145
	Abbildungsverzeichnis	147
	Tabellenverzeichnis	152
	Literaturverzeichnis	155

Teil I

EINLEITUNG

1 | EINLEITUNG

Das heutige Automobil stellt eines der komplexesten technischen Systeme dar. Es besteht aus einer Vielzahl integrierter Funktionseinheiten, welche weit über die grundlegende Funktionalität des Fahrens hinausgehen und Aspekte wie Komfort und Unterhaltung sowie Umweltfreundlichkeit und maximale Sicherheit umfassen. Der Großteil dieser Funktionen wird mithilfe von Elektrik/Elektronik (E/E) realisiert, da diese als flexibler, effizienter und schließlich zuverlässiger im Gegensatz zu überwiegend mechanischen Lösungen gilt. So sind bereits heute zwischen 50 und 100 elektronischer Steuereinheiten, welche auch als eingebettete Systeme bezeichnet werden, in einem Fahrzeug der gehobenen Mittelklasse integriert [100].

Rund 90% aller Innovationen im Automobil werden auf Basis der Elektrik und Software umgesetzt - Tendenz steigend [91]. Eingebettete Systeme fungieren somit als entscheidende Technologie für Weiterentwicklungen sowie Produktneuerungen und sind daher als wichtige Innovationstreiber zu sehen [127]. In diesem Kontext zählen beispielsweise das unfallfreie und das autonome Fahren zu den Zukunftsvisionen der Fahrzeugentwicklung, deren Umsetzung ohne Elektronik und Software undenkbar wäre. Zudem tragen Softwarekomponenten von E/E-Systemen in hohem Maße zur Produktdifferenzierung bei [127] und spielen daher für die Wettbewerbsfähigkeit eines Unternehmens eine zentrale Rolle.

Ein besonders wichtiges Instrument im Zusammenhang mit der Wettbewerbsfähigkeit stellt die Diversifizierung von Automobiltypen dar. Durch die Vielfalt soll jedem Kunden die Möglichkeit gegeben werden, das Fahrzeug gemäß seiner Präferenzen bezüglich Größe, Motorisierung, Ausstattung usw. individuell zu bestimmen. Dies steht im Gegensatz zur Einzelfertigung, welche auf die Entwicklung von Unikaten ausgerichtet und noch bis etwa Mitte des 19. Jahrhunderts vorherrschend war. In dem anschließenden Zeitalter der Industrialisierung kam es zunehmend zur maschinellen Erzeugung von Gütern. Dadurch wurde es möglich die Anzahl der hergestellten Produkte unter Berücksichtigung von verschiedenen Kunden und Märkten stark zu steigern. Im Rahmen der daraus resultierenden individualisierten Massenfertigung werden heute ganze Fahrzeugfamilien, auch als (Software-)Produktlinien bezeichnet, entwickelt und hergestellt [90].

Produktlinien ermöglichen es nicht nur, eine zunehmend heterogene Kundenbasis zu bedienen, sondern auch Länderspezifika zu adressieren, technologische Umsetzungen zu erleichtern und strategische Entscheidungen umzusetzen [49]. Im Bezug auf das Automobil entsteht jedoch eine Situation, bei welcher die Endprodukte zahlreiche individuelle Zusammensetzungen aus Hardware und Software darstellen, sodass kaum ein Fahrzeug identisch zu einem vorherigen das Werk verlässt [83]. Als Folge davon steht ein OEM nicht nur vor der Aufgabe die komplexer werdende Vernetzung zwischen den elektronischen Steuereinheiten, sondern auch die steigende Variabilität der Produkte und deren Bestandteile beherrschen zu müssen [76].

Um die Herausforderungen erfolgreich bewältigen zu können, wird der methodische Ansatz der Produktlinienentwicklung eingesetzt. Der zentrale Grundgedanke dieses Konzeptes beruht auf der systematischen Wiederverwendung [90]. Mithilfe der Wiederverwendung können Gemeinsamkeiten zwischen Produkten ausgenutzt werden, sodass Produkte mit ähnlichen Eigenschaften nicht mehr einzeln, sondern ausgehend von einer Basis von wiederverwendbaren Artefakten (z.B. Anforderungen, Architekturelementen oder Testfällen) entwickelt werden. Die Gemeinsamkeiten und Unterschiede zwischen den Produkten werden in einem Variabilitätsmodell in Form von Merkmalen dargestellt. Mittels des Variabilitätsmodells können alle validen Produkte der Produktlinie bestimmt bzw. konfiguriert werden, sodass in diesem Zusammenhang auch von Konfigurationen gesprochen wird. Als Ergebnis führt die Produktlinienentwicklung zu einer Reihe von Vorteilen, welche durch kürzere Produktionszeiten, geringere Kostenaufwände und höhere Qualitätseigenschaften gekennzeichnet sind und in zahlreichen Industrieprojekten bestätigt werden konnten [23, 104].

Jedoch ist die Thematik des Testens in der Produktlinienentwicklung noch nicht ausreichend entwickelt. Insbesondere im industriellen Umfeld besteht nach wie vor die Problematik einer effizienten Absicherung aller Konfigurationen, welche aus einer Produktlinie abgeleitet werden können. Da die hohe Anzahl an Konfigurationen die praktische Absicherung jeder einzelnen davon unmöglich macht, müssen repräsentative Umfänge identifiziert werden, welche innerhalb der zeitlich begrenzten Testphase sinnvollerweise abgesichert werden sollten. Darüber hinaus können sich eventuelle Fehler aufgrund der Entwicklung ganzer Produktlinien in zahlreiche Produkte ausbreiten und zu unerwünschten Auswirkungen führen. Aus diesem Grund ist ein systematisches und transparentes Vorgehen beim Produktlinientest zwingend erforderlich.

1.1 Problembeschreibung

Testen ist eine der wichtigsten Maßnahmen zur Steigerung des Vertrauens in die korrekte Funktionalität sowie zur Bewertung des angestrebten Qualitätsgrades von Softwaresystemen. Der hohe Stellenwert spiegelt sich unter anderem in den hohen Aufwendungen, welche in der Einzelsystementwicklung mit 25% bis 50% der Entwicklungszeit und -kosten beziffert werden, wider [111]. Dabei wird ein möglichst frühzeitiges Aufdecken von potentiellen Fehlern in Softwaresystemen angestrebt. Denn je später ein Fehler erkannt wird, umso höhere Ausbesserungskosten zieht dieser nach sich [63].

Ausgehend davon kann eine Produktlinie als eine Menge von Einzelsystemen, d.h. Konfigurationen, verstanden werden. Die daraus folgende, naheliegende Idee ist es, jede dieser Konfigurationen während der Testphase einer Produktlinie isoliert und vollständig abzusichern. Insbesondere im Automobilsektor kann aufgrund des ausgeprägten Variantenreichtums die ableitbare Anzahl an Konfigurationen aus einer Produktlinie jedoch derart hoch sein, dass ein solches umfassendes Testen unter Einhaltung von Terminplänen und Kostenbudgets praktisch nicht realisierbar ist. Aus dieser Problematik erwachsen folgende zentrale Herausforderungen an einen erfolgreichen und wirtschaftlichen Produktlinientest:

- **Bereitstellung einer wiederverwendbaren Artefaktbasis.** Um die Testumfänge für jede zu testende Konfiguration nicht erneut definieren zu müssen, können Testfälle innerhalb einer für die gesamte Produktlinie geltenden Testspezifikation festgelegt werden. Hiermit wird der Aufwand an die Dokumentation und Bereitstellung der entsprechenden Testfälle reduziert. Jedoch ergibt sich die Frage, wie die Variabilität der Produktlinie in den Testfällen (und weiteren Entwicklungsartefakten) explizit abzubilden ist, um die Artefakte jederzeit und an jede Konfiguration anpassen zu können. Hierbei müssen wiederverwendbare Artefakte besonders sorgsam spezifiziert und verwaltet werden, um ein Fortpflanzen von möglichen Fehlern in die Artefakte einzelner Konfigurationen zu verhindern.
- **Durchführung der Testfälle für eine Produktlinie.** Die Tatsache, dass alle Konfigurationen einer Produktlinie gewisse Ähnlichkeiten aufweisen, suggeriert die Möglichkeit, die gemeinsamen Eigenschaften nicht in allen, sondern nur in ausgewählten Konfigurationen zu testen. Dadurch können Redundanzen vermieden und erhebliche Kosten- sowie Zeiteinsparungen erreicht werden. Gleichzeitig stellt sich die Frage, welche, wie viele und auf Basis welcher Testfälle die Konfigurationen getestet werden müssen, um eine ausreichende Absicherung sicherzustellen.

Der erste Aspekt beschreibt die notwendige Voraussetzung zur Entwicklung und Anwendung einer effizienten Teststrategie für Produktlinien. Diesbezüglich existieren zahlreiche Ansätze, die sich mit der generellen Erfassung und Modellierung der Variabilität und deren Abbildung in Artefakten beschäftigen. Einige davon empfehlen den Einsatz *eines* zentralen Variabilitätsmodells zur durchgängigen Verwaltung der Variabilität über die Entwicklungsphasen hinweg [6, 101]. Jedoch fehlt es an weitergehenden Konzepten, mit deren Hilfe die einheitliche und konsistente Definition der Variabilität über die Phasen geprüft und unterstützt wird, um dadurch Fehler in den für die gesamte Produktlinie geltenden Entwicklungsdokumenten möglichst gering zu halten.

Mit Hilfe einer wiederverwendbaren Artefaktbasis kann die zweite Fragestellung adressiert werden. Hierfür werden häufig, analog zu den klassischen Testvorgehen, Abdeckungskriterien definiert, mit deren Hilfe die zu testenden Konfigurationen ermittelt werden. Die Abdeckungskriterien nutzen dabei die Ähnlichkeiten zwischen den Konfigurationen, um durch möglichst wenige Testfälle eine möglichst umfassende Testabdeckung einer Produktlinie zu erreichen.

So existieren Ansätze, welche verschiedene Abdeckungsgrade im Bezug auf die Wechselwirkungen zwischen Merkmalen im Variabilitätsmodell definieren [25, 83], um durch den Test entsprechender Konfigurationen insbesondere Fehler an Schnittstellen, an denen Merkmale Informationen austauschen, aufzudecken. Weitere Ansätze verwenden die Abdeckungskriterien der klassischen modellbasierten Entwicklung [86, 96] mit dem Fokus auf der Erweiterung von Testmodellen zu solchen, welche die Beschreibung der gesamten Produktlinie ermöglichen. Zudem werden Kriterien zur Priorisierung von zu testenden Konfigurationen genutzt [15, 69]. Schließlich dienen Anforderungen und Architekturelemente als Basis zur Festlegung der optimalen Konfigurationsmenge [100].

Aus der hohen Anzahl an Literaturbeiträgen geht hervor, dass die Herausforderungen an den Produktlinientest gut verstanden sind und die Thematik einen insgesamt hohen Stellenwert hat [38]. Gleichzeitig weisen viele Methoden Schwachstellen auf. So handelt es sich häufig um *Konzeptvorschläge*, deren Praxistauglichkeit begrenzt ist [77]. Die fehlende Praxisnähe hängt damit zusammen, dass die Methoden für sehr spezielle Problemstellungen ausgearbeitet werden. Der Übertrag auf weitere Projekte wird damit stark eingeschränkt. Ein weiteres Defizit ist durch einen zu hohen Aufwand gegeben, der aufgebracht werden muss, um eine Methode anwenden zu können. Dabei steht der Aufwand zur Identifikation der repräsentativen Umfänge oft nicht im Verhältnis zu dem Nutzen.

Für einen erfolgreichen Entwicklungs- und Testprozess von software-basierten Produktlinien sind daher weiterführende Konzepte notwendig, welche im Sinne der Produktlinienentwicklung das Konzept der Wiederverwendung umsetzen und in der Praxis anwendbar machen.

1.2 Ziele und Beiträge

Ausgehend von den identifizierten Herausforderungen sowie Grenzen existierender Beiträge sind für die vorliegende Arbeit drei zentrale Zielsetzungen abgeleitet worden:

Ziel 1. *Entwicklung einer werkzeug-unterstützten Methodik zur konsistenten Abbildung und Verwaltung der Variabilität in Spezifikationsdokumenten.*

Das erste Ziel adressiert die Notwendigkeit einer wiederverwendbaren Artefaktbasis. Hierfür wird ein Konzept, welches bei der Daimler AG zur Abbildung der Variabilität in Anforderungsartefakten bereits erfolgreich verwendet wird, für den Einsatz in der Umgebung von Testspezifikationen geprüft und integriert. Wichtig ist dabei, dass die Variabilität *einer* Produktlinie zwecks eines durchgängigen Entwicklungsprozesses sowohl für die Anforderungs- als auch Testspezifikation auf Basis *eines* zentralen Variabilitätsmodells definiert und verwaltet wird. Eine Konsistenzprüfung wird ausgearbeitet, um die Aufdeckung und Korrektur von möglichen Inkonsistenzen zwischen wiederverwendbaren Artefakten unterschiedlicher Entwicklungsphasen zu unterstützen.

Ziel 2. *Entwicklung einer werkzeug-unterstützten Methodik zur systematischen Bestimmung einer repräsentativen Menge an Konfigurationen für einen effizienten Produktlinientest.*

Um die zweite Zielsetzung zu erreichen, werden Kriterien definiert, mit deren Hilfe die zu testenden Konfigurationen für eine Produktlinie systematisch ermittelt werden. Hierbei dient die zuvor ausgearbeitete wiederverwendbare Artefaktbasis als zentrale Grundlage für die Selektionsmethodik. Die zu testenden Konfigurationen werden auf Basis einer optimierten Abdeckung aller Anforderungen der wiederverwendbaren Anforderungsspezifikation sowie aller Merkmale des Merkmalmodells bestimmt. Die Selektion wird mithilfe von zwei Optimierungsverfahren, nämlich des Greedy-Algorithmus und der Simulierten Abkühlung, realisiert.

Ziel 3. *Entwicklung einer werkzeug-unterstützten Methodik zur Identifikation der für die selektierten Konfigurationen zugehörigen Testfälle.*

Mit dem letzten Aspekt soll die Anpassung der Testartefakte an die selektierten Konfigurationen ermöglicht werden. Hierfür werden zwei mögliche Strategien der Testfallselektion untersucht und gegenübergestellt.

Die ausgearbeiteten Konzepte werden zwecks Anwendbarkeit innerhalb eines Werkzeugs prototypisch umgesetzt. Schließlich wird die Effektivität der vorgeschlagenen Methoden im Rahmen von zwei Fallstudien der Daimler AG nachgewiesen.

1.3 Struktur der Arbeit

Ein Überblick über den Aufbau der Arbeit ist in Abbildung 1 gegeben. Die dunkelgrau hervorgehobenen Kapitel weisen auf die zentralen Beiträge der Arbeit hin.

Die Arbeit beginnt mit einer Beschreibung der relevanten Grundlagen (vgl. Kapitel 2), welche die Themengebiete des Requirements Engineerings und der Qualitätssicherungsmaßnahmen innerhalb des Software-Entwicklungsprozesses von eingebetteten Systemen umfassen. Es folgt eine Einführung in die Produktlinienentwicklung und die damit verbundenen Besonderheiten, welche im Gegensatz zur Entwicklung von Einzelsystemen stehen (vgl. Kapitel 3). Ein wesentliches Thema bildet dabei die Variabilität sowie die zu deren Modellierung und Verwaltung verwendete Methoden. Mit dem folgenden Kapitel 4, welches eine ausführliche Vorstellung und Erläuterung der existierenden verwandten Arbeiten zum Produktlinientest sowie deren Potentiale liefert, wird der zweite Teil der Dissertation abgeschlossen und der Übergang zu den neuartigen ausgearbeiteten Konzepten ermöglicht.

Im Kapitel 5 wird die Methode zur einheitlichen Darstellung der Variabilität in Anforderungs- und Testartefakten beschrieben und der Lösungsansatz einer Konsistenzprüfung entwickelt. Auf Basis der dadurch wiederverwendbaren und widerspruchsfreien Artefakte wird das Konzept zur Selektion von zu testenden Konfigurationen ausgearbeitet und mithilfe von zwei Optimierungsalgorithmen umgesetzt (vgl. Kapitel 6). Zudem wird die Testfallselektion für die entsprechenden Konfigurationen behandelt. Im abschließenden Kapitel 7 des dritten Teils steht die Praxistauglichkeit der entwickelten Konzepte im Fokus, die anhand von Fallstudien demonstriert und der gewonnenen Ergebnisse diskutiert wird.

Die Arbeit wird mit einer zusammenfassenden Betrachtung sowie einem Ausblick auf weiterführende Arbeiten abgeschlossen (vgl. Kapitel 8).

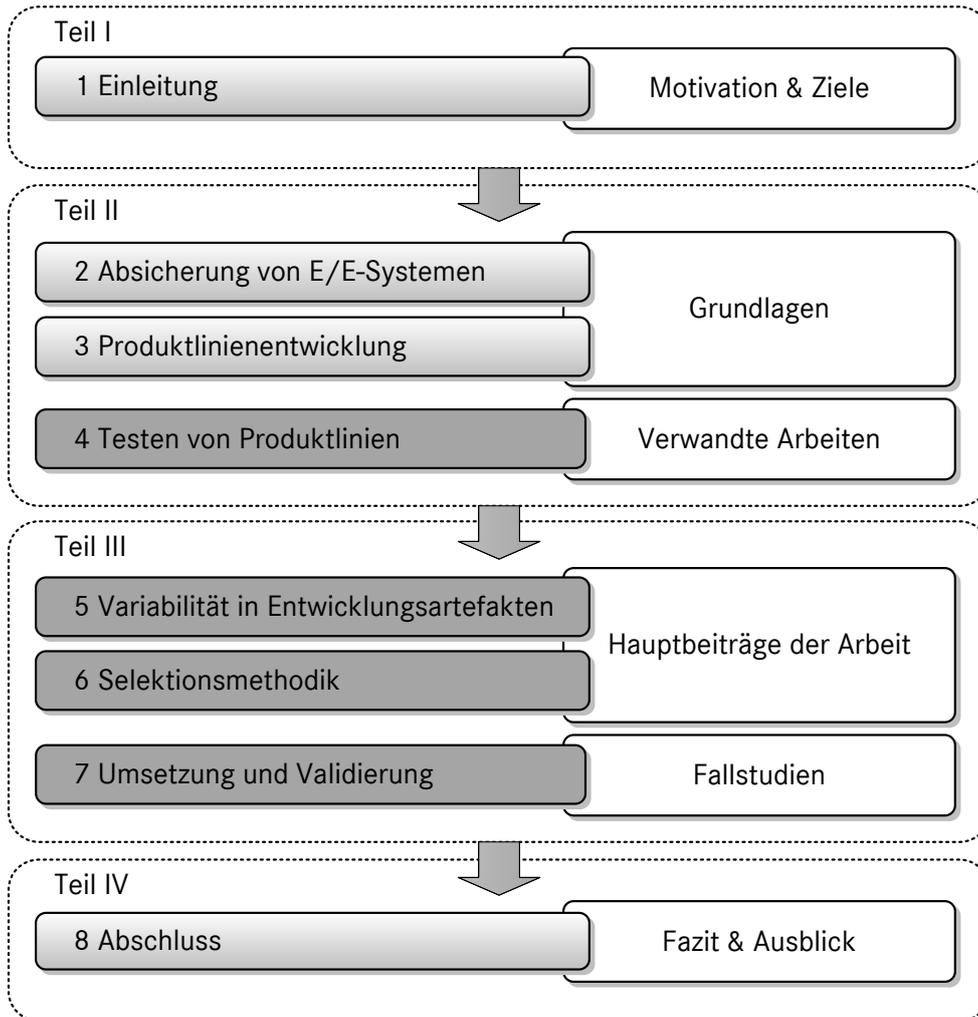


Abbildung 1: Überblick über den Aufbau der Arbeit

Teil II

GRUNDLAGEN UND VERWANDTE ARBEITEN

2 | ABSICHERUNG VON EINGEBETTETEN SYSTEMEN

Das vorliegende Kapitel beschäftigt sich mit dem Stand der Technik zur Absicherung von eingebetteten Systemen und liefert damit eine Eingliederung der vorliegenden Arbeit in das entsprechende Umfeld. Nach einer einleitenden Erläuterung des Begriffs „Eingebettetes System“ in Abschnitt 2.1, behandelt Abschnitt 2.2 die Phasen des Entwicklungsprozesses nach dem V-Modell. In Abschnitt 2.3 werden die verschiedenen Arten von Anforderungsspezifikationen, welche zu Beginn des Entwicklungsprozesses eine zentrale Rolle spielen, diskutiert. Nachfolgend wird das zur Qualitätssicherung von eingebetteten Systemen eingesetzte statische und dynamische Testen in Abschnitt 2.4 vorgestellt. Darauf aufbauend beschreibt Abschnitt 2.5 die systematische Testfallableitung, welche auf Basis von Anforderungen sowie einer Testkonzeption geschieht.

2.1 Eingebettetes System

Eine Vielzahl der Innovationen in der Automobilindustrie, aber auch in Bereichen wie der Luft- und Raumfahrt oder Medizintechnik, wird mithilfe von Software-basierten, *eingebetteten Systemen* (engl. *embedded systems*) realisiert [64]. Ein eingebettetes System besteht aus einem oder mehreren (Mikro-)Prozessoren, welche in einen technischen Kontext eingebettet sind und mit deren Hilfe vordefinierte Steuerungs-, Regelungs- oder Überwachungsaufgaben verrichtet werden [12]. Über Schnittstellen, nämlich Sensoren an den Eingängen und Aktoren an den Ausgängen, interagiert ein eingebettetes System mit der Außenwelt, wie in Abbildung 2 schematisch dargestellt.

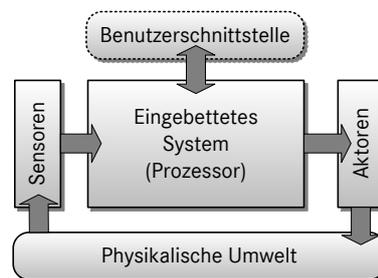


Abbildung 2: Schematische Darstellung eines eingebetteten Systems nach [12]

Mithilfe von Sensoren werden Informationen über den Zustand der physikalischen Umgebung aufgenommen, vom Prozessor verarbeitet sowie ausgewertet und anschließend über Signale an Aktoren weitergeleitet. Letztere manipulieren je nach Situation das zu steuernde Objekt (z.B. durch mechanische Eingriffe). Gegebenenfalls kann eine

Benutzerschnittstelle, etwa die Taster eines DVD-Players, für die ergänzende Bedienung des Systems durch einen Nutzer vorhanden sein. Abgesehen von einigen Funktionen, welche das System für einen Nutzer zur Verfügung stellt, bleibt dessen Funktionsweise nach außen hin in der Regel unsichtbar.

Definition 1 (Eingebettetes System nach [127]). *Unter einem eingebetteten System wird eine Einheit aus Hardware und Software verstanden, welche über Sensoren seine Umgebung erfasst und das Objekt, in welches es eingebettet ist, über Aktoren regelt, steuert oder überwacht. Häufig spricht man bei eingebetteten Systemen auch von elektrisch/elektronischen (E/E)-Systemen.*

Ein Beispiel für ein eingebettetes System im Fahrzeug stellt die Scheibenwischenanlage dar. Diese registriert über einen Regensensor die Benetzung der Scheibenoberfläche und steuert mithilfe eines Elektromotors die Scheibenwischer an. Im Automobilkontext werden Steuergeräte (engl. *electronic control unit*, ECU) als eingebettete Systeme eingesetzt. Mehrere Steuergeräte werden über Bussysteme miteinander verbunden, wodurch eine Kommunikation im Fahrzeug, d.h. ein Austausch von Informationen über relevante Daten, ermöglicht wird [8]. Im Verbund können Steuergeräte als (Teil-)Systeme unterschiedliche Funktionalitäten realisieren. So kann eine elektronisch gesteuerte Bremse über ihre Basisaufgabe, die Umsetzung der vom Fahrer ausgehenden Bremsanforderungen, ebenfalls im Kontext des Elektronisches Stabilitätsprogramms (ESP) eingesetzt werden. Aus der daraus resultierenden funktionalen Vernetzung folgt als Konsequenz eine stark erhöhte Komplexität im Fahrzeug. Diese Komplexität muss im Entwicklungs- und insbesondere Absicherungsprozess berücksichtigt werden.

2.2 Entwicklungsprozess

In der vorliegenden Arbeit steht der zur Sicherstellung der Qualität notwendige Softwaretest von eingebetteten Systemen im Fokus. Daher werden zunächst der allgemeine Entwicklungsprozess sowie die darin integrierte Testphase diskutiert.

Definition 2 (Qualität nach [119]). *Qualität gibt den Grad an, inwieweit ein Bezugsobjekt die Anforderungen erfüllt, welche in der Spezifikation definiert sind.*

Für einen organisierten Entwicklungsprozess von IT-Projekten hat sich in Deutschland das sogenannte *V-Modell* als Standard-Vorgehensmodell durchgesetzt. Dieses wurde erstmals 1979 von Boehm eingeführt [47]. 1997 wurde die überarbeitete Version, das V-Modell 97 veröffentlicht, welche wiederum erweitert und 2005 vom V-Modell XT abgelöst wurde. Zusätzlich zu den publizierten Standards existiert eine Reihe problemspezifischer Derivate des Vorgehensmodells. Ein Überblick über die Entwicklungsschritte des V-Modells, wie es in der Automobilindustrie bei der Entwicklung von eingebetteten Systemen typischerweise verwendet wird, ist in Abbildung 3 gegeben.

Grundsätzlich ist im V-Modell eine Unterteilung in die Spezifikationsphase (linker V-Ast) sowie Testphase (rechter V-Ast) klar erkennbar. Dabei steht zu Beginn eines Entwicklungsprozesses die Definition von Anforderungen, welche die Wünsche und Erwartungen an das zu realisierende Gesamtsystem (Fahrzeug) ausdrücken. In den

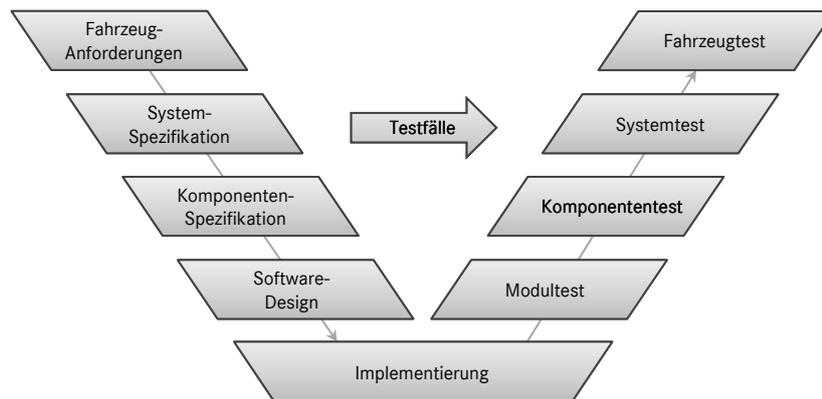


Abbildung 3: Übersicht über den sequenziellen Entwicklungsprozess nach dem V-Modell im Automobilsektor

darunterliegenden Phasen werden Anforderungen an die in dem Gesamtsystem enthaltenen Systeme, Komponenten, Software-Module usw. verfeinert und in den jeweiligen Anforderungsspezifikationen dokumentiert (vgl. Abschnitt 2.3).

Auf Basis der festgelegten Anforderungen werden Testfälle für die gegenüberstehenden Testphasen abgeleitet. Hierbei wird im rechten V-Ast in umgekehrter Reihenfolge, d.h. Bottom-Up, über die *Teststufen* vorgegangen.

Definition 3 (Teststufe nach [43]). *Eine Teststufe beschreibt eine Gruppe von Testaktivitäten, die gemeinsam ausgeführt und verwaltet werden sowie mit Zuständigkeiten in einem Projekt verknüpft sind. Siehe auch den rechten Ast des V-Modells in Abbildung 3.*

Auf der untersten Teststufe werden einzelne Module in Form von Softwarebausteinen isoliert getestet. Unter der Voraussetzung, dass die Software einer ausreichenden Absicherung unterzogen und vorhandene Defekte möglichst beseitigt wurden, werden die Module zu Komponenten zusammengesetzt. Beim Komponententest wird unter anderem die Funktionalität der Hardware und Software im Verbund getestet, weshalb auch oft vom (Hardware-/Software-)Integrationstest gesprochen wird. Sofern die Komponenten (Steuergeräte) alle Anforderungen erfüllen, werden diese zu einer Gruppe, d.h. einem System zusammengefügt und beim Systemtest integrativ auf die korrekte Systemfunktionalität sowie insbesondere die korrekte Interaktion an Schnittstellen hin untersucht [16]. Abschließend wird im Fahrzeugtest das Endprodukt (Fahrzeug) hinsichtlich der zu Beginn des Entwicklungsprozesses definierten Anforderungen getestet.

Das V-Modell spiegelt zudem die Verbindung der Zuständigkeitsbereiche zwischen einem Lieferanten und einem OEM wider. Da der Lieferant oft eine Komponente zu einem System beiträgt, ist dieser für die Testdurchführung auf unteren Teststufen, bis hin zum Komponententest, zuständig. Ein Automobilhersteller konzentriert sich verstärkt auf die korrekte Funktionalitätserbringung der Komponente im Verbund mit anderen, wodurch sein Fokus eher auf höheren Teststufen (z.B. System- und Fahrzeugtest) liegt.

2.3 Anforderungsspezifikation

Für die in der vorliegenden Arbeit vorgestellte Selektionsmethodik von testrelevanten Varianten werden unter anderem Anforderungen als Basis verwendet. Daher werden diesbezüglich die wichtigsten Begriffe sowie Arten von Anforderungsspezifikationen vorgestellt. Wie im vorangegangenen Abschnitt bereits erwähnt, steht zu Beginn eines Entwicklungsprozesses die Definition des Produktverhaltens sowie der Produkteigenschaften, welche als *Anforderungen* bezeichnet werden.

Definition 4 (Anforderung nach [119]). *Eine Anforderung ist eine dokumentierte Darstellung einer Beschaffenheit oder Fähigkeit, die*

1. *von einem Benutzer zur Lösung eines Problems oder Erreichung eines Ziels benötigt wird oder*
2. *ein System oder Systemteile erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen.*

Dabei wird das Erheben, Dokumentieren und Verwalten von Anforderungen durch das *Requirements Engineering* unterstützt. Das *Requirements Engineering* gibt den methodischen Rahmen vor, wie man von einer Projektidee bis hin zu einer Menge vollständiger und korrekter Anforderungen sowie dessen strukturierter Beschreibung innerhalb einer Anforderungsspezifikation gelangt [108]. Darüber hinaus definiert das *Requirements Engineering* Kriterien, welche sowohl von den Anforderungen, als auch der gesamten Anforderungsspezifikation, erfüllt werden müssen, um einen gewissen Qualitätsgrad zu gewährleisten [115].

Grundsätzlich wird zwischen zwei Gruppen von Anforderungen unterschieden, den *funktionalen* und den *nicht-funktionalen* [98]. Funktionale Anforderungen beschreiben das Verhalten bzw. die Funktionalität, welche das entsprechende Produkt bereitstellen soll. Dagegen drücken nicht-funktionale Anforderungen Qualitätsanforderungen sowie Randbedingungen aus. Qualitätsanforderungen kennzeichnen Vorgaben bspw. bezüglich Zuverlässigkeit, Leistung oder Zeitverhalten des entsprechenden Produktes, während Randbedingungen weitere Vorgaben in Form von Normen und Standards festlegen [98]. Nicht-funktionale Anforderungen beziehen sich in der Regel auf andere funktionale Anforderungen und ergänzen diese [108].

Allgemein können Anforderungen in *formaler* (Modelle), *semiformaler* (z.B. Satzschablonen) und *informaler* (natürliche Sprache) *Form* dokumentiert werden. Alle drei Dokumentationstechniken haben Vor- und Nachteile. So hat natürliche Sprache gegenüber der formalen Form den Vorzug, dass kein Expertenwissen notwendig ist, um diese lesen und verstehen zu können. Auf der anderen Seite ist natürliche Sprache oft nicht eindeutig und kann Interpretationsspielräume offen lassen [115]. Darüber hinaus lassen sich formale Dokumente maschinell leichter verarbeiten. In der vorliegenden Arbeit werden Spezifikationsdokumente (Anforderungs- und Testspezifikationen) betrachtet, welche in natürlicher Sprache verfasst sind. Im Kontext der Anforderungsspezifikationen spricht man auch häufig von *Lastenheften*¹. Bei der Daimler AG differenziert man dabei zwischen Fahrzeug-, System- und Komponentenlastenheften.

¹ Demgegenüber stehen Pflichtenhefte, die auf Basis der Lastenhefte erstellt werden und vom Zulieferer zu erfüllende Anforderungen enthalten.

Fahrzeuglastenheft

Im *Fahrzeuglastenheft* werden die allgemeinen vom Fahrzeug zu erbringenden Leistungsmerkmale festgelegt und dokumentiert. Dies umfasst die zu erreichenden Zielgrößen aus allen Funktionalbereichen, nämlich Entwicklung inkl. Design, Marketing bzw. Vertrieb, Einkauf, Produktion, Betriebswirtschaft, Qualität und After Sales [115].

Systemlastenheft

Das *Systemlastenheft* stellt das Bindeglied zwischen einem Fahrzeug- und den Komponentenlastenheften dar [10]. Es beschreibt überwiegend Anforderungen an die (kundererlebbaren) Funktionen, welche in dem zu realisierenden Fahrzeug vom System bereitgestellt werden sowie das Zusammenspiel einzelner Komponenten an den Schnittstellen. Für eine einheitliche Dokumentation solcher Anforderungen wird bei der Daimler AG die Methodik der sogenannten Funktionsorientierung verwendet [94]. Zum einen werden dadurch die Anforderungen jeder Fahrzeugfunktion innerhalb einer hierarchisch angeordneten Funktionsstruktur dokumentiert. Die Funktionsstruktur enthält vordefinierte Elemente (z.B. Voraussetzung, Auslöseereignis, Endbedingung) und erhöht auf diese Weise den Grad der Einheitlichkeit während der Anforderungsdokumentation. Zum anderen wird durch das Komponentenmapping zugeordnet, welche Funktionen durch die Beiträge welcher Komponenten realisiert werden. Diese Abhängigkeit bezeichnet man als Funktionsbeitrag. Die Funktionsorientierung trägt erheblich zur eindeutigen und vollständigen Erstellung von Systemanforderungen bei und unterstützt dadurch das nachfolgende Spezifizieren von Testfällen, welche eine gleichartige Struktur (mit den Elementen: Eingabe, Verarbeitung, Ausgabe) aufweisen.

Komponentenlastenheft

Das *Komponentenlastenheft* stellt eine umfangreiche Beschreibung aller an eine Komponente gestellten Anforderungen dar. Dies umfasst Anforderungen an die Hardware, die technischen Schnittstellen, die physikalischen Rahmenbedingungen sowie die eingesetzten Bauteile [79]. Zentral ist hierbei die Festlegung, welche Funktionsbeiträge in welchen Systemen von den jeweiligen Komponenten bereitgestellt werden. Solche Anforderungen sind in der Regel bereits im Systemlastenheft definiert und werden in Komponentenlastenheften weiter verfeinert. Zudem fungieren Komponentenlastenhefte häufig als Grundlage für den Anfrage- und Vergabeprozess für Lieferanten. Eine Unterkategorie des Komponentenlastenheftes stellt das Software-Lastenheft dar, welches explizite Anforderungen an die Software(-Module) enthält [115].

DOORS

Zur Dokumentation und Verwaltung von Anforderungen wird bei der Daimler AG das Werkzeug *DOORS* (engl. *Dynamic Object Oriented Requirements System*) von IBM

(Telelogic) eingesetzt [46]. Dokumente in DOORS-Datenbanken werden in Form von sogenannten Modulen gespeichert. Die Verwaltung der in jedem Modul enthaltenen Informationen wird mithilfe von Attributen und Objekten unterstützt. Objekte (Zeilen) sind dabei in einer hierarchischen Struktur organisiert. Durch Attribute (Spalten) werden Objekte mit zusätzlichen Informationen, wie Status, Typ oder Priorität angereichert. Die Vergabe von Identifikationsnummern (IDs) ermöglicht eine eindeutige Kennzeichnung von Objekten. Bei Änderungen kann über die sogenannte History auf verschiedene Stände/Versionen (engl. *baselines*) der Module zurückgegriffen werden. Mithilfe von Links werden zusammengehörige Informationen innerhalb eines Moduls, aber auch über verschiedene Dokumente hinweg miteinander verknüpft. Die durch Verlinkungen ermöglichte Rückverfolgbarkeit (engl. *traceability*) ist insbesondere zwischen Anforderungen sowie den daraus abgeleiteten Testfällen, welche ebenfalls in DOORS dokumentiert werden, wichtig. Zum einen kann dadurch gewährleistet werden, dass es zu jeder Anforderung einen Testfall gibt, zum anderen können Änderungsauswirkungen leichter verfolgt werden.

2.4 Softwaretest

Im Folgenden werden die zur Qualitätssicherung von eingebetteten Systemen verwendeten, gängigen Methoden des Softwaretests diskutiert. Klassischerweise differenziert man dabei zwischen dem *statischen* und *dynamischen Test*. Der wesentliche Unterschied besteht darin, dass bei dynamischem Testen das Prüfobjekt mit entsprechenden Testfällen ausgeführt wird, bei statischem hingegen nicht.

2.4.1 Statischer Test

Bei *statischem Test* steht die Prävention von (Spezifikations-)Fehlern und Abweichungen im Vordergrund. Dabei werden die für die Entwicklung des entsprechenden Bezugsobjektes relevanten Dokumente und Arbeitsergebnisse einer eingehenden Prüfung unterzogen. Auf diese Weise werden Ursachen von Fehlern so früh wie möglich aufgedeckt und damit das wesentlich kostenintensivere Ausbessern von Fehlerwirkungen in späteren Entwicklungsphasen vermieden. Zum statischen Test zählen *analysierende Prüftechniken* (z.B. Review, Inspektion, Walkthrough und statische (Code-)Analyse) sowie *verifizierende Prüftechniken* (z.B. Model-Checking) [63, 111].

Definition 5 (Statischer Test nach [43, 119]). *Statisches Testen beschreibt den Prozess zur Bewertung eines Prüfobjektes auf Basis seiner Form, Struktur, Inhalt oder Dokumentation und ohne dessen Ausführung.*

Im Rahmen von Reviews und Walkthroughs werden (Entwicklungs-)Dokumente, bspw. Anforderungsspezifikationen (vgl. Abschnitt 2.3), Verträge, Testkonzeptionen (vgl. Abschnitt 2.5.1) usw., von einer oder mehreren Personen manuell geprüft [63]. Die Prüfung kann im Hinblick auf die Einhaltung von Template- oder Prozessvorgaben, aber auch

hinsichtlich der Semantik eines Dokumentes erfolgen. Im Falle von Inspektionen werden analoge Zielsetzungen verfolgt. Jedoch stellen sie die formalste Reviewart dar, da Inspektionen eine streng definierte Vorgehensweise zugrunde liegt.

Im Gegensatz dazu lässt sich die statische Analyse in der Regel leichter automatisieren. Ein Dokument, welches eine formale Struktur aufweist, wird dabei auf Verletzungen der Syntax, Abweichungen von Konventionen und (Programmier-)Standards, Kontrollfluss- sowie Datenflussanomalien hin analysiert [16]. Typische Fehler, welche von statischer Analyse aufgedeckt werden, sind Endlosschleifen, ein undefinierter Wert einer Variable oder nicht erreichbarer Code. Aus Kostengründen ist es sinnvoller, statische Analyse *vor* manuellen Reviews, Walkthroughs oder Inspektionen durchzuführen [111].

Schließlich werden im Rahmen von verifizierenden Prüftechniken formale Beweisverfahren eingesetzt. So wird beim Model-Checking geprüft, ob eine gegebene Systembeschreibung die Eigenschaften einer entsprechenden Spezifikation erfüllt. Hierfür müssen eine hinreichend formale Systembeschreibung und Spezifikation vorliegen.

2.4.2 Dynamischer Test

Unter *dynamischem Test* werden typischerweise *funktionsorientierte, strukturorientierte und diversifizierende Prüftechniken* zusammengefasst [63]. Hierbei erfolgt eine Ausführung von Testfällen für das Prüfobjekt sowie die Untersuchung der erzielten Ergebnisse bezüglich deren Übereinstimmung mit den erwarteten Werten. In der Regel können nicht alle möglichen Kombinationen von Eingabedaten (erschöpfend) getestet werden, sodass aufgrund des Stichprobencharakters die Abwesenheit von Fehlern nicht garantiert werden kann. Daher ist die Bestimmung eines geeigneten Testumfangs mithilfe von entsprechenden Prüftechniken für den Erfolg des dynamischen Tests entscheidend.

Definition 6 (Dynamischer Test nach [43, 119]). *Dynamisches Testen beschreibt den Prozess zur Bewertung des Verhaltens eines Prüfobjektes während seiner Nutzung bzw. Ausführung.*

Bei der dynamischen Qualitätssicherung ist eine weitere Unterteilung in *White-Box-* und *Black-Box-Techniken* üblich [62]. Beim White-Box-Test wird die interne Struktur, d.h. der Sourcecode, zur Ableitung von Testfällen vorausgesetzt. Daher wird dieser häufig auf unteren Teststufen (z.B. Modultest) angewendet [56]. Zum White-Box-Test zählen alle *strukturorientierten Prüftechniken*. Diese messen die Abdeckung der Strukturelemente (z.B. Zweige oder Anweisungen) im Programmcode zur Bewertung der Testvollständigkeit [63].

Definition 7 ((Test-)Abdeckungskriterium nach [79]). *Ein Abdeckungskriterium gibt ein Element (z.B. Zweig oder Anforderung) an, für das ein prozentualer Grad an Abdeckung durch die Testdurchführung gemessen werden kann.*

Beim Black-Box-Test hingegen bleibt die innere Funktionsweise verborgen und der Fokus liegt dadurch auf dem, nach außen sichtbaren Verhalten des Testobjektes. Aus diesem Grund wird der Black-Box-Test tendenziell auf höheren Teststufen (z.B. Systemtest) durchgeführt [56]. Zum Black-Box-Test zählt der *funktionsorientierte Test*. Das Ziel ist es,

auf Basis der in den zugehörigen Anforderungen spezifizierten Soll-Funktionalität, das Ein- und Ausgabeverhalten des Prüfobjektes zu testen. Ein Testfallermittlungsverfahren des funktionsorientierten Tests ist die Äquivalenzklassenanalyse. Dabei werden die Eingabedaten in Klassen, bei welchen das gleiche Testergebnis erwartet wird, unterteilt [16]. Ergänzend dazu wird häufig die Grenzwertanalyse eingesetzt, mit deren Hilfe Testfälle zur Untersuchung der Grenzwertbereiche von Eingabedaten bestimmt werden. Als Abdeckungskriterium für den funktionsorientierten Test gilt die Berücksichtigung jeder Anforderung (Anforderungsabdeckung) in der zugehörigen Spezifikation (vgl. Abschnitt 2.3). Der funktionsorientierte Test wird daher auch oft als anforderungs- bzw. spezifikationsbasierter Test bezeichnet.

Definition 8 (Testfallermittlungsverfahren nach [79]). *Testfallermittlungsverfahren beschreiben Analyseverfahren, die zur Ableitung eines Testfalls verwendet werden können (z.B. Äquivalenzklassenanalyse oder Anforderungsanalyse).*

Ergänzend dazu ist der *nicht-funktionale Test* zu berücksichtigen. Dieser bildet zwar keine eigenständige Kategorie der dynamischen Prüftechniken nach [63], ist jedoch als eine wichtige Ergänzung des funktionsorientierten Tests zu betrachten. Mithilfe der *nicht-funktionalen Testtechniken* wird geprüft, wie gut eine Funktionalität erbracht wird. Dies umfasst das Testen von Qualitätsmerkmalen wie Zuverlässigkeit, Benutzbarkeit und Effizienz [111]. So wird im Rahmen des Nachweises der Robustheit das Testobjekt extremen Umgebungsbedingungen (z.B. sehr hoher/niedriger Temperatur) ausgesetzt. Die entsprechenden Testfälle werden aus den nicht-funktionalen Anforderungen abgeleitet.

Schließlich werden im Rahmen der *diversifizierenden Prüftechniken* verschiedene Versionen eines Prüfobjektes gegeneinander getestet [111]. Ziel ist es, gleiche Testergebnisse für verschiedene Versionen zu erhalten. Der bekannteste Vertreter der diversifizierenden Prüfmethoden ist der Regressionstest, welcher in Kapitel 4.2.3 aufgegriffen wird.

Keine Prüftechnik ist so universell, dass sie alle Aspekte des Testens abdeckt. Aus diesem Grund wird für eine aussagekräftige Absicherung eines Prüfobjektes, grundsätzlich eine Kombination verschiedener Techniken genutzt. Der Schwerpunkt der vorliegenden Arbeit liegt auf dynamischen Tests. Die in diesem Zusammenhang notwendige methodische Bestimmung von Testfällen wird im folgenden Abschnitt 2.5 ausführlich behandelt.

2.5 Methodische Testfallableitung

Wie im vorhergehenden Abschnitt 2.4.2 thematisiert, können aufgrund des Stichprobencharakters von dynamischen Tests, Restfehler nicht ausgeschlossen werden. Daraus folgt, dass die Bestimmung von geeigneten Stichproben, d.h. Testfällen, eine wesentliche Rolle für die Effektivität des Testens spielt. Dazu sollten Testfälle einerseits repräsentativ und fehlersensitiv, gleichzeitig jedoch redundanzarm sowie ökonomisch sein [63]. Dieser Abschnitt beschreibt detailliert, wie bei der Daimler AG, auf Basis von Anforderungen sowie einer Testkonzeption, solche Testfälle systematisch abgeleitet

werden. Abbildung 4 zeigt einen schematischen Überblick über den Testprozess mit den genannten Phasen².

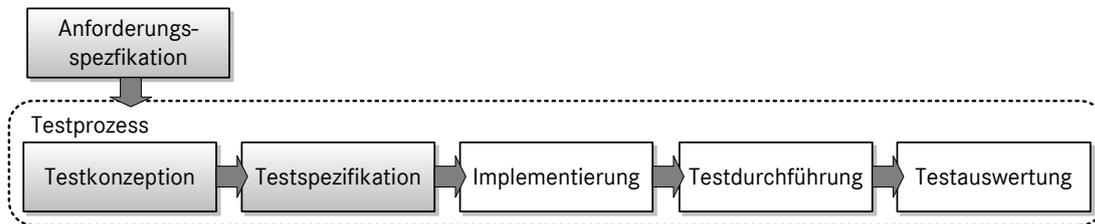


Abbildung 4: Übersicht über die aufeinanderfolgenden Phasen des Testprozesses sowie die vorhergehende Spezifikationsphase von Anforderungen nach [107]

2.5.1 Testkonzeption

Die *Testkonzeption* stellt die vorbereitende Phase vor der eigentlichen Testfallableitung dar [112]. Die wesentliche Aufgabe besteht darin, das konkrete Testvorgehen für das abzusichernde Objekt mit den Testaktivitäten, Testschwerpunkten, anzuwendenden Testfallermittlungsverfahren usw. festzulegen. Mithilfe der daraus resultierenden Teststrategie sowie den darin enthaltenen quantifizierbaren Testgrößen (z.B. den Testzielen) kann eine verlässlichere Aussage bezüglich der Qualität des entsprechenden E/E-Systems gemacht werden.

Darüber hinaus adressiert die Testkonzeption die von der Norm ISO 26262 gestellten Anforderungen an ein Entwicklungsvorgehen für sicherheitsrelevante E/E-Systeme in Kraftfahrzeugen bereits vor der Testfallermittlung [48]. Hierbei werden auf Basis einer Gefahren- und Risikoanalyse (engl. *hazard analysis and risk assessment*) Situationen untersucht, in welchen Fehlfunktionen des E/E-Systems zu potentiellen Gefährdungen führen können. Jede Gefährdung kann innerhalb der entsprechenden Anforderung mit einer sogenannten ASIL-Einstufung (engl. *automotive safety integrity level*) zwischen A und D klassifiziert werden. ASIL D kennzeichnet die sicherheitskritischste Einstufung (z.B. bei Anforderungen an ein Airbag). Nicht sicherheitsrelevante Anforderungen werden von der ISO 26262 mit der QM-Einstufung (engl. *quality management*) deklariert.

Abbildung 5 zeigt die aufeinanderfolgenden Prozessschritte der Testkonzeption. Zu berücksichtigen ist, dass die hier beschriebene Testkonzeption eine spezifische Interpretation der korrespondierenden Anforderungen der ISO 26262 an die Prozesse der Daimler AG darstellt.

Zu Beginn gilt es, den Umfang des *Testobjektes* klar zu definieren und abzugrenzen. Dazu müssen sämtliche Bestandteile des abzusichernden eingebetteten Systems festgelegt werden, da dieses in der Regel in weitere abzusichernde Testobjekte (Komponenten, Funktionen, Software-Module usw.) zerlegt werden kann. In diesem Zusammenhang werden ebenfalls organisatorische Aspekte wie die Definition von Verantwortlichen für

² Die Phasen des Implementierens von Testfällen, der Testdurchführung sowie der Auswertung sind für diese Arbeit nicht relevant und werden daher nicht behandelt.



Abbildung 5: Übersicht über die Prozessschritte der Testkonzeption nach [81]

die jeweiligen Testobjekte sowie die Schnittstellen zwischen Test- und Entwicklungspartnern, geklärt.

Im nächsten Schritt werden die zu erreichenden *Testziele* (z.B. der Nachweis der korrekten Funktionalitätserbringung oder der Nachweis der Benutzbarkeit) bestimmt. Bei der Daimler AG sind insgesamt acht standardisierte Testziele vorgegeben, welche jedoch je nach Projekt angepasst, d.h. verfeinert, erweitert oder reduziert, werden können.

Der darauf folgende Schritt legt fest, auf welchen *Teststufen* die zuvor definierten Testziele zu erreichen sind. Diesbezüglich ist zu beachten, dass der Aufwand sowie die Kosten für eine Fehlerbehebung wachsen je höher die Teststufe ist, auf der der Fehler gefunden wurde. Aus diesem Grund ist es wichtig eine effiziente Verteilung des Testaufwandes auf die Teststufen zu gewährleisten.

Im Rahmen der nachfolgenden Phase der *Teststrategie* wird abgeleitet, welche konkreten Testfallermittlungsverfahren (z.B. die, in Abschnitt 2.4.2 beschriebenen, Äquivalenzklassenanalyse oder Grenzwertanalyse) für die Erreichung welcher Testziele anzuwenden sind. Ferner werden Abdeckungskriterien festgelegt, anhand derer der Nachweis erbracht wird, dass das jeweilige Testziel erfüllt wurde. Dazu zählen unter anderem die Schnittstellenabdeckung oder die Variantenabdeckung. Letztere ist für die vorliegende Arbeit von besonderer Relevanz und wird in Kapitel 6 näher betrachtet. Dabei wird von E/E-Systemen mit einer sicherheitskritischeren Einstufung der Nachweis strengerer/stärkerer Abdeckungskriterien gefordert [48].

Die Erstellung der Testkonzeption erfolgt auf Basis einer Vorlage, sodass schließlich ein Planungsdokument, das Testkonzept, vorliegt. Dieses wird abschließend unter Teilnahme aller Beteiligten einem *Review* unterzogen sowie bei Akzeptanz *freigegeben*.

2.5.2 Testspezifikation

Liegt die, im vorigen Abschnitt behandelte Teststrategie vor, werden *Testfälle* systematisch abgeleitet und innerhalb einer *Testspezifikation* dokumentiert. In Abbildung 6 ist ein Überblick über die sequentiellen Schritte einer Testspezifikationserstellung gegeben.

Anfangs werden während der *Testbasisanalyse* alle für die Testfallermittlung relevanten Dokumente (die Testbasis) untersucht. In erster Linie zählen dazu Anforderungen, welche die im Testkonzept definierten Testobjekte beschreiben und in entsprechenden Komponenten- und/oder Systemlastenheften dokumentiert sind³. Das wesentliche Ziel

³ Darüber hinaus können mitgeltende Unterlagen (MGU) oder das Testkonzept im Rahmen einer Testbasisanalyse geprüft werden.

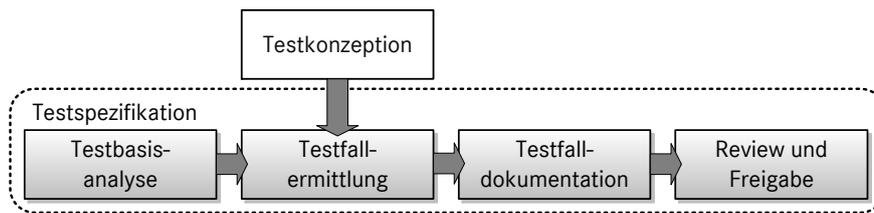


Abbildung 6: Übersicht über die Prozessschritte der Testspezifikation nach [79]

der Testbasisanalyse ist es, die Anforderungen hinsichtlich des Testbarkeitskriteriums zu bewerten, d.h der Tatsache, ob sie durch dynamische Tests oder andere Maßnahmen abgesichert werden müssen. Dies ist ein wichtiger Schritt, da nach der Testfallermittlung untersucht wird, ob zu jeder Anforderung mindestens ein Link zu einem Testfall existiert und umgekehrt, was auch als Nachweis der Anforderungsabdeckung bezeichnet wird. Geht man dabei von ausschließlich testbaren Anforderungen aus, kann eher eine repräsentative Aussage bezüglich der Anforderungsabdeckung gemacht werden. Ist eine Anforderung testbar, wird weiterhin ermittelt, ob diese

- verständlich (hinsichtlich des Inhalts eindeutig),
- widerspruchsfrei (hinsichtlich des Inhalts zu anderen bereits vorliegenden Anforderungen sowie zu sich selbst widerspruchsfrei),
- atomar (nur einen Aspekt beschreibend)
- und messbar (hinsichtlich der Umsetzung bewertbar) ist.

Nachdem alle durch die Testbasisanalyse gefundenen Unstimmigkeiten behoben wurden, werden die Testfälle abgeleitet. Neben den Anforderungsdokumenten bildet dabei die Testkonzeption die zentrale Grundlage für die *Testfallermittlung*. Gemäß der darin festgelegten Strategie werden die Testtechniken (vgl. Abschnitt 2.4.2) umgesetzt. Abschließend wird die Vollständigkeit der Testfälle und die Erreichung der Testziele über die (ebenfalls im Testkonzept definierten) Abdeckungskriterien nachgewiesen.

Definition 9 (Testfall nach [43, 119]). *Ein Testfall wird definiert über eine Menge von Vorbedingungen, Eingabewerten, den erwarteten Ergebnissen sowie Nachbedingungen, welche im Hinblick auf ein Ziel (z.B. Pfadabdeckung im Code) entwickelt werden.*

Der folgende Schritt der *Testfalldokumentation* bezieht sich auf die Erfassung der ermittelten Testfälle innerhalb einer *Testspezifikation*. Zur Veranschaulichung ist in Tabelle 1 ein beispielhafter Testfall in natürlicher Sprache angelehnt an die Struktur in DOORS dargestellt.

Jeder Testfall wird mit einem vordefinierten Ausgangszustand *initialisiert*. Der Übersichtlichkeit halber und um die Dokumentation von redundanten Informationen zu vermeiden, wird die Initialisierung häufig in Form eines Basisszenarios (engl. *base scenario*), zusammengefasst. Nach der Initialisierung folgen weitere Testschritte (engl. *test steps*), welche jeweils aus einer Menge von Eingabewerten sowie der zu erwarteten Reaktion bestehen. Durch Setzen eines bzw. mehrerer Links wird gekennzeichnet,

ID	Ebene	Objekttyp	Beschreibung	Aktion	Nachbedingung
TS-0	1	Basisszenario	(1) Motor an		
TS-1	1	Testfall	Sitzbereich vorne links heizen		
TS-2	2	Testschritt	Initialisierung	Basisszenario (1)	Basisszenario (1) erreicht
TS-3	2	Testschritt	Sitzheizung ein	Knopf SH-v-l drücken	Sitzheizung vorne links ist eingeschaltet
TS-4	2	Testschritt	Sitzheizung aus	Knopf SH-v-l drücken	Sitzheizung vorne links ist ausgeschaltet

Tabelle 1: Darstellung der Dokumentationsbestandteile eines beispielhaften Testfalls

welche Anforderung(en) durch den jeweiligen Testfall abgedeckt ist/sind. Da die Testfälle möglichst abstrakt beschrieben werden, lassen sie sich auf mehreren Teststufen wiederverwenden. Die empfohlenen Teststufen werden dabei pro Testfall im entsprechenden Attribut angegeben. Analog dazu werden die Testziele, die der jeweilige Testfall adressiert, über ein Attribut zugeordnet. Um die Umsetzung der abstrakten Testfälle auf der jeweiligen Testumgebung zu ermöglichen, werden diese implementiert (vgl. Abbildung 4).

Abschließend wird die Testspezifikation einem oder mehreren *Reviewprozess(en)* unterzogen, unter Umständen überarbeitet bis sich der gewünschte Qualitätsgrad einstellt und schließlich *freigegeben*.

Zusammenfassung

Dieses Kapitel beschäftigt sich mit der Definition des Umfelds sowie des Einsatzgebietes für die in der vorliegenden Arbeit vorgestellte Methodik. Zu diesem Zweck wird ein Überblick über die gängigen Prüftechniken für eingebettete Systeme, die eine Vielzahl teils sicherheitsrelevanter Aufgaben in Automobilen übernehmen, gegeben. Dabei ist der dynamische Test, welcher in der Praxis von zentraler Bedeutung ist, als eine besonders wichtige Qualitätssicherungsmaßnahme identifiziert worden. Dessen Ziel ist es zu prüfen, ob die in der Spezifikation festgelegten Anforderungen erfüllt werden und somit das Vertrauen in das Testobjekt zu erhöhen.

Der Erfolg des dynamischen Testens hängt stark von der zugrunde liegenden Methodik ab. Um die Ableitung der Testumfänge, in Form von Testfällen, nach einem systematischen Vorgehen zu ermöglichen, ist zum einen eine einheitliche Anforderungsdokumentation in entsprechenden Lastenheften sowie zum anderen eine Teststrategie notwendig. Darüber hinaus trägt die strukturierte Dokumentation innerhalb einer Testspezifikation zur Korrektheit und Vollständigkeit der Testfälle bei. Die hier diskutierte methodische Testfallableitung und -dokumentation bildet die Grundlage für die, in Kapitel 6, ausgearbeitete Testmethodik von hoch variablen E/E-Systemen, auch bezeichnet als Produktlinien.

3 | PRODUKTLINIENENTWICKLUNG

Dieses Kapitel stellt die Grundlagen und Besonderheiten der Entwicklung von Produktlinien vor. Abschnitt 3.1 führt das Kapitel mit einer Beschreibung der Grundsätze sowie der Vorteile einer Produktlinienentwicklung ein. Ausgehend davon werden die beiden Aktivitäten der Produktlinienentwicklung, nämlich die Domänen- sowie die Applikationsentwicklung in Abschnitt 3.2 beschrieben. Dabei werden die im Kontext verwendeten Begrifflichkeiten definiert, um eine einheitliche Terminologiebasis für die nachfolgenden Kapitel zu schaffen. Anschließend wird in Abschnitt 3.3 untersucht, welche geeigneten Möglichkeiten zur Erfassung, Dokumentation und Verwaltung der Variabilität in Produktlinien existieren.

3.1 Einführung in die Produktlinienentwicklung

Der Trend zur Individualisierung von Software-intensiven Systemen lässt sich in der Automobilindustrie besonders gut beobachten. So hat jeder Kunde beim Kauf eines Fahrzeugs die Möglichkeit dieses gemäß seiner Präferenzen bezüglich Größe, Motorisierung, Leistung usw. individuell zu bestimmen. Bei der aktuellen A-Klasse, W 176, von Mercedes-Benz sind über 10^{19} Möglichkeiten gegeben, die Ausstattungsmerkmale¹ des Fahrzeugs zu konfigurieren. In diesem Zusammenhang ist die A-Klasse als eine Produktlinie (engl. *product line*, *product family*) und das konfigurierte Fahrzeug als ein Produkt dieser Produktlinie zu verstehen.

Definition 10 (Produktlinie nach [23]). *Als Produktlinie wird eine Menge von ähnlichen Produkten bezeichnet, welche sowohl gemeinsame, in jedem Produkt vorkommende Eigenschaften, als auch individuelle Ausprägungen besitzen.*

Definition 11 (Produkt nach [90]). *Ein Produkt p_i ist eine Ware oder eine Dienstleistung, welche die konkreten Bedürfnisse eines Kunden adressiert.*

Um die mit der Entwicklung von hoch variablen Systemen einhergehenden Herausforderungen bewältigen zu können, wird der Ansatz der *Produktlinienentwicklung* (engl. *product line engineering*, PLE) herangezogen. Dieser ist deswegen erfolgreich, da die einzelnen Produkte unter Ausnutzung von gemeinsamen Eigenschaften entstehen. Durch die daraus resultierende hohe Quote an Wiederverwendung ist es möglich, entscheidende Vorteile gegenüber der konventionellen Einzelsystementwicklung zu erzielen.

¹ Dies umfasst die Merkmale bezüglich „Motor“, „Ausstattungsline“, „Pakete“, „Farben & Räder“, „Polster & Zierelemente“ und „Ausstattung“ [35].

Diese Vorteile spiegeln sich in einer Reihe von wirtschaftlichen, technischen und organisatorischen Effizienzen wider [29, 54]. So können mithilfe der Produktlinienentwicklung erhebliche Einsparungen der Entwicklungskosten und der Entwicklungszeit erreicht werden [90, 121], dargestellt in Abbildung 7. Diesbezüglich lassen sich sowohl die Kosten als auch die Zeit um den Faktor zwei bis vier verringern bzw. verkürzen [104]. Um die Gewinnschwelle zu erreichen, muss ein initialer Aufwand bei der Bereitstellung von gemeinsamen Entwicklungselementen aufgebracht werden. Empirische Studien haben ergeben, dass sich für Software-Produktlinien die Investitionen bereits nach drei Produkten amortisiert haben [23]. Des Weiteren lässt sich der Mehrwert einer Produktlinienentwicklung anhand von Faktoren, wie höhere Produktqualität, halbierte Fehlerrate sowie Reduzierung des Wartungsaufwandes belegen [19].

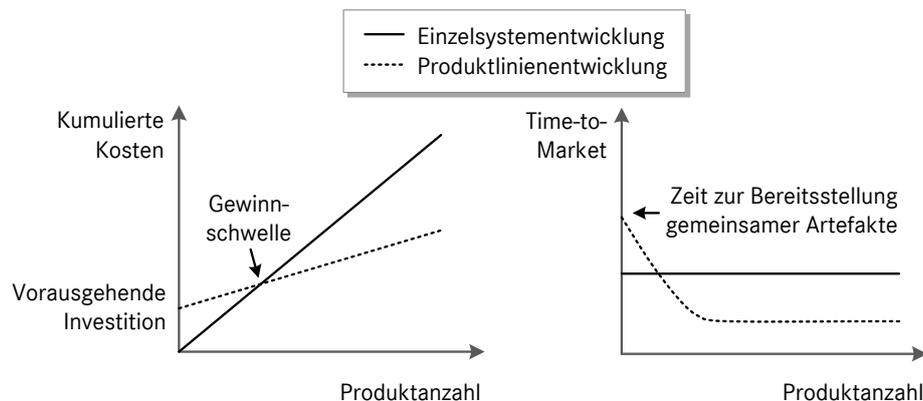


Abbildung 7: Qualitative Gegenüberstellung von Entwicklungskosten (links) sowie Time-to-Market (rechts) mit und ohne Produktlinienentwicklung nach [4]

3.2 Domänen- und Applikationsentwicklung

Produktlinienentwicklung ist ein methodischer Ansatz, welcher eine organisierte Wiederverwendung von gemeinsamen Artefakten im Entwicklungsprozess von Produktlinien umsetzt [4].

Definition 12 ((Entwicklungs-)Artefakt nach [90]). *Artefakt ist ein kontextabhängiger Begriff, welcher die während des Entwicklungsprozesses bereitgestellten Elemente, wie Anforderungen, Architekturelemente, Komponenten, Codefragmente, Testfälle u. Ä. zusammenfasst.*

Die Methodik der Produktlinienentwicklung hat sich insbesondere in der Softwareentwicklung etabliert [53]. Zwei wesentliche Eigenschaften bzw. Grundprinzipien unterscheiden die Produktlinienentwicklung von der Einzelsystementwicklung, nach [4]:

1. Analyse und Modellierung der Variabilität entsprechender Produktlinie (Variantenmanagement),

2. Trennung zwischen den Aktivitäten der Domänen- und der Applikationsentwicklung.

Der erste Aspekt betrifft die Notwendigkeit einer systematischen Identifikation und Erfassung der Gemeinsamkeiten sowie Unterschiede. Dies ist essentiell, um ein einheitliches Verständnis im Bezug auf die Produktlinie zu ermöglichen und gleichzeitig die Verfügbarkeit der entsprechenden Informationen für die Phasen des Entwicklungsprozesses sicherzustellen. Relevante Techniken zur Variabilitätsmodellierung werden in Abschnitt 3.3 ausführlich diskutiert.

Mithilfe des Variantenmanagements kann das zweite Kriterium adressiert werden. Hierfür werden während der sogenannten *Domänenentwicklung* (engl. *domain engineering, core asset development*) Artefakte generiert, welche für die gesamte Produktlinie gelten. Hierbei wird auch von produktübergreifenden bzw. generischen Artefakten gesprochen. Die Ansammlung dieser wiederverwendbaren Artefakte, welche die Entwicklungsbasis darstellt, wird als Plattform (engl. *platform, core asset*) oder auch (Produktlinien-) Infrastruktur bezeichnet. Die in dieser Arbeit verwendete Produktlinieninfrastruktur wird in Kapitel 5 schrittweise ausgearbeitet.

Definition 13 (Domänenentwicklung nach [90]). *Domänenentwicklung ist der Prozess der Produktlinienentwicklung, in welchem sowohl die gemeinsamen als auch die variablen Anteile der Produktlinie für die Wiederverwendung ausgearbeitet werden.*

Definition 14 (Plattform, (Produktlinien-)Infrastruktur). *Eine Plattform ist eine Akkumulation von wiederverwendbaren Entwicklungsartefakten für eine Produktlinie.*

In der nachfolgenden *Applikationsentwicklung* (engl. *application engineering, product development*) werden die aus der Domänenentwicklung bereitgestellten Artefakte für die Entwicklung der einzelnen Produkte wiederverwendet.

Definition 15 (Applikationsentwicklung nach [90]). *Applikationsentwicklung ist der Prozess der Produktlinienentwicklung, in welchem die Produkte durch Wiederverwendung der Domänenartefakte und Ausnutzung von Variabilität erzeugt werden.*

Abbildung 8 zeigt den Referenzprozess der Produktlinienentwicklung. Die Domänenentwicklung ist dabei in fünf Subprozesse unterteilt [90]:

- Das **Produktmanagement** legt die Marktstrategie fest. Mithilfe von sogenannten *Scoping*-Techniken wird unter Berücksichtigung der ökonomischen und technischen Randbedingungen das Produktportfolio mit gemeinsamen und variablen Eigenschaften der zukünftigen Produktlinie beschlossen.
- Das **Domänen-Requirements-Engineering** fasst die Aktivitäten zur Identifikation und Dokumentation von gemeinsamen und variablen Anforderungen der Produktlinie zusammen. Zusätzlich ist die Erstellung des Variabilitätsmodells Teil des Subprozesses.
- Im **Domänendesign** wird die Architektur der Produktlinie entworfen.

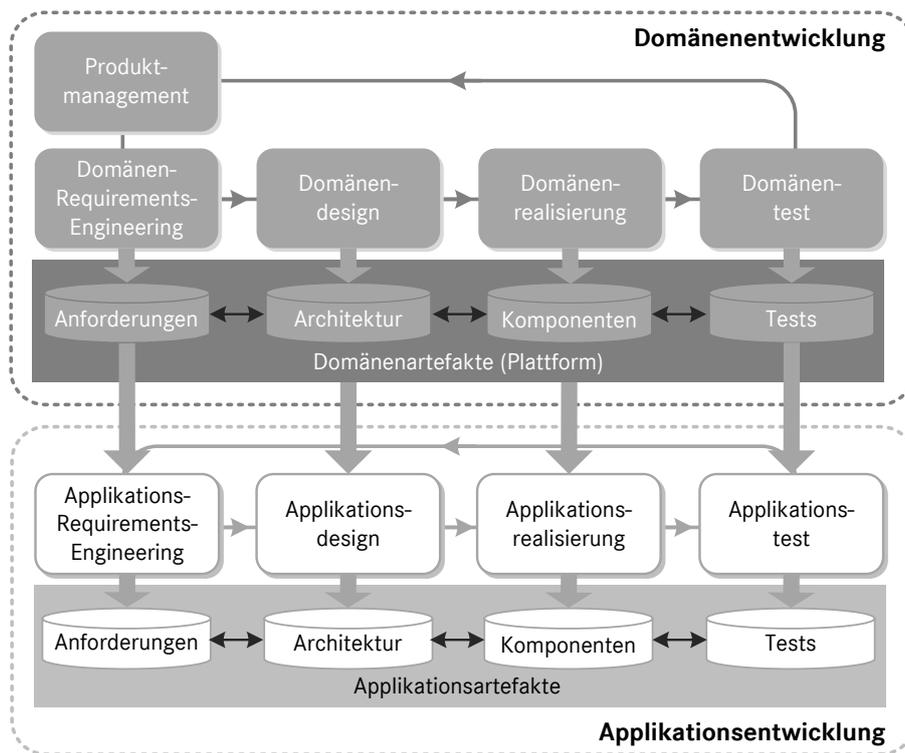


Abbildung 8: Übersicht über den Referenzprozess für die Produktlinienentwicklung von Software [90]

- Die **Domänenrealisierung** beschreibt die wiederverwendbaren Komponenten der Produktlinie.
- Im **Domänentest** werden wiederverwendbare Testartefakte spezifiziert, welche das Testen der Komponenten gegen deren Spezifikation ermöglichen.

Während der Applikationsentwicklung werden die Produkte entwickelt. Hierfür werden die Produkte konfiguriert. Darüber hinaus werden Entwicklungsartefakte aus der Plattform ausgeleitet und wiederverwendet, sodass produktspezifische Anpassungen bzw. Ergänzungen nur in geringem Maße notwendig sind. Die Applikationsentwicklung setzt sich aus vier Subprozessen zusammen [90]:

- Im **Applikations-Requirements-Engineering** werden aus den Domänenanforderungen wiederverwendbare sowie unter Umständen zusätzliche Anforderungen für die Produkte bestimmt.
- Das **Applikationsdesign** spezialisiert die Architektur der einzelnen Produkte.
- In der **Applikationsrealisierung** werden die Komponenten konfiguriert sowie anhand der wiederverwendbaren und produktspezifischen Anteile realisiert.
- Der **Applikationstest** beschreibt die Aktivitäten zur Validierung und Verifikation der Produkte inklusive Testreports.

Als konkretes Beispiel für die Aktivität der Domänenentwicklung lässt sich das Spezifizieren eines Lastenheftes, welches die Eigenschaften des ESP in allen Baureihen beschreibt, angeben. Das baureihenübergreifende Lastenheft ist somit ein Teil der wiederverwendbaren Plattform. In der anschließenden Applikationsentwicklung wird das ESP für eine konkrete Baureihe (d.h. ein Produkt) entwickelt. Hierfür wird das Lastenheft aus der Domänenentwicklung in angepasster Form für die Baureihe (wieder-) verwendet.

Die Domänenartefakte sind miteinander verknüpft (vgl. Abbildung 8) um eine konsistente Verfolgbarkeit bzw. Durchgängigkeit der Variabilität zwischen den Artefakten sicherzustellen [90]. Dazu zählt beispielsweise die Verknüpfung zwischen den wiederverwendbaren Anforderungs- und Testartefakten. Analog dazu sind die Applikationsartefakte für eine einheitliche Bindung der Variabilität untereinander gekoppelt. Diese konsistente Kopplung der Artefakte wird in Abschnitt 5.4 detailliert behandelt. Schließlich können Links zwischen Domänen- und Applikationsartefakten eine konsistente Evolution der Produktlinie gewährleisten.

Die Prozesse der Domänen- und Applikationstests, welche für die vorliegende Arbeit von besonderer Relevanz sind, werden in Kapitel 4.2.1 eingehender erläutert.

3.3 Variabilität in Produktlinien

Variabilität stellt ein zentrales Konzept dar, mit dessen Hilfe eine Individualisierung von Produkten ermöglicht wird. Für eine Produktlinie bedeutet dies, dass jedes Produkt sich über bestimmte variable Eigenschaften von allen weiteren Produkten unterscheidet, während gleichzeitig alle Produkte einer Produktlinie über einen gemeinsamen, invarianten Teil verfügen [90].

Variabilität kann in unterschiedlichen Phasen sowie auf unterschiedlichen Stufen der Produktlinienentwicklung eine Rolle spielen. Während für einen Systemverantwortlichen die variablen Anteile des Systems Relevanz haben, sind für einen Software-Entwickler die Code-Varianten von Bedeutung. Auf der anderen Seite sind die während der Spezifikationsphase auftretenden variablen Elemente eines Systems auch in der Testphase ausschlaggebend. Variabilität lässt sich somit nicht immer isoliert betrachten, sondern kann Auswirkungen auf andere Entwicklungsstufen bzw. -phasen haben.

Um die damit einhergehende Komplexität beherrschen zu können und das Wissen um die Variabilität für andere Beteiligten im Entwicklungsprozess verfügbar zu machen, ist es wichtig, die Variabilität eines Objektes in einer einheitlichen Art und Weise zu dokumentieren. Dieser Prozess der Beschreibung von Varianten wird als Variabilitätsmodellierung bezeichnet. Die Information, was in der entsprechenden Produktlinie variiert und welche Abhängigkeiten oder Restriktionen es zwischen den Varianten gibt wird in einem Variabilitätsmodell festgehalten. Im Bereich der Software-Produktlinien existieren zahlreiche Methoden für die Modellierung der Variabilität. Die am häufigsten eingesetzten werden im Folgenden vorgestellt.

3.3.1 Variante vs. Version

Bevor auf die Methoden der Variabilitätsmodellierung eingegangen wird, muss zwischen den Begriffen Version und Variante klar differenziert werden. Dies ist notwendig, um Missverständnisse vorzubeugen, da beide Begriffe im Sprachgebrauch häufig als Synonyme verwendet werden. Unter *Varianten*, welche auch als Produkte bezeichnet werden, werden zeitlich parallel existierende Ausprägungen einer Produktlinie verstanden. Varianten beschreiben somit eine statische Variabilität.

Definition 16 (Variante [90]). *Eine Variante eines variablen Objektes stellt eine Form dieses Objektes dar, welches sich von allen weiteren Objekten durch mindestens eine Eigenschaft unterscheidet.*

Aufgrund von technischer Weiterentwicklung der Systeme oder Komponenten, steigenden Sicherheitsanforderung bzw. Einführung von neuen Funktionen sind die Objekte auch zeitlich variabel. Solche Objekte stellen über die Zeit veränderte Gegenstände dar und werden als *Versionen* bezeichnet.

Definition 17 (Version nach [90]). *Versionen stellen zeitlich nacheinander entstehende Objekte dar, wobei eine neue Version durch eine Weiterentwicklung aus einer alten hervorgeht und diese in der Regel dadurch ersetzt.*

Als Beispiel für eine Version lässt sich die aktuelle A-Klasse, W 176, von Mercedes-Benz anführen, welche die Vorgängerversion, nämlich die Baureihe W 169, im Jahr 2012 abgelöst hat. Zwei Varianten dagegen stellen der W 176 mit einem Diesel-Motor sowie alternativ einem Benzin-Motor dar.

Die in der vorliegenden Arbeit vorgestellte Methodik richtet sich auf Varianten und geht nicht auf die zeitliche Variabilität ein.

3.3.2 Merkmalmodellierung

Die am besten erforschte und am häufigsten verwendete Methode zur Modellierung von Variabilität ist die *Merkmalmodellierung* (engl. *feature modeling*) [20, 21, 34, 93, 105]. Ein Merkmalmodell dient dabei einer kompakten Darstellung aller Produkte einer Produktlinien mithilfe von Merkmalen (engl. *features*) sowie den Beziehungen zwischen diesen.

Definition 18 (Merkmal nach [31, 95]). *Ein Merkmal f_i kennzeichnet einen charakteristischen Aspekt bzw. eine Eigenschaft, welche/r in einem Produkt der entsprechenden Produktlinie auftreten kann.*

Die Merkmale eines Merkmalmodells sind hierarchisch angeordnet. Zwischen den daraus resultierenden Elternmerkmalen und zugehörigen Kindermerkmalen bestehen Abhängigkeiten, die in Tabelle 2 zusammengefasst sind. Darüber hinaus legen sogenannte Constraints weitere Merkmal-Beziehungen fest, welche in Tabelle 3 abgebildet sind. Sowohl die Abhängigkeiten als auch die Constraint-Beziehungen definieren semantische Regeln, nach denen die Merkmale zu validen Konfigurationen kombiniert werden dürfen.

Merkmal (-Gruppe)	Implizite Kardinalität	Interpretation
obligatorisches Merkmal f_{obl}	[1..1]	f_{obl} ist Bestandteil jeder Konfiguration, welche das Elternmerkmal von f_{obl} enthält
optionales Merkmal f_{opt}	[0..1]	f_{opt} kann in einer Konfiguration enthalten sein
alternative Merkmalgruppe F_{alt}	$\langle 1 - 1 \rangle$	eine Konfiguration enthält genau ein Merkmal aus der Merkmalgruppe, $f_i \in F_{alt}$
oder-Merkmalgruppe F_{oder}	$\langle 1 - n \rangle$	eine Konfiguration kann beliebige Kombinationen von Merkmalen aus der Gruppe, mindestens jedoch eins $f_i \in F_{oder}$ beinhalten

Tabelle 2: In einem Merkmalmodell verwendete Merkmaltypen zur Abbildung von Abhängigkeiten zwischen Merkmalen nach [32]

Beziehung	Formalismus	Interpretation
Implikation (unidirektional)	$f_i \rightarrow f_j$	eine Konfiguration enthält immer f_j , wenn es f_i enthält
Exklusion (bidirektional)	$\neg f_i \vee \neg f_j$	f_i und f_j sind niemals Bestandteil <i>einer</i> Konfiguration

Tabelle 3: In einem Merkmalmodell verwendete Constraint-Beziehungen zwischen Merkmalen

Definition 19 (Konfiguration). *Eine Konfiguration c_i beschreibt eine Kombination von Merkmalen, welche ein konkretes Produkt der entsprechenden Produktlinie kennzeichnet und die, durch das Merkmalmodell vorgegebenen, semantischen Abhängigkeiten und Constraint-Beziehungen beachtet. Als Synonyme² sind die Begriffe Variante und Produkt gebräuchlich.*

Abbildung 9 zeigt ein beispielhaftes Merkmalmodell für das fiktive System *Klimatisierung* aus dem Automobilbereich, das die genannten Beziehungen aus Tabelle 3 und Abhängigkeiten aus Tabelle 2 enthält. Das System variiert in den *Klimamaßnahmen* sowie der Tatsache, ob eine und wenn ja, welche *HV-Batterie* verbaut ist. Dabei erlaubt die Semantik der alternativen Merkmalgruppe *HV-Batterie* genau ein Kindermerkmal pro Fahrzeug, nämlich eine *große*, eine *kleine* bzw. *keine HV-Batterie*. Aus der oder-Gruppe der *Klimamaßnahmen* können alle Kindermerkmale im Fahrzeug verbaut sein, mindestens jedoch eins. Die optionale *Klimaanlage Automatik* sowie *Sitzkühlung* können Bestandteile eines Fahrzeugproduktes sein, Letztere jedoch nur, wenn eine *Sitzheizung* ebenfalls enthalten ist. Darüber hinaus schränken die Constraint-Beziehungen den Konfigurationsraum weiter ein, da die Exklusion das Vorkommen einer *Panelheizung* sowie einer *kleinen HV-Batterie* in einer Konfiguration verbietet. Zusätzlich kann die *Lenkradheizung* nur mit der *Panelheizung* vertreten sein (wobei die *Panelheizung* nicht

² Die Begriffe Konfiguration, Produkt und Variante lassen sich nicht immer scharf abgrenzen und werden daher je nach Kontext synonymisch verwendet.

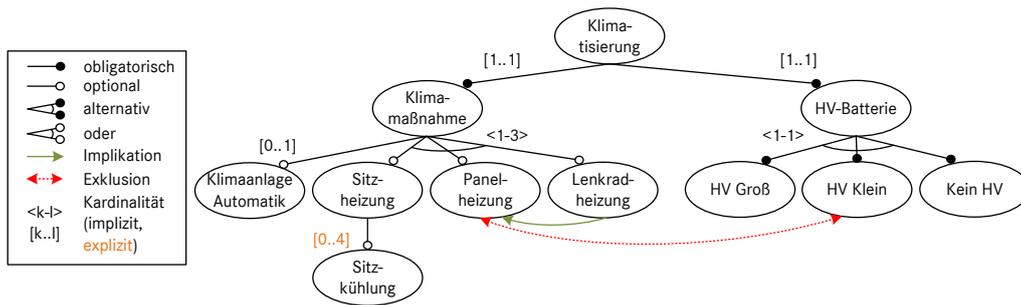


Abbildung 9: Merkmalmodell für das beispielhafte System *Klimatisierung*

zwingend die *Lenkradheizung* benötigt). Die Merkmale *Klimamaßnahmen* und *HV-Batterie* beschreiben Variationspunkte des Systems. An diesen Stellen führt die Auswahl einer oder mehrerer Kindermerkmale zu unterschiedlichen Konfigurationen.

Definition 20 (Variationspunkt nach [4]). *Ein Variationspunkt ist eine Stelle im Variabilitätsmodell (oder auch in Artefakten), an der die Auswahl einer oder mehrerer Ausprägungen für unterschiedliche Konfigurationen bzw. Produkte möglich ist.*

Ursprünglich wurde die Merkmalmodellierung von Kang et al. in [53] unter der Bezeichnung FODA (engl. *feature-oriented domain analysis*) eingeführt. Heute sind unzählige Tools und Techniken [5, 9] sowie zahlreiche Ergänzungen und Anpassungen [93] zur FODA-Modellierung verfügbar.

Eine von Czarnecki et al. in [33] vorgeschlagene Erweiterung ist die kardinalitätsbasierte Notation der Merkmalmodellierung. Diese ermöglicht eine Attributierung von Merkmalen sowie Merkmalgruppen mit Multiplizitäten (engl. *multiplicity*).

- Die **Kardinalität eines Merkmals** gibt dabei ein Intervall $[k..l]$, mit der oberen l und unteren k Grenze, vor. Das Intervall bestimmt die Anzahl der Merkmale, welche in einem Produkt enthalten sein kann. Für die bereits vorgestellten Merkmalbeziehungen eines FODA-Merkmalmodells sind die Kardinalitäten implizit vorgegeben (vgl. Tabelle 2 und Abbildung 9). So ist die Kardinalität eines obligatorischen Merkmals $[1..1]$, während diese für ein optionales Merkmal $[0..1]$ beträgt.
- Analog dazu legt die **Kardinalität einer Merkmalgruppe** ein Intervall $<k-l>$ fest, wobei l die Anzahl der in einem Produkt enthaltenen Merkmale nach oben und k nach unten limitiert. Eine alternative Merkmalgruppe besitzt demnach die Kardinalität $<1-1>$, was für die Gruppe *HV-Batterie* in Abbildung 9 der Fall ist. Eine oder-Gruppe entspricht der Kardinalität $<1-L>$, während L die Anzahl an Kindermerkmalen der oder-Gruppe repräsentiert.

Zusätzlich lassen sich die impliziten Kardinalitäten durch explizite erweitern. So zeigt Abbildung 9 eine Kardinalität $[0..4]$ für das Merkmal *Sitzkühlung*. Diese sagt aus, dass die Sitzkühlung viermal im Fahrzeug, d.h. am Fahrer-, Beifahrersitz sowie an den hinteren Sitzen links und rechts, verbaut sein kann. Die expliziten Kardinalitäten werden an dieser Stelle nur kurz angeführt, da in den bei der Daimler AG verwendeten

Variabilitätsmodellen auf deren Angabe verzichtet wird (vgl. Abschnitt 5.2). Weitere Details können jedoch in [32, 33] nachgelesen werden.

Schließlich besteht die Möglichkeit zur Erweiterung der Merkmalmodelle in Form von Merkmalattributen. Diese erlauben das Hinterlegen von weiteren relevanten Informationen, wie Kosten, Wertebereichen, Domänen usw. Für Genaueres sei auf weitere Literatur [2, 3] verwiesen.

3.3.3 Orthogonale Variabilitätsmodellierung

Ergänzend zur Merkmalmodellierung wird im Folgenden ein weiterer Ansatz zur Verwaltung der Variabilität vorgestellt, da dieser unter anderem bei der Analyse der verwandten Arbeiten (vgl. Kapitel 4) relevant ist. Dieser Ansatz ist die flache *orthogonale Variabilitätsmodellierung*, welche von Pohl et al. eingeführt wurde [90]. Ein orthogonales Variabilitätsmodell besteht aus zwei Typen von Merkmalen. Dies sind zum einen Variationspunkte (engl. *variation points*, VP), welche zur Beschreibung von variablen Aspekten dienen und zum anderen Varianten (engl. *variants*, V), welche zur Darstellung möglicher Ausprägungen der jeweiligen variablen Aspekte genutzt werden [75]. In Analogie zur Merkmalmodellierung können (obligatorische, alternative und optionale) Abhängigkeiten (engl. *variability dependences*) mit Kardinalitäten sowie Constraint-Beziehungen (Implikation, Exklusion) abgebildet werden [97]. Da Merkmalmodelle und orthogonale Variabilitätsmodelle semantisch äquivalent zu einander sind [75], kann das Merkmalmodell für das System *Klimatisierung* (vgl. Abbildung 9) in ein orthogonales Variabilitätsmodell überführt werden (siehe Abbildung 10).

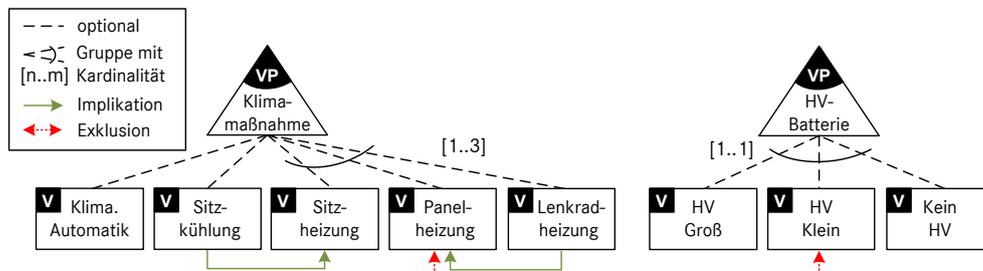


Abbildung 10: Orthogonales Variabilitätsmodell für das beispielhafte System *Klimatisierung* aus Abbildung 9

Der Ansatz wird als orthogonal bezeichnet, da dieser Variabilität in einem eigenständigen Modell, d.h. unabhängig von konkreten Entwicklungsartefakten beschreibt und gleichzeitig die Zuordnung der Variationspunkte zu Entwicklungsartefakten ermöglicht [61]. Dies wird häufig als Vorteil bei der Verwendung der orthogonalen Variabilitätsmodellierung genannt. Viele Beiträge haben jedoch gezeigt, dass Merkmalmodelle ebenfalls, über die Variabilitätsbeschreibung auf Merkmalebene hinaus, zu diversen Entwicklungsartefakten in Beziehung gesetzt werden können [83].

3.3.4 Entscheidungsmodellierung

Um den Überblick über die Methoden der Variabilitätsmodellierung zu vervollständigen, wird abschließend das *Entscheidungsmodell* vorgestellt. Dieses, als Teil von Synthesis³ eingeführte Vorgehen [17] gehört, im Gegensatz zu vielen anderen, nicht zur Gruppe der FODA-basierten Ansätze. Typischerweise werden in einem Entscheidungsmodell Entscheidungen in Form von Fragen mit einer definierten Menge an Antwortmöglichkeiten repräsentiert. Hierbei lässt sich jede Entscheidung mit Entwicklungsartefakten verknüpfen. Der wesentliche Unterschied zu Merkmalmodellen besteht darin, dass in einem Entscheidungsmodell keine obligatorischen, d.h. in allen Konfigurationen vorhandene, sondern nur variable Anteile abgebildet werden [93]. Das Modell ist dabei häufig in Tabellenform dargestellt und kann Informationen wie Kardinalitäten und Constraint-Beziehungen enthalten. Tabelle 4 zeigt das Entscheidungsmodell angelehnt an das System *Klimatisierung* aus Abbildung 9. Für eine ausführliche Diskussion zu Entscheidungsmodellen siehe [102].

ID	Frage	Antwort	Kardinalität	Constraint-Beziehungen
Klima-Automatik	Soll Klima-Automatik vorhanden sein?	Ja, Nein	0..1	-
Sitzkühlung	Soll Sitzkühlung vorhanden sein?	Ja, Nein	0..4	Sitzkühlung.Ja <i>benötigt</i> (Klimamaßnahme.Sitzheizung)
Heizmaßnahme	Welche Heizmaßnahme(n) vorhanden sein?	Sitzheizung, Panelheizung, Lenkradheizung	1 : 3	Klimamaßnahme.Panelheizung <i>schließt aus</i> HV.Klein
HV	Welche Batterie soll vorhanden sein?	Groß, Klein, Keine	1 : 1	HV.Klein <i>schließt aus</i> Klimamaßnahme.Panelheizung

Tabelle 4: Vereinfachtes Entscheidungsmodell angelehnt an das beispielhafte System *Klimatisierung* aus Abbildung 9

Zusätzlich zu den in diesem Kapitel vorgestellten Ansätzen zur Merkmalmodellierung (vgl. Abschnitt 3.3.2), der orthogonalen Variabilitätsmodellierung (vgl. Abschnitt 3.3.3) sowie der Entscheidungsmodellierung existiert eine Reihe weiterer Methoden. Da diese in Zusammenhang der vorliegenden Arbeit nicht grundlegend sind, sei hier auf weiterführende Literatur in [20, 21, 93] verwiesen.

Zusammenfassung

Das vorliegende Kapitel gibt einen Überblick über die Grundlagen der Produktlinienentwicklung. Durch deren Anwendung lässt sich eine Reihe von wirtschaftlichen und technischen Vorteilen gegenüber der Einzelsystementwicklung erzielen. Dies ist möglich, da der Fokus bei der Entwicklung von Produktlinien auf der Herausarbeitung und Ausnutzung von gleichartigen Elementen zwischen den Produkten liegt.

³ Synthesis ist ein Vorgehensmodell zur Unterstützung der Wiederverwendung, welches 1990 vom Software Productivity Consortium für den Einsatz in der Industrie entwickelt wurde.

Weiterhin wird mithilfe der Differenzierung zwischen der Domänenentwicklung, in welcher die Identifikation der genannten Gemeinsamkeiten geschieht sowie der Applikationsentwicklung, in der die diversen Produkte erzeugt werden, ein strukturierter, organisatorischer Rahmen für die Entwicklung vorgegeben.

Um das Konzept der Produktlinienentwicklung umsetzen zu können, spielt die Variabilitätsmodellierung eine wesentliche Rolle. Ein Variabilitätsmodell unterstützt eine systematische und damit einheitliche Dokumentation der Variabilität mit Beziehungen und Abhängigkeiten zwischen den Variationspunkten. Nachfolgend ermöglicht es die Konfigurierbarkeit der Artefakte aus der Domänen- für die Applikationsentwicklung zu systematisieren. Dies ist für die vorliegende Arbeit von zentraler Bedeutung, da in Kapitel 5 eine solche Produktlinieninfrastruktur in Anforderungs- und Testartefakten umgesetzt und in Kapitel 6 die darauf aufbauende Selektionsmethodik angewendet wird.

4 | TESTEN VON PRODUKTLINIEN

Im Mittelpunkt dieses Kapitels steht die ausführliche Behandlung der verwandten Arbeiten zum Testen von Produktlinien. Durch die Beschreibung der allgemeinen Herausforderungen und Ziele des Produktlinientests wird in Abschnitt 4.1 zunächst eine Einführung in die Thematik gegeben. Davon ausgehend werden in Abschnitt 4.2 die einzelnen methodischen Ansätze detailliert behandelt und jeweils kritisch bewertet. Abschließend werden in Abschnitt 4.3 die als relevant befundenen Methoden auf definierte Kriterien hin zusammenfassend untersucht und evaluiert. Zum einen wird dadurch eine Abgrenzung der vorliegenden Arbeit zu den bereits existierenden Verfahren ermöglicht. Zum anderen werden die Kriterien für einen erfolgreichen Produktlinientest identifiziert und zur Ausarbeitung der Methodik für die vorliegende Arbeit in den folgenden Kapiteln 5 und 6 verwendet.

4.1 Herausforderungen des Produktlinientests

Der methodische Ansatz der Produktlinienentwicklung kann eine signifikante Verbesserung des Entwicklungsprozesses im Bezug auf die Kosten und Zeit, aber auch die Qualität der Produkte bewirken (vgl. Kapitel 3). Die Vorteile resultieren dabei aus der Wiederverwendung von Gemeinsamkeiten zwischen den Produkten. Dieses Prinzip lässt sich auch in der Phase des Testens von Produktlinien für einen effizienten Testprozess einsetzen.

Eine der größten Herausforderungen betrifft dabei die große Anzahl an Tests, die durchgeführt werden müssten, um eine Produktlinie vollständig abzusichern. Aufgrund der Tatsache, dass die Anzahl der gültigen Konfigurationen, die aus einer Produktlinie abgeleitet werden kann, exponentiell mit der Anzahl der möglichen Optionen wächst [38], ist das Testen aller Konfigurationen sowohl aus ökonomischer als auch praktischer Sicht nicht realisierbar. Ein weiterer Aspekt betrifft die Entwicklungskosten und -zeit, von welchen alleine schon in der Einzelsystementwicklung bis zu 50% für die Testphase beansprucht werden [86, 111]. Geht man von Produktlinien mit einer Vielzahl an Produkten aus, kann der Aufwand extrem schnell ansteigen.

Über die genannten Herausforderungen hinaus spiegelt sich die Bedeutung des Produktlinientests unmittelbar in der hohen Anzahl an wissenschaftlichen Untersuchungen sowie Literaturrecherchen [18, 38, 59, 77] wider. Die im nächsten Kapitel vorgestellte Bestandsaufnahme der Testmethoden zum Produktlinientest stellt die für die vorliegende Arbeit essentiellen Arbeiten vor. Der Fokus der diskutierten Beiträge liegt dabei auf solchen, welche eine oder mehrere der folgenden unterstützenden Maßnahmen bereitstellen:

- adäquate Abbildung der Variationspunkte in Testartefakten,

- systematische Bestimmung einer Untermenge zu testender Konfigurationen sowie
- Vermeidung von redundanten Testfällen durch Wiederverwendung.

4.2 Methoden des Produktlinientests

Die Methoden des Produktlinientests lassen sich in vier Kategorien unterteilen:

1. In der ersten Kategorie wird in Abschnitt 4.2.1 das Konzept der *Aufteilung nach Verantwortlichkeiten* vorgestellt. Dieses nimmt eine Aufteilung der Testaktivitäten in Domänen- und Applikationsentwicklung vor und kann ergänzend zu allen, in den folgenden drei Kategorien diskutierten Verfahren eingesetzt werden.
2. Der zweiten Kategorie werden Vorgehen zugeordnet, welche das individuelle Testen jedes einzelnen Produktes einer Produktlinie vorsehen. Die Verfahren sind unter dem Begriff des *Product by Product Testens* bekannt und werden in Abschnitt 4.2.2 diskutiert.
3. In der dritten Kategorie werden Methoden, welche sich das Prinzip der Wiederverwendung zu Nutze machen, aufgeführt. Diese finden beim *Inkrementellen Testen* Anwendung, zusammengefasst in Abschnitt 4.2.3, bei welchem Ansätze des Regressionstests herangezogen werden. Des Weiteren werden die Methoden im Bereich des *Modellbasierten Testens* von Produktlinien eingesetzt, wenn Tests aus der Domänen- in der Applikationsentwicklung in angepasster Form wiederverwendet werden, siehe Abschnitt 4.2.4.
4. Abschließend fasst die vierte Kategorie Verfahren zusammen, bei denen die zu testenden Konfigurationen basierend auf verschiedenen (Abdeckungs-)Kriterien selektiert werden. Die Kriterien können sich auf *Gewichtungsfaktoren*, dargestellt in Abschnitt 4.2.5, die Abdeckung von *Merkmalleraktionen*, beschrieben in Abschnitt 4.2.6, und/oder die Abdeckung von *Anforderungen*, ausgearbeitet in Abschnitt 4.2.7, beziehen. Anstelle des vollständigen Testens jeder möglichen Konfiguration wird nur die selektierte, repräsentative Untermenge an Konfigurationen abgesichert.

Im Folgenden werden die Methoden jeweils inhaltlich vorgestellt sowie anschließend kritisch bewertet. Der Schwerpunkt liegt auf Verfahren der vierten Kategorie, zu welcher die vorliegende Arbeit gehört.

4.2.1 Aufteilung nach Verantwortlichkeiten

Die *Aufteilung nach Verantwortlichkeiten* (engl. *division of responsibilities*) stellt ein Konzept dar, welches begleitend bzw. ergänzend zu allen im Weiteren diskutierten Testmethoden des Produktlinientests angewendet werden kann. Das Konzept legt fest, auf welchen Teststufen des V-Modells (vgl. Abschnitt 2.2) die Testaktivitäten der Domänen-, der Applikationsentwicklung oder beider Phasen stattfinden.

Gemäß der in Kapitel 3.2 beschriebenen Strategie der Produktlinienentwicklung werden Testaktivitäten in Domänen- sowie Applikationstests unterteilt. Im Domänentest werden dabei wiederverwendbare, unter Umständen unfertige, Komponenten abgesichert, während im Applikationstest konkrete und vollständige Applikationen getestet werden. Dabei wird häufig empfohlen, die Domänentests sehr frühzeitig und daher tendenziell auf den unteren, dagegen die Applikationstests auf den höheren Teststufen durchzuführen [85, 118]. Das Vorgehen ist in Abbildung 11 schematisch dargestellt. Eine einheitliche Zuordnung der Teststufen zu den beiden Prozessen der Domänen- sowie Applikationsentwicklung existiert derweilen nicht, denn diese hängt stark davon ab, welche Methode für den Produktlinientest verwendet wird. In der Literatur werden unterschiedliche Ansätze vorgestellt.

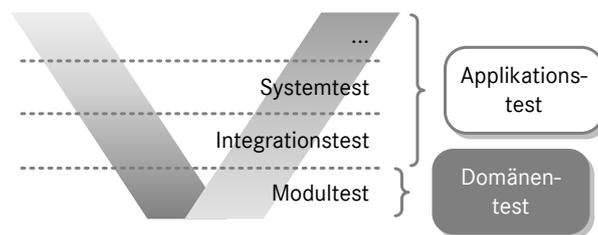


Abbildung 11: Abstrahierte Darstellung einer beispielhaften *Aufteilung nach Verantwortlichkeiten*

So konstatieren Stricker et al. [116], dass das Testen in der Applikationsentwicklung das wichtigste Mittel zur verlässlichen Absicherung von Produktlinien darstellt und beschreiben eine Methodik, welche das unnötige Testen von redundanten Umfängen auf Basis der Datenflussabdeckung in der Applikationsentwicklung vermeidet. In [78] wird ebenfalls eine Methodik vorgestellt, welche sich auf Applikationstests beschränkt. Dabei werden die Anforderungen aus der Domänenentwicklung zur automatisierten Ableitung von Testfällen auf Systemebene für konkrete Produkte in der Applikationsentwicklung verwendet.

Im Gegensatz dazu schlagen Al-Dallal et al. [1] das Testen in beiden Phasen der Produktlinienentwicklung vor und präsentieren ein Verfahren zur Identifikation von Anwendungsfällen, welche in der Domänenentwicklung noch nicht abgedeckt wurden. Diese Anwendungsfälle sollen mithilfe einer Wiederverwendungstechnik konfigurationsspezifisch angepasst und in der Applikationsentwicklung berücksichtigt werden. Auch in [50] bewerten die Autoren beide Phasen für den Produktlinientest als relevant. Davon ausgehend beschreiben sie das Konzept des W-Modells, in welchem das klassische V-Modell (vgl. Abschnitt 2.2) jeweils in der Domänen- und Applikationsentwicklung für alle Teststufen durchlaufen wird.

Bewertung und Relevanz des Konzeptes Aufteilung nach Verantwortlichkeiten

Das Konzept der *Aufteilung nach Verantwortlichkeiten* lässt sich als begleitende Maßnahme zu den vorgestellten Methoden des Produktlinientests ansehen. Das Verfahren ist eng mit dem Referenzprozess der Produktlinienentwicklung verzahnt. Somit trägt

es zu einer höheren Durchgängigkeit zwischen der Domänen- und Applikationsentwicklung bei und kann das Zustandekommen von Synergieeffekten aus beiden Phasen begünstigen.

Gleichzeitig bringt das Verfahren zusätzlichen Aufwand mit sich, welcher sich insbesondere in organisatorischen Aspekten widerspiegelt. Denn bei einer festen Aufteilung der Teststufen auf Domänen- und Applikationstests ist eine erhöhte Abstimmungs- und Koordinationsarbeit zwischen den Fraktionen der beiden genannten Phasen notwendig. Der Informationsaustausch ist wichtig, um einerseits die bereits in der Domänenentwicklung abgesicherten Umfänge nicht redundant in der Applikationsentwicklung zu testen, andererseits um Lücken zu vermeiden. Gegebenenfalls können dabei Methoden des Regressionstests unterstützend eingesetzt werden.

4.2.2 Product by Product Testen

Die elementarste Methodik, um eine ausreichende Absicherung einer Produktlinie zu gewährleisten, ist es, jedes einzelne Produkt der Produktlinie individuell zu testen [50, 118], dargestellt in Abbildung 12. Ein solches Vorgehen wird als *Product by Product Testen* bezeichnet. Da hierbei Gemeinsamkeiten zwischen den Produkten nicht genutzt werden, entstehen viele redundante Testumfänge. Zusätzlich ist der Aufwand in der Bereitstellung von Testartefakten für jedes Produkt ohne Wiederverwendung enorm. Die Testaktivitäten des *Product by Product Testens* finden allein in der Applikationsentwicklung statt, in welcher nur konkrete Produkte getestet werden. Ein ähnliches Vorgehen wird in [90] als sogenannte *Brute Force Strategy* beschrieben, welche das umfassende Testen aller Konfigurationen vorsieht. Der Unterschied zum *Product by Product Testen* besteht darin, dass die Testaktivitäten der *Brute Force Strategy* noch in der Domänenentwicklung stattfinden.

Der Vorteil solcher Verfahren liegt darin, dass eine hohe Produktqualität garantiert werden kann. Jedoch weisen diese eine starke Ähnlichkeit zur Einzelsystementwicklung auf und stehen somit im Widerspruch zum eigentlichen Kerngedanken der Produktlinienentwicklung. Dennoch werden die Verfahren behandelt, um einen gesamtseitlichen Überblick zu ermöglichen.

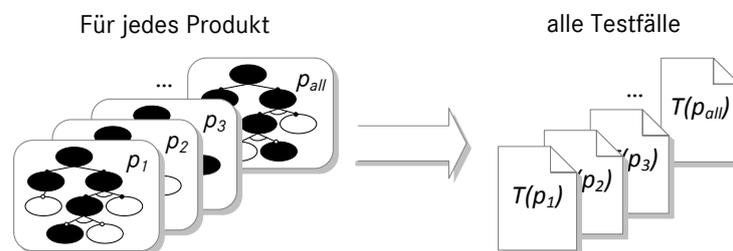


Abbildung 12: Abstrahierte Darstellung des *Product by Product Testens*

Der Ansatz des *Product by Product Testens* wurde bei Nokia zur Entwicklung der Produktlinie von mobilen Browsern angewandt [49]. Jaaksi berichtet, dass durch die

Umgestaltung des Entwicklungsprozesses von der Einzelsystem- zur Produktlinienentwicklung insbesondere in den Bereichen des Anforderungsmanagements und der Architektur hohe Wiederverwendungsquoten erreicht wurden. Einzig in der Testphase wurde bewusst darauf verzichtet, gemeinsame Anteile verschiedener Produkte auszunutzen, sodass stattdessen die Produkte vollständig in jeder verfügbaren Konfiguration abgesichert wurden. Mit etwa 30% der benötigten Entwicklungszeit für den Test eines Produktreleases von mobilen Browsern, definiert Jaaksi das Testen einer Produktlinie als eine kostspielige und komplexe Aufgabe und schlägt für das Erreichen einer garantierten Qualität das Absichern jeder Produktkonfiguration vor.

Bewertung und Relevanz des Product by Product Testens

Aus Sicht der Anwendbarkeit weist das *Product by Product Testen* einen hohen Grad an Generalität auf und setzt dabei wenig methodisches Hintergrundwissen voraus. Dadurch, dass jede Konfiguration und damit jede mögliche Interaktion zwischen Merkmalen getestet wird, können alle aus den Wechselwirkungen resultierenden Fehler aufgedeckt werden. Aus ökonomischer Sicht sind jedoch erhebliche Mehrkosten zu erwarten, da viele redundante Testumfänge vorliegen.

In einigen Studien wurde ein Vergleich zwischen Strategien des Einzel- und Produktlinientests durchgeführt. So führen Ganesan et al. anhand einer, aus zehn Produkten bestehenden, Produktlinie der Testo AG¹ die Kostenberechnung für beide Teststrategien durch [39]. Das Ergebnis zeigt, dass mithilfe des Produktlinientests im Schnitt eine Kostenersparnis von 13% mit einer 87%igen Gewissheit zu erwarten ist [51]. Je höher dabei die Anzahl der Produkte in der Produktlinie liegt, desto höher ist die Kostenreduktion. In [104] wird dieses positive Ergebnis von Philips Medical Systems mit einer empirischen Studie aus dem medizinischen Bereich bestätigt. Darin wird von einer 50%igen Senkung der gefundenen Fehler für eine aus zehn Produkten bestehende Produktlinie berichtet. Auch Dialect Solutions stellt eine Verbesserung bei der Anwendung des Produktlinientests in der Entwicklung von sechs Softwareprogrammen für Zahlungssysteme fest [113]. Die Verbesserung ist qualitativer Art und wird nicht genauer quantifiziert.

Der Ansatz einer produktspezifischen Absicherung ist somit generell als nicht effizient zu bewerten und kann nur bei sehr kleinen Produktlinien zum Erfolg führen. Dies wird von Jaaksi, welcher von einer Produktlinie bei Nokia mit insgesamt vier Produkten berichtet, untermauert [49]. Ganz offensichtlich würde eine solche Methodik bei Produktlinien mit einer wesentlich höheren Anzahl an ausleitbaren Produkten, an den dadurch potentiell sehr hohen Kosten und Zeitaufwänden, scheitern.

4.2.3 Inkrementelles Testen

Im Gegensatz zum *Product by Product Testen* strebt das *Inkrementelle Testen* eine möglichst effiziente Wiederverwendung der Testfälle. Hierfür werden Methoden, welche zum Regressionstest verwandt sind, eingesetzt. Der Regressionstest wird im Allgemeinen

1 Die Testo AG präsentiert eine Produktlinie von Messgeräten für Abgase und Emissionen.

genutzt, um solche Testfälle zu ermitteln, welche nach Modifikation eines Testobjektes zu dessen *Re*-Verifikation wiederholt durchgeführt werden müssten. Das modifizierte Testobjekt stellt oft eine Weiterentwicklung des ursprünglichen Testobjektes dar und ist somit eine Version davon. Beim Produktlinientest geht man dagegen von parallel bzw. gleichzeitig existierenden Versionen, nämlich Varianten (eigentlich Produkten) aus (vgl. Abschnitt 3.3.1). Analog zu Versionen gibt es zwischen Produkten Gemeinsamkeiten und Unterschiede, welche durch das jeweilige Verfahren effizient ausgenutzt werden. Setzt man Produkte mit Versionen gleich, kann beim Produktlinientest ausgehend von einem vollständig getesteten Produkt p_{init} , die Regressionstestmethodik zur Identifikation der Testumfänge weiterer Produkte eingesetzt werden. Das Vorgehen ist in Abbildung 13 schematisch dargestellt. In der Literatur finden sich verschiedene Ansätze, in denen sich dieser Zusammenhang widerspiegelt und Methoden des Regressionstests auf die Herausforderungen und Ziele des Produktlinientests übertragen werden [37, 118].

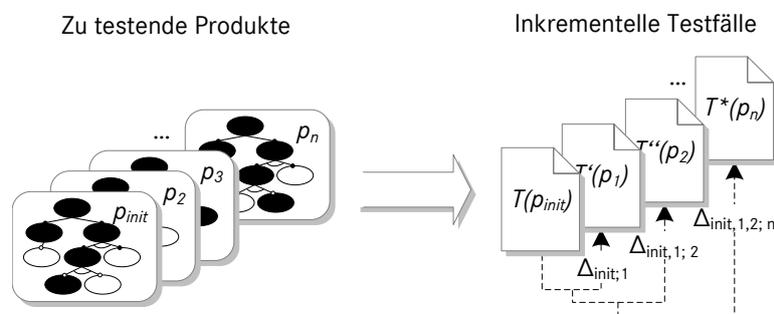


Abbildung 13: Abstrahierte Darstellung des *Inkrementellen Testens*

So stellen Uzuncaova et al. einen spezifikationsbasierten Ansatz zur automatisierten, inkrementellen Generierung von Testfällen für Produktlinien vor [123]. Dabei ist ein Merkmal als ein Inkrement des jeweiligen Produktes und ein Produkt als eine Komposition von Merkmalen definiert. Die Generierung der Testfälle erfolgt für jedes Produkt der Produktlinie durch eine inkrementelle Anpassung der Testumfänge.

Ein ähnlicher Ansatz wird von Lity et al. in [66] vorgestellt. Die Basis bildet ein wiederverwendbares Testmodell der jeweiligen Produktlinie. Dieses ist durch ein Kernprodukt p_0 (engl. *core product*) repräsentiert sowie eine Menge von Deltas δ , durch die Änderungen von p_0 zur Darstellung weiterer Produkte beschrieben sind. Ausgehend davon wird zuerst ein initialer Testumfang mithilfe von herkömmlichen Techniken des modellbasierten Testens für ein Produkt generiert. Die Testartefakte weiterer Produkte entstehen danach inkrementell durch Wiederverwendung bereits betrachteter Testartefakte.

Einen stärker methodisch ausgerichteten Ansatz behandelt Nörenberg in [80], welcher von Anforderungen und den damit assoziierten Testfällen in natürlichsprachlichen Spezifikationen auf Systemebene ausgeht. Basierend auf einer standardisierten Teststrategie sowie einer vordefinierten Systemdarstellung des Testobjektes wird eine Regressionstestanalyse zur Identifikation aller potentiell gefährdeter Testobjekte nach der Änderung eingesetzt. Anschließend lassen sich verschiedene Regressionstestmechanismen anhand der Testziele, des Funktionsablaufs und/oder der Testintensität zur weiteren Reduzierung der selektierten Testfälle anwenden.

Bewertung und Relevanz des Inkrementellen Testens

Die Grundidee des *Inkrementellen Testens* basiert auf der Methode des Regressionstests. Da der Regressionstest ein in der Anwendung bewährtes Standardtestverfahren darstellt, kann der Produktlinientest die bereits ausgearbeitete Methode im neuen Kontext nutzen und somit stark davon profitieren. Dabei können hohe Wiederverwendungsquoten erreicht werden. So wird in [122] von einer Ersparnis zwischen 50% und 75% der durchzuführenden Testfälle sowie in [80] von rund 70% berichtet.

Eine Schwierigkeit, die sich aus dem Vorgehen ergibt, liegt in der Identifikation einer geeigneten initialen Produktkonfiguration p_{init} , welche einem vollständigen Test unterzogen wird, um auf deren Basis alle weiteren Testumfänge zu definieren. Hier ist Potential durch die Festlegung von konkreten Kriterien gegeben, um die Findung des initialen Produktes zu unterstützen. Ebenfalls kritisch zu bewerten ist die Tatsache, dass die Methodik des *Inkrementellen Testens* nach einer Reduktion der Testumfänge für die Produkte strebt, jedoch die Frage danach, welche repräsentativen, selektierten Konfigurationen für die Absicherung der gesamten Produktlinie genügen, nicht in den Fokus rückt. Es wird entweder von einer kleinen Produktlinie ausgegangen oder einer bereits vorliegenden Konfigurationsuntermenge, welche mithilfe anderer Verfahren (z.B. durch eine Pairwise-Abdeckung, siehe Abschnitt 4.2.6) bestimmt wird. Werden die Methoden des Inkrementellen Testens sowie die einer Variantenselektion zusammengeführt, kann dies zu einem hohen Synergiepotenzial führen.

4.2.4 Modellbasierter Produktlinientest

Ähnlich zum *Inkrementellen Testen* wird beim *Modellbasierten Produktlinientest* das Prinzip der Wiederverwendung genutzt. Die in der Domänenentwicklung initial erstellten Testmodelle werden in der anschließenden Applikationsentwicklung in angepasster Form wiederverwendet. Das Soll-Verhalten des entsprechenden Testobjektes wird in Zustandsübergangsdiagrammen (engl. *state charts*), Aktivitätsdiagrammen (engl. *activity diagrams*) oder Sequenzdiagrammen (engl. *sequence diagrams*) spezifiziert. Der Fokus der Methodik liegt dabei auf der automatisierten Ableitung von Testfällen für die einzelnen Produkte der Produktlinie auf Basis von Modellen. Eine Übersicht über das Vorgehen ist in Abbildung 14 gegeben.

In [82] beschreibt Olimpiew das modellbasierte Testverfahren CADeT (engl. *customizable activity diagrams, decision tables and test specifications*). Darin werden aus textuellen Anwendungsfällen (engl. *use cases*) zunächst Aktivitätsdiagramme manuell erstellt. Basierend auf einem Aktivitätsdiagramm sowie einem Merkmalmodell, welches die verschiedenen Anwendungsfälle darstellt, werden Testfälle automatisiert abgeleitet. Dabei sind in CADeT einige Abdeckungen bezüglich der Merkmalinteraktionen (vgl. Abschnitt 4.2.6) möglich. Über eine Entscheidungstabelle werden die Testfälle aus der Domänenentwicklung produktspezifisch anpasst.

In [96] wird eine ähnliche Methode beschrieben. ScenTED (engl. *scenario-based test case derivation*) ist ebenfalls zur automatisierten Generierung von Testfällen ausgehend von Aktivitätsdiagrammen konzipiert. Darin wird die Variabilität über gekennzeichnete

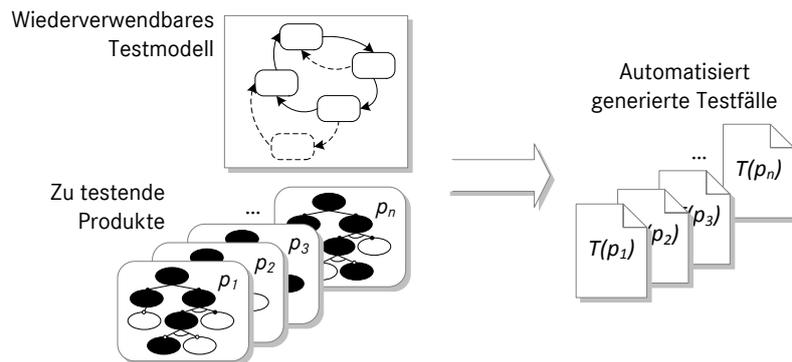


Abbildung 14: Abstrahierte Darstellung des *Modellbasierten Produktlinientests*

Verzweigungen im Kontrollfluss des Aktivitätsdiagramms beschrieben. Die Testfallableitung geschieht im Hinblick auf das Kriterium der Zweigüberdeckung. Nicht nur Aktivitätsdiagramme, sondern auch Zustandsautomaten (engl. *state machines*) werden beim modellbasierten Produktlinientest verwendet. Im Beitrag, beschrieben in [86], setzt Kishi et al. das Verfahren des Model Checkings zur Verifikation der formalen Systembeschreibung von Produktlinien ein. Dadurch sollen Fehler noch während der frühen Entwicklungsphase des Spezifizierens aufgedeckt werden. Anstelle einer Prüfung der Modelle für die einzelnen Produkte der Produktlinie, schlägt Kishi vor, die Modelle wiederverwendbar zu gestalten. Dazu wird in Zustandsautomaten eines Umgebungsmodells (engl. *environment model*) Variabilität abgebildet, sodass diese produktspezifisch auf das Verhaltensmodell (engl. *target model*) des Testobjektes angewandt werden können.

Bewertung und Relevanz des Modellbasierten Produktlinientests

Gemäß der klassischen modellbasierten Verfahren gibt es beim modellbasierten Produktlinientest eine Reihe von Vor- und Nachteilen. So ist die Erreichbarkeit eines hohen Automatisierungsgrades, durch welchen ein wirtschaftlicherer Testprozess realisiert werden kann, eine wesentliche Stärke der Methodik. Darüber hinaus ist durch das formale Spezifizieren eines Testobjektes zwar ein Zusatzaufwand, jedoch auch mehr Systematisierung beim Ableiten von Testfällen gegeben.

Die Herausforderung, welche sich bei der Anwendung des modellbasierten Testens von Produktlinien stellt, betrifft die Darstellung der Variabilität in den Testmodellen sowie die Zuordnung der Modellartefakte zu den Variationspunkten. Dies kann entweder unter Anknüpfung an ein externes Variabilitätsmodell oder direkt im Testmodell geschehen. In diesem Aspekt unterscheiden sich viele Beiträge. Ein weiterer Schwerpunkt der Untersuchungen liegt auf der automatisierten Testfallgenerierung, welche nach verschiedenen Abdeckungskriterien, ablaufen kann. Analog zum *Inkrementellen Testen* steht jedoch die Frage nach der methodischen Selektion der repräsentativen, zu testenden Konfigurationen nicht im Fokus.

Der Aspekt der Wiederverwendung bei der Testfallgenerierung für Produktlinien wird in vielen Beiträgen behandelt. Da diese Thematik nicht Schwerpunkt der vorliegenden

Arbeit ist, sind nur einige repräsentative Beiträge behandelt worden. Für eine detaillierte Beschreibung sei auf [86] verwiesen.

4.2.5 Priorisierung

Bei Methoden der *Priorisierung* bzw. Gewichtung wird, durch Analyse entsprechender Kriterien, den Konfigurationen ein Rang zuordnet. Die Konfigurationen werden folglich mit einer hohen oder niedrigen Test-Dringlichkeit bewertet. Das mit Abstand am meisten verwendete Kriterium, nach dem eine Priorisierung vorgenommen wird, ist die Häufigkeit, mit der eine Produktkonfiguration verkauft wird bzw. geplant ist zu verkaufen [15, 69, 70]. Ebenfalls werden Risiken, die sich aus einer Fehlfunktion der entsprechenden Produktkonfiguration ergeben könnten zur Priorisierung herangezogen [55, 69, 103]. Darüber hinaus existieren Ansätze, welche Kostenmodelle zur Einschätzung des Testaufwandes nutzen [126] oder Fehlermodelle zur Identifikation der wahrscheinlichsten Fehlerquellen aufstellen [74]. Im Kontext der hier präsentierten verwandten Arbeiten wird die Priorisierung als ein Spezialfall einer Konfigurationsselektion verstanden, da mithilfe von Priorisierungskriterien eine Selektionsentscheidung getroffen werden kann.

4.2.5.1 Priorisierungsmethodik von Manicke

In [69] stellt Manicke einen Ansatz zum durchgängigen Variantenmanagement von Fahrzeugfunktionen vor. Hierfür werden die Funktionsvarianten mittels einer definierten Struktur in XML Schema modelliert. Die Priorisierung erfolgt automatisiert durch das Tool *VIS* (Varianten-Informationen-System). Für die Evaluierung der Methodik werden zwei Funktionen bei Fahrzeugen der Dr. Ing. h.c. F. Porsche AG herangezogen. In Abbildung 15 ist der Ablauf der Priorisierungsmethodik schematisch dargestellt.

Die Methodik von *VIS* ist ausschließlich auf die Ebene von Fahrzeugfunktionen im mechatronischen Bereich anwendbar. Für die Modellierung einer Produktlinien als Funktionsfamilie ist eine einheitliche Funktionsschema-Struktur definiert, mit eindeutigen Attributen wie Steuergerät, Sensor, Aktor, Signal usw. Ein solches Modell wird als Superfunktion bezeichnet. Mithilfe von Wahrheitstabellen werden alle gültigen Konfigurationen der entsprechenden Funktionsfamilie ermittelt und daraufhin für den Test in eine Reihenfolge gebracht. Manicke sieht dafür mehrere Priorisierungskriterien vor, welche entweder einzeln oder in Kombination miteinander anwendbar sind:

- Priorisierung nach Verbauhäufigkeit h ,
- Abdeckung von Komponenten und Signalen a_{KS} ,
- Gewichtungen nach:
 - a) Zuverlässigkeit w_z (resultierend aus Analysen der Überlebens- und Ausfallwahrscheinlichkeiten),
 - b) Kundenrelevanz w_{Ku} (anhand des kundennahen Verhaltens),

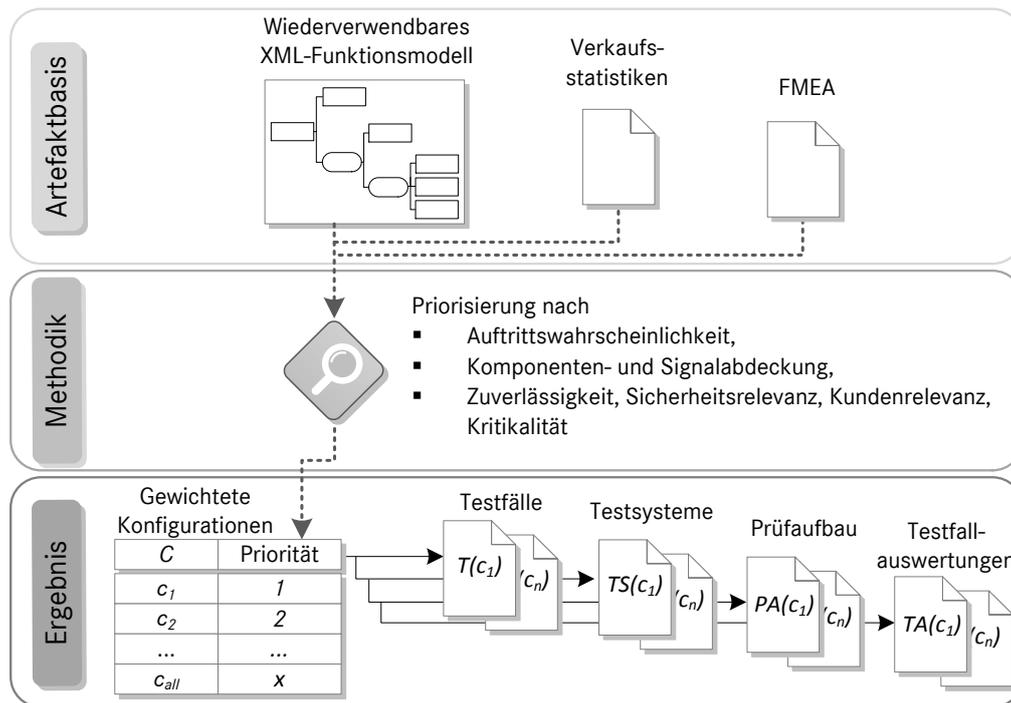


Abbildung 15: Abstrahierte Übersicht über die *Methode der Priorisierung* von Manicke

- c) Sicherheitsrelevanz w_s (basierend auf der Auswirkungsanalyse (engl. *failure mode and effects analysis*, FMEA)) und
- d) Kritikalität w_{K_r} (basierend auf der FMEA)

Um Konfigurationen nach der Verbauhäufigkeit h zu priorisieren, werden Bayes'sche Netze eingesetzt. Mit deren Hilfe lassen sich aus den bekannten Wahrscheinlichkeiten der verkauften Sonderausstattungen konditionale Auftretswahrscheinlichkeiten der Fahrzeugfunktionen errechnen und bei der Priorisierung berücksichtigen. Für die Abdeckung von Komponenten und Signalen α_{KS} wird als Bewertungsmaß die Distanz zwischen einer Funktion und der entsprechenden Superfunktion genutzt. Dabei wird eine niedrige Distanz angestrebt, um eine möglichst hohe Abdeckung durch eine Funktion zu erreichen. Des Weiteren besteht die Möglichkeit zur Gewichtung nach den Kriterien der Zuverlässigkeit, Kundenrelevanz, Sicherheitsrelevanz sowie der Kritikalität einer Fehlfunktion. Abschließend können strukturorientierte Abdeckungskriterien zur Beurteilung der Testvollständigkeit genutzt werden. Für die dadurch ermittelten Konfigurationen lassen sich die passenden Testfälle konfigurieren, dargestellt in Abbildung 15. Dazu werden bspw. Informationen zur Kodierung der Steuergeräte aus dem Variantenmanagement an die sogenannte Pre-Sequenz eines Testfalls übergeben. Durch Verzweigungen innerhalb eines Testfalls werden die Spezifika der jeweiligen Variante aufgerufen.

Bewertung und Relevanz der Priorisierungsmethodik von Manicke

Die Arbeit von Manicke liefert eine Methode zur Erfassung und Priorisierung von Funktions-Produktlinien mit dem Fokus auf E/E-Integrations- und Fahrzeugtests. Durch das vorgegebene Schema zur Funktionsmodellierung wird eine einheitliche Strukturierung jeder Funktionsfamilie sichergestellt. Gleichzeitig ist das Verfahren streng auf Funktionen zugeschnitten, wodurch keine hohe Allgemeingültigkeit des Ansatzes erreicht werden kann, weder über die Teststufen noch die Testobjekttypen von Produktlinien hinweg.

Bei der Auswahl der zu testenden Konfigurationen werden dem Anwender unterschiedliche Priorisierungskriterien zur Verfügung gestellt. Hierbei ist ein nicht unerheblicher initialer Aufwand in der Beschaffung und Bereitstellung der notwendigen Informationen (z.B. Verbauhäufigkeit, Kundenrelevanz usw.) zu berücksichtigen. Liegen die Informationen erst einmal vor, ist mithilfe des Tools *VIS* ein hoher Grad an Automatisierbarkeit erreichbar. *VIS* wurde anhand der Funktionsfamilien Tagfahrlicht mit 24 Konfigurationen und der automatischen Leuchtweitenregulierung mit 8 Konfigurationen evaluiert. Eine Aussage über die Effizienz ist, aufgrund der fehlenden konkreten Angaben zu Einsparungen an Testumfängen sowie der relativ geringen Anzahl an Konfigurationen in den Fallstudien, nur bedingt möglich. Die Konfigurierbarkeit und somit Wiederverwendbarkeit der Testfälle ist umgesetzt.

4.2.5.2 Priorisierungsmethodik von Burgdorf

Burgdorf behandelt in seiner Dissertation [15] einen Ansatz zur Bestimmung abzusichernder Konfigurationen von Sonderausstattungen resultierend aus Prognosen der nach Produktionsstart am häufigsten verkauften Fahrzeuge. Dabei werden die Konfigurationen mithilfe von parametrisierten Eigenschaftsvektoren modelliert. Die Absicherungsstrategie wird anhand der Produktlinie von Bordnetzen bei Fahrzeugen der BMW AG evaluiert. In Abbildung 16 ist das Vorgehen schematisch skizziert.

Die Grundlage der Methodik bildet ein angenommener Zusammenhang zwischen Test- und Feldfahrzeugen. Dieser wird durch eine empirische Korrelationsanalyse bestätigt, bei der Burgdorf feststellt, dass Fahrzeuge mit ähnlicher oder identischer Sonderausstattung (z.B. abgesicherte Prototypen) eine höhere Qualität² während ihrer Lebensdauer aufweisen als solche, die stärker abweichen. Davon ausgehend erfolgt eine sogenannte Varianten-Clusteranalyse, durch welche eine Partitionierung von bereits existierenden Feldkonfigurationen mit ähnlichen Eigenschaften in Cluster vorgenommen wird. Die Selektion repräsentativer Kundenkonfigurationen kann dadurch anhand der Cluster und somit anhand der verkauften Fahrzeuge ausgerichtet werden. Um jedoch eine Bestimmung der abzusichernden Konfigurationen noch *während* des Produktentstehungsprozesses zu ermöglichen, werden zusätzlich wahrscheinlichkeitstheoretische Aussagen gemäß den Absatzerwartungen und mithilfe des Hidden-Markov-Modells gemacht. Das Prognosemodell wird durch statistisches Hochrechnen erstellt und lässt sich als eine mehrdimensionale Bernoulli-Verteilung interpretieren. Zusätzlich dazu selektiert

2 Die Qualität der Feldfahrzeuge wird mittels der entstehenden Service- bzw. Werkstattkosten der jeweiligen Fahrzeugkonfiguration bewertet.

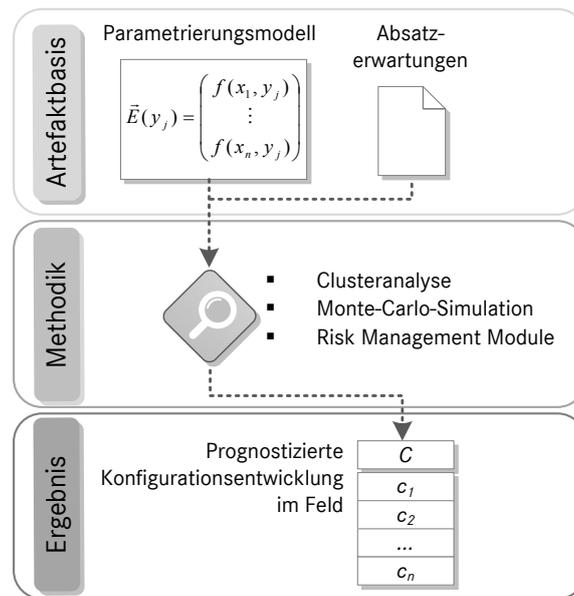


Abbildung 16: Abstrahierte Übersicht über die *Methode der Priorisierung* von Burgdorf

Burgdorf die Steuergeräte, welche die Funktionen der abzusichernden Sonderausstattungen realisieren. Dies wird im sogenannten Risk Management Module durch eine Gegenüberstellung von Risikoreduktion (wiederum auf Basis der oben erwähnten Korrelationsanalyse) zu Absicherungskosten bestimmt, um das Optimum zwischen den beiden Faktoren zu ermitteln. Abschließend wird die Möglichkeit einer vollständigen Prognose der Konfigurationsentwicklung im Feld durch eine Monte-Carlo-Simulation gezeigt. Eine Übersicht über das Vorgehen bietet Abbildung 16.

Bewertung und Relevanz der Priorisierungsmethodik von Burgdorf

Die von Burgdorf vorgestellte Absicherungsstrategie richtet sich nach der Prognose verkaufter Fahrzeugkonfigurationen. Die Berechnung erfolgt mithilfe einer Monte-Carlo-Simulation. Nachteilig bei diesem Verfahren sind der hohe Entwicklungs- und Pflegeaufwand der Simulationsmodelle sowie die oftmals nichttriviale Interpretation der Ergebnisse. Darüber hinaus ist die Aussagekraft des Prognosemodells an möglichst korrekte Eingangsdaten, wie Absatzerwartungen, gebunden, dessen Vorhersage häufig schwer möglich ist.

Die Evaluierung ist anhand der Produktlinie von E/E-Fahrzeugbordnetzen durchgeführt worden. Durch das sehr abstrakte zugrunde liegende Modell lässt sich das Vorgehen ebenfalls auf andere Testobjekttypen übertragen. Die Optimierung ist auf die kundennahe Absicherung ausgerichtet, wodurch sich die Methode auf die Ebene von Integrations- und Fahrzeugtests konzentriert. Toolseitig wird das Vorgehen durch den sogenannten *Virtual Vehicle Simulator (VVS)* unterstützt. Eine konkrete Methode zur Wiederverwendung von Testfällen ist nicht erwähnt.

4.2.6 Kombinatorisches Testen

Kombinatorische Testverfahren (engl. *combinatorial interaction testing*) basieren auf der Annahme, dass Fehler verstärkt an Schnittstellen auftreten, an denen Informationen bzw. Daten ausgetauscht werden. Um möglichst viele Fehler aufdecken zu können, muss somit ein gewisser Grad der Interaktionen abgedeckt werden. Relevante Abdeckungsgrade sind:

- 1-wise: Abdeckung jedes Eingabewertes³ durch mindestens eine Konfiguration,
- Pairwise: Abdeckung aller Paare der Eingabewerte durch mindestens eine Konfiguration,
- t-wise: beliebige t-Tupel Abdeckung der Eingabewerte durch mindestens eine Konfiguration,
- N-wise: Abdeckung jeder möglichen Kombination N von Eingabewerten und damit jeder Interaktion bzw. Konfiguration (entspricht dem *Product by Product Testen*, siehe Abschnitt 4.2.2)

Das kombinatorische Verfahren kommt ursprünglich aus der statistischen Versuchsplanung (engl. *design of experiments*, DOE) [36], welches in den 1920er Jahren von Sir R. A. Fisher vorgestellt wurde [25]. Obwohl dabei nur ein Bruchteil an Konfigurationen getestet wird, weisen empirische Untersuchungen einen hohen Grad an Testabdeckung (engl. *code coverage*) und Fehlerrückmeldung nach. Besonders häufig wird die Pairwise-Methode genutzt [13, 88], aufgrund der starken Komprimierung des zu testenden Konfigurationsraums und einer gleichzeitigen Fehlerrückmeldungsrates von über 70% [14, 57, 73]. Eine ähnliche Idee wird bei der Klassifikationsbaum-Methode zur Testfallgenerierung verwendet. Dabei können die Eingabedaten in Form von Klassen für alle Klassifikationen gemäß verschiedener Abdeckungsmaße zu Testfällen kombiniert werden [7, 40].

4.2.6.1 Kombinatorische Testmethodik von Oster

In [83] präsentiert Oster einen Ansatz, mit welchem entsprechend der Pairwise-Methode eine repräsentative Untermenge an zu testenden Konfigurationen selektiert wird. Die Konfigurationen einer Produktlinie werden hierfür im Merkmalmodell dargestellt. Die Selektion erfolgt automatisiert mittels des Tools *MoSo-PoLiTe* (engl. *model-based software product line testing*). Durch eine Verknüpfung zwischen Merkmalen und einem wiederverwendbaren Testmodell, können produktspezifische Testfälle generiert werden. Die Methodik wird anhand von drei industriellen Fallstudien evaluiert. In Abbildung 17 ist das Vorgehen schematisch veranschaulicht.

Die Basis des kombinatorischen Testens von Oster bildet das Merkmalmodell einer Produktlinie. Um die Methodik auf alle möglichen Merkmalmodelle anwendbar zu machen, wird durch eine Graphentransformation die hierarchische Struktur des Merkmalmodells zunächst in ein binäres Constraint Satisfaction Problem (CSP) übersetzt [89].

³ Der abstrakte Begriff *Eingabewert* kann je nach Zusammenhang als Test-Parameter, Merkmal, Komponente usw. verstanden werden.

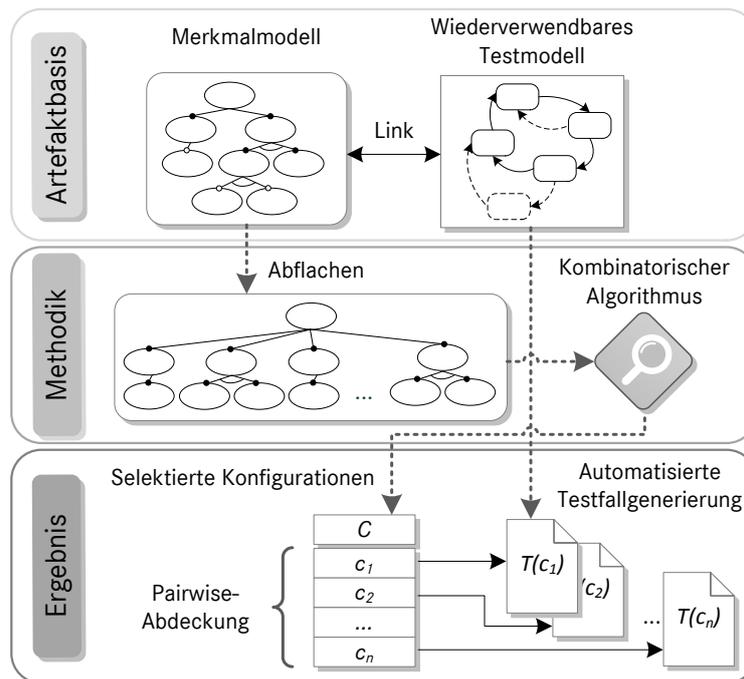


Abbildung 17: Abstrahierte Übersicht über das *Kombinatorische Testvorgehen* von Oster

Dabei wird das Merkmalmodell abgeflacht, sodass es nur noch aus Parametern (engl. *parameters*) und Parameterwerten (engl. *parameter values*) besteht, siehe Abbildung 17. Ein kombinatorischer Algorithmus selektiert die zu testenden Konfigurationen, durch die alle Merkmalpaare abdeckt werden. Gleichzeitig wird mithilfe eines Constraint Solver (Forward Checking) die Validität der ausgewählten Konfigurationen sichergestellt [84]. Die Verknüpfung zwischen Merkmalen und den Elementen eines wiederverwendbaren Testmodells der Produktlinie erlaubt die Generierung von Testfällen für jede selektierte Konfiguration [22]. Das Testmodell liegt in Form eines Zustandsübergangsdiagramms (engl. *state chart*) vor.

Bewertung und Relevanz der Kombinatorischen Testmethodik von Oster

Als Ausgangspunkt der Selektionsmethodik verwendet Oster das Merkmalmodell. Da dies die häufigste Form der Variabilitätsmodellierung darstellt, weist der Ansatz einen hohen Grad an Generalität auf. Dies wird dadurch bestätigt, dass die Fallstudien in verschiedenen Bereichen, nämlich Automotive sowie Leistungselektronik, durchgeführt wurden. Die Effizienz der Pairwise-Methode wird ebenfalls in den Fallstudien deutlich. So werden für ein sogenanntes Body Comfort System 17 Konfigurationen (aus 11616 validen) und für das Kombiinstrument (engl. *industrial panel cluster*) für Fahrzeuge der Opel AG 22 Konfigurationen (aus 19680 validen) selektiert. Die Idee des Pairwise-Testens ist zwar nicht neu, wird jedoch durch das Werkzeug *MoSo-PoLiTe* in der Anwendung praktikabel und automatisiert gemacht [87].

Das Verfahren sieht die Verknüpfung von Variationspunkten mit Testartefakten innerhalb eines Testmodells vor. Der Aufwand zur Erstellung eines wiederverwendbaren Test-

modells für ein Testobjekt aus der Praxis ist sehr hoch. Es wird vom Autor angemerkt, dass eine vollständige Testsuite schwer zu finden sei, wodurch die Wiederverwendung von Testfällen in dieser Arbeit nur für kleinere Testmodellausschnitte nachgewiesen wurde und daher eher konzeptioneller Art bleibt. Die Selektion stützt sich einzig auf die Variabilitätsinformationen aus dem Merkmalmodell und lässt sämtliche weiteren Kriterien unberücksichtigt.

4.2.6.2 Kombinatorische Testmethodik von Cohen

In [25] entwickelt Cohen ein mathematisches Modell zur Selektion von Konfigurationen basierend auf variablen Abdeckungen (engl. *variable interaction strengths*) von Eingabewerten für eine Produktlinie. Die Variabilitätsmodellierung erfolgt sehr vereinfacht auf zwei Ebenen, ähnlich zum orthogonalen Variabilitätsmodell, welches in Kapitel 3.3.3 vorgestellt wurde. Die Evaluierung der Ergebnisse wird im Bezug auf die Effizienz unterschiedlicher Suchalgorithmen durchgeführt. Ein unterstützendes Werkzeug ist nicht erwähnt.

Mithilfe von orthogonalen Matrizen (engl. *covering arrays*) erweitert Cohen die bekannten Methoden der kombinatorischen t -Abdeckungen um die Möglichkeit Konfigurationen zu selektieren, welche unterschiedliche t -Abdeckungen für eine Produktlinie liefern. Dadurch kann bspw. eine Pairwise-Abdeckung für die gesamte Produktlinie und zusätzlich eine Abdeckung mit $t > 2$ für einen ausgewählten Bereich generiert werden. Cohen argumentiert, dass dies notwendig sei, um gezielt Gewichtungen in der Selektion einbauen zu können. Bei der Methode wird von Produktlinien ausgegangen, dessen Variabilität sich auf zwei Ebenen, durch Faktoren k (engl. *factors*) sowie dessen mögliche Werte v (engl. *values*), darstellen lässt. In späteren Beiträgen wird das Verfahren auf orthogonale Variabilitätsmodelle übertragen [26], in welchen zusätzlich die Möglichkeit zur Erfassung von Constraint-Beziehungen mithilfe eines SAT-Solvers besteht [27, 28]. Für die Generierung der Konfigurationen mit unterschiedlichen t -Abdeckungen werden Greedy-Algorithmen, der Bergsteigeralgorithmus (engl. *hill climbing*) sowie die Metaheuristik Simulierte Abkühlung (engl. *simulated annealing*) genauer untersucht. Bei der Analyse erweist sich die Simulierte Abkühlung als das effizienteste Suchverfahren. Eine Übersicht über das Verfahren ist in Abbildung 18 dargestellt.

Bewertung und Relevanz der Kombinatorischen Testmethodik von Cohen

Die Arbeit von Cohen liefert ein Verfahren zur Generierung von Konfigurationen auf Basis von variablen t -Abdeckungen. Das zugrundeliegende mathematische Modell macht die Methodik allgemeingültig für jede Art von Testobjekt sowie jede Testebene. Die bereits bekannte Effizienz des kombinatorischen Testens wird erreicht. Zusätzlich können stärkere t -Abdeckungen für ausgewählte Werte gefordert werden. Allerdings werden weder explizite Gewichtungskriterien genannt, noch nach welcher Methodik diese in die Selektion einfließen.

Nachteilig ist ebenfalls, dass – analog zu Oster (vgl. Abschnitt 4.2.6.1) – keine Informationen herangezogen werden, abgesehen vom Variabilitätsmodell. Der dadurch gegebene Aufwand an die Bereitstellung der Produktlinieninfrastruktur ist gering.

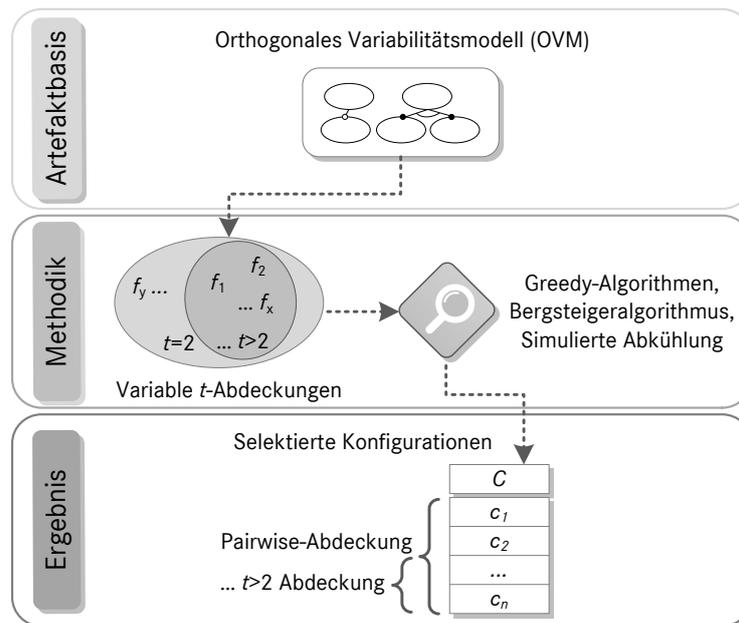


Abbildung 18: Abstrahierte Übersicht über das *kombinatorische Testvorgehen* von Cohen

Gleichzeitig wird keine Testfall-Wiederverwendung ermöglicht. Schließlich kann aufgrund des theoretischen Charakters der Arbeit sowie der fehlenden Toolunterstützung keine einfache Anwendbarkeit gewährleistet werden.

4.2.7 Anforderungsbasierte Selektion

Die Methodik der *Anforderungsbasierten Selektion* sieht Anforderungen als zentrale Grundlage für die Bestimmung der testrelevanten Konfigurationen vor und ist dadurch der Idee der vorliegenden Arbeit am ähnlichsten.

Anforderungsbasierte Selektionsmethodik von Scheidemann

In [100] stellt Scheidemann einen Ansatz zur Selektion der zu testenden Konfigurationen eines Systems basierend auf einer optimierten Abdeckung der Anforderungen sowie der Architekturelemente vor. Hierfür werden Informationen zur Variabilität, welche mittels eines vereinfachten orthogonalen Variabilitätsmodells (engl. *simple orthogonal variability model*, SOVM) modelliert werden, sowie Anforderungen und Informationen zur Architektur der Systeme herangezogen. Die Evaluierung findet anhand von drei Fallstudien im Automotive-Bereich statt. Ein unterstützendes Tool ist nicht erwähnt. Das Vorgehen ist in Abbildung 19 schematisch dargestellt.

Für die Umsetzung der Methode wird die Variabilität einer Produktlinie im SOVM als ein sogenanntes Familienmodell (engl. *family model*) erfasst. Das SOVM bietet eingeschränkte Möglichkeiten zur Abbildung der Semantik zwischen Merkmalen, denn dieses erlaubt nur optionale Merkmale sowie die Constraint-Beziehungen Implikation,

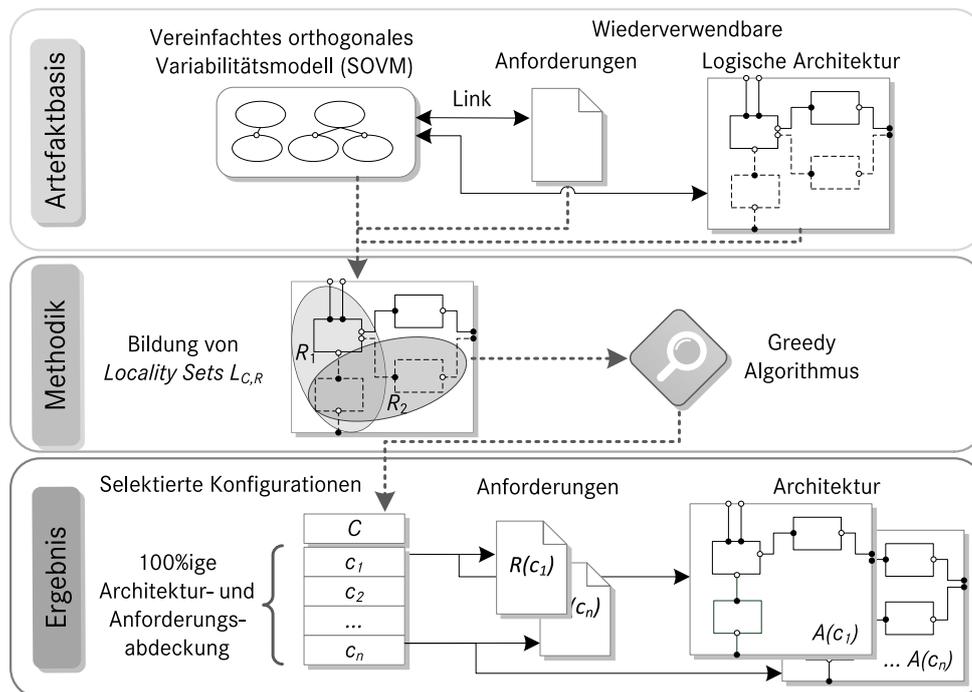


Abbildung 19: Abstrahierte Übersicht über die *Anforderungsbasierte Selektion* von Scheidemann

Exklusion und Kontravalenz (XOR) zwischen diesen. Über Variationspunkte (engl. *variation point*) wird definiert, welche Merkmale für welche Anforderungen und welche Architekturanteile gültig sind. Darüber hinaus werden als Konsequenz der Evolution der Systeme verschiedene Versionen eines Familienmodells angenommen. Um eine optimierte Absicherung aller Versionen des Familienmodells zu gewährleisten, wird zunächst die Menge der Architekturelemente in einer Konfiguration C , welche in einer Anforderung R realisiert wird, zu einem sogenannten *locality set* $L_{C,R}$ zusammengefasst [99]. Auf Basis der $L_{C,R}$ generiert ein Greedy-Algorithmus eine optimierte Selektion der Konfigurationen, welche die Abdeckung aller Anforderungen sicherstellen. Die Idee ist dabei, dass solche Anforderungen, welche in zwei verschiedenen Konfigurationen durch dieselbe Menge der Architekturelemente realisiert werden, d.h. ein gemeinsames $L_{C,R}$ besitzen, in nur einer Konfiguration abgesichert werden müssen.

Bewertung und Relevanz der Anforderungsbasierten Selektionsmethodik von Scheidemann

Die Arbeit von Scheidemann liefert einen anforderungsbasierten Ansatz zur Selektion von zu testenden Konfigurationen auf Systemebene. Die Basis der Selektion bildet die Modellierung des entsprechenden Familienmodells mithilfe des SOVM. Diese Art der Variabilitätsbeschreibung gilt insbesondere bei sehr umfangreichen Produktlinien als recht aufwändig.

Bei der Evaluierung der Methodik stellt Scheidemann für ein kleines Fahrerassistenzsystem (engl. *driver assistance functions*) eine vielversprechende Reduktion von zu testenden

Konfigurationen von rund 30% fest. Eine zweite Fallstudie für ein reales und wesentlich größeres System zeigt, dass aufgrund der Komplexität die Methodik nur schlecht skaliert und somit für große Produktlinien angepasst bzw. erweitert werden muss. Die Konfigurierbarkeit und somit Wiederverwendung von Testfällen wird nicht erwähnt.

4.3 Evaluierung der Methoden

Im vorangegangenen Abschnitt wurden die Methoden des Produktlinientests ausführlich beschrieben und auf deren Relevanz für die vorliegende Arbeit hin analysiert.

Dabei stellt die *Aufteilung nach Verantwortlichkeiten* ein Konzept dar, welches Effizienzen durch eine sinnvolle Aufteilung der Domänen- und Applikationstests auf die Teststufen des V-Modells verspricht (vgl. Abschnitt 4.2.1). Da in der vorliegenden Arbeit der Fokus auf höheren Teststufen und somit nur Applikationstests gelegt ist, hat das Konzept der *Aufteilung nach Verantwortlichkeiten* eine geringere Relevanz.

Bedingt durch die Tatsache, dass eine Methode des Produktlinientests die Wiederverwendung zwischen den Produkten ermöglichen sollte, ist das *Product by Product Testen* (vgl. Abschnitt 4.2.2) ebenfalls als weniger effektiv zu beurteilen. Insbesondere für große Produktlinien führt das Verfahren zu extrem hohen Zeit- und Kostenaufwänden. Im Gegensatz dazu fokussieren sich das *Inkrementelle Testen* (vgl. Abschnitt 4.2.3) sowie der *Modellbasierte Produktlinientest* (vgl. Abschnitt 4.2.4) sehr stark auf die Wiederverwendung von gemeinsamen Artefakten zwischen den Produkten. Allerdings beschäftigen sich beide Methoden weniger intensiv mit der Identifikation von zu testenden Konfigurationen und greifen an dieser Stelle häufig auf andere Methoden zurück.

Solche Methoden, welche sich mit der Selektion von zu testenden Konfigurationen auseinandersetzen, sind in den letzten drei Abschnitten beschrieben. Die darin behandelten Ansätze der *Priorisierung* (vgl. Abschnitt 4.2.5), des *Kombinatorischen Testens* (vgl. Abschnitt 4.2.6) sowie der *Anforderungsbasierten Selektion* (vgl. Abschnitt 4.2.7) sind als besonders bedeutend zu bewerten. Die drei Verfahren haben unter anderem das Ziel, mithilfe von gesetzten Abdeckungskriterien zu einer repräsentativen Menge an Konfigurationen zu gelangen, welche für die Absicherung der gesamten Produktlinie genügt. Dieser Grundgedanke ist dem der vorliegenden Dissertation sehr ähnlich, weshalb die drei genannten Selektionsmethoden im Folgenden genauer betrachtet werden.

Insgesamt weisen die Selektionsmethoden eine starke Heterogenität auf [18]. Diese spiegelt sich zum einen darin wider, auf welche Art und Weise Variabilität beschrieben wird. So werden entweder Merkmalmodelle, die (vereinfachte) orthogonale Variabilitätsmodellierung oder sogar neuartig entwickelte, stark auf den Kontext der Arbeit angepasste, Methoden verwendet. Zum anderen unterscheiden sich die Verfahren darin, welche Artefakte als Basis für eine Methode herangezogen werden. Letzteres bestimmt unmittelbar das Ergebnis, d.h. den Output der Methodik, welcher selektierte, priorisierte, prognostizierte Konfigurationen usw. darstellen kann. Aufgrund dieser Heterogenität müssen einheitliche Kriterien für eine objektive und aussagekräftige Evaluierung der Methoden definiert werden, um deren Vergleichbarkeit zu ermöglichen. Diese setzen sich einerseits aus den in der Literatur identifizierten Anforderungen an eine erfolgreiche Produktlinien-Testmethodik sowie andererseits den eigenen Erkenntnissen zusammen und sind wie folgt definiert:

- **Allgemeingültigkeit** (betrifft die Anwendungsmöglichkeit des Verfahrens auf verschiedene Testobjekttypen (z.B. System, Komponente, Modell))
- **Automatisierbarkeit** (bezieht sich auf die Unterstützung des Verfahrens mithilfe eines Tools)
- **Aufwand an die Produktlinieninfrastruktur** (beschreibt den Aufwand bei der Bereitstellung der wiederverwendbaren Artefakte sowie weiteren für die Methode notwendigen Informationen)
- **Nutzen** (betrifft die Effektivität der von der Methode erzielten und innerhalb von Fallstudien nachgewiesenen Ergebnisse im Bezug auf die selektierten Konfigurationen und die wiederverwendeten Testfälle)
- **Domänen-Testartefakte** (bezieht sich auf die Möglichkeit einer Wiederverwendung von Testfällen)

Alle Selektionsmethoden der vierten Kategorie sind anhand der genannten Kriterien verglichen worden, sofern die notwendigen Informationen vorlagen. Tabelle 5 zeigt eine zusammengefasste Darstellung der Kriterien sowie dessen Zuordnung zu den einzelnen Arbeiten.

	Manicke	Burgdorf	Oster	Cohen	Scheidemann
Fokus (Testobjekttyp)	Funktion	System	alle	alle	System
Automatisierbarkeit (Werkzeug)	VIS	VVS	MoSo-PoLiTe	k.A.	k.A.
Geringer Aufwand (Produktlinieninfrastruktur)	o	-	o	+	o
Nutzen (Fallstudien)	o	k.A.	o	k.A.	o
Wiederverwendung (Testfälle)	+	-	+	-	-

Tabelle 5: Vergleich sowie Evaluierung der verwandten Arbeiten, behandelt in den Abschnitten 4.2.5, 4.2.6 sowie 4.2.7.

Legende: (+) erfüllt das Kriterium, (o) erfüllt das Kriterium teils, (-) erfüllt das Kriterium nicht.

Der Vergleich der Selektionsmethoden in Tabelle 5 zeigt, dass alle Arbeiten Stärken, gleichzeitig aber auch Grenzen in den gegebenen Geltungsbereichen aufweisen. Zentral ist dabei der Aspekt des Nutzens, welcher letztendlich die Güte bzw. die Effektivität einer Methode aufzeigt. Um eine adäquate Aussage über den Nutzen treffen zu können, sind umfangreiche Fallstudien notwendig, welche zum einen die Anwendbarkeit im realen (Industrie-)Umfeld und zum anderen die (Ergebnis-)Effizienz der Methode belegen. Allzu häufig weisen jedoch die Arbeiten eingeschränkte Randbedingungen auf, in welchen die Fallstudien durchgeführt wurden und verhindern somit eine Generalisierung der Ergebnisse. Nicht selten findet auch gar keine Evaluierung im Rahmen einer realen Fallstudie statt.

Dieses Ergebnis wird von weiteren Studien zu Produktlinien-Testmethoden bestätigt [38, 77]. Darin wird argumentiert, dass sich die Mehrzahl der Konzepte und Ansätze häufig noch im Forschungsstadium befinden. Die wenigen Untersuchungen aus der Praxis beziehen sich oft auf sehr kleine Fallstudien oder solche, welche an den spezifischen Kontext *einer* Anwendung angepasst sind. Infolgedessen können die Ergebnisse solcher Studien nicht ohne Weiteres auf andere, umfangreiche Projekte aus dem Industriekontext übertragen werden. Daraus geht hervor, dass deutlich mehr empirische Untersuchungen notwendig sind, um bessere Ergebnisse beim Produktlinientest zu erzielen.

Ein weiteres wesentliches Kriterium, welches für eine erfolgreiche Methodik des Produktlinientests charakteristisch ist, stellt die Verknüpfung der Variationspunkte zu Testartefakten dar. Denn zusätzlich zur selektierten Konfigurationsmenge müssen ebenfalls die Testumfänge angepasst an jede einzelne Konfiguration vorliegen. Auch dieses Kriterium wird nicht von allen Arbeiten erfüllt.

Nicht zuletzt ist die Unterstützung durch ein Werkzeug für eine Selektionsmethodik als wichtig zu erachten, welche ebenfalls nicht bei allen Methoden verfügbar ist. Aufgrund der Konfigurierbarkeit der Produkte, der Artefakte sowie deren Zusammenhänge können komplexe Aufgaben entstehen, dessen manuelle Bearbeitung extrem fehleranfällig, langsam und daher nicht sinnvoll ist. Das systematische Lösen mithilfe eines Tools ist folglich unverzichtbar.

Zusammenfassung

In diesem Kapitel erfolgt eine eingehende Untersuchung der vorhandenen Testmethoden für Produktlinien. Durch die Analyse erfolgt eine Abgrenzung der vorliegenden Arbeit zu den bereits existierenden Beiträgen. Gleichzeitig werden Grenzen sowie Potentiale existierender Verfahren des Produktlinientests identifiziert und aufgezeigt.

Eine wichtige Erkenntnis aus der kritischen Auseinandersetzung ist die notwendige Entwicklung einer adäquaten Produktlinieninfrastruktur. Diese bildet eine einheitliche Grundlage, welche einerseits für alle Beteiligten als Referenzplattform gilt, um ein gemeinsames Verständnis über die Variabilität in Artefakten aufzubauen. Andererseits sollte diese ohne großen Aufwand in die bereits gegebene Dokumentationsstruktur integrierbar sein. Dies ist insbesondere für Testfälle essentiell, da sich in diesen der methodisch festgelegte Testumfang widerspiegelt. Hierzu sollte eine Anpassung der Testspezifikation an das entsprechende Testobjektes durch die Konfigurierbarkeit der Testfälle ermöglicht werden. Ebenso zentral für eine erfolgreiche Testmethodik von Produktlinien ist die Umsetzung und Unterstützung mithilfe eines Werkzeugs. Schließlich ist ein Nachweis der Praxistauglichkeit sowie des Nutzens anhand von Fallstudien im Industriekontext erforderlich.

Ausgehend von der Analyse, werden die gewonnen Erkenntnisse innerhalb der in dieser Arbeit entwickelten Methodik berücksichtigt und die erkannten Potentiale adressiert.

Teil III

OPTIMIERTE VARIANTENSELEKTION FÜR EFFIZIENTEN E/E-PRODUKTLINIENTEST

5 | VARIABILITÄT IN ENTWICKLUNGSARTEFAKTEN

Dieses Kapitel liefert eine Einführung in die Produktlinieninfrastruktur, welche die Grundlage für eine systematische Selektion von zu testenden Konfigurationen bildet. Abschnitt 5.1 diskutiert die allgemeinen Anforderungen bezüglich einer solchen Produktlinieninfrastruktur und gibt einen Überblick über deren wesentliche Bestandteile. In Abschnitt 5.2 wird die Modellierung der Variabilität beschrieben. Die darauf folgende Abbildung der Merkmale in entsprechenden Entwicklungsartefakten wird in Abschnitt 5.3 eingehend behandelt. Der Schwerpunkt des Kapitels liegt auf dem letzten Abschnitt 5.4, in welchem eine Methodik zur Sicherstellung einer konsistenten Merkmalabbildung zwischen Entwicklungsartefakten ausgearbeitet wird. Die mithilfe der *Konsistenzprüfung* unterstützte und somit widerspruchsfreie Variabilitätsabbildung in Anforderungen und Testfällen bildet die Basis für die nachfolgende Selektionsmethodik.

5.1 Überblick über die Produktlinieninfrastruktur

Der Ansatz der Produktlinienentwicklung für hoch variable Systeme unterstützt ein systematisches und proaktives Wiederverwenden von Entwicklungsartefakten (vgl. Kapitel 3). Die hierfür notwendige konkrete Produktlinieninfrastruktur wird in diesem Kapitel schrittweise aufgebaut und ist in Abbildung 20 veranschaulicht. Die daraus resultierende einheitliche und durchgängige Artefaktbasis bildet eine zwingende Voraussetzung für die anschließende systematische Bestimmung der Konfigurationen für den Test (vgl. Kapitel 6).

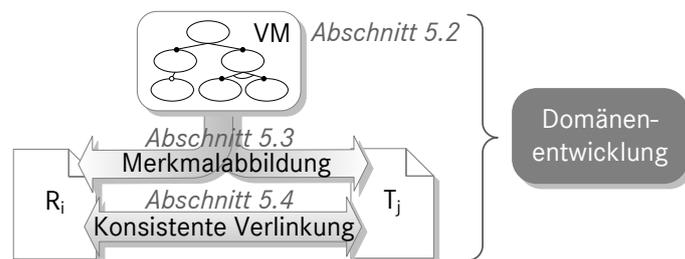


Abbildung 20: Zusammenhang zwischen dem Variabilitätsmodell (VM), Anforderungen (R_i) und Testfällen (T_j) sowie eine Übersicht über die jeweils diskutierten Elemente der Produktlinieninfrastruktur innerhalb der entsprechenden Abschnitte des vorliegenden Kapitels

Der in der vorliegenden Arbeit verwendeten Produktlinieninfrastruktur liegt das Konzept eines sogenannten universellen Variantenmanagements [6] zugrunde. Zu diesem

Zweck wird *ein gemeinsames* Merkmalmodell, für ein bestimmtes Bezugsobjekt, d.h. eine Produktlinie, über die verschiedenen Phasen des Entwicklungsprozesses hinweg (wieder-)verwendet. Auf diese Weise wird es ermöglicht die im Merkmalmodell dargestellte Variabilität in Anforderungs- und Testspezifikationen abzubilden und somit die Artefaktbasis bzw. Plattform der Domänenentwicklung (vgl. Abschnitt 3.2) bereitzustellen. Dieser Zusammenhang ist in Abbildung 20 schematisch aufgezeigt. Auf Basis der resultierenden wiederverwendbaren Spezifikationsdokumente (Domänenartefakte) können alle produktspezifischen Applikationsartefakte mithilfe des Variabilitätsmodells konfiguriert und abgeleitet werden. Auf die methodische Ausleitung von produktspezifischen Entwicklungsumfängen wird im nächsten Kapitel 6 näher eingegangen.

Die Anwendung eines universellen Variabilitätsmodells bringt eine Reihe von Vorteilen mit sich. So wird das Verständnis bezüglich einer Produktlinie für verschiedene Beteiligte aus diversen Entwicklungsphasen, wie dem Requirements Engineer, Entwickler oder Tester, vereinheitlicht und dadurch verbessert. Zusätzlich wird der Dokumentations-, Verwaltungs- und damit der Entwicklungsaufwand erheblich reduziert.

Gleichzeitig bringt ein solches Konzept besondere Herausforderungen mit sich. Aufgrund des generischen Charakters der Spezifikationsdokumente würden sich potentielle Inkongruenzen aus den Domänen- in die Applikationsartefakte fortpflanzen. Um daher Fehler, die höchstwahrscheinlich mehr als nur ein Produkt betreffen würden, zu vermeiden, ist es notwendig die Variabilität auf eine konsistente Art und Weise in Entwicklungsartefakten abzubilden. Darüber hinaus könnten Inkonsistenzen Einfluss auf die nachfolgenden Entwicklungsphasen haben. Würde bspw. die integrative Testphase auf Basis einer inkonsistenten Testspezifikation durchgeführt werden, könnten die Testergebnisse zu Widersprüchen oder sogar Fehlern in der Produktion führen. Schließlich, erlaubt eine Konsistenzprüfung im Falle von Änderungen im Variabilitätsmodell oder innerhalb der Domänenartefakte, eine systematische und effiziente Unterstützung der Evolution und Organisation einer Produktlinie.

Im Folgenden werden die expliziten Elemente der Produktlinieninfrastruktur vorgestellt, formalisiert sowie die zur Sicherstellung der Konsistenz notwendige Prüfung präsentiert. Um die Zusammenhänge wirkungsvoller zu verdeutlichen, wird eine fiktive Produktlinie als fortlaufendes Beispiel verwendet.

5.2 Merkmalmodellierung im Variantenmodul

Zur Darstellung der Variabilität werden bei der Daimler AG Merkmalmodelle verwendet. Die Dokumentation und Verwaltung der Merkmalmodelle wird mithilfe des bereits in Abschnitt 2.3 vorgestellten Werkzeugs – DOORS – unterstützt. Darin wird ein Merkmalmodell in Form eines eigenständigen (*Variante(n)-Moduls*) angelegt [11]. Dies bietet den Vorteil, dass sich entsprechende Entwicklungsartefakte, nämlich Anforderungen und Testfälle, ebenfalls in DOORS befinden und sich ohne zusätzlichen tooltechnischen Aufwand an das Merkmalmodell anknüpfen lassen. Dabei ist das Variantenmodul nach

einer vorgegebenen Struktur, die in Abbildung 21 anhand der fiktiven Produktlinie *Klimatisierung*¹ dargestellt ist, aufgebaut.

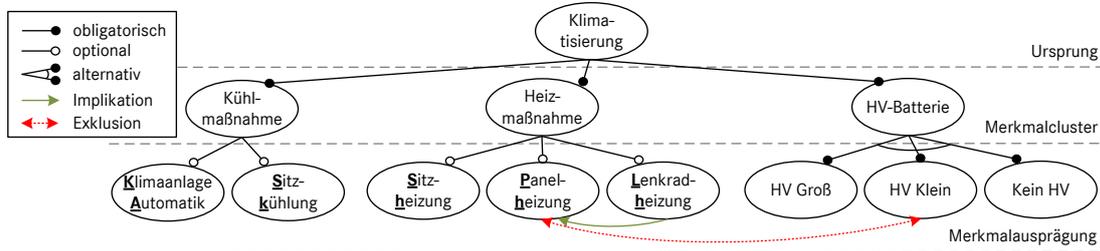


Abbildung 21: Merkmalsmodell mit der zugrundeliegenden Struktur eines Variantenmoduls für das beispielhafte System *Klimatisierung*

Das allgemeine Variantenmodul besteht aus einer Menge von *Merkmalen*, *Abhängigkeiten* sowie *Constraint-Beziehungen*, welche in Abschnitt 3.3.2 thematisiert wurden. Die Merkmale des Variantenmoduls sind in drei Ebenen unterteilt. Das Merkmal der obersten Ebene beschreibt die Produktlinie und stellt gleichzeitig den Ursprungsknoten (engl. *root*) des Modells dar. Die darunterliegenden Kindermerkmale kennzeichnen die *Merkmalscluster* und sind durch obligatorische Merkmale f_{obl} (vgl. Tabelle 2) repräsentiert. Die unterste Ebene spiegelt die eigentlichen variierenden Eigenschaften einer Produktlinie mithilfe der *Ausprägungen der Merkmalscluster* wider. Letztere können entweder mittels optionaler Merkmale f_{opt} oder einer alternativen Gruppe F_{alt} (vgl. Tabelle 2) dargestellt werden. In diesem Zusammenhang wird im Folgenden auch von (Merkmalausprägungen der) optionalen und alternativen Merkmalscluster(n) gesprochen. Gleichzeitig können die Implikation sowie die Exklusion (vgl. Tabelle 3) zwischen den Merkmalausprägungen den Konfigurationsraum weiter eingrenzen. Die aussagenlogische Beschreibung der Abhängigkeiten und Constraints in einem Variantenmodul ist in Tabelle 6 anhand der Produktlinie *Klimatisierung* gezeigt.

Aus dem alternativen Merkmalscluster *HV-Batterie* muss exakt eine Merkmalausprägung in jeder Konfiguration vorhanden sein. Dagegen können beliebige Kombinationen der optionalen *Kühl-* und *Heizmaßnahmen* pro Konfiguration enthalten sein. Aufgrund der Struktur des Variantenmoduls ist die Konfigurierbarkeit nur auf unterster Ebene der Merkmalausprägungen gegeben. Das Variantenmodul kann somit als eine Menge der in Tabelle 6 aufgeführten aussagenlogischen Ausdrücke verstanden werden. Folglich ist der gesamte Konfigurationsraum KR für das System *Klimatisierung* aus Abbildung 21 aufgespannt durch:

$$\begin{aligned}
 KR(\text{Klima}) = & [(KA) \dot{\vee} (\neg KA)] \wedge [(SK) \dot{\vee} (\neg SK)] \wedge \\
 & [(SH) \dot{\vee} (\neg SH)] \wedge [(PH) \dot{\vee} (\neg PH)] \wedge [(LH) \dot{\vee} (\neg LH)] \wedge \\
 & [(HV \text{ Groß}) \dot{\vee} (HV \text{ Klein}) \dot{\vee} (\text{Kein HV})] \wedge \\
 & [(\neg LH) \vee (PH)] \wedge \\
 & [(\neg PH) \vee (\neg HV \text{ Klein})]
 \end{aligned} \tag{5.1}$$

¹ Das Variantenmodul für das System *Klimatisierung* bezieht sich auf das Merkmalsmodell aus Abbildung 9. Aufgrund von Demonstrationszwecken ist das hier präsentierte Variantenmodul leicht angepasst worden.

Abhängigkeit, Constraint	Graphische Notation	Logischer Ausdruck
optionales Merkmalcluster		$[(SH) \dot{\vee} (\neg SH)] \wedge [(PH) \dot{\vee} (\neg PH)] \wedge [(LH) \dot{\vee} (\neg LH)]$
alternatives Merkmalcluster		$(HV\text{ Groß}) \dot{\vee} (HV\text{ Klein}) \dot{\vee} (Kein\text{ HV})$
Implikation		$(\neg LH) \vee (PH)$
Exklusion		$(\neg PH) \vee (\neg HV\text{ Klein})$

Tabelle 6: Logische Definition der Constraint-Beziehungen und Abhängigkeiten eines Variantenmoduls anhand des Systems *Klimatisierung* aus Abbildung 21. Für die Darstellung der logischen Kontravalenz (XOR) wird das Symbol ($\dot{\vee}$) verwendet.

Jede mögliche Lösung der Formel (5.1) ist als eine valide Konfiguration c_v des Konfigurationsraumes zu interpretieren. Als Beispiel lässt sich ein Produkt der fiktiven A-Klasse, W 176, anführen, mit der Konfiguration:

$$c_v(W\ 176) = (KA) \wedge (\neg SK) \wedge (SH) \wedge (PH) \wedge (\neg LH) \wedge (Kein\text{ HV}) \quad (5.2)$$

Diese Produktkonfiguration der A-Klasse enthält die *Klimaanlage Automatik*, *Sitzheizung*, *Panelheizung* (jedoch keine *Sitzkühlung* sowie keine *Lenkradheizung*) und hat *keine HV-Batterie* verbaut.

Auf den ersten Blick bildet die Struktur des Variantenmoduls einen einschränkenden Rahmen. Jedoch lässt sich die in Abschnitt 3.3.2 behandelte Semantik eines allgemeinen Merkmalmodells und die eines Variantenmoduls vereinheitlichen. In [83] zeigt Oster, dass jedes Merkmalmodell mithilfe einer sogenannten Abflachung in eine einheitliche Struktur überführt werden kann (vgl. Abschnitt 4.2.6.1). Hierfür werden für alle möglichen Permutationen der Eltern- und Kinderabhängigkeiten sowie für die Constraint-Beziehungen semantische Regeln definiert, nach denen die Merkmale bei der untersten Ebene angefangen hochgezogen werden. Dieses bewirkt eine Vereinheitlichung jedes Modells, welches schließlich aus drei Ebenen aufgebaut ist. Die erste Ebene bildet den Ursprung, die zweite wird durch obligatorische Merkmale repräsentiert, unter denen in der dritten Gruppe entweder ebenfalls obligatorische Merkmale oder die eigentlichen variierenden Eigenschaften mithilfe von alternativen Gruppen sowie gegebenenfalls Implikationen und Exklusionen definiert sind. Da das Variantenmodul

ein mögliches Merkmalmodell ist, lässt es sich somit mithilfe der Abflachung auf die gleiche semantische Struktur vereinheitlichen.

5.3 Merkmalabbildung in Entwicklungsartefakten

Um aus der Plattform der Domänenentwicklung die nachfolgenden Applikationsartefakte für die einzelnen Produkte ableiten zu können, wird die im Variantenmodul modellierte Variabilität in den Entwicklungsartefakten abgebildet. Der Kontext der vorliegenden Arbeit umfasst zum einen die in natürlicher Sprache verfassten Anforderungsartefakte in entsprechenden Lastenheften sowie zum anderen die ebenfalls natürlichsprachlichen Testartefakte in korrespondierenden Testspezifikationen. Die zentrale Grundidee ist dabei für *eine* Produktlinie, d.h. also *ein* Bezugsobjekt, auch *ein* Variantenmodul zu verwenden. Dass der Einsatz einer einheitlichen Quelle mit entsprechenden Variabilitätsinformationen sowohl für Anforderungen als auch Testfälle möglich ist, ist bei der Daimler AG für mindestens zwei Systeme gezeigt worden (vgl. Kapitel 7). Der Übertrag auf weitere Entwicklungsphasen (z.B. modellbasierte Entwicklung) wird unter anderem im Rahmen des Forschungsprojektes SPES_XT betrachtet [41, 110].

Die Abbildung der Variabilität in Domänenartefakten erfolgt über die im Variantenmodul definierten Merkmalcluster. Dazu werden die Merkmalcluster als Spalten in das entsprechende Dokument importiert, sodass innerhalb dieser Spalten die geltenden Merkmalausprägungen für jedes Artefakt ausgewählt werden können. Das Zuordnen von Merkmalausprägungen zu den Artefakten wird im Folgenden auch als *Merkmal-mapping* bezeichnet. Ein solches Merkmalmapping ist ursprünglich zur Anwendung in Anforderungsspezifikationen ausgearbeitet worden [11]. Im Rahmen der vorliegenden Arbeit wird der geeignete Einsatz des Konzeptes ebenfalls im Rahmen der Testspezifikation geprüft und bestätigt.

In Tabelle 7 und 8 ist ein beispielhaftes Merkmalmapping innerhalb eines Lastenheftes sowie einer Testspezifikation dargestellt. In beiden Dokumenten sind die Merkmalcluster *Kühlmaßnahme*, *Heizmaßnahme* sowie *HV-Batterie* als Spalten eingefügt sowie innerhalb der Spalten entsprechende Merkmalausprägungen auf ein Artefakt abgebildet. Wird innerhalb einer Spalte keine Auswahl an Merkmalausprägungen getroffen, gelten für dieses Artefakt alle Ausprägungen des entsprechenden Merkmalclusters.

Grundsätzlich besteht im Lastenheft und in der Testspezifikation die Möglichkeit jede Zeile, d.h. jeden Objekttyp, mit Merkmalausprägungen zu attributieren. Dadurch können Merkmale nicht nur auf Anforderungen und Testfälle, sondern auch Überschriften, Zusatzinformationen, Basisszenarien usw. abgebildet werden. So sind der Information AS-1, den Signalen AS-4, AS-5 im Lastenheft (vgl. Tabelle 7) und den Testschritten TS-2, TS-3, TS-4 in der Testspezifikation (vgl. Tabelle 8) Merkmale zugeordnet. Die Informationsergänzung der Zeilen erlaubt es, ein einheitliches Konstrukt der Produktlinieninfrastruktur aufzubauen, welches die schnelle Konfigurierbarkeit von gesamten Entwicklungsdokumenten ermöglicht.

Die Semantik zwischen den zugeordneten Merkmalen *eines* Artefakts, welche als Assoziationslogik bezeichnet wird, spielt beim Merkmalmapping eine wesentliche

ID	Ebene	Objekttyp	Beschreibung	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie
AS-0	3	Anforderung	Die HV-Batterie muss die Energie für die Klimatisierung zur Verfügung stellen.			HV Groß HV Klein
AS-1	1	Information	Das System Klimatisierung umfasst alle Heizmaßnahmen im Fahrzeug.		Panelheizung Sitzheizung Lenkradheizung	
AS-2	2	Überschrift	Funktionsausführung Heizung.		Panelheizung Sitzheizung Lenkradheizung	
AS-3	3	Anforderung	Ist $T_{ist} < T_{soll}$ muss die Panelheizung eingeschaltet werden.		Panelheizung	
AS-4	3	Signal-gesendet	Klima-Master sendet Signal PH-ein an Panelheizung.		Panelheizung	
AS-5	3	Signal-empfangen	Panelheizung empfängt Signal PH-ein von Klima-Master.		Panelheizung	

Tabelle 7: Prinzip des Merkmalmappings innerhalb eines Auszugs aus einer beispielhaften Anforderungsspezifikation

Rolle. In den Spezifikationsdokumenten werden die Merkmalausprägungen *eines* Merkmalclusters über die nicht-ausschließende Disjunktion (ODER-Operator) und die Merkmalausprägungen *zwischen* verschiedenen Merkmalclustern über die Konjunktion (UND-Operator) verknüpft. Dieser Zusammenhang bewirkt eine Assoziationslogik in konjunktiver Normalform (KNF). Folglich drückt das Merkmalmapping MM für Anforderung AS-3 in Tabelle 7 aus²:

$$MM(AS-3) = [(KA) \vee (SK)] \wedge (PH) \wedge [(HV \text{ Groß}) \vee (Kein HV)] \quad (5.3)$$

Demnach ist AS-3 für jede Konfiguration, welche mindestens das Merkmal *Panelheizung* enthält, gültig.

Unter Verwendung des Merkmalmappings werden alle Artefakte mit der Information gekennzeichnet, ob diese generischen Charakters, d.h. für die gesamte Produktlinie oder produktspezifisch sind. Folglich umfassen die Spezifikationsdokumente alle Produktspezifikationen der korrespondierenden Produktlinie. Jede Produktspezifikation kann durch die Selektion der relevanten Merkmale im Bezug auf eine bestimmte Produktkonfiguration ausgeleitet werden. Für die im vorherigen Abschnitt genannte Produktkonfiguration 5.2 einer fiktiven A-Klasse, W 176, wären demnach die Artefakte AS-1 bis AS-5 im Lastenheft (vgl. Tabelle 7) und alle Artefakte der Testspezifikation (vgl. Tabelle 8) gültig.

Eine zu der hier vorgestellten alternative Assoziationslogik, welche ausschließlich die Disjunktion zur Verknüpfung von zugeordneten Merkmalen bei allgemeinen Merkmalmodellen verwendet, wird in Anhang A vorgestellt.

² In dem Ausdruck ist (*HV Klein*) nicht aufgeführt, da sich (*PH*) und (*HV Klein*) laut Variantenmodul aus Abbildung 21 gegenseitig ausschließen.

ID	Ebene	Objekttyp	Beschreibung	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie
TS-0	1	Basisszenario	(1) Bereich vorne links heizen voreingestellt		Sitzheizung Panelheizung Lenkradheizung	
TS-1	1	Testfall	Funktionsausführung Heizung		Sitzheizung Panelheizung Lenkradheizung	
TS-2	2	Testschritt	Initialisierung		Sitzheizung Panelheizung Lenkradheizung	
TS-3	2	Testschritt	Heizmaßnahmen ein		Sitzheizung Panelheizung Lenkradheizung	
TS-4	2	Testschritt	Heizmaßnahmen aus		Sitzheizung Panelheizung Lenkradheizung	

Tabelle 8: Prinzip des Merkmalmappings innerhalb eines Auszugs aus einer beispielhaften Testspezifikation (Aus Gründen der Übersichtlichkeit wird in der Darstellung auf die Elemente „Aktion“ und „Nachbedingung“ der Testspezifikation verzichtet)

5.4 Konsistenzmanagement

Wie bereits zu Beginn des Kapitels motiviert, ist bei Verwendung der produktübergreifenden Spezifikationen eine hohe Wiederverwendungsquote der darin enthaltenen Artefakte erreichbar. Gleichzeitig müssen die Dokumente besonders sorgfältig gepflegt werden, da aus den generischen Domänenartefakten alle Applikationsartefakte abgeleitet werden. Fehler würden sich nicht nur auf eine, sondern sehr wahrscheinlich auf mehrere Produktspezifikationen ausbreiten. Aus diesem Grund wird ein *Konsistenzmanagement für die Variabilität* eingeführt.

Der Fokus eines allgemeinen Variabilitäts-Konsistenzmanagements lässt sich in drei Kategorien unterteilen:

1. Konsistenz der Variabilität innerhalb des Variabilitätsmodells,
2. Konsistenz der Variabilität innerhalb von Artefakten *einer* Entwicklungsphase (z.B. in Anforderungen) und
3. Konsistenz der Variabilität zwischen Artefakten verschiedener Entwicklungsphasen (z.B. zwischen Anforderungen und Testfällen).

Die erste Kategorie hat zum Ziel, mögliche Inkonsistenzen in der Semantik eines Variabilitätsmodells aufzudecken. Eine Inkonsistenz kann bspw. durch eine Implikation zwischen zwei Merkmalen einer alternativen Merkmalgruppe gegeben sein. Daraus würde resultieren, dass alle Konfigurationen, welche das Merkmal mit der ausgehenden Implikation enthalten, nicht valide wären. Auch Merkmale, die im Modell nie erreichbar sind und somit „tote“ Elemente darstellen, gehören zu Unstimmigkeiten [72]. Darüber hinaus existieren Ansätze, welche die Variabilität einer Produktlinie aufgeteilt in verschiedene Merkmalmodelle betrachten und die automatisierte Fortpflanzung von

logischen Constraints aus dem einen in das andere Variabilitätsmodell unterstützen [52]. Im weitesten Sinne können sämtliche Analysen von Variabilitätsmodellen zu deren qualitätsverbessernden Methoden gezählt werden [5]. Eine solche Überprüfung der Konsistenz ist innerhalb des in Abschnitt 5.2 beschriebenen Variantenmoduls in DOORS bereits im Einsatz. Die Prüfung analysiert das Modell hinsichtlich der Widersprüche in der Semantik. Darüber hinaus prüft sie die semantische Validität der definierten Konfigurationen, welche ebenfalls im Variantenmodul definiert werden.

Die zweite Kategorie des Konsistenzmanagements untersucht die Variabilität innerhalb eines Artefakttyps. Mögliche Inkongruenzen werden durch Verletzung der im Variabilitätsmodell definierten Abhängigkeiten und Constraint-Beziehungen hervorgerufen. Sind bspw. einem Artefakt zwei sich ausschließende Merkmale zugeordnet, handelt es sich um ein nicht-valides Merkmalmapping [125]. Eine Folge davon wäre, dass das entsprechende Artefakt in keiner Konfiguration berücksichtigt wird. Auch bei der Ausleitung der Applikationsartefakte muss die Konsistenz eingehalten werden, damit etwa keine widersprüchlichen Artefakte Teil *einer* Produktspezifikation werden [60]. Für die in Abschnitt 5.3 vorgestellten Spezifikationsdokumente existiert eine Prüfungsmöglichkeit des Merkmal mappings bezüglich der semantischen Korrektheit. Dies ist notwendig, um sicherzustellen, dass es sich bei den konjunktiv verknüpften Merkmalen nicht um exklusive Merkmale (definiert im Variantenmodul) handelt.

Die meisten Beiträge in der Literatur konzentrieren sich auf Analysemethoden der ersten beiden Kategorien. Darüber hinaus existieren Arbeiten, welche sich mit der generellen Durchgängigkeit (engl. *traceability*) von Variabilität auf Basis eines universellen Variantenmanagements über verschiedene Entwicklungsphasen hinweg oder sogar den gesamten Entwicklungszyklus beschäftigen [6, 101]. Allerdings gibt es nach bestem Wissen keine Methode, die die Konsistenz des Merkmal mappings zwischen verschiedenen Entwicklungsphasen explizit untersucht und damit die Durchgängigkeit unterstützt.

Da Artefakte unterschiedlicher Entwicklungsphasen miteinander zusammenhängen oder sogar aufeinander aufbauen, müssen für eine konsistente Produktlinieninfrastruktur die Verknüpfungen zwischen den Artefakten geprüft werden. So wird bei der Testfallermittlung durch Verlinkungen gekennzeichnet, welche Testfälle aus welchen Anforderungen entstehen (vgl. Abschnitt 2.5.2). Soll durch einen Testfall eine Funktion geprüft werden, so muss ein Link zu der bzw. den Anforderungen gesetzt werden, welche dieselbe Funktion definieren. Die Verlinkungen geschehen ausschließlich zwischen Testfällen³ auf der einen sowie Anforderungen und gesendeten/empfangenen Signalen auf der anderen Seite, wobei letztere eine Definition entsprechender Anforderung auf Signalebene liefern. Demnach müssten in den beispielhaften Spezifikationen aus Abschnitt 5.3 die Anforderungen AS-3, AS-4 und AS-5 (vgl. Tabelle 7) mit dem Testfall TS-1 (vgl. Tabelle 8) verlinkt werden. Allerdings sind für eine vollständige Abdeckung des Testfalls TS-1 weitere Verlinkungen zu Anforderungen notwendig, welche das Ein- und Ausschaltverhalten der *Sitz- und Lenkradheizung* sowie das Ausschaltverhalten der *Panelheizung* charakterisieren.

Aus dem Beispiel geht hervor, dass die Verlinkungen keine eindeutigen 1 : 1-Beziehungen zeigen, sondern in $n : m$ -Verknüpfungen resultieren. Ein Testfall deckt in der Regel

³ Die Tatsache, dass die Verlinkungen nicht zwischen einzelnen Testschritten und Anforderungsartefakten zugelassen ist, entspricht einer Festlegung durch die Daimler AG.

eine Gruppe von Anforderungen ab und eine Anforderung bezieht sich in den meisten Fällen auf mehr als einen Testfall. Die dadurch entstehende Komplexität muss bei der Überprüfung der Konsistenz berücksichtigt werden. Im Folgenden wird ein solches Konzept zur Herstellung der Variabilitäts-Konsistenz zwischen Anforderungen und Testfällen vorgestellt. Hierfür wird eine Kategorisierung möglicher Inkongruenzen eingeführt und deren semi-automatische Behebung ausgearbeitet. Die Automatisierung des Vorgehens mithilfe eines Tools wird in Abschnitt 7.1 näher erläutert.

5.4.1 Konsistenzprüfung

Im Folgenden wird die zur Analyse der Variabilität zwischen zusammenhängenden Anforderungs- und Testartefakten notwendige *Konsistenzprüfung* sukzessive aufgebaut. In Abbildung 22 ist ein Überblick über die einzelnen Schritte der Prüfung dargestellt. In Abschnitt 5.4.1.1 wird zunächst die *Basis* der Prüfung eingeführt. In Rahmen dieser

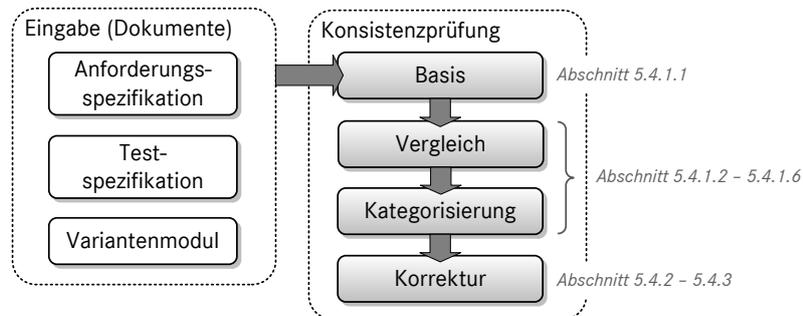


Abbildung 22: Überblick über die aufeinanderfolgenden Schritte der Konsistenzprüfung unter Angabe der Abschnitte, in denen entsprechende Elemente diskutiert werden

Phase werden die für die Prüfung notwendigen Informationen bezüglich der Variabilität aus den Spezifikationen sowie dem Variantenmodul einheitlich aufbereitet. Die nächsten beiden Schritte, nämlich der *Vergleich* sowie die *Kategorisierung*, kennzeichnen den Schwerpunkt der Konsistenzprüfung. Beide Phasen werden in Abschnitten 5.4.1.2 bis 5.4.1.6 detailliert behandelt und anhand von ausgewählten Beispielen veranschaulicht. Der letzte Schritt beschreibt die Behebung von möglichen auftretenden Inkonsistenzen. Dazu werden in Abschnitt 5.4.2 die für das Entstehen der Inkonsistenzen verantwortlichen Ursachen untersucht und in Abschnitt 5.4.3 explizite *Korrekturmaßnahmen* definiert.

5.4.1.1 Basis der Konsistenzprüfung

Bevor auf die Konsistenzprüfung genauer eingegangen wird, wird die *Grundlage*, auf der die Prüfung basiert, erläutert. Seien die Merkmalausprägungen Merkmalmengen F_{MC_z} der jeweiligen Merkmalcluster MC_z . Aufgrund der konjunktiven Normalform der Assoziationslogik ist die Einschränkung gegeben, dass in den Artefakten explizit

nur das kartesische Produkt aller Merkmalmengen $F_{MC_1} \times \dots \times F_{MC_n}$ eines Variantenmoduls abgebildet werden kann (vgl. das exemplarische Merkmalmapping 5.3). Entsprechend gilt für das System *Klimatisierung* aus Abbildung 21:

$$\begin{aligned} &F_{Kühlm.} \times F_{Heizm.} \times F_{HV-Batterie}, \text{ mit} \\ &F_{Kühlm.} = \{(KA), (SK)\}, \\ &F_{Heizm.} = \{(SH), (PH), (LH)\} \text{ und} \\ &F_{HV-Batterie} = \{(HV \text{ Groß}), (HV \text{ Klein}), (Kein HV)\} \end{aligned} \quad (5.4)$$

Daraus resultieren insgesamt 18 Konfigurationsausdrücke c_x , wovon 14 *valide* und in Tabelle 9 dargestellt sind. Für jedes Merkmal f_y , welches in der Konfiguration enthalten ist, wird der Eintrag 1 vorgenommen. Für die Abdeckung der Merkmale durch jeweils eine Konfiguration $F_{cov}(c_x)$ gilt somit:

$$\text{cov}(c_x, f_y) = \begin{cases} 1 & \text{falls } f_y \in F_{cov}(c_x) \\ \emptyset & \text{sonst.} \end{cases} \quad (5.5)$$

c_x	$\text{cov}(c_x, f_1),$ $f_1 = KA$	$\text{cov}(c_x, f_2),$ $f_2 = SK$	$\text{cov}(c_x, f_3),$ $f_3 = SH$	$\text{cov}(c_x, f_4),$ $f_4 = PH$	$\text{cov}(c_x, f_5),$ $f_5 = LH$	$\text{cov}(c_x, f_6),$ $f_6 = HV \text{ Groß}$	$\text{cov}(c_x, f_7),$ $f_7 = HV \text{ Klein}$	$\text{cov}(c_x, f_8),$ $f_8 = \text{Kein HV}$
c_1	1		1			1		
c_2	1		1				1	
c_3	1		1					1
c_4	1			1		1		
c_5	1			1				1
c_6	1				1	1		
c_7	1				1			1
c_8		1	1			1		
c_9		1	1				1	
c_{10}		1	1					1
c_{11}		1		1		1		
c_{12}		1		1				1
c_{13}		1			1	1		
c_{14}		1			1			1

Tabelle 9: Abbildbare Konfigurationen c_x für das System *Klimatisierung* durch das Merkmalmapping in konjunktiver Normalform (gemäß 5.5). Die Berechnung erfolgt mittels des kartesischen Produktes von F_{MC_z} .

Grundsätzlich ergeben sich aus dem Kreuzprodukt der Clusterausprägungen 18 Konfigurationsausdrücke. Allerdings verletzen vier davon die im Variantenmodul definierten Exklusionen (bspw. c_x , welche die sich ausschließenden Merkmale *PH* und *HV Klein* enthalten). Weitere Zuordnungen, welche sich implizit über die vom Merkmalmodell vorgegebenen Abhängigkeiten und Constraints ergeben, werden dagegen nicht berücksichtigt (vgl. mit Tabelle 11, in der die implizite Semantik des Variantenmoduls erfasst ist). Der Grund dafür liegt darin, dass die semantische Korrektheit des Merkmalmapping, welche mithilfe der zweiten Kategorie des Konsistenzmanagements geprüft und hergestellt wird, als gegebene Voraussetzung angenommen wird. Weiterhin sind in

einigen Feldern der Tabelle 9 keine expliziten Angaben gemacht, ob ein Merkmal in der jeweiligen Konfiguration enthalten ist oder nicht. In dieser Darstellung entsprechen c_x lediglich dem aus dem Merkmalmapping hervorgehenden kartesischen Produkt der Clusterausprägungen und können daher unvollständige Konfigurationen darstellen, in welchen offene Entscheidungen bestehen. Diese sind für die Konsistenzprüfung irrelevant, sodass auf diese in Kapitel 6 im Rahmen der Selektionsmethodik näher eingegangen wird. Werden dementsprechend die für das in Abschnitt 5.3 beispielhaft vorgestellte Anforderungsartefakt AS-3 gültigen Konfigurationen bestimmt, so ergeben sich:

$$\begin{aligned}
 (KA) \wedge (\mathbf{PH}) \wedge (HV \text{ Groß}) &= c_4, \\
 (KA) \wedge (\mathbf{PH}) \wedge (\text{Kein HV}) &= c_5, \\
 (SK) \wedge (\mathbf{PH}) \wedge (HV \text{ Groß}) &= c_{11}, \\
 (SK) \wedge (\mathbf{PH}) \wedge (\text{Kein HV}) &= c_{12},
 \end{aligned}
 \tag{5.6}$$

während

$$C(\text{AS-3}) = \{c_4, c_5, c_{11}, c_{12}\} \tag{5.7}$$

die Menge aller möglichen Konfigurationen, die für Anforderung AS-3 Gültigkeit haben, darstellt. Ausgehend von dieser Definition wird die Prüfmethode eingeführt.

5.4.1.2 Konzept der Konsistenzprüfung

Seien eine Anforderung $r_i \in R_{\text{all}}$ (siehe bspw. AS-0 in Tabelle 7) und ein Testfall $t_j \in T_{\text{all}}$ (siehe bspw. TS-1 in Tabelle 8) miteinander verlinkt $r_i \leftrightarrow t_j$, wobei R_{all} und T_{all} den Mengen aller Anforderungs- bzw. Testartefakte in der jeweiligen Spezifikation entsprechen. Die Analyse der Konsistenz zwischen den beiden Artefakten geschieht durch einen *Vergleich* (siehe Abbildung 22) der auf die Anforderung sowie der auf den Testfall abgebildeten Variabilität. Da die Artefakte in Beziehung stehen, müssen sie (mindestens zum Teil) gleiche Variabilität abbilden. Bedingt durch das Merkmalmapping in konjunktiver Normalform, reicht es nicht aus, lediglich Merkmale verlinkter Artefakte miteinander zu vergleichen⁴, sondern die zugeordnete Variabilität wird zunächst in einen einheitlichen Ausdruck übersetzt (im vorhergehenden Abschnitt 5.4.1.1 beschrieben). Jedes Artefakt ist somit für eine Menge von Konfigurationen $c_x \in C_{\text{all}}$ mit $C_{\text{all}} = \{c_1 \cdots c_{14}\}$ gültig. Zur Überprüfung der Konsistenz der abgebildeten Variabilität wird die Konfigurationsmenge der Anforderung $C(r_i)$ sowie die des Testfalls $C(t_j)$ miteinander verglichen. Der entsprechende Link wird in eine der vier definierten *Kategorien*, welche in den nächsten Abschnitten näher erläutert werden, einsortiert:

- Konsistenz
- Inkonsistenz:

4 Die Tatsache, dass beim konjunktiven Merkmalmapping nicht einfach Merkmalmengen miteinander verglichen werden dürfen, kann unter anderem anhand der in Abbildung 27 oder 28 gezeigten Beispiele nachvollzogen werden.

- Übervollständigkeit
- Unvollständigkeit
- Widerspruch.

Da die Anforderungsspezifikation vor der Testfallableitung einem Reviewprozess in Form der Testbasisanalyse (vgl. Abschnitt 2.5.2) unterzogen wird, wird das Merkmalmapping innerhalb des Anforderungsdokumentes als korrekt angenommen. Dieses gilt als Referenz, zu der das Merkmalmapping in der Testspezifikation konsistent sein muss.

In Tabelle 10 ist das gesamte *Konzept der Konsistenzprüfung* komprimiert dargestellt. Darin kennzeichnet die erste Spalte die Klassen-Nummerierung des entsprechenden Vergleichs. In der zweiten Spalte sind die Möglichkeiten aufgeführt, wie sich die Konfigurationsmengen der Anforderungen sowie die der Testfälle zueinander verhalten können. Die dritte Spalte zeigt, welche verlinkten Artefakte welcher Kategorie (in der vierten Spalte) zuzuordnen sind. Die letzten beiden Spalten definieren die Differenzen der Variabilität zwischen verglichenen Artefakten und beziehen sich damit auf die Behebung der Inkonsistenzen. Alle Kategorien(-Klassen) werden im Folgenden diskutiert und exemplarisch veranschaulicht.

5.4.1.3 Kategorie Konsistenz

Verlinkte Artefakte werden in sechs Fällen der Kategorie *Konsistenz* zugeordnet (vgl. Klasse 1, 2.1, 2.2, 3.1, 3.2 und 4.2.1 der Tabelle 10). Die Zuordnung in die Klasse 1 ist die trivialste und tritt dann auf, wenn das Merkmalmapping einer Anforderung mit dem Merkmalmapping des entsprechenden Testfalls übereinstimmt.

Definition 21 (Konsistenz). *Wird die der/den Anforderung(en) zugeordnete Konfigurationsmenge von der Konfigurationsmenge des/der mit der/den Anforderung(en) verlinkten Testfalls/Testfälle vollständig abgedeckt, handelt es sich um (eine) konsistente und damit korrekte Verknüpfung(en) zwischen den Anforderungs- und Testartefakten.*

Zur Veranschaulichung ist in Abbildung 23 ein Beispiel gezeigt. Darin ist die Anforderung r_1 mit dem Testfall t_1 verlinkt. Beide Artefakte weisen das identische Merkmalmapping, welches den Konfigurationen c_4 und c_{11} gemäß Tabelle 9 entspricht, auf. Die Verlinkung zwischen r_1 und t_1 ist somit konsistent:

$$\begin{aligned} MM(r_1) &= MM(t_1) = [(KA) \vee (SK)] \wedge (PH) \wedge (HV \text{ Groß}) \\ \Rightarrow \text{Klasse 1: } C(r_1) &= C(t_1) = \{c_4, c_{11}\} \end{aligned} \quad (5.8)$$

Durch die grau hervorgehobenen Pfeile in Abbildung 23 wird angedeutet, dass zusätzlich Links zwischen r_1 und weiteren Testfällen sowie zwischen t_1 und weiteren Anforderungen existieren können. Diese haben jedoch keine Auswirkung auf die Kategorisierung des Links zwischen r_1 und t_1 und würden von der Konsistenzprüfung separat analysiert werden.

Klasse	Formalismus	Verlinkte Artefakte	Kategorie	Differenz der Konfigurationsmengen C	Fehlende bzw. überschüssige Merkmale $f_y \in F$
1	$C(r_i) = C(t_j)$	$r_i \leftrightarrow t_j$	konsistent	-	-
2	$C(r_i) \supset C(t_j)$:				
2.1	$C(r_i) = C(T_{r_i})$	$\forall t \in T_{r_i} [r_i \leftrightarrow t]$	konsistent	-	-
2.2	$C(r_i) \subset C(T_{r_i})$	$r_i \leftrightarrow t_j$	konsistent	-	-
2.3	$C(r_i) \supset C(T_{r_i})$	$\forall t \in T_{r_i} [r_i \leftrightarrow t]$	unvollständig	$C_{\text{def}}(T_{r_i}) = C(r_i) \setminus C(T_{r_i})$	$F_{\text{def}}(t) = F(C_{\text{def}}(T_{r_i})) \setminus F(t)$
2.4	$[C(r_i) \not\subset C(T_{r_i})] \wedge [C(r_i) \not\supset C(T_{r_i})]$	$r_i \leftrightarrow t_j$	unvollständig	$C_{\text{def}}(T_{r_i}) = C(r_i) \setminus C(T_{r_i})$	$F_{\text{def}}(t_j) = F(C_{\text{def}}(T_{r_i})) \setminus F(t_j)$
3	$C(r_i) \subset C(t_j)$:				
3.1	$C(R_{t_j}) = C(t_j)$	$\forall r \in R_{t_j} [r \leftrightarrow t_j]$	konsistent	-	-
3.2	$C(R_{t_j}) \supset C(t_j)$	$r_i \leftrightarrow t_j$	konsistent	-	-
3.3	$C(R_{t_j}) \subset C(t_j)$	$\forall r \in R_{t_j} [r \leftrightarrow t_j]$	überevllständig	$C_{\text{exc}}(t_j) = C(t_j) \setminus C(R_{t_j})$	$F_{\text{exc}}(t_j) = F(C_{\text{exc}}(t_j)) \setminus F(r)$
3.4	$[C(R_{t_j}) \not\subset C(t_j)] \wedge [C(R_{t_j}) \not\supset C(t_j)]$	$r_i \leftrightarrow t_j$	überevllständig	$C_{\text{exc}}(t_j) = C(t_j) \setminus C(R_{t_j})$	$F_{\text{exc}}(t_j) = F(C_{\text{exc}}(t_j)) \setminus F(r_i)$
4	$[C(r_i) \not\subset C(t_j)] \wedge [C(r_i) \not\supset C(t_j)]$:				
4.1	$C(r_i) \cap C(t_j) = \emptyset$	$r_i \leftrightarrow t_j$	widersprüchlich	-	-
4.2	$C(r_i) \cap C(t_j) \neq \emptyset$:				
4.2.1	$[C(r_i) \subset C(T_{r_i})] \wedge [C(R_{t_j}) \supset C(t_j)]$	$r_i \leftrightarrow t_j$	konsistent	-	-
4.2.2	$[C(r_i) \not\subset C(T_{r_i})] \wedge [C(R_{t_j}) \supset C(t_j)]$	$r_i \leftrightarrow t_j$	unvollständig	$C_{\text{def}}(T_{r_i}) = C(r_i) \setminus C(T_{r_i})$	$F_{\text{def}}(t_j) = F(C_{\text{def}}(T_{r_i})) \setminus F(t_j)$
4.2.3	$[C(r_i) \subset C(T_{r_i})] \wedge [C(R_{t_j}) \not\supset C(t_j)]$	$r_i \leftrightarrow t_j$	überevllständig	$C_{\text{exc}}(t_j) = C(t_j) \setminus C(R_{t_j})$	$F_{\text{exc}}(t_j) = F(C_{\text{exc}}(t_j)) \setminus F(r_i)$
4.2.4	$[C(r_i) \not\subset C(T_{r_i})] \wedge [C(R_{t_j}) \not\supset C(t_j)]$	$r_i \leftrightarrow t_j$	unvollständig und überevllständig	$C_{\text{def}}(T_{r_i}) = C(r_i) \setminus C(T_{r_i})$ und $C_{\text{exc}}(t_j) = C(t_j) \setminus C(R_{t_j})$	$F_{\text{def}}(t_j) = F(C_{\text{def}}(T_{r_i})) \setminus F(t_j)$ und $F_{\text{exc}}(t_j) = F(C_{\text{exc}}(t_j)) \setminus F(r_i)$

Tabelle 10: Konzept der Konsistenzprüfung über den Vergleich der Konfigurationsmengen C zwischen verlinkten Anforderungen r und Testfällen t bei einer Assoziationslogik in konjunktiver Normalform

Anforderung	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie		Testfall	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie
r_1		PH	HV Groß		t_1		PH	HV Groß
... r					... t			

Abbildung 23: Beispiel für eine konsistente Verlinkung (grün hervorgehoben) der Klasse 1 (vgl. Tabelle 10). Weitere (grau angedeutete) Verlinkungen sind für die Analyse und Kategorisierung von $r_1 \leftrightarrow t_1$ irrelevant.

Ein zweites Beispiel für das Auftreten von konsistenten Verlinkungen ist in Abbildung 24 gezeigt. Dort sind die Anforderung r_2 und der Testfall t_2 miteinander verlinkt. Die von r_2 aufgespannte Konfigurationsmenge $C(r_2)$ wird von t_2 nur teilweise abgedeckt (vgl. Term 5.9). Dies entspricht der Klasse 2 in Tabelle 10. Weitere Testfälle, die mit r_2 verlinkt sind, könnten die fehlenden Konfigurationen abdecken. Aus diesem Grund wird geprüft, ob solche Testfälle T_{r_2} existieren mit $T_{r_2} = \{t \in T_{\text{all}} \mid r_2 \leftrightarrow t\}$. Im vorliegenden Beispiel ist ein einziger weiterer mit r_2 verlinkter Testfall t_3 gegeben, wodurch $T_{r_2} = \{t_2, t_3\}$ gilt. Die gesamte Konfigurationsmenge der Testfälle wird nun mit der der Anforderung verglichen. Durch t_3 wird die Abdeckung vervollständigt (vgl. Term 5.10). Dementsprechend sind beide Verlinkungen $r_2 \leftrightarrow t_2$ und $r_2 \leftrightarrow t_3$ als konsistent zu bewerten.

Anforderung	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie		Testfall	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie
r_2		SH	HV Groß		t_2	KA	SH	HV Groß
... r					t_3	SK	SH	HV Groß

Abbildung 24: Beispiel für konsistente Verlinkungen (grün hervorgehoben) der Klasse 2.1 (vgl. Tabelle 10). Weitere (grau angedeutete) Verlinkungen sind für die Analyse und Kategorisierung von $r_2 \leftrightarrow T_{r_2}$, mit $T_{r_2} = \{t_2, t_3\}$ irrelevant.

$$MM(r_2) = [(KA) \vee (SK)] \wedge (SH) \wedge (HV \text{ Groß})$$

$$\Rightarrow C(r_2) = \{c_1, c_8\}$$

$$MM(t_2) = (KA) \wedge (SH) \wedge (HV \text{ Groß})$$

$$\Rightarrow C(t_2) = \{c_1\}$$

$$\textbf{Klasse 2:} \quad C(r_2) \supset C(t_2) \tag{5.9}$$

$$MM(t_3) = (SK) \wedge (SH) \wedge (HV \text{ Groß})$$

$$\Rightarrow C(t_3) = \{c_8\}$$

$$\textbf{Klasse 2.1:} \quad C(r_2) = C(T_{r_2}), \text{ mit } T_{r_2} = \{t_2, t_3\} \tag{5.10}$$

In Analogie zur Klasse 2.1 ist gemäß Tabelle 10 die Klasse 3.1 definiert. Da es sich um einen spiegelverkehrten Fall von Klasse 2.1 handelt, wird dieser ohne ein konkretes Beispiel kurz diskutiert. Bei Klasse 3.1 wird anstelle von einer unvollständig $C(r_i) \supset C(t_j)$

umgekehrt von einer übergroß abgedeckten Konfigurationsmenge seitens eines Testfalls ausgegangen $C(r_i) \subset C(t_j)$. Existieren weitere Anforderungen, die eine vollständige Abdeckung von $C(t_j)$ generieren $C(R_{t_j}) = C(t_j)$, mit $R_{t_j} = \{r \in R_{\text{all}} \mid r \leftrightarrow t_j\}$, handelt es sich um konsistente Verlinkungen zwischen t_j und R_{t_j} .

Ein weiterer Fall für eine konsistente Verknüpfung ist in Abbildung 25 dargestellt. Dabei liegt der Ausgangslink $r_3 \leftrightarrow t_4$ vor, bei dem die Konfigurationsmenge des Testfalls $C(t_4)$ die der Anforderung r_3 mehr als notwendig abdeckt (vgl. Term 5.11). Weitere mit t_4 verlinkte Anforderungsartefakte könnten $C(t_4)$ komplettieren. Diese sind gegeben mit $R_{t_4} = \{r \in R_{\text{all}} \mid r \leftrightarrow t_4\}$. Entsprechende Artefakte liegen durch $R_{t_4} = \{r_3, r_4\}$ vor. $C(t_4)$ deckt $C(R_{t_4})$ bis auf die fehlenden Elemente c_{11}, c_{12} ab (vgl. Term 5.12). Dadurch ist nur der Ausgangslink zwischen r_3 und t_4 konsistent, während $r_4 \leftrightarrow t_4$ gesondert analysiert werden müsste.

Anforderung	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie	Testfall	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie
r_3	KA	PH	HV Groß	t_4	KA	PH	
r_4		PH		... t			

Abbildung 25: Beispiel für eine konsistente Verlinkung (grün hervorgehoben) der Klasse 3.2 (vgl. Tabelle 10). Schwarz markierte Verlinkung wird berücksichtigt, während weitere (grau angedeutete) Verknüpfungen bei der Analyse und Kategorisierung von $r_3 \leftrightarrow t_4$ irrelevant sind.

$$MM(r_3) = (KA) \wedge (PH) \wedge (HV \text{ Groß})$$

$$\Rightarrow C(r_3) = \{c_4\}$$

$$MM(t_4) = (KA) \wedge (PH) \wedge [(HV \text{ Groß}) \vee (Kein HV)]$$

$$\Rightarrow C(t_4) = \{c_4, c_5\}$$

$$\textbf{Klasse 3:} \quad C(r_3) \subset C(t_4) \quad (5.11)$$

$$MM(r_4) = [(KA) \vee (SK)] \wedge (PH) \wedge [(HV \text{ Groß}) \vee (Kein HV)]$$

$$\Rightarrow C(r_4) = \{c_4, c_5, c_{11}, c_{12}\}$$

$$\textbf{Klasse 3.2:} \quad C(R_{t_4}) \supset C(t_4), \text{ mit } R_{t_4} = \{r_3, r_4\} \quad (5.12)$$

Wie bereits beim vorhergehenden Beispiel diskutiert, existiert ebenfalls ein zu Klasse 3.2 spiegelsymmetrischer Fall, gegeben durch Klasse 2.2. Bei dieser wird anders als bei Klasse 3.2 die teilweise Abdeckung einer Anforderung vorausgesetzt $C(r_i) \supset C(t_j)$. Wird durch weitere mit r_i verknüpfte Testfälle die Konfigurationsmenge der Anforderung mehr als notwendig abgedeckt $C(r_i) \subset C(T_{r_i})$, kann nur $r_i \leftrightarrow t_j$ als konsistent klassifiziert werden. Alle weiteren Links zwischen r_i und T_{r_i} müssten anderweitig untersucht werden.

Der letzte mögliche Fall einer konsistenten Verknüpfung ist anhand des Beispiels in Abbildung 26 gezeigt. Dabei besitzen die Konfigurationsräume der verlinkten Artefakte r_5 und t_5 eine Schnittmenge. Gleichzeitig verfügt $C(r_5)$ über Elemente, die nicht in

$C(t_5)$ vertreten sind und umgekehrt (vgl. Term 5.13). Die Konsistenz kann dennoch vorliegen, wenn zum einen weitere mit r_5 verknüpfte Testfälle die noch nicht abgedeckten Elemente von $C(r_5)$ ergänzen. Dies ist für $r_5 \leftrightarrow T_{r_5}$ der Fall. Zum anderen trägt r_6 zur Vervollständigung der Abdeckung des Testfalls t_5 bei (vgl. Term 5.14). Damit ist der Link zwischen r_5 und t_5 konsistent. Die Klassifikation von $r_5 \leftrightarrow t_6$ sowie $r_6 \leftrightarrow t_5$ kann an dieser Stelle noch nicht vorgenommen werden. Diese müssten gesondert untersucht werden.

Anforderung	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie		Testfall	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie
r_5		PH	HV Groß	↔	t_5	SK		HV Groß
r_6	SK		HV Groß	↔	t_6	KA		
... r				↔	... t			

Abbildung 26: Beispiel für eine konsistente Verlinkung (grün hervorgehoben) der Klasse 4.2.1 (vgl. Tabelle 10). Schwarz markierte Verlinkungen werden berücksichtigt, während weitere (grau angedeutete) Verknüpfungen bei der Analyse und Kategorisierung von $r_5 \leftrightarrow t_5$ irrelevant sind.

$$MM(r_5) = [(KA) \vee (SK)] \wedge (PH) \wedge (HV \text{ Groß})$$

$$\Rightarrow C(r_5) = \{c_4, c_{11}\}$$

$$MM(t_5) = (SK) \wedge [(SH) \vee (PH) \vee (LH)] \wedge (HV \text{ Groß})$$

$$\Rightarrow C(t_5) = \{c_8, c_{11}, c_{13}\}$$

$$\text{Klasse 4.2: } [C(r_5) \cap C(t_5) \neq \emptyset] \wedge [C(r_5) \not\subseteq C(t_5)] \wedge [C(r_5) \not\supseteq C(t_5)] \quad (5.13)$$

$$MM(r_6) = (SK) \wedge [(SH) \vee (PH) \vee (LH)] \wedge (HV \text{ Groß})$$

$$\Rightarrow C(r_6) = \{c_8, c_{11}, c_{13}\}$$

$$MM(t_6) = (KA) \wedge [(SH) \vee (PH) \vee (LH)] \wedge [(HV \text{ Groß}) \vee (HV \text{ Klein}) \vee (Kein HV)]$$

$$\Rightarrow C(t_6) = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7\}$$

$$\text{Klasse 4.2.1: } [C(r_5) \subset C(T_{r_5})] \wedge [C(R_{t_5}) \supset C(t_5)], \quad (5.14)$$

mit $R_{t_5} = \{r_5, r_6\}$ und $T_{r_5} = \{t_5, t_6\}$

5.4.1.4 Widerspruch

Eine von drei Kategorien für inkonsistente Verknüpfungen ist durch den *Widerspruch* gegeben (vgl. Klasse 4.1 der Tabelle 10). Diese ist folgendermaßen definiert:

Definition 22 (Widerspruch). *Besitzen die Konfigurationsmenge der Anforderung sowie die des mit der Anforderung verlinkten Testfalls keine gemeinsamen Elemente, handelt es sich um eine widersprüchliche Verknüpfung zwischen dem Anforderungs- und Testartefakt.*

In Abbildung 27 ist eine exemplarische widersprüchliche Verlinkung dargestellt. Darin sind die Artefakte r_7 und t_7 miteinander verknüpft. Die Schnittmenge der Konfigurationsräume $C(r_7) \cap C(t_7)$ ist leer (vgl. Term 5.15), wodurch beide Artefakte unberechtigt teilweise verlinkt sind. Sämtliche Verknüpfungen von r_7 und t_7 zu weiteren Artefakten würden den Widerspruch nicht beseitigen können, weshalb $r_7 \leftrightarrow t_7$ klassifiziert wird ohne weitere Links in die Analyse einzubeziehen.

Anforderung	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie		Testfall	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie
r_7		PH	HV Groß		t_7		PH	Kein HV
... r					... t			

Abbildung 27: Beispiel für eine widersprüchliche Verlinkung (rot hervorgehoben) der Klasse 4.1 (vgl. Tabelle 10). Weitere (grau angedeutete) Verlinkungen sind bei der Analyse und Kategorisierung von $r_7 \leftrightarrow t_7$ irrelevant.

$$\begin{aligned}
 MM(r_7) &= (PH) \wedge (HV \text{ Groß}) \\
 \Rightarrow C(r_7) &= \{c_4, c_{11}\} \\
 MM(t_7) &= (PH) \wedge (\text{Kein HV}) \\
 \Rightarrow C(t_7) &= \{c_5, c_{12}\} \\
 \textbf{Klasse 4.1:} \quad C(r_7) \cap C(t_7) &= \emptyset
 \end{aligned} \tag{5.15}$$

5.4.1.5 Unvollständigkeit

Die zweite Möglichkeit für inkonsistente Verknüpfungen liegt im Falle *unvollständig* abgedeckter Variabilität der Anforderungs- durch entsprechende Testartefakte vor. Unvollständigkeit tritt in den Klassen 2.3, 2.4, 4.2.2 und 4.2.4 (vgl. Tabelle 10) auf.

Definition 23 (Unvollständigkeit). *Wird die der/den Anforderung(en) zugeordnete Konfigurationsmenge von der Konfigurationsmenge des/der mit der/den Anforderung(en) verlinkten Testfalls/Testfälle teilweise abgedeckt, handelt es sich um (eine) unvollständige Verknüpfung(en) zwischen den Anforderungs- und Testartefakten.*

Abbildung 28 präsentiert ein Beispiel für die Kategorie der Unvollständigkeit. Die Ausgangssituation ist durch die Verlinkung $r_8 \leftrightarrow t_8$ gegeben, bei welcher der Testfall nur einen Teil des Konfigurationsraumes von r_8 abdeckt (vgl. Term 5.16). t_9 ist das einzige weitere Testartefakt, welches mit r_8 verknüpft ist und die Abdeckung von $C(r_8)$ ergänzen könnte. Entsprechende Analyse zeigt, dass der Konfigurationsraum der Anforderung dennoch eine Obermenge des Konfigurationsraumes aller Testfälle bildet und damit ein Defizit in der Abdeckung bleibt. Daher werden alle Verlinkungen zwischen r_8 und entsprechenden Testartefakten als unvollständig klassifiziert (vgl. Term 5.17).

Anforderung	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie		Testfall	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie
r_8	SK	SH PH			t_8	SK	SH	HV Groß
... r					t_9	SK	PH	

Abbildung 28: Beispiel für unvollständige Verlinkungen (rot hervorgehoben) der Klasse 2.3 (vgl. Tabelle 10). Weitere (grau angedeutete) Verlinkungen sind bei der Analyse und Kategorisierung von $r_8 \leftrightarrow C(T_{r_8})$ mit $T_{r_8} = \{t_8, t_9\}$ irrelevant.

$$MM(r_8) = (SK) \wedge [(SH) \vee (PH)] \wedge [(HV \text{ Groß}) \vee (HV \text{ Klein}) \vee (Kein HV)]$$

$$\Rightarrow C(r_8) = \{c_8, c_9, c_{10}, c_{11}, c_{12}\}$$

$$MM(t_8) = (SK) \wedge (SH) \wedge (HV \text{ Groß})$$

$$\Rightarrow C(t_8) = \{c_8\}$$

$$\textbf{Klasse 2:} \quad C(r_8) \supset C(t_8) \quad (5.16)$$

$$MM(t_9) = (SK) \wedge (PH) \wedge [(HV \text{ Groß}) \vee (Kein HV)]$$

$$\Rightarrow C(t_9) = \{c_{11}, c_{12}\}$$

$$\textbf{Klasse 2.3:} \quad C(r_8) \supset C(T_{r_8}), \text{ mit } T_{r_8} = \{t_8, t_9\} \quad (5.17)$$

Ausgehend von dem hier präsentierten Beispiel ist es nachvollziehbar, warum es nicht möglich ist Merkmale direkt, d.h. ohne die Ermittlung von c_x , zu vergleichen. Die Merkmalmenge der Anforderung $F(r_8)$ sowie die der Testfälle $F(T_{r_8})$ sind identisch und würden demnach als konsistent charakterisiert werden. Dies ist jedoch nicht korrekt, da bei der Ausleitung einer Produktkonfiguration, welche bspw. *SH* und *HV Klein* enthält, r_8 gültig wäre, allerdings keine der mit r_8 verlinkten Testfälle t_8 oder t_9 gelten würde.

Das Beispiel $r_8 \leftrightarrow t_8$ beschreibt eine Ausgangssituation, bei welcher der Konfigurationsraum des Testfalls eine Untermenge des Konfigurationsraumes der Anforderung bildet. Existieren weiterhin Links zwischen r_8 und zusätzlichen Testfällen, bei deren Analyse die Beziehung $[C(r_8) \not\supseteq C(T_{r_8})] \wedge [C(r_8) \not\subseteq C(T_{r_8})]$ festgestellt wird, handelt es sich um Klasse 2.4. Das bedeutet, dass die Anforderung insgesamt unvollständig abgedeckt ist und gleichzeitig weitere in $C(r_8)$ nicht enthaltene Elemente innerhalb der zusätzlich analysierten Testfälle vorkommen. Dadurch ist ausschließlich der Ausgangslink zwischen r_8 und t_8 als unvollständig zu klassifizieren. Für eine Kategorisierung aller weiteren Verknüpfungen müssen diese gesondert untersucht werden.

Ein weiteres Beispiel für die Kategorie der Unvollständigkeit ist in Abbildung 29 dargestellt. Der Ausgangslink zwischen r_9 und t_{10} weist eine Schnittmenge der Konfigurationsräume beider Artefakte auf, welche jeweils überschüssige Elemente enthalten (vgl. Term 5.18). Durch Hinzunahme aller weiterer mit t_{10} verknüpfter Anforderungsartefakte r_{10} , r_{11} wird eine Abdeckung von $C(t_{10})$ identifiziert. Allerdings bleibt r_9 trotz weiterer Verlinkung zu t_{11} unvollständig abgedeckt. Daraus resultiert die Klassifikation des Ausgangslinks als unvollständig. Die Kategorisierung weiterer genannter Artefakte muss separat analysiert werden.

Anforderung	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie		Testfall	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie
r ₉	SK	SH LH		←→	t ₁₀		LH	Kein HV
r ₁₀	KA	SH	Kein HV	←→	t ₁₁	KA		Kein HV
r ₁₁	KA	PH LH	Kein HV	←→				

Abbildung 29: Beispiel für eine unvollständige Verlinkung (rot hervorgehoben) der Klasse 4.2.2 (vgl. Tabelle 10). Schwarz markierte Verlinkungen werden bei der Analyse und Kategorisierung von $r_9 \leftrightarrow t_{10}$ berücksichtigt. Auf die Darstellung von möglichen Verlinkungen zwischen r_{10} , r_{11} und weiteren Testfällen sowie zwischen t_{11} und weiteren Anforderungen wird zwecks Übersichtlichkeit verzichtet. Diese wären für die Analyse des diskutierten Falls irrelevant.

$$\begin{aligned}
MM(r_9) &= (SK) \wedge [(SH) \vee (LH)] \wedge [(HV \text{ Groß}) \vee (HV \text{ Klein}) \vee (Kein HV)] \\
\Rightarrow C(r_9) &= \{c_8, c_9, c_{10}, c_{13}, c_{14}\} \\
MM(t_{10}) &= [(KA) \vee (SK)] \wedge (LH) \wedge (Kein HV) \\
\Rightarrow C(t_{10}) &= \{c_7, c_{14}\} \\
\textbf{Klasse 4.2:} & [C(r_9) \cap C(t_{10}) \neq \emptyset] \wedge [C(r_9) \not\supseteq C(t_{10})] \wedge [C(r_9) \not\subseteq C(t_{10})] \\
MM(r_{10}) &= (KA) \wedge (SH) \wedge (Kein HV) \\
\Rightarrow C(r_{10}) &= \{c_3\} \\
MM(r_{11}) &= (KA) \wedge [(PH) \vee (LH)] \wedge (Kein HV) \\
\Rightarrow C(r_{11}) &= \{c_5, c_7\} \\
MM(t_{11}) &= (KA) \wedge [(SH) \vee (PH) \vee (LH)] \wedge (Kein HV) \\
\Rightarrow C(t_{11}) &= \{c_3, c_5, c_7\} \\
\textbf{Klasse 4.2.2:} & [C(r_9) \not\subseteq C(T_{r_9})] \wedge [C(R_{t_{10}}) \supset C(t_{10})], \\
& \text{mit } R_{t_{10}} = \{r_9, r_{10}, r_{11}\} \text{ und } T_{r_9} = \{t_{10}, t_{11}\}
\end{aligned} \tag{5.18}$$

5.4.1.6 Übervollständigkeit

Die dritte und letzte Möglichkeit für Inkonsistenzen ist durch *übervollständige* Verknüpfungen gegeben. Übervollständigkeit tritt in den Klassen 3.3, 3.4, 4.2.3 und 4.2.4 (vgl. Tabelle 10) auf. Die Kategorie wird folgendermaßen definiert:

Definition 24 (Übervollständigkeit). *Wird die der/den Anforderung(en) zugeordnete Konfigurationsmenge von der Konfigurationsmenge des/der mit der/den Anforderung(en) verlinkten Testfalls/Testfälle mehr als vollständig abgedeckt, handelt es sich um (eine) übervollständige Verknüpfung(en) zwischen den Anforderungs- und Testartefakten.*

Ein Beispiel für eine übervollständige Verknüpfung ist in Abbildung 30 gezeigt. Es wird der Ausgangslink zwischen r_{12} und t_{12} vorausgesetzt. $C(t_{12})$ bildet die Obermenge

von $C(r_{12})$, was zunächst eine übervollständige Abdeckung der Anforderung charakterisiert. Durch die ergänzende Betrachtung weiterer mit dem Testfall verknüpften Anforderungsartefakte wird eine Schnittmenge der Konfigurationsräume $C(t_{12})$ und $C(R_{t_{12}})$ mit jeweils überschüssigen Elementen festgestellt (vgl. Term 5.20). Dadurch bleibt der Ausgangslink $r_{12} \leftrightarrow t_{12}$ als übervollständig klassifiziert. Weitere erwähnte Links müssen separat geprüft werden.

Der hierzu spiegelsymmetrische Fall – die Klasse 2.4 – wurde im vorhergehenden Abschnitt 5.4.1.5 ohne Anführung eines konkreten Beispiel kurz diskutiert.

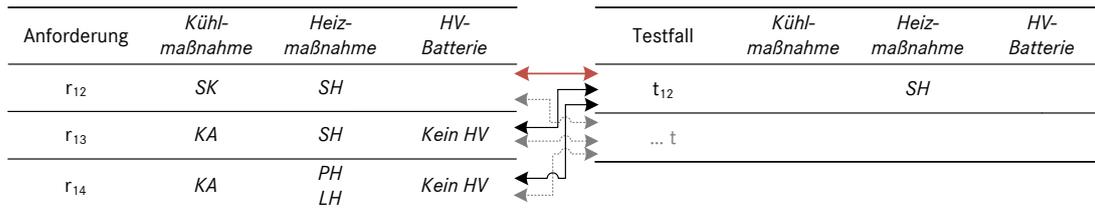


Abbildung 30: Beispiel für eine übervollständige Verlinkung (rot hervorgehoben) der Klasse 3.4 (vgl. Tabelle 10). Schwarz markierte Verlinkungen werden berücksichtigt, während weitere (grau angedeutete) Verknüpfungen bei der Analyse und Kategorisierung von $r_{12} \leftrightarrow t_{12}$ irrelevant sind.

$$\begin{aligned}
 MM(r_{12}) &= (SK) \wedge (SH) \wedge [(HV \text{ Groß}) \vee (HV \text{ Klein}) \vee (Kein HV)] \\
 \Rightarrow C(r_{12}) &= \{c_8, c_9, c_{10}\} \\
 MM(t_{12}) &= [(KA) \vee (SK)] \wedge (SH) \wedge [(HV \text{ Groß}) \vee (HV \text{ Klein}) \vee (Kein HV)] \\
 \Rightarrow C(t_{12}) &= \{c_1, c_2, c_3, c_8, c_9, c_{10}\} \\
 \textbf{Klasse 3:} \quad C(r_{12}) &\subset C(t_{12}) & (5.20) \\
 MM(r_{13}) &= (KA) \wedge (SH) \wedge (Kein HV) \\
 \Rightarrow C(r_{13}) &= \{c_3\} \\
 MM(r_{14}) &= (KA) \wedge [(PH) \vee (LH)] \wedge (Kein HV) \\
 \Rightarrow C(r_{14}) &= \{c_5, c_7\} \\
 \textbf{Klasse 3.4:} \quad [C(R_{t_{12}}) \not\subset C(t_{12})] \wedge [C(R_{t_{12}}) \not\supset C(t_{12})], & (5.21) \\
 \text{mit } R_{t_{12}} &= \{r_{12}, r_{13}, r_{14}\}
 \end{aligned}$$

Der letzte vorgestellte Fall stellt eine Besonderheit dar, da dabei eine Verknüpfung zwei Kategorien gleichzeitig zugeordnet wird. Dieser wird anhand eines Beispiels in Abbildung 31 demonstriert. Die Konfigurationsräume der Artefakte r_{15} und t_{13} besitzen eine Schnittmenge mit jeweils überschüssigen Elementen (vgl. Term 5.22). Die Analyse der Konfigurationsräume weiterer mit r_{15} und t_{13} verlinkten Artefakte ergibt weder eine vollständige Abdeckung von $C(r_{15})$ (Defizit) noch die von $C(t_{13})$ (Überschuss). Dadurch ist die Verlinkung $r_{15} \leftrightarrow t_{13}$ als unvollständig sowie zur gleichen Zeit übervollständig zu bezeichnen. Die Klassifikation der Verlinkungen zwischen weiteren genannten Artefakten muss gesondert untersucht werden.

Anforderung	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie	Testfall	Kühl- maßnahme	Heiz- maßnahme	HV- Batterie
r ₁₅		PH	HV Groß	t ₁₃	SK	PH	
r ₁₆	SK		HV Groß	t ₁₄	KA		HV Klein
... r				... t			

Abbildung 31: Beispiel für eine unvollständige und gleichzeitig übervollständige Verlinkung (rot hervorgehoben) der Klasse 4.2.4 (vgl. Tabelle 10). Schwarz markierte Verlinkungen werden berücksichtigt, während weitere (grau angedeutete) Verknüpfungen bei der Analyse und Kategorisierung von $r_{15} \leftrightarrow t_{13}$ irrelevant sind.

$$MM(r_{15}) = [(KA) \vee (SK)] \wedge (PH) \wedge (HV \text{ Groß})$$

$$\Rightarrow C(r_{15}) = \{c_4, c_{11}\}$$

$$MM(t_{13}) = (SK) \wedge (PH) \wedge [(HV \text{ Groß}) \vee (\text{Kein HV})]$$

$$\Rightarrow C(t_{13}) = \{c_{11}, c_{12}\}$$

$$\textbf{Klasse 4.2:} \quad [C(r_{15}) \cap C(t_{13}) \neq \emptyset] \wedge \quad (5.22)$$

$$[C(r_{15}) \not\supseteq C(t_{13})] \wedge [C(r_{15}) \not\subseteq C(t_{13})]$$

$$MM(r_{16}) = (SK) \wedge [(SH) \vee (PH) \vee (LH)] \wedge (HV \text{ Groß})$$

$$\Rightarrow C(r_{16}) = \{c_8, c_{11}, c_{13}\}$$

$$MM(t_{14}) = (KA) \wedge (SH) \wedge (HV \text{ Klein})$$

$$\Rightarrow C(t_{14}) = \{c_2\}$$

$$\textbf{Klasse 4.2.4:} \quad [C(r_{15}) \not\subseteq C(T_{r_{15}})] \wedge [C(R_{t_{13}}) \not\supseteq C(t_{13})], \quad (5.23)$$

$$\text{mit } R_{t_{13}} = \{r_{15}, r_{16}\} \text{ und } T_{r_{15}} = \{t_{13}, t_{14}\}$$

5.4.2 Ursachen und Folgen von Inkonsistenzen

Die oben vorgestellte Prüfmethode identifiziert verschiedenen Typen von Inkonsistenzen (spezifiziert in Tabelle 10 und diskutiert im vorhergehenden Abschnitt) und unterstützt bei deren Korrektur. In Abbildung 32 ist ein Überblick über die möglichen auftretenden Inkonsistenzen schematisch dargestellt. Darüber hinaus ist neben den aufgelisteten Kategorien der (von unten nach oben) zunehmende Grad der Kritikalität visualisiert. Die Kritikalität ist mittels der möglichen aus der jeweiligen Inkonsistenz hervorgehenden Konsequenzen bzw. Auswirkungen definiert. Sowohl die *Ursachen* als auch die *Folgen* werden im Folgenden für die einzelnen Fälle der Inkonsistenz genauer erläutert.

Die Gründe, welche für das Auftreten der oben diskutierten Inkonsistenzen verantwortlich sein können, sind vielfältig und lassen sich folgendermaßen zusammenfassen:

- **falsches Merkmalmapping** durch unvollständig und/oder überschüssig zugeordnete Merkmale und

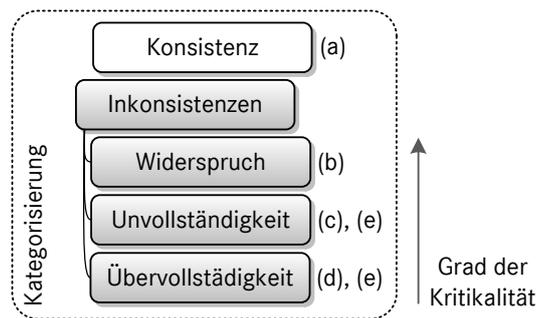


Abbildung 32: Überblick über mögliche Inkonsistenzen (identifizierbar mithilfe der im vorhergehenden Abschnitt 5.4.1.2 diskutierten Prüfmethodik), der Kritikalitätsgrad sowie die Zuordnung zu den in Abbildung 33 dargestellten Fällen (a)–(e).

- **falsche Verlinkungen** durch fehlende und/oder überflüssige Links (unter anderem aufgrund von nicht bzw. überflüssig spezifizierten (Test)Artefakten).

Der Widerspruch charakterisiert den kritischsten Typ einer Inkonsistenz. Dieser hat die Bedeutung, dass zwei Artefakte, in welchen gänzlich unterschiedliche Variabilität abgebildet ist, miteinander in Beziehung gesetzt sind. Offensichtlich handelt es sich dabei um eine nicht gerechtfertigte Verknüpfung, welche entweder unnötigerweise bzw. an der falschen Stelle gesetzt wurde. Eine ebenfalls mögliche Ursache für die Inkonsistenz ist ein falsches Merkmalmapping. Eine sehr wahrscheinliche Folge einer widersprüchlichen Verknüpfung wäre, dass das entsprechende Anforderungsartefakt durch keinen Testfall in der korrekten Konfiguration berücksichtigt wäre. Die darauf folgende Durchführung des Testfalls in der falschen Konfiguration könnte ihrerseits zu widersprüchlichen Testergebnissen führen. Letztendlich kann die fehlende (korrekte) Überprüfung der entsprechenden Anforderung schwerwiegende Auswirkungen bis hin in die Produktion haben.

Weiterhin liegt eine kritische Bedingung im Falle von unvollständig definierten Verknüpfungen vor. Die hierfür verantwortlichen Gründe können einerseits nicht gesetzte Links sowie andererseits fehlende Merkmale sein. Insbesondere Letzteres kennzeichnet einen kritischen Fall, da entsprechende Tests unzureichend und damit unvollständig durchgeführt werden könnten. Analog zum oben beschriebenen Fall können hieraus widersprüchliche Testergebnisse erwachsen und gegebenenfalls einen negativen Einfluss auf den weiteren Entwicklungs- bis hin zum Produktionsprozess haben.

Übervollständige Verknüpfungen charakterisieren weniger kritische Fälle, da das Kriterium der Anforderungsabdeckung erfüllt ist. Gleichzeitig ist es jedoch übererfüllt. Hierbei sind überschüssige Merkmale sowie fehlende Links als Ursache zu nennen. Die aus der Kategorie hervorgehenden Konsequenzen können die Durchführung der Testfälle in falschen Konfigurationen – zusätzlich zu denen in korrekten Konfigurationen – bewirken. Dennoch ist auch in diesem Fall die Gefahr von abweichenden und dadurch irreführenden Testergebnissen präsent. Zudem sind übervollständig definierte Testfälle generell als überschüssige Testumfänge und die damit verbundenen Korrekturen im Sinne einer Effizienzsteigerung zu bewerten.

Aufgrund der beschriebenen Kritikalitätsstufen ist es sinnvoll festgestellte Inkonsistenzen in der in Abbildung 32 dargestellten Reihenfolge (angefangen bei widersprüchlichen, über unvollständige und zuletzt übervollständige Verknüpfungen) zu beheben. Explizite Handlungsmaßnahmen werden im nächsten Abschnitt vorgestellt.

5.4.3 Behebung von Inkonsistenzen

Das Ziel der vorgestellten Prüfmethodik ist es alle in Kapitel 5.4.1.2 definierten Inkonsistenzen und die daraus resultierenden Fehler in den Spezifikationsdokumenten zu beheben. Im vorliegenden Abschnitt werden konkrete Maßnahmen definiert, welche dem Anwender zur gezielten *Korrektur* der Fehler bereitgestellt werden.

Wie bereits in Abschnitt 5.4.1.2 diskutiert, wird bei der Prüfmethodik das Merkmal-mapping innerhalb einer Anforderungsspezifikation als fehlerfrei vorausgesetzt. Die Merkmale in der Testspezifikation müssen somit konsistent zu dieser Referenz abgebildet sein. Aus dieser Eigenschaft/Bedingung folgt, dass sich sämtliche Korrekturmaßnahmen ausschließlich auf Testartefakte beziehen.

Anhand der Abbildung 33 sind alle möglichen Fälle, wie sich Mengen der abgebildeten Variabilität zu einander verhalten können, veranschaulicht. Fall (a) repräsentiert die Kategorie der Konsistenz, die entweder genau zwischen einer Anforderung r sowie einem Testfall t bzw. zwischen mehreren Artefakten r , T und R , t gleichzeitig identifiziert werden kann. Die Konfigurationsmengen entsprechender Artefakte liegen übereinander und stimmen daher überein. Demnach sind keine Korrekturmaßnahmen erforderlich.

Im Fall (b), welcher den Widerspruch beschreibt, besitzen die Konfigurationsräume der Anforderung sowie die des Testfalls keine gemeinsamen Elemente bzw. keine Schnittmenge. Zum einen ist es möglich, dass das Merkmal-mapping des Testfalls falsch ist und gemäß des Merkmal-mappings in der entsprechenden Anforderung angepasst werden sollte. Hierbei wird keine Differenzmenge zur Formulierung einer Handlungsmaßnahme ermittelt, da das Merkmal-mapping der Anforderung direkt abgelesen und auf den Testfall übertragen werden kann. Zum anderen sollte bei widersprüchlich verlinkten Artefakten der Link an sich in Frage gestellt werden. Dieser kann entweder an der falschen Stelle gesetzt worden sein oder gänzlich ohne Berechtigungs-dasein existieren.

Die Fälle (c) bis (d) (vgl. Abbildung 33) kennzeichnen die Kategorie der Un- und Übervollständigkeit, bei welchen die verknüpften Artefakte gemeinsame Elemente in den Konfigurationsräumen aufweisen. Hierbei gilt es ebenfalls die Verlinkungen zu hinterfragen. Im Falle der Unvollständigkeit sollte geprüft werden, ob fehlende Links, im Falle der Übervollständigkeit hingegen überschüssige Links zu entsprechenden Testartefakten vorliegen und diese gegebenenfalls ergänzen oder entfernen. Darüber hinaus ist es vorstellbar, dass die Merkmale nicht korrekt den Testfällen zugeordnet und dadurch nicht die gleichen Konfigurationen wie in den Anforderungen abgebildet sind.

Hierfür können die Differenzen im Bezug auf die zugeordnete Variabilität ermittelt werden (vgl. vorletzte Spalte in Tabelle 10). Im Falle der Unvollständigkeit bezieht sich eine Differenzmenge auf defizitäre, von Testfällen nicht abgedeckte Konfigurationen

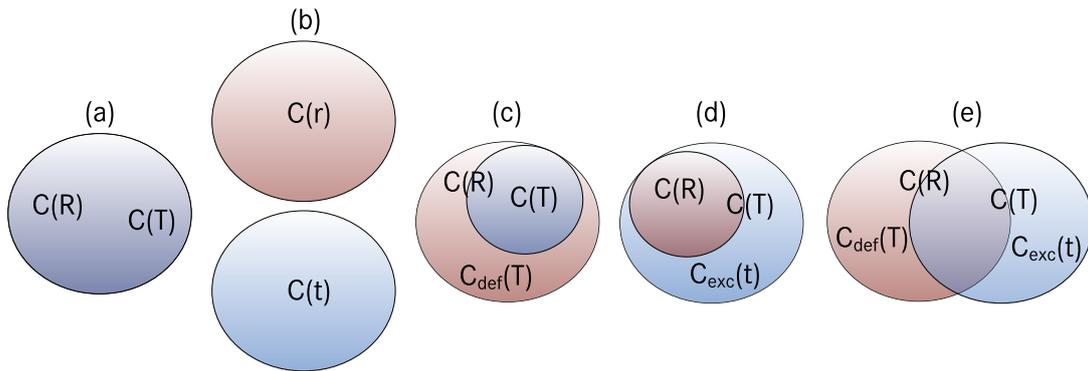


Abbildung 33: Verhältnisse der Konfigurationsräume zwischen Anforderungen und Testfällen (ausgehend von Tabelle 10) sowie die damit einhergehenden und zur Behebung von Inkonsistenzen erforderlichen Differenzmengen C_{def} , C_{exc}

C_{def} (siehe Fall (c) in Abbildung 33). Bei der Kategorie Übervollständigkeit dagegen repräsentiert diese überschüssige, durch den Testfall mehr als notwendig abgedeckte Konfigurationen C_{exc} (engl. *excess*) (siehe Fall (d) in Abbildung 33).

Zum besseren Verständnis werden die Fälle (c)-(d) anhand von bereits vorgestellten Beispielen erläutert. Dazu wird als erstes die in Abschnitt 5.4.1.5 diskutierte Klasse 2.3 der Unvollständigkeit betrachtet. Die Konfigurationsmenge des Anforderungsartefaktes $C(r_8)$ ist nicht vollständig abgedeckt durch die Konfigurationsmenge aller mit der Anforderung verlinkten Testartefakte $T_{r_8} = \{t_8, t_9\}$. Die entsprechende Differenz der von den Testfällen nicht abgedeckten Konfigurationen ist gegeben durch

$$\begin{aligned}
 \text{Klasse 2.3:} \quad & C(r_8) \supset C(T_{r_8}) \\
 \Rightarrow C_{def}(T_{r_8}) &= C(r_8) \setminus C(T_{r_8}) & (5.24) \\
 &= \{c_8, c_9, c_{10}, c_{11}, c_{12}\} \setminus \{c_8, c_{11}, c_{12}\} \\
 &= \{c_9, c_{10}\}
 \end{aligned}$$

Würden c_9 , c_{10} direkt als Korrekturmaßnahme angegeben werden, würde es sich um schwer überschaubare Ausdrücke handeln, insbesondere wenn es sich um deutlich mehr als zwei fehlende/überschüssige Konfigurationen handelt. Auch die Tatsache, dass für größere Variabilitätsmodelle die einzelnen c_x sehr lang werden können, macht sie für den Anwender als Maßnahmen schwer umsetzbar. Die Konfigurationsmengen eignen sich nicht als Korrekturvorschläge und sollten aus diesem Grund in einzelne Merkmale übersetzt werden, welche dann an entsprechender Stelle ergänzt bzw. entfernt werden sollten (vgl. letzte Spalte der Tabelle 10). Bei fehlenden Konfigurationsmengen wird hierfür mithilfe der Tabelle 9 geprüft, welche Merkmale der Differenzmenge $F_{def}(C_{def}(T_{r_i}))$ in den jeweiligen *kategorisierten* Testartefakten nicht vorkommen. Für das oben betrach-

tete Beispiel wird konkret untersucht, welche Merkmale aus $F_{\text{def}}(C_{\text{def}}(T_{r_8}))$ in den Testfällen t_8 bzw. t_9 fehlen:

$$\begin{aligned}
 F_{\text{def}}(C_{\text{def}}(T_{r_8})) &= \{f_2, f_3, f_7, f_8\} \\
 F(t_8) &= \{f_2, f_3, f_6\} \\
 \Rightarrow \mathbf{F}_{\text{def}}(t_8) &= F_{\text{def}}(C_{\text{def}}(T_{r_8})) \setminus F(t_8) \\
 &= \{f_7, f_8\}
 \end{aligned} \tag{5.25}$$

$$\begin{aligned}
 F(t_9) &= \{f_2, f_4, f_6, f_7, f_8\} \\
 \Rightarrow \mathbf{F}_{\text{def}}(t_9) &= F_{\text{def}}(C_{\text{def}}(T_{r_8})) \setminus F(t_9) \\
 &= \{f_3\}
 \end{aligned} \tag{5.26}$$

Um die Konsistenz zwischen r_8 und den damit verknüpften Testartefakten herzustellen, sollten somit in t_8 die Merkmale $f_7=HV$ Klein und $f_8=Kein HV$ oder in t_9 das Merkmal $f_3=SH$ ergänzt werden.

Weiterhin wird die in Abschnitt 5.4.1.6 diskutierte Klasse 3.4 der Übervollständigkeit herangezogen. Die Konfigurationsmenge des Testartefaktes $C(t_{12})$ deckt die Konfigurationsmenge aller mit t_{12} verlinkten Anforderungsartefakte $R_{t_{12}} = \{r_{12}, r_{13}, r_{14}\}$ mehr als vollständig ab (vgl. Fall (d) in Abbildung 33). Die entsprechende Differenz der von den Testfällen überflüssig abgedeckten Konfigurationen $C_{\text{exc}}(t_{12})$ und Merkmale $F_{\text{exc}}(t_{12})$ ist gegeben durch:

$$\begin{aligned}
 \text{Klasse 3.4: } & [C(R_{t_{12}}) \not\subseteq C(t_{12})] \wedge [C(R_{t_{12}}) \not\supseteq C(t_{12})] \\
 \Rightarrow \mathbf{C}_{\text{exc}}(t_{12}) &= C(t_{12}) \setminus C(R_{t_{12}}) \\
 &= \{c_1, c_2, c_3, c_8, c_9, c_{10}\} \setminus \{c_3, c_5, c_7, c_8, c_9, c_{10}\} \\
 &= \{c_1, c_2\}
 \end{aligned} \tag{5.27}$$

$$\begin{aligned}
 F_{\text{exc}}(C_{\text{exc}}(t_{12})) &= \{f_1, f_3, f_6, f_7\} \\
 F(r_{12}) &= \{f_2, f_3, f_6, f_7, f_8\} \\
 \Rightarrow \mathbf{F}_{\text{exc}}(t_{12}) &= F_{\text{exc}}(C_{\text{exc}}(t_{12})) \setminus F(r_{12}) \\
 &= \{f_1\}
 \end{aligned} \tag{5.28}$$

Zum Wiederherstellen der Konsistenz sollte das Merkmal $f_1=KA$ aus dem Merkmal-mapping von t_{12} entfernt werden. Für Klasse 3.4 können nur überflüssige Merkmale für den Testfall des Ausgangslinks t_{12} angegeben werden. Alle weiteren zur Untersuchung herangezogenen Verknüpfungen müssten gesondert analysiert und kategorisiert werden, bevor für diese Handlungsempfehlung formuliert werden können.

Der letzte Fall stellt eine gleichzeitige Un- sowie Übervollständigkeit dar. Demnach existiert sowohl die defizitäre Konfigurationsmenge $C_{\text{def}}(T_{r_i})$ mit entsprechenden fehlenden Merkmalen $F_{\text{def}}(t_j)$, als auch die überflüssige Konfigurationsmenge $C_{\text{exc}}(t_j)$

mit den überschüssigen Merkmalen $F_{exc}(t_j)$ (vgl. Fall (e) in Abbildung 33). Dies wird mittels der in Abschnitt 5.4.1.6 eingeführten Klasse 4.2.4 erläutert:

$$\begin{aligned}
 \text{Klasse 4.2.4:} \quad & [C(r_{15}) \cap C(t_{13}) \neq \emptyset] \wedge \\
 & [C(r_{15}) \not\supseteq C(t_{13})] \wedge [C(r_{15}) \not\subseteq C(t_{13})] \\
 \Rightarrow \mathbf{C}_{def}(T_{r_{15}}) = & C(r_{15}) \setminus C(T_{r_{15}}) & (5.29) \\
 = & \{c_4, c_{11}\} \setminus \{c_2, c_{11}, c_{12}\} \\
 = & \{c_4\}
 \end{aligned}$$

$$\begin{aligned}
 \Rightarrow \mathbf{C}_{exc}(t_{13}) = & C(t_{13}) \setminus C(R_{t_{13}}) & (5.30) \\
 = & \{c_{11}, c_{12}\} \setminus \{c_4, c_8, c_{11}, c_{13}\} \\
 = & \{c_{12}\}
 \end{aligned}$$

$$\begin{aligned}
 F_{def}(C_{def}(T_{r_{15}})) = & \{f_1, f_4, f_6\} \\
 F(t_{13}) = & \{f_2, f_4, f_6, f_7, f_8\} \\
 \Rightarrow \mathbf{F}_{def}(t_{13}) = & C_{def}(T_{r_{15}}) \setminus F(t_{13}) & (5.31) \\
 = & \{f_1\}
 \end{aligned}$$

$$\begin{aligned}
 F_{exc}(C_{exc}(t_{13})) = & \{f_2, f_4, f_8\} \\
 F(r_{15}) = & \{f_1, f_2, f_4, f_6\} \\
 \Rightarrow \mathbf{F}_{exc}(t_{13}) = & F_{exc}(C_{exc}(t_{13})) \setminus F(r_{15}) & (5.32) \\
 = & \{f_8\}
 \end{aligned}$$

In dem Beispiel aus Abbildung 31 fehlt somit in t_{13} das Merkmal $f_1=KA$ und gleichzeitig ist $f_8=Kein HV$ überschüssig. Auch hier kann nur ein Korrekturvorschlag bezüglich des klassifizierten Ausgangslinks $r_{15} \leftrightarrow t_{13}$ gemacht werden. Da alle anderen Links zur Kategorisierung separat betrachtet werden müssen, werden für diese keine Korrekturvorschläge gemacht.

Die Konsistenzprüfung unterstützt einen Anwender bei der Korrektur semi-automatisch. Hierfür wird die Stelle, an welcher die Inkonsistenz auftritt, identifiziert und aufgezeigt. Gegebenenfalls werden die zur vollständigen Abdeckung fehlenden bzw. überschüssigen Merkmale pro Verlinkung berechnet sowie dem Anwender präsentiert. Auch weitere, oben beschriebene Korrekturvorschläge, welche sich auf vergessene, unnötig oder falsch gesetzte Links sowie mangelnde bzw. überflüssig spezifizierte (Test-)Artefakte beziehen, werden angebracht. Eine vollautomatische Umsetzung der Korrekturvorschläge ist nicht möglich, weil aus den bereitgestellten die *richtige(n) Maßnahme(n)* ausgewählt sowie das/die *konkrete(n) Artefakt(e)* identifiziert werden müssen. Zu diesem Zweck muss der Inhalt und damit die Semantik der Artefakte studiert und verstanden werden, was nicht automatisiert geschehen kann.

Komplexitätsanalyse der Konsistenzprüfung

Die *Zeitkomplexität* der Konsistenzprüfung hängt von mehreren Faktoren ab. Vor der Durchführung der eigentlichen Prüfung wird zunächst die Basis (vgl. Abschnitt 5.4.1.1) bereitgestellt. Zum einen wird hierfür Tabelle 9 mit den Konfigurationen C_{all} ermittelt. Gleichzeitig werden für jede Konfiguration die darin enthaltenen Merkmale bestimmt.

Die Bereitstellung der Tabelle 9 wird für die Durchführung der Konsistenzprüfung als Voraussetzung angenommen und geht daher nicht in die anschließende Laufzeitdiskussion der Prüfungsmethodik ein.

Die *Laufzeit* der Konsistenzprüfung hängt im Wesentlichen von der Anzahl der zu analysierenden Verlinkungen ab. Dies ist gegeben durch $\mathcal{O}(\sum_{L_i} |C_{\text{Vergleich}, L_i}|)$, wobei $C_{\text{Vergleich}, L_i}$ der Menge aller Konfigurationen, welche bei den jeweiligen Verlinkungen L_i miteinander verglichen werden, entspricht. Die Vergleichsoperation wird durch die Verwendung der Indexstruktur einer sogenannten Hashtabelle unterstützt, welche zur Suche eines Eintrages im Mittel einen konstanten Zeitaufwand benötigt. Maximal müssen pro Verlinkung nahezu alle Konfigurationen miteinander verglichen werden, sodass gilt $|C_{\text{Vergleich}, L_i}| \approx |C_{\text{all}}|$.

Zudem trägt die Berechnung der Differenzmengen im Bezug auf die Konfigurationen sowie der fehlenden bzw. überschüssigen Merkmale für alle inkonsistenten Verlinkungen L_{ink} mit $\mathcal{O}(\sum_{L_{\text{ink}}} |C_{\text{def/exc}, L_{\text{ink}}}| \cdot |F_{\text{def/exc}, L_{\text{ink}}}|)$ zu der Laufzeit bei. Für den Fall, dass alle Verlinkungen L_{all} zu einer Kategorie der Inkonsistenz gehören, pro Link nahezu alle Konfigurationen sowie pro Link nahezu alle Merkmale überflüssig sind bzw. fehlen, gilt $|L_{\text{ink}}| = |L_{\text{all}}|$, $|C_{\text{def/exc}}| \approx |C_{\text{all}}|$ und $|F_{\text{def/exc}}| \approx |F_{\text{all}}|$.

Die Konsistenzprüfung besitzt somit eine maximale Laufzeit von:

$$\mathcal{O}(|L_{\text{all}}| \cdot |C_{\text{all}}| + |L_{\text{all}}| \cdot |C_{\text{all}}| \cdot |F_{\text{all}}|). \quad (5.33)$$

Beide Summanden der Formel 5.33 hängen direkt von der Anzahl der vorliegenden Verknüpfungen zwischen Anforderungen und Testfällen ab. Für den oben beschriebenen Extremfall dominiert der letzte Term, welcher die Berechnung der Korrekturmaßnahmen repräsentiert.

Im Normalfall ist die Laufzeit der Konsistenzprüfung abhängig von:

$$\mathcal{O}\left(\sum_{L_i} |C_{\text{Vergleich}, L_i}| + \sum_{L_{\text{ink}}} |C_{\text{def/exc}, L_{\text{ink}}}| \cdot |F_{\text{def/exc}, L_{\text{ink}}}| \right). \quad (5.34)$$

Ob der erste oder zweite Term dominiert, hängt davon ab, wie viele inkonsistente Verknüpfungen vorliegen. Zusätzlich kann es eine Rolle spielen, zu welcher Kategorien-Klasse die analysierte Verlinkung gehört. So ist die Rechenzeit kürzer, wenn mehrere Verlinkungen gleichzeitig einer Kategorien-Klasse zugeordnet werden können. Dies ist für die Klassen 2.1, 2.3, 3.1, 3.3 der Fall. Dagegen braucht die Prüfung länger, wenn die Verlinkungen einzeln kategorisiert werden müssen.

Der *Speicherbedarf* der Konsistenzprüfung wird bestimmt von der anfänglichen Erstellung der Listen, in denen die von den Artefakten abgedeckten Konfigurationen $\mathcal{O}(|C_{\text{all}}| \cdot |R_{\text{all}}| + |C_{\text{all}}| \cdot |T_{\text{all}}|)$ sowie von den Konfigurationen abgedeckten Merkmale $\mathcal{O}(|C_{\text{all}}| \cdot |F_{\text{all}}|)$ berechnet werden. Darüber hinaus werden pro inkonsistenten Link die fehlenden bzw. überschüssigen Merkmale gespeichert mit $\mathcal{O}(\sum_{L_{\text{ink}}} |F_{\text{def/exc}}|)$, wobei

maximal alle Links inkonsistent $|L_{ink}| = |L_{all}|$ sowie nahezu alle Merkmale fehlen bzw. überschüssig $|F_{def/exc}| \approx |F_{all}|$ sein können.

Die Konsistenzprüfung besitzt somit einen maximalen Speicherbedarf von:

$$\mathcal{O}(|C_{all}| \cdot |R_{all}| + |C_{all}| \cdot |T_{all}| + |C_{all}| \cdot |F_{all}| + |L_{all}| \cdot |F_{all}|). \quad (5.35)$$

Für den Normalfall hängt der Speicherbedarf der Konsistenzprüfung ab von:

$$\mathcal{O}(|C_{all}| \cdot |R_{all}| + |C_{all}| \cdot |T_{all}| + |C_{all}| \cdot |F_{all}| + \sum_{L_{ink}} |F_{def/exc, L_{ink}}|). \quad (5.36)$$

Im Normalfall wird die Speicherkomplexität dominiert von $\mathcal{O}(|C_{all}| \cdot |R_{all}|)$, da in der Regel gilt $|F_{all}| \ll |T_{all}| < |R_{all}|$. Bei zahlreichen vorliegenden Inkonsistenzen sowie einem Merkmalmodell mit entsprechend vielen Merkmalen kann jedoch auch der letzte Term vorherrschend sein.

Zusammenfassung

Dieses Kapitel stellt eine einheitliche Grundlage im Umgang mit der Variabilität in Anforderungs- sowie Testspezifikationen vor. Ausgehend von einem universellen Variantenmanagement, dessen zentrales Element dem gemeinsamen Variantenmodul entspricht, werden die Variabilitätsinformationen den Entwicklungsartefakten zugeordnet. Aus den somit generischen Spezifikationsdokumenten lassen sich sämtliche produktspezifische Artefakte schnell ableiten und wiederverwenden.

Darüber hinaus wird das Konzept eines Konsistenzmanagements ausgearbeitet, welches die abgebildete Variabilität zwischen Anforderungen sowie den damit verlinkten Testfällen untersucht. Die schrittweise vorgestellte Methode unterstützt bei der Analyse komplexer Zusammenhänge, welche sich durch die verwendete Assoziationslogik in konjunktiver Normalform⁵ ergeben. Zusätzlich werden konkrete Handlungsempfehlungen zur Beseitigung der Inkonsistenzen identifiziert und dem Anwender an entsprechender Stelle bereitgestellt. Die Tool-gestützte Konsistenzprüfung unterstützt die Durchgängigkeit zwischen Anforderungs- und Testartefakten bzw. verschiedenen Entwicklungsphasen. Das abstrakte Konzeptgerüst lässt sich zur Überprüfung der Konsistenz zwischen weiteren Artefakttypen übertragen und anwenden. Mithilfe des universellen Variantenmanagements sowie des konsistenten Merkmalmappings ist eine widerspruchsfreie Basis gegeben, die für die Selektion von zu testenden Varianten und zur Identifikation der durchzuführenden Testfälle im anschließenden Kapitel verwendet wird.

⁵ Die bei disjunktiver Assoziationslogik notwendige Konsistenzprüfung, bei der die zugeordneten Merkmale der Artefakte direkt miteinander verglichen werden, wird in Anhang A vorgestellt.

6 | SELEKTIONSMETHODIK

Das vorliegende Kapitel beschreibt eine Methodik zur Bestimmung von zu testenden Konfigurationen für einen effizienten Produktlinientest auf Basis der zuvor definierten Produktlinieninfrastruktur. In Abschnitt 6.1 wird das Kapitel mit einer Übersicht über das zugrunde liegende Konzept, welches auf einer optimierten Anforderungs- und Merkmalabdeckung beruht, eingeführt. Daraufaufgehend wird in Abschnitt 6.2.1 die für die Selektionsmethodik notwendige Basis ausgearbeitet. Für die Optimierung werden zwei verschiedene Verfahren eingesetzt: einerseits der *Greedy-Algorithmus*, dargestellt in Abschnitt 6.2.2 sowie andererseits die *Simulierte Abkühlung*, vorgestellt in Abschnitt 6.2.3. Anschließend an die Beschreibung beider Algorithmen werden die Vorgehen, jeweils adaptiert an die in Kapitel 5 behandelte Produktlinieninfrastruktur, präsentiert. Schließlich werden in Abschnitt 6.3 Methoden zur Selektion und Wiederverwendung von zu den Konfigurationen passenden Testfällen diskutiert.

6.1 Konzept der Optimierung

Ausgehend von der in Kapitel 5 ausgearbeiteten widerspruchsfreien Produktlinieninfrastruktur wird ein Vorgehen zur Bestimmung einer Untermenge an zu testenden Konfigurationen mittels zweier Optimierungsalgorithmen definiert. Entsprechende Testfälle werden auf Basis der generischen Testspezifikation abgeleitet. In Abbildung 34 ist ein Überblick über die Bestandteile der Selektionsmethodik schematisch dargestellt.

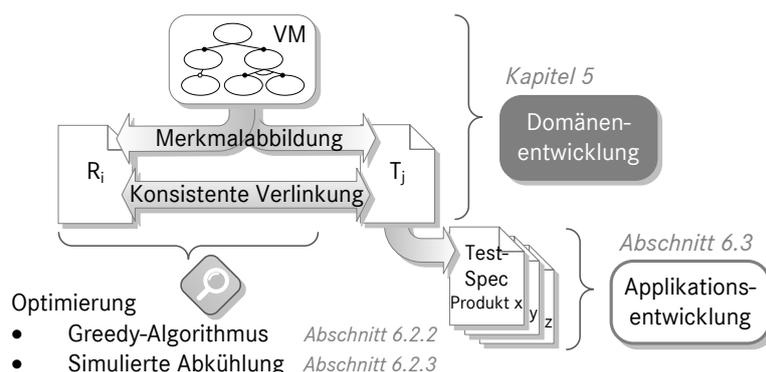


Abbildung 34: Übersicht über die Produktlinieninfrastruktur sowie die darauf aufbauende Selektionsmethodik mit den jeweils diskutierten Elementen innerhalb der entsprechenden Abschnitte des vorliegenden Kapitels

Die Kernidee der Selektionsmethodik beruht auf der Ermittlung von zu testenden Konfigurationen auf Basis einer optimierten Abdeckung von generischen Anforderungen, welche im Lastenheft dokumentiert sind, und einer optimierten Abdeckung der im Merkmalmodell enthaltenen Merkmale. Durch das Testen dieser Konfigurationen wird eine repräsentative Aussage bezüglich der Absicherung der gesamten Produktlinie ermöglicht.

Durch eine Anforderungsabdeckung wird gewährleistet, dass jede Anforderung in mindestens einer Konfiguration berücksichtigt und dadurch getestet ist.

Definition 25 (Anforderungsabdeckung). *Die Anforderungsabdeckung ist definiert als das Verhältnis der Anforderungen, welche von mindestens einer selektierten Konfiguration abgedeckt werden, zu der Anzahl aller Anforderungen in der generischen Spezifikation.*

Grundsätzlich spiegeln Anforderungen sämtliche Funktions-, Qualitätseigenschaften sowie Randbedingungen in Bezug auf eine Produktlinie wider (vgl. Abschnitt 2.3). Resultierend aus dem Variantenmanagement enthalten sie zusätzlich die Information, welche dieser Eigenschaften für die gesamte Produktlinie, d.h. generischen Charakters sind, und welche nur für spezifische Konfigurationen Gültigkeit haben. Dies ist in Form des Merkmal mappings in den Anforderungen explizit abgebildet (vgl. Abschnitt 5.3). Auf dieser Informationsgrundlage kann einerseits die vollständige Anforderungsabdeckung durch möglichst wenige Konfigurationen erreicht und andererseits können insbesondere solche Konfigurationen, welche spezielle Merkmalkombinationen in den Anforderungen abbilden, berücksichtigt werden. Dadurch wird insgesamt eine effiziente Abdeckung aller in der Spezifikation dargestellten Eigenschaften der Produktlinie ermöglicht.

Die generische Anforderungsspezifikation ist als Selektionsbasis besonders gut geeignet, da diese als initiales Dokument im Entwicklungsprozess vorliegt. Die Anforderungsartefakte fungieren somit als zentrale Elemente, welche zum allgemeinen Verständnis über das Bezugsobjekt für alle weiteren Entwicklungsphasen zur Verfügung stehen. Darüber hinaus weisen Anforderungsspezifikationen in der Regel einen guten Entwicklungsstand, aufgrund von zahlreichen Reviews wie etwa der Testbasisanalyse (vgl. Abschnitt 2.5.2), auf.

Da sich die häufig eingesetzten kombinatorischen Testmethoden in der Praxis bewährt haben, wird zusätzlich zur Anforderungsabdeckung eine Merkmalabdeckung (1-wise bzw. $t = 1$) angestrebt (vgl. Abschnitt 4.2.6).

Definition 26 (Merkmalabdeckung). *Die Merkmalabdeckung ist definiert als das Verhältnis der Merkmale, welche in mindestens einer selektierten Konfiguration enthalten sind, zu der Anzahl aller Merkmale im Variabilitätsmodell.*

Auf diese Weise soll jedes im Variabilitätsmodell vorhandene Merkmal in mindestens einer selektierten Konfiguration berücksichtigt und abgesichert werden. In der Regel liegen in realen Projekten (vgl. Abschnitt 7.2) weitaus mehr Anforderungen als Merkmale vor, sodass durch eine Anforderungsabdeckung gleichzeitig eine Merkmalabdeckung erreicht werden kann.

Die selektierten Konfigurationen sind als eine minimale Menge zu betrachten, welche zur vollständigen Anforderungs- und Merkmalabdeckung notwendigerweise getestet werden müssen. Gegebenenfalls können zusätzlich zur anforderungsbasierten Vorauswahl höhere t -Abdeckungen $t > 1$, z.B. mithilfe der Pairwise-Methode, generiert werden. Dies kann etwa für sicherheitsrelevante Testobjekte sinnvoll sein, welche gemäß der ASIL C bzw. ASIL D Einstufung getestet werden.

Durch die Anforderungs- und Merkmalabdeckung ist sichergestellt, dass die Eigenschaften der gesamten Produktlinie berücksichtigt sind. Aus Sicht der Norm ISO 26262 ist die Anforderungsabdeckung bereits als ein zu erfüllendes Testabdeckungskriterium für alle (Sicherheits-)Einstufungen vorgegeben (vgl. Abschnitt 2.5.1). Weiterhin ist das Kriterium der Variantenabdeckung, welches ebenfalls für alle (Sicherheits-)Einstufungen gilt, zu erfüllen. Hierbei erlaubt jedoch die Norm ISO 26262 die Selektion einer sinnvoll zu testenden Untermenge aufgrund der enormen Anzahl an Konfigurationen (4-8.4.1.4) [48]. Was dabei als sinnvoll erachtet werden kann, wird mit der oben aufgeführten Argumentation adressiert.

Durch die optimierte Anforderungs- und Merkmalabdeckung liegt eine fundierte, systematische und effiziente Methode zur Bestimmung der Konfigurationen für den Test vor. Da die vorhandene Produktlinieninfrastruktur als Basis für die Selektion genutzt wird, entsteht kein Zusatzaufwand in der Bereitstellung bzw. Aufbereitung von weiteren Informationen. Auch ist eine sofortige Aktualisierung der Testumfänge bei Veränderung oder Anpassung der Produktlinieneigenschaften in den Anforderungen möglich. Das explizite Vorgehen wird in den nächsten Abschnitten detailliert vorgestellt.

6.2 Optimierungsvorgehen

Im Folgenden wird das *Optimierungsvorgehen* zur Selektion von zu testenden Varianten sukzessive aufgebaut. In Abbildung 35 ist ein Überblick über die einzelnen Schritte der Methode dargestellt.

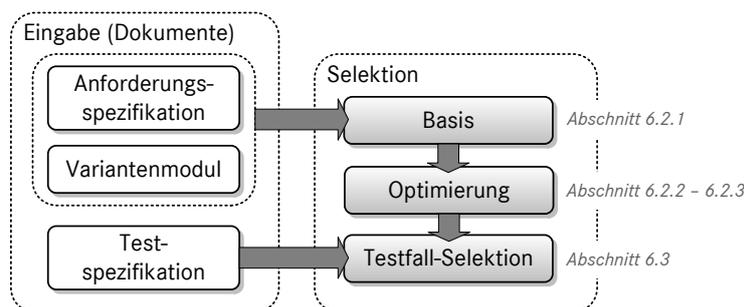


Abbildung 35: Überblick über die aufeinanderfolgenden Schritte der Selektionsmethodik unter Angabe der Abschnitte, in denen entsprechende Elemente diskutiert werden

In Abschnitt 6.2.1 wird zunächst die *Basis* der Selektionsmethode eingeführt. In Rahmen dieser Phase werden die für die Selektion notwendigen Informationen bezüglich der

Variabilität aus der Anforderungsspezifikation sowie dem Variantenmodul einheitlich aufbereitet. Der nächste Schritt der *Optimierung* kennzeichnet den Schwerpunkt der Selektionsmethodik. Dieser wird in den Abschnitten 6.2.2 bis 6.2.3 diskutiert und anhand eines Beispiels veranschaulicht. Der letzte Schritt beschreibt die Auswahl von passenden Testfällen für die selektierten Konfigurationen ausgehend von der generischen Testspezifikation und wird in Abschnitt 6.3 vorgestellt.

6.2.1 Basis des Optimierungsvorgehens

Analog zur Beschreibung der Konsistenzprüfung im vorhergehenden Kapitel wird mit einer Definition der für das Optimierungsvorgehen erforderlichen *Basis* begonnen. Ausgehend von dieser Basis erfolgt die Selektion mithilfe der Algorithmen. Zur Veranschaulichung der Zusammenhänge wird das in Abschnitt 5.2 bereits eingeführte Variantenmodul des fiktiven Systems *Klimatisierung* verwendet.

Einen zentralen Bestandteil der Basis bilden die ausgehend von Tabelle 9 erweiterten Konfigurationsausdrücke c_x . Diese sind in Tabelle 11 für das System *Klimatisierung* dargestellt und gehen aus dem Merkmalmapping in konjunktiver Normalform hervor. Dadurch, dass nur diese Konfigurationen als Basis für die Selektion zur Verfügung gestellt und nicht alle möglichen bestimmt werden, wird das Problem eingegrenzt und eine effizientere Lösung ermöglicht.

c_x	$\text{cov}(c_x, f_1),$ $f_1 = KA$	$\text{cov}(c_x, f_2),$ $f_2 = SK$	$\text{cov}(c_x, f_3),$ $f_3 = SH$	$\text{cov}(c_x, f_4),$ $f_4 = PH$	$\text{cov}(c_x, f_5),$ $f_5 = LH$	$\text{cov}(c_x, f_6),$ $f_6 = HV \text{ Groß}$	$\text{cov}(c_x, f_7),$ $f_7 = HV \text{ Klein}$	$\text{cov}(c_x, f_8),$ $f_8 = \text{Kein HV}$	$ F_{\text{cov}}(c_x) $
c_1	1		1			1	0	0	3
c_2	1		1	0	0	0	1	0	3
c_3	1		1			0	0	1	3
c_4	1			1		1	0	0	3
c_5	1			1		0	0	1	3
c_6	1			1	1	1	0	0	4
c_7	1			1	1	0	0	1	4
c_8		1	1			1	0	0	3
c_9		1	1	0	0	0	1	0	3
c_{10}		1	1			0	0	1	3
c_{11}		1		1		1	0	0	3
c_{12}		1		1		0	0	1	3
c_{13}		1		1	1	1	0	0	4
c_{14}		1		1	1	0	0	1	4

Tabelle 11: Beim Merkmalmapping in konjunktiver Normalform abbildbare Konfigurationen c_x (gemäß 5.5) für das System *Klimatisierung* sowie die Anzahl $|F_{\text{cov}}(c_x)|$ wie viele Merkmale von jeweils einer Konfiguration abgedeckt werden.

In Tabelle 11 wird für jedes Merkmal f_y , welches in der Konfiguration enthalten ist, der Eintrag 1, für jedes nicht enthaltene Merkmal der Eintrag 0 vorgenommen. Analog dazu ließe sich für $\neg f_y$, welches in der Konfiguration enthalten ist, der Eintrag 0 sowie für jedes nicht enthaltene Merkmal $\neg f_y$ der Eintrag 1 vornehmen (in Term 6.1 grau

hervorgehoben). Für die Abdeckung der Merkmale durch jeweils eine Konfiguration $F_{\text{cov}}(c_x)$ gilt somit:

$$\text{cov}(c_x, f_y) \text{ bzw. } \text{cov}(c_x, \neg f_y) = \begin{cases} 1 \text{ bzw. } 0 & \text{falls } f_y \in F_{\text{cov}}(c_x) \\ 0 \text{ bzw. } 1 & \text{falls } f_y \notin F_{\text{cov}}(c_x) \\ \emptyset & \text{noch nicht definiert.} \end{cases} \quad (6.1)$$

Zusätzlich werden in Tabelle 11 die durch das Variantenmodul vorgegebenen Abhängigkeiten und Constraints, welche als graue Einträge hervorgehoben sind, berücksichtigt. Diese sind notwendig, um die Selektion von ausschließlich validen Konfigurationen sicherzustellen. So enthält die Konfiguration c_6 den Eintrag 1 für das Merkmal $f_4 = PH$, da das durch das kartesische Produkt vorgegebene Merkmal $f_5 = LH$ dieses benötigt. In einigen Feldern ist keine explizite Angabe gemacht, ob ein Merkmal in der jeweiligen Konfiguration enthalten ist oder nicht. Dies ist bspw. für c_4 der Fall, in der das Merkmal $f_2 = SK$ enthalten sein kann oder nicht. Ein c_x kann somit eine unvollständige Konfiguration darstellen, in der Entscheidungen im Rahmen der Selektion noch zu treffen sind. Dieser Freiheitsgrad wird zwecks einer optimierten Merkmalabdeckung genutzt, was in den Abschnitten 6.2.2.2 sowie 6.2.3.2 näher behandelt wird. Schließlich ist in der letzten Spalte mithilfe von $|F_{\text{cov}}(c_x)|$ zusammengefasst, wie viele Merkmale eine Konfiguration jeweils explizit abdeckt.

In Tabelle 12 ist eine beispielhafte Anforderungsspezifikation dargestellt. Anhand der den Anforderungen zugeordneten Merkmale wird das Vorgehen der Greedy-Optimierung in Abschnitt 6.2.2.2 demonstriert.

Anforderungen, r_i	Kühlmaßnahme	Heizmaßnahme	HV-Batterie
r_1			HV Klein
r_2	Klimaanlage Automatik	Panelheizung Lenkradheizung	
r_3			
r_4	Sitzkühlung	Sitzheizung	HV Klein
r_5	Sitzkühlung		Kein HV
r_6	Klimaanlage Automatik		HV Groß

Tabelle 12: Merkmalmapping in der beispielhaften Anforderungsspezifikation für das System *Klimatisierung*

Hierfür wird für jede Konfiguration bestimmt, welche die in der Spezifikation (vgl. Tabelle 12) aufgelisteten Anforderungen $R_{\text{cov}}(c_x)$ abgedeckt werden. Dies ist in Tabelle 13 gezeigt, wobei für jede Konfiguration c_x , welche eine Anforderung r_i abdeckt, der Eintrag 1 und sonst 0 vorgenommen wird. Damit gilt:

$$\text{cov}(c_x, r_i) = \begin{cases} 1 & \text{falls } r_i \in R_{\text{cov}}(c_x) \\ 0 & \text{sonst.} \end{cases} \quad (6.2)$$

So ist die Anforderung r_1 für c_2 und c_9 gültig. Des weiteren ist in der letzten Spalte der Tabelle 13 die Kardinalität der Menge $R_{\text{cov}}(c_x)$ dargestellt, welche die Anzahl der von der jeweiligen Konfiguration, abgedeckten Anforderungen kennzeichnet. Schließlich zeigt die unterste Zeile mithilfe von $C_{\text{cov}}(r_i)$, wie viele Konfigurationen eine Anforderung abdecken.

c_x	$\text{cov}(c_x, r_1)$	$\text{cov}(c_x, r_2)$	$\text{cov}(c_x, r_3)$	$\text{cov}(c_x, r_4)$	$\text{cov}(c_x, r_5)$	$\text{cov}(c_x, r_6)$	$ R_{\text{cov}}(c_x) $
c_1	0	0	1	0	0	1	2
c_2	1	0	1	0	0	0	2
c_3	0	0	1	0	0	0	1
c_4	0	1	1	0	0	1	3
c_5	0	1	1	0	0	0	2
c_6	0	1	1	0	0	1	3
c_7	0	1	1	0	0	0	2
c_8	0	0	1	0	0	0	1
c_9	1	0	1	1	0	0	3
c_{10}	0	0	1	0	1	0	2
c_{11}	0	0	1	0	0	0	1
c_{12}	0	0	1	0	1	0	2
c_{13}	0	0	1	0	0	0	1
c_{14}	0	0	1	0	1	0	2
$ C_{\text{cov}}(r_i) $	2	4	14	1	3	3	

Tabelle 13: Abbildung welche Konfigurationen für welche Anforderungen gelten $\text{cov}(c_x, r_i)$, wie viele Anforderungen von jeweils einer Konfiguration abgedeckt werden $|R_{\text{cov}}(c_x)|$ sowie wie viele Konfigurationen jeweils eine Anforderung abdecken $|C_{\text{cov}}(r_i)|$.

Ausgehend von der vorgestellten Basis (vgl. Tabelle 11 sowie 13) erfolgt die Selektion der zu testenden Konfigurationen mittels der im Folgenden behandelten Optimierungsverfahren.

6.2.2 Optimierung mittels Greedy-Algorithmus

Die Selektion der repräsentativen, zu testenden Konfigurationsmenge, welche eine optimierte Anforderungs- sowie Merkmalabdeckung gewährleistet, stellt ein kombinatorisches (Minimierungs-)Problem dar. Da das vollständige Ausprobieren aller Lösungsmöglichkeiten, d.h. Konfigurationskombinationen, einen enormen Rechenaufwand bedeuten würde, werden heuristische Lösungsverfahren eingesetzt. Diese können zwar nicht garantieren, dass eine optimale Lösung gefunden wird, jedoch kann eine Heuristik das Problem näherungsweise lösen. Als erster Ansatz wird der weitgehend deterministische Greedy-Algorithmus als Lösungsverfahren angewendet, um den Lösungsraum zu erkunden.

6.2.2.1 Greedy-Algorithmus

Greedy-Algorithmen (engl. *greedy*: gierig) gehören zu einer speziellen Klasse von Algorithmen zur Lösung von Optimierungsproblemen. Sie bestimmen das Optimum durch eine Folge von Entscheidungen, welche zum Zeitpunkt der Wahl jeweils die bestmögliche darstellen. Durch ein iteratives Erweitern der Lösung mit diesen Entscheidungen wird die Gesamtlösung ermittelt [124]. Dabei geschieht die Betrachtung der besten Auswahlmöglichkeit lokal, sodass keine Analyse der globalen Situation erfolgt. Darüber hinaus werden bereits getroffene Entscheidungen nicht mehr revidiert. Dies kann dazu führen, dass das Verfahren lediglich ein lokales anstatt des eigentlich gewünschten globalen Optimums liefert.

Der große Vorteil von Greedy-Algorithmen liegt darin, dass sie mit wenig Zusatzaufwand entworfen und implementiert werden können [45]. Dadurch sind sie besonders gut geeignet, um vor der Implementierung eines möglicherweise umfangreicheren Algorithmus das Optimierungsproblem prüfen und besser beurteilen zu können. Zusätzlich liefern sie im Allgemeinen gute bis sehr gute Ergebnisse in einer annehmbaren Ausführungszeit. Insbesondere für NP-vollständige Probleme, zu dem das Vorliegende zählt, ist die zeitliche Effizienz wichtig, da eine exakte Berechnung der optimalen Lösung zu aufwändig wäre¹. Aufgrund ihrer vergleichsweise einfachen Implementierung und schnellen Ausführung wird die Greedy-Heuristik zur Ermittlung der approximativen Lösung in der Arbeit eingesetzt.

Der Entwurf und die Implementierung des Greedy-Algorithmus für die maximierte Anforderungs- und Merkmalabdeckung wird im Folgenden am Beispiel des Systems *Klimatisierung* erläutert.

6.2.2.2 Selektionsvorgehen mittels Greedy-Algorithmus

Zur Ermittlung der optimalen Menge an Konfigurationen wählt der Greedy-Algorithmus schrittweise den Folgezustand aus, welcher zum Zeitpunkt der Wahl die größte Anforderungs- und Merkmalabdeckung liefert. Hierfür werden die im Rahmen der Basis (vgl. Abschnitt 6.2.1) erstellten Tabellen 11 und 13 für die Selektion herangezogen. Zusätzlich werden die in den Tabellen abgebildeten Konfigurationen einerseits mithilfe von $|R_{\text{cov}}(c_x)|$ nach der Anforderungsabdeckung sowie andererseits mithilfe von $|F_{\text{cov}}(c_x)|$ nach der Merkmalabdeckung sortiert. Die Sortierung ist jeweils innerhalb der Datenstruktur mit einer sogenannten Prioritätswarteschlange (engl. *priority queue*) umgesetzt.

Das Vorgehen des Greedy-Algorithmus ist anhand des Pseudocodes in Algorithmus 1 veranschaulicht². In der *Phase der Initialisierung*, wird zunächst geprüft, ob Anforderungen R_{min} existieren, welche von ausschließlich einer Konfiguration abgedeckt werden (vgl. Zeile 6 in Algorithmus 1). Da solche Konfigurationen in jedem Fall gewählt werden müssen, um eine vollständige Anforderungsabdeckung zu gewährleisten, ist es

¹ Hierzu müssten alle möglichen Kombinationen der Konfigurationen ($2^{|\text{Cat}|} - 1$) berechnet und miteinander verglichen werden.

² In Anhang B sind alle innerhalb des Greedy-Algorithmus und der Simulierten Abkühlung verwendeten Abkürzungen zusammenfassend aufgelistet und erklärt.

Algorithmus 1 Greedy Configuration Selection

```

function GREEDYCONFIGSELECTION( $C_{\text{all}}, R_{\text{all}}, F_{\text{all}}$ )
2    $C_{\text{sel,filled}} \leftarrow \emptyset$ 
    $C_{\text{unsel}} \leftarrow C_{\text{all}}$   $\triangleright$  Priority Queue bzgl. Anforderungs- und Merkmalabdeckung
4    $R_{\text{uncovered}} \leftarrow R_{\text{all}}$ 
    $F_{\text{uncovered}} \leftarrow F_{\text{all}}$ 
    $\triangleright$  Phase 1 – Initialisierung:
6    $R_{\text{min}} \leftarrow \{r_{\text{min}} \in R_{\text{uncovered}} \mid |C_{\text{cov}}(r_{\text{min}})| = 1\}$ 
   while  $|R_{\text{min}}| > 0$  do
    $\triangleright$  Für Anforderungen, welche von einem  $c$  abgedeckt werden, selektiere solche  $c$ :
8    $c_s \leftarrow \mathbf{randomSelect}(\bigcup_{r_{\text{min}} \in R_{\text{min}}} C_{\text{cov}}(r_{\text{min}}) \cap C_{\text{unsel}})$ 
    $c_{s,\text{filled}} \leftarrow c_s\text{-FEATURE FILLING}(c_s, C_{\text{sel,filled}}, F_{\text{all}}, F_{\text{uncovered}})$ 
10   $C_{\text{sel,filled}} \leftarrow C_{\text{sel,filled}} \cup \{c_{s,\text{filled}}\}$ 
    $C_{\text{unsel}} \leftarrow C_{\text{unsel}} \setminus \{c_s\}$ 
12   $R_{\text{min}} \leftarrow R_{\text{min}} \setminus R_{\text{cov}}(c_s)$ 
    $R_{\text{uncovered}} \leftarrow R_{\text{uncovered}} \setminus R_{\text{cov}}(c_s)$ 
14   $F_{\text{uncovered}} \leftarrow F_{\text{uncovered}} \setminus F_{\text{cov}}(c_{s,\text{filled}})$ 
end while
    $\triangleright$  Phase 2 – Greedy-Algorithmus:
16  while  $|R_{\text{uncovered}}| > 0$  or  $|F_{\text{uncovered}}| > 0$  do
    $C_{\text{max}} = \{c_{\text{max}} \in C_{\text{unsel}} \mid \forall c_d \in C_{\text{unsel}} (|R_{\text{cov}}(c_{\text{max}})| \geq |R_{\text{cov}}(c_d)|)\}$ 
18  if  $|C_{\text{max}}| = 1$  then
    $\triangleright$  Selektiere  $c$ , welches die meisten Anforderungen abdeckt:
    $c_s \leftarrow c_{\text{max}} \in C_{\text{max}}$ 
20  else
    $C_{\text{fmax}} = \{c_{\text{fmax}} \in C_{\text{max}} \mid \forall c_d \in C_{\text{max}} (|F_{\text{cov}}(c_{\text{fmax}})| \geq |F_{\text{cov}}(c_d)|)\}$ 
    $\triangleright$  Selektiere  $c$ , welches die meisten Merkmale abdeckt:
22   $c_s \leftarrow \mathbf{randomSelect}(C_{\text{fmax}})$ 
end if
24   $c_{s,\text{filled}} \leftarrow c_s\text{-FEATURE FILLING}(c_s, C_{\text{sel,filled}}, F_{\text{all}}, F_{\text{uncovered}})$ 
    $C_{\text{sel,filled}} \leftarrow C_{\text{sel,filled}} \cup \{c_{s,\text{filled}}\}$ 
26   $C_{\text{unsel}} \leftarrow C_{\text{unsel}} \setminus \{c_s\}$ 
    $R_{\text{uncovered}} \leftarrow R_{\text{uncovered}} \setminus R_{\text{cov}}(c_s)$ 
28   $F_{\text{uncovered}} \leftarrow F_{\text{uncovered}} \setminus F_{\text{cov}}(c_{s,\text{filled}})$ 
end while
30  return  $C_{\text{sel,filled}}$ 
end function

```

sinnvoll diese zu Beginn zu selektieren. Für das System *Klimatisierung* wird die Bedingung $|C_{\text{cov}}(r_{\text{min}})| = 1$ von der Anforderung r_4 erfüllt (vgl. Tabelle 13). Diese kann ausschließlich von c_9 abgedeckt werden, weshalb c_9 zu der Menge der selektierten Konfigurationen C_{sel} hinzugefügt wird.

Da der Wert von $|C_{\text{cov}}(r_i)|$ nur zwischen dem zu Beginn berechneten und Null (nachdem die entsprechende Anforderung r_i abgedeckt wurde) variieren kann, können keine R_{min} nachträglich während der Phase der Greedy-Suche auftreten. Aus diesem Grund genügt es, die Suche nach R_{min} vor der Phase des Greedy-Algorithmus einmalig durchzuführen. Die Initialisierungsphase wird mit der Aktualisierung der noch nicht abgedeckten Anforderungen $R_{\text{uncovered}}$ sowie der noch nicht selektierten Konfigurationen C_{unsel} abgeschlossen.

In der darauf folgenden *Phase des Greedy-Algorithmus* wird nach Konfigurationen C_{max} gesucht, welche die meisten Anforderungen abdecken (vgl. Zeile 17 in Algorithmus 1). Für das System *Klimatisierung* sind es c_4 und c_6 (vgl. Tabelle 13). Aufgrund der Tatsache, dass mehrere Konfigurationen, welche zu einem maximalen Beitrag der Anforderungsabdeckung führen, vorliegen, wird aus diesen die Konfiguration mit der größten Merkmalabdeckung C_{fmax} gewählt (vgl. Zeile 21 in Algorithmus 1). Für das Beispielsystem decken die Konfigurationen c_4 und c_6 jeweils ein noch nicht berücksichtigtes Merkmal ab (vgl. Tabelle 11). Da mehrere Kandidaten mit der gleichen maximalen Merkmalabdeckung existieren, wird von diesen eine zufällig ausgewählt und als c_s gespeichert (vgl. Zeile 22 in Algorithmus 1). Schließlich werden die Mengen $R_{\text{uncovered}}$ und C_{unsel} aktualisiert. Das Vorgehen wird so lange wiederholt und die Menge der selektierten Konfigurationen vervollständigt, bis alle Anforderungen und Merkmale mindestens einmal abgedeckt sind.

Gegebenenfalls können selektierte Konfigurationen unbestimmte Merkmale enthalten und somit unvollständig sein (vgl. Tabelle 11). Solche offenen Entscheidungen werden dazu genutzt eine optimierte Merkmalabdeckung zu erreichen. Hierfür wird jede selektierte Konfiguration c_s mittels des sogenannten *Fillings* vervollständigt (vgl. Zeile 9 und 24 in Algorithmus 1). Das Befüllen der offenen Merkmaleinträge ist anhand des Pseudocodes in Algorithmus 2 dargestellt. Darüber hinaus ist der Sachverhalt sowie das iterative Erweitern der Lösung durch den Greedy-Algorithmus in Abbildung 36 schematisch verdeutlicht.

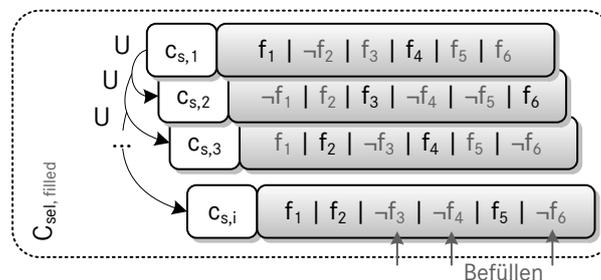


Abbildung 36: Abstrahiertes Vorgehen des Greedy-Algorithmus zur optimierten Anforderungs- und Merkmalabdeckung

Algorithmus 2 c_s -Feature Filling

```

function FILLING( $c_s, C_{sel, filled}, F_{all}, F_{uncovered}$ )
2    $c_{s, filled} \leftarrow c_s$ 
   while  $\exists f_y \in F_{all}, \text{cov}(c_{s, filled}, f_y) = \emptyset$  do
        $\triangleright$  Wurde  $f_y$  durch keine der bereits selektierten Konfigurationen abgedeckt,
       befülle undefiniertes Merkmal mit  $f_y$ :
4   if  $f_y \in F_{uncovered}$  then
        $\text{cov}(c_{s, filled}, f_y) \leftarrow 1$ 
6   else
        $\triangleright$  Berechne wie oft  $f_y$  durch bereits selektierte Konfigurationen  $c_{x, filled}$  abdeckt
       wird:
        $\text{sumCov}(f_y) \leftarrow \sum_{c_{x, filled} \in C_{sel, filled}} \text{cov}(c_{x, filled}, f_y)$ 
        $\triangleright$  Berechne wie oft  $\neg f_y$  durch bereits selektierte Konfigurationen  $c_{x, filled}$  ab-
       deckt wird:
8    $\text{sumCov}(\neg f_y) \leftarrow \sum_{c_{x, filled} \in C_{sel, filled}} \text{cov}(c_{x, filled}, \neg f_y)$ 
        $\triangleright$  Wurde  $\neg f_y$  durch bereits selektierte Konfigurationen häufiger abgedeckt als
        $f_y$ , befülle undefiniertes Merkmal mit  $f_y$ :
       if  $\text{sumCov}(f_y) < \text{sumCov}(\neg f_y)$  then
10       $\text{cov}(c_{s, filled}, f_y) \leftarrow 1$ 
        $\triangleright$  Wurde  $f_y$  durch bereits selektierte Konfigurationen häufiger abgedeckt als
        $\neg f_y$ , befülle undefiniertes Merkmal mit  $\neg f_y$ :
       else if  $\text{sumCov}(f_y) > \text{sumCov}(\neg f_y)$  then
12       $\text{cov}(c_{s, filled}, f_y) \leftarrow 0$ 
        $\triangleright$  Wurde  $\neg f_y$  durch bereits selektierte Konfigurationen genauso häufig abge-
       deckt wie  $f_y$ , befülle undefiniertes Merkmal zufällig:
       else
14       $\text{cov}(c_{s, filled}, f_y) \leftarrow \text{randomSelect}\{0, 1\}$ 
       end if
16   end if
   end while
18   return  $c_{s, filled}$ 
end function

```

Im Rahmen des Fillings wird ein Merkmal f_y , welches von keiner bis dahin selektierten Konfiguration abgedeckt ist, zugunsten einer maximierten Merkmalabdeckung dem betrachteten c_s hinzugefügt und es gilt $\text{cov}(c_{s, filled}, f_y) = 1$ (vgl. Zeile 4-5 in Algorithmus 2). Falls ein Merkmal f_y bereits abgedeckt wurde, wird zwecks einer gleichmäßigen Merkmalselektion danach entschieden, wie oft dieses Merkmal in den selektierten Konfigurationen enthalten f_y bzw. nicht enthalten $\neg f_y$ ist (vgl. Zeile 7-8 in Algorithmus 2). Ist $\neg f_y$ in $C_{sel, filled}$ häufiger vertreten als f_y , wird f_y eingefügt (vgl. Zeile 9-10 in Algorithmus 2). Verhält es sich dagegen andersherum, wird $\neg f_y$ hinzugefügt (vgl. Zeile 11-12 in Algorithmus 2). Wenn f_y und $\neg f_y$ gleich oft auftreten, wird zufällig entschieden (vgl. Zeile 14 in Algorithmus 2). Das Befüllen der offenen Merkmaleinträge wird mit der Aktualisierung der noch nicht abgedeckten Merkmale $F_{uncovered}$ abgeschlossen und die vollständige Konfiguration $c_{s, filled}$ der Menge der selektierten Konfigurationen $C_{sel, filled}$ hinzugefügt.

Für das in Tabelle 12 spezifizierte Beispielsystem selektiert der Greedy-Algorithmus drei Konfigurationen. In Tabelle 14 ist ein Überblick über die Entscheidungsgrundlage und die schrittweise aufgebaute Lösung für das Beispielsystem dargestellt.

Durchlauf	r_{\min}	C_{\max}	$C_{f\max}$	$R_{\text{uncovered}}$	$F_{\text{uncovered}}$	$c_{s,\text{filled}}$
Initialisierung	r_4			R_{all}	F_{all}	c_9
Greedy 1		c_4, c_6	c_4, c_6	r_2, r_5, r_6	f_6, f_8	c_6
Greedy 2		c_{10}, c_{12}, c_{14}	c_{10}, c_{12}, c_{14}	r_5	f_8	c_{10}

Tabelle 14: Durchläufe sowie das Ergebnis des Greedy-Vorgehens für das System *Klimatisierung*

Die mit mithilfe des Fillings (grau hervorgehoben) vollständig bestimmte Konfigurationen sind:

$$\begin{aligned}
 c_{9,\text{filled}} &= (KA) \wedge (SK) \wedge (SH) \wedge (PH) \wedge (LH) \wedge (HV \text{ Klein}), & (6.3) \\
 c_{6,\text{filled}} &= (KA) \wedge (\neg SK) \wedge (\neg SH) \wedge (PH) \wedge (LH) \wedge (HV \text{ Groß}) \text{ und} \\
 c_{10,\text{filled}} &= (\neg KA) \wedge (SK) \wedge (SH) \wedge (PH) \wedge (LH) \wedge (\text{Kein HV}).
 \end{aligned}$$

Dabei sind alle Merkmale von $c_{9,\text{filled}}$ zugunsten einer maximierten Merkmalabdeckung, alle Merkmale von $c_{6,\text{filled}}$ sowie das Merkmal $(\neg KA)$ von $c_{10,\text{filled}}$ zugunsten einer gleichmäßigen Merkmalabdeckung und schließlich die Merkmale (PH) , (LH) von $c_{10,\text{filled}}$ zufällig befüllt worden.

Komplexitätsanalyse des Greedy-Algorithmus

Vor der Durchführung der eigentlichen Greedy-Selektion wird zunächst die Basis (vgl. Abschnitt 6.2.1) bereitgestellt. Zum einen wird hierfür Tabelle 11 mit den Konfigurationen C_{all} ermittelt. Zum anderen werden in Tabelle 13 für jede Konfiguration die für sie gültigen Anforderungen sowie die darin enthaltenen Merkmale bestimmt. Die Bereitstellung beider Tabellen wird für die Durchführung des Greedy-Algorithmus sowie der im Folgenden vorgestellten Simulierten Abkühlung als Voraussetzung angenommen und geht daher nicht in die anschließende Laufzeitdiskussion der Optimierungen ein.

Die *Zeitkomplexität* des Greedy-Algorithmus hängt von mehreren Faktoren ab. Der Durchlauf der Initialisierungsphase des Greedy-Algorithmus hängt davon ab, wie viele Anforderungen existieren, die von exakt einer Konfiguration abgedeckt werden und somit von $\mathcal{O}(|R_{\min}|)$. Diese können maximal allen Anforderungen $|R_{\min}| = |R_{\text{all}}|$ entsprechen.

Während der Phase des Greedy-Algorithmus werden die Konfigurationen mit der maximalen Anforderungsabdeckung sowie gegebenenfalls die mit der größten Merkmalabdeckung gesucht. Um die Operation zu unterstützen, wird eine Sortierung der Konfigurationen nach den maximalen Abdeckungen jeweils in $\mathcal{O}(|C_{\text{all}}| \cdot \log_2(|C_{\text{all}}|))$ durchgeführt [30]. Nach der Extraktion einer Konfiguration mit der größten Anforderungs-

bzw. Merkmalabdeckung werden die Prioritätswarteschlangen aktualisiert. Dies ist von der Größenordnung $\mathcal{O}(\log_2(|C_{\text{unsel}}|) + \log_2(|C_{\text{max}}|))$ [30]. Hierbei ist der zweite Term vernachlässigbar gegenüber dem ersten, da $C_{\text{max}} \subseteq C_{\text{unsel}}$ gilt. Die Anzahl der nicht selektierten Konfigurationen entspricht im Normalfall ungefähr der Anzahl aller Konfigurationen $|C_{\text{unsel}}| = |C_{\text{all}}|$. Wie oft die Schleife durchlaufen wird, hängt von $|C_{\text{sel}}|$ abzüglich der selektierten Konfigurationsanzahl aus der Initialisierungsphase ab. Letztere werden durch $|R_{\text{min}}|$ definiert, für die im Normalfall $|R_{\text{min}}| \approx 0$ gilt. Im Extremfall wird für jede abzudeckende Anforderung jeweils eine Konfiguration benötigt, sodass $|C_{\text{sel}}| = |R_{\text{all}}|$ gilt.

Das Befüllen der offenen Entscheidungen hängt von $\mathcal{O}(\sum_{C_{\text{sel},i}} |F_{\text{filled},C_{\text{sel},i}}|)$ ab, wobei $F_{\text{filled},C_{\text{sel},i}}$ der Menge aller Merkmale, welche innerhalb der jeweils selektierten Konfiguration $C_{\text{sel},i}$ befüllt werden, entspricht. Die Anzahl der maximal selektierten Konfigurationen ist durch $|C_{\text{sel}}| = |R_{\text{all}}|$ begrenzt. Zudem müssen mithilfe des Filling nahezu alle Konfigurationen vervollständigt werden $|F_{\text{filled},C_{\text{sel},i}}| \approx |F_{\text{opt},\text{all}}|$, was jedoch nur für optionale Merkmale möglich ist.

Im Normalfall besitzt der Greedy-Algorithmus somit eine Rechenzeit von:

$$\mathcal{O}(|R_{\text{min}}| + |C_{\text{all}}| \cdot \log_2(|C_{\text{all}}|) + |C_{\text{sel}}| \cdot \log_2(|C_{\text{all}}|) + \sum_{C_{\text{sel},i}} |F_{\text{filled},C_{\text{sel},i}}|). \quad (6.4)$$

Der letzte Term von 6.4 kann nur dann dominieren, wenn für jede abzudeckende Anforderung eine Konfiguration selektiert werden würde. Hierfür müsste in einer Spezifikation für jedes c_x explizit eine Anforderung existieren. Jedoch würde eine solche Spezifikation gar nicht erst mit dem Gedanken einer Produktlinienentwicklung erstellt werden, sodass $|C_{\text{sel}}| = |R_{\text{all}}|$ einen theoretischen Fall darstellt und im Normalfall $|C_{\text{sel}}| \ll |R_{\text{all}}|$ sowie $|C_{\text{sel}}| \ll |C_{\text{all}}|$ gilt. Aus diesem Grund wird die Laufzeit im Regelfall vom Sortierverfahren mittels Prioritätswarteschlange mit $\mathcal{O}(|C_{\text{all}}| \cdot \log_2(|C_{\text{all}}|))$ dominiert. Anhand von Formel 6.5 ist ersichtlich, dass auch die maximale Rechenzeit von der Sortierung dominiert wird. Denn dabei gilt $|C_{\text{all}}| > |R_{\text{all}}| > |F_{\text{opt},\text{all}}|$.

Die maximale Rechenzeit des Greedy-Algorithmus hängt ab von:

$$\mathcal{O}(|R_{\text{all}}| + |C_{\text{all}}| \cdot \log_2(|C_{\text{all}}|) + |R_{\text{all}}| \cdot \log_2(|C_{\text{all}}|) + |R_{\text{all}}| \cdot |F_{\text{opt},\text{all}}|). \quad (6.5)$$

Die empirisch ermittelte Laufzeitabhängigkeit ist in Abbildung 47 dargestellt und wird im Rahmen der Vorstellung der Fallstudienenergebnisse in Abschnitt 7.4.1 diskutiert.

Die *Speicherkomplexität* des Greedy-Algorithmus wird bestimmt von der anfänglichen Erstellung der Listen, welche jede Konfiguration zu jeder Anforderung sowie zu jedem Merkmal zuordnen (vgl. die Basis innerhalb der Tabellen 11 und 13), und somit von $\mathcal{O}(|C_{\text{all}}| \cdot |R_{\text{all}}| + |C_{\text{all}}| \cdot |F_{\text{all}}|)$. Zusätzlich werden befüllte Merkmale gespeichert mit $\mathcal{O}(\sum_{C_{\text{sel},i}} |F_{\text{filled},C_{\text{sel},i}}|)$. Im Extremfall existiert eine Konfiguration pro Anforderung $|C_{\text{sel}}| = |R_{\text{all}}|$ und es werden nahezu alle Merkmale mithilfe des Filling vervollständigt $|F_{\text{filled},C_{\text{sel},i}}| \approx |F_{\text{opt},\text{all}}|$.

Im Regelfall besitzt der Greedy-Algorithmus somit einen Speicherbedarf von:

$$\mathcal{O}(|C_{all}| \cdot |R_{all}| + |C_{all}| \cdot |F_{all}| + \sum_{C_{sel,i}} |F_{filled,C_{sel,i}}|). \quad (6.6)$$

Im Normalfall wird die Speicherkomplexität dominiert von $\mathcal{O}(|C_{all}| \cdot |R_{all}|)$, da wie bereits diskutiert $|C_{sel}| \ll |C_{all}|$ sowie $|R_{all}| \gg |F_{all}| > |F_{filled,C_{sel,i}}|$ gilt. Der maximale Speicherbedarf wird ebenfalls vom ersten Term dominiert.

6.2.3 Optimierung mittels Simulierter Abkühlung

Der Greedy-Algorithmus stellt eine lokale Suche dar, deren Ergebnis unter Umständen nicht dem eigentlich gesuchten Optimum entspricht. Damit verbunden ist die Tatsache, dass der Greedy-Algorithmus die Lösung nach der Strategie der momentanen Gewinnmaximierung und damit nach einem fest definierten Suchmuster aufbaut. Um den Suchraum zu erweitern ist ein zufallsbasiertes bzw. stochastisches Optimierungsverfahren sinnvoll. Prinzipiell kommen hierfür eine Reihe möglicher Ansätze in Frage. Die meisten gebräuchlichen stochastischen Verfahren basieren auf Prinzipien, die auf der Nachahmung der Natur basieren [42]. Dazu gehören insbesondere biologische und physikalische Vorgänge. In diesem Zusammenhang werden häufig Evolutionäre Algorithmen (engl. *evolutionary algorithms*) angewandt. Dabei wird die Lösung in Analogie zur biologischen Evolution mittels Reproduktion, Mutation, Rekombination und Selektion ermittelt. Die Menge der Kandidaten, welche Individuen in einer Population darstellen, werden anhand einer Fitnessfunktion bewertet [65]. Auf Basis der genannten Operationen wird die Evolution nachgeahmt und das Ergebnis dem Optimum angenähert. Darüber hinaus wird häufig die Simulierte Abkühlung, welche auf Beobachtungen des Abkühlungsprozesses bspw. von Flüssigkeiten oder Festkörpern basiert, genutzt. Die Effizienz der Evolutionären Algorithmen ist mit der der Simulierten Abkühlung anhand von unterschiedlichen Kriterien wie Komplexität, Voraussetzungen oder Konvergenz verglichen worden [114]. Je nach Anwendungsumfeld liefern beide Verfahren ähnlich gute Ergebnisse [58, 71]. Da jedoch die Simulierte Abkühlung eine im Allgemeinen kürzere Laufzeit und eine niedrigere Komplexität bei der Implementierung aufweist [58, 106, 117], wird die Heuristik für das vorliegende Problem herangezogen.

6.2.3.1 Simulierte Abkühlung

Die *Simulierte Abkühlung* (engl. *simulated annealing*) ist ein metaheuristisches Verfahren zur Lösung von kombinatorischen Optimierungsproblemen mit großen Suchräumen. Im Gegensatz zum Greedy-Algorithmus akzeptiert die Simulierte Abkühlung auch schlechtere Lösungen und kann auf diese Weise lokale Optima wieder verlassen. Dadurch ist es möglich eine umfassendere Suche durchzuführen [117].

Die Grundidee der Simulierten Abkühlung basiert auf dem physikalischen Abkühlprozess von Materialien, bspw. Metallen, aus dem geschmolzenen in den festen Zustand.

Die Energie eines Systems setzt sich dabei aus den Energien seiner einzelnen Teilchen zusammen. Wird ein Metall erhitzt, wird zunächst eine Unordnung in der Kristallstruktur verursacht und die Gesamtenergie des Metalls stark erhöht. Durch ein *langsames* Abkühlen haben die Atome des Metalls wieder die Möglichkeit sich zu ordnen und eine regelmäßige Kristallstruktur aufzubauen. Dadurch kann schließlich ein energieärmer Zustand erreicht werden. Die Übergangswahrscheinlichkeit von einem Energiezustand E in einen anderen E_{new} wird beschrieben (ausgehend von der Boltzmann-Verteilung) durch [106]:

$$P(E, E_{new}, T) = \exp\{(E - E_{new}) / (k_B \cdot T)\} \quad (6.7)$$

wobei T die Temperatur des Systems und k_B die Boltzmann-Konstante kennzeichnen.

Die Analogie des Abkühlprozesses wird bei der Simulierten Abkühlung verwendet, um die Akzeptanz einer Lösung anhand von $P(E, E_{new}, T)$ zu bewerten. Dabei wird zum Zeitpunkt einer Entscheidung eine bessere Lösung, welche eine niedrigere Energie E_{new} besitzt als die vorherige, immer akzeptiert. Dagegen wird eine schlechtere Lösung nur mit der Wahrscheinlichkeit $P(E, E_{new}, T)$ angenommen. Das Entscheidungsprinzip der Simulierten Abkühlung ist in Abbildung 37 schematisch dargestellt.

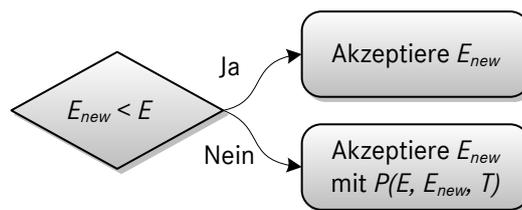


Abbildung 37: Entscheidungsprinzip der Simulierten Abkühlung für die Akzeptanz einer Lösung

Die Optimierung der Simulierten Abkühlung wird bei einer hohen Temperatur T_{init} und einem zufällig gewählten Startzustand bzw. -lösung initialisiert. Der Übergang in einen neuen Zustand geschieht innerhalb von wohldefinierten Zuständen der Nachbarschaft, um die Lösung nur minimal zu modifizieren. Auf diese Weise wird der Algorithmus dabei unterstützt bessere Teile der Lösung beizubehalten und tendenziell die schlechteren zu verändern. Wird die Lösung bei den Übergängen zu stark verändert kann keine schnelle bis gar keine Konvergenz des Algorithmus gewährleistet werden [117].

Um bewerten zu können, ob die Lösung sich nach einem Übergang verbessert oder verschlechtert hat, wird für jeden Zustand ein Energiewert berechnet. Die Energie enthält eine Aussage bezüglich der zu optimierenden Größe(n) und stellt somit ein Maß für die Qualität der Lösung dar. Die effiziente Bestimmung des Energiewertes ist ein wichtiges Kriterium, da dieser bei jedem Iterationsschritt berechnet wird und somit die Laufzeit stark beeinflussen kann.

Die Wahrscheinlichkeit $P(E, E_{new}, T)$ dafür, ob eine schlechtere Lösung akzeptiert wird, wird anhand der Formel 6.7 berechnet. Diese hängt einerseits von der Energie der

ursprünglichen E und der neuen Lösung E_{new} sowie andererseits von dem zeitabhängigen Parameter Temperatur T ab. Je höher dabei die Temperatur, desto wahrscheinlicher ist es, dass auch eine schlechtere Lösung akzeptiert wird. Wenn die Temperatur schließlich gegen Null geht, werden zunehmend bessere Lösungen mit niedrigeren Energien akzeptiert und das Verfahren geht in die lokale Greedy-Suche über [44].

Der Kühlprozess kann verschiedenen Temperaturverläufen unterliegen. Der einfachste ist gegeben durch $T_{\text{next}} = \alpha \cdot T$, mit einer exemplarischen Kühlrate $0,8 \leq \alpha \leq 0,99$ [106]. Zusätzlich kann eine Anzahl an Iterationen festgelegt werden, wie oft der Algorithmus die Suche bei einem Temperaturwert wiederholen soll. Wichtig ist dabei die Kühlung entsprechend langsam verlaufen zu lassen, damit der Algorithmus genügend Iterationsschritte durchführen und damit die Möglichkeit hat näher an das globale Optimum zu gelangen. Gleichzeitig darf nicht zu langsam gekühlt werden, da dies zu Lasten der Rechenzeit geht.

Schließlich wird ein Abbruchkriterium, bei welchem der Algorithmus terminieren soll, definiert. Dieses kann bspw. als eine bestimmte Anzahl an Iterationsschritten oder als ein erreichter Temperaturwert festgelegt werden. Darüber hinaus wird häufig eine feste Anzahl an Iterationen, nach welchen keine Verbesserung der Lösung festgestellt wurde, als Abbruchgrenze definiert [106].

Die Parameter, welche einen Einfluss auf das Ergebnis der Simulierten Abkühlung haben und bei der Implementierung festgelegt werden müssen, sind unter dem Begriff *Annealing Schedule* zusammengefasst [117]. Dazu gehören: T_{init} , der Kühlprozess, die Anzahl der Iterationsschritte bei einer festen Temperatur sowie das Abbruchkriterium. Der *Annealing Schedule* muss empirisch ermittelt werden, um abhängig vom jeweiligen Problem die Effizienz des Algorithmus gewährleisten zu können.

Im Folgenden wird das Verfahren der Simulierten Abkühlung angepasst an das Problem der optimierten Anforderungs- sowie Merkmalabdeckung vorgestellt.

6.2.3.2 Selektionsvorgehen mittels Simulierter Abkühlung

Bei der Selektion mittels Simulierter Abkühlung werden wie auch beim Greedy-Algorithmus die in Abschnitt 6.2.1 vorgestellten Tabellen 11 und 13 als Basis verwendet.

Die Optimierung beginnt mit der *Initialisierungsphase*, welche anhand des Pseudocodes in Algorithmus 3 dargestellt ist. Darin wird die initiale Energie mit $E_{\text{init}} = 1$ festgelegt. Dies ist der (theoretisch) maximale Wert, welcher aus der später diskutierten Energiefunktion (vgl. Algorithmus 5) errechnet werden kann. Zusätzlich wird die Größe der selektierten Konfigurationssets N bestimmt. Hierfür wird die Größe des vom Greedy-Algorithmus zuvor ermittelten Ergebnisses $|\text{GreedySolution}| = G$ als Orientierungswert herangezogen³. Dies ist sinnvoll, da mittels Simulierter Abkühlung nach einem möglichst besseren Ergebnis mit kleinerem Konfigurationsset als den vom Greedy-Algorithmus ermittelten gesucht wird. Um die Suche „in der Nähe“ der Greedy-Lösung durchzuführen, wird die Anzahl der Konfigurationssets unterhalb

³ In Abschnitt B sind alle innerhalb des Greedy-Algorithmus und der Simulierten Abkühlung verwendeten Abkürzungen zusammenfassend aufgelistet und erklärt.

$G - y$, aber auch oberhalb der Greedy-Lösung $G + x$ ermöglicht. Somit werden $x + y + 1$ Simulierte Abkühlungen mit festen Setgrößen parallel durchgeführt. Das Vorgehen ist in Abbildung 38 schematisch dargestellt. Die Entscheidung die Größen der Konfigurationssets konstant zu lassen beruht auf empirischen Erfahrungswerten, da dies durch eine bessere Konvergenz des Algorithmus zu besseren Ergebnissen führt.

Algorithmus 3 Simulated Annealing Initialisation

```

global Cglobal, Eglobal
2 function ANNEALINGCONFIGSELECTION(Call, |GreedySolution| = G)
    Eglobal ← Einit
4   Cglobal ← ∅
    N ← G + x
        ▷ Bilde feste Konfigurationsset-Größen gleich der Greedy-Lösung sowie x
          Größen oberhalb und y unterhalb der Greedy-Lösung:
6   while N ≥ G - y do
        SIMULATEDANNEALING(N, Call)
8   N ← N - 1
    end while
10 end function

```

Während der Phase der *Konfigurationsselektion*, welche in Algorithmus 4 dargestellt ist, folgt das eigentliche Optimierungsvorgehen. Zu Beginn wird der zunächst leeren Menge C_{sel} zufällig ausgewählte Konfigurationen c_x (vgl. Tabelle 11) aus der Menge C_{unsel} hinzugefügt (vgl. Zeile 5-7 in Algorithmus 4). Dies wird für alle Sets zwischen $G - y \leq N \leq G + x$ durchgeführt und zuletzt der Energiewert eines jeden Konfigurationssets berechnet (vgl. Zeile 10 in Algorithmus 4).

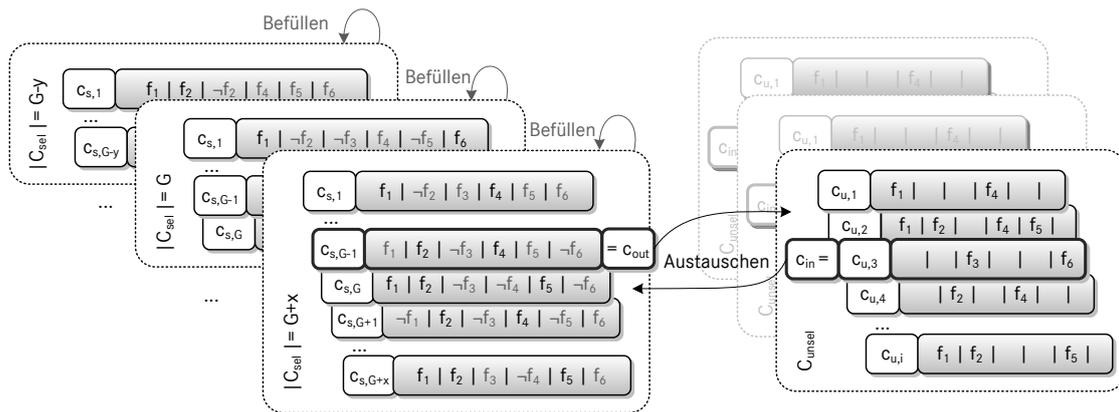


Abbildung 38: Überblick über das Vorgehen der parallel durchgeführten Simulierten Abkühlungen zur optimierten Anforderungs- und Merkmalabdeckung. Die Lösungen mit festen Konfigurationssetgrößen (links) werden durch das Austauschen der Konfigurationen aus den selektierten mit denen aus den nicht selektierten Konfigurationsmengen (rechts) ermittelt.

Algorithmus 4 Simulated Annealing Configuration Selection

```

function SIMULATEDANNEALING( $N, C_{all}$ )
2    $C_{sel} \leftarrow \emptyset$ 
    $C_{unsel} \leftarrow C_{all}$ 
   ▷ Befülle die Konfigurationssets zufällig:
4   for  $n = 1$  to  $N$  do
        $c_s \leftarrow \text{randomSelect}(C_{unsel})$ 
6        $C_{sel} \leftarrow C_{sel} \cup \{c_s\}$ 
        $C_{unsel} \leftarrow C_{unsel} \setminus \{c_s\}$ 
8   end for
    $C_{sel, \text{filled}}, F_{uncovered} \leftarrow C_{sel}\text{-FEATURE FILLING}(C_{sel}, F_{all})$ 
10   $E_{local} \leftarrow \text{ENERGY}(C_{sel, \text{filled}}, C_{all}, R_{all}, F_{all}, F_{uncovered})$ 
    $T \leftarrow T_{init}$ 
12  while  $T > T_{min}$  do
       ▷ Tausche eine Konfiguration aus der Menge der selektierten zufällig mit einer
       Konfiguration aus der Menge der nicht selektierten Konfigurationen:
        $c_{out} \leftarrow \text{randomSelect}(C_{sel})$ 
14       $c_{in} \leftarrow \text{randomSelect}(C_{unsel})$ 
        $C_{new} \leftarrow C_{sel} \setminus \{c_{out}\}$ 
16       $C_{new} \leftarrow C_{new} \cup \{c_{in}\}$ 
       ▷ Befülle die unbestimmten Merkmale des selektierten Konfigurationssets neu
        $C_{new, \text{filled}}, F_{uncovered} \leftarrow C_{new}\text{-FEATURE FILLING}(C_{new}, F_{all})$ 
18       $E_{new} \leftarrow \text{ENERGY}(C_{new, \text{filled}}, C_{all}, R_{all}, F_{all}, F_{uncovered})$ 
        $P \leftarrow \exp\left(-\frac{E_{new} - E_{local}}{T}\right)$ 
       ▷ Ist die Energie des neuen selektierten Konfigurationssets besser als die des
       vorigen bzw. ist  $P$  höher als eine zufällige Zahl zwischen 0 und 1, speichere
       die neue als beste Lösung:
20      if  $E_{new} < E_{local}$  or  $P > \text{randomSelect}[0, 1)$  then
            $E_{local} \leftarrow E_{new}$ 
22            $C_{sel, \text{filled}} \leftarrow C_{new, \text{filled}}$ 
            $C_{sel} \leftarrow C_{new}$ 
24            $C_{unsel} \leftarrow C_{all} \setminus C_{sel}$ 
           ▷ Ist das neu selektierte Konfigurationsset besser als alle bisher bewerteten
           Konfigurationssets, speichere dieses als global beste Lösung:
           if  $E_{local} < E_{global}$  then
26                $E_{global} \leftarrow E_{local}$ 
                $C_{global} \leftarrow C_{sel, \text{filled}}$ 
28           end if
           end if
30        $T \leftarrow \frac{T}{1 + \beta \cdot T}$ 
   end while
32 end function

```

Die *Energiefunktion*, anhand derer die Qualität jeder Lösung bewertet wird, ist in Algorithmus 5 gezeigt. Diese setzt sich aus der Summe der drei zu optimierenden Größen zusammen. Die erste Größe beschreibt die prozentual noch abzudeckenden Anforderungen $|R_{\text{uncovered}}|/|R_{\text{all}}|$, während die zweite der prozentual abzudeckenden Merkmalen $|F_{\text{uncovered}}|/|F_{\text{all}}|$ entspricht. Dabei erfolgt die Berechnung der Merkmalabdeckung für befüllte Konfigurationssets $C_{\text{sel, filled}}$ ohne offene Merkmaleinträge. Das Filling wird analog zu dem des Greedy-Algorithmus durchgeführt und weiter unten näher erläutert. Da maximale Artefaktabdeckungen angestrebt werden, wird der Energiewert kleiner und somit die Lösung besser, je höher die Abdeckungen sind. Die dritte Größe $|C_{\text{sel, filled}}|/|C_{\text{all}}|$ stellt das Verhältnis der selektierten zu allen verfügbaren Konfigurationen dar. Es wird ein minimales Set an selektierten Konfigurationen gesucht, sodass der Energiewert mit kleiner werdenden Konfigurationssetgrößen sinkt.

Algorithmus 5 Energy Calculation

```

function ENERGY( $C_{\text{sel, filled}}, C_{\text{all}}, R_{\text{all}}, F_{\text{all}}, F_{\text{uncovered}}$ )
   $R_{\text{uncovered}} \leftarrow R_{\text{all}} \setminus \bigcup_{c_s \in C_{\text{sel, filled}}} R_{\text{cov}}(c_s)$ 
  return  $w_r \frac{|R_{\text{uncovered}}|}{|R_{\text{all}}|} + w_f \frac{|F_{\text{uncovered}}|}{|F_{\text{all}}|} + w_c \frac{|C_{\text{sel, filled}}|}{|C_{\text{all}}|}$ 
end function

```

Jede der drei zu optimierenden Größen wird mit einem Faktor w (engl. *weight*) gewichtet, um diesen unterschiedliche Prioritäten zuzuordnen. In der Regel müssen weitaus mehr Anforderungen als Merkmale berücksichtigt werden, sodass durch eine Anforderungsabdeckung gleichzeitig die Merkmalabdeckung erreicht werden kann. Aus diesem Grund wird die Anforderungsabdeckung wesentlich höher priorisiert als die Merkmalabdeckung: $w_r > w_f$. Der dritte Term der Energiefunktion hat mit w_c die kleinste Priorität, aufgrund der zu Beginn festgelegten konstanten Größen der Konfigurationssets. Dieser wird in erster Linie benötigt, um das kleinste Set aus allen zu bestimmen. Die konkreten Gewichtungsfaktoren betragen $w_r = 127/210$, $w_f = 80/210$, $w_c = 3/210$ und stellen vom vorliegenden Problem abhängige, empirisch ermittelte Werte dar.

Nach der Energieberechnung der zu Beginn zufällig hinzugefügten Konfigurationssets, beginnt die Optimierung bei einer initialen Temperatur $T_{\text{init}} = 100$ (vgl. Zeile 11 in Algorithmus 4). Der *Kühlprozess* verläuft dabei nach

$$T_{\text{next}} = T/(1 + \beta T), \quad (6.8)$$

wobei die Aktualisierung der Temperatur in Zeile 30 des Algorithmus 4 stattfindet. Die Funktion 6.8 simuliert einen anfänglich schnellen Temperaturabfall und eine kontinuierlich langsame Annäherung an den Nullpunkt (vgl. Abbildung 49). Für die Kühlrate β wurden Werte zwischen $0,0001 \leq \beta \leq 0,1$ experimentell verglichen. Je kleiner dabei die Kühlrate, desto langsamer verläuft die Kühlung und desto häufiger iteriert der Algorithmus. Aufgrund des sehr langsamen Temperaturabfalls kann darauf verzichtet werden Iterationsschritte bei einer festen Temperatur einzuführen, sodass stattdessen bei jedem Temperaturwert nur einmal iteriert wird [68]. Der Vorteil eines solchen

Kühlprozesses ist es, dass dieser im Wesentlichen von nur einem Parameter, nämlich β , abhängt und anhand dieses modifiziert werden kann.

Der *Übergang* von einer Lösung zu einer anderen erfolgt dadurch, dass eine zufällig gewählte Konfiguration c_{out} (aus der Menge der selektierten Konfigurationen C_{sel}) mit einer zufällig gewählten Konfiguration c_{in} (aus der Menge der nicht selektierten Konfigurationen C_{unsel}) vertauscht wird (vgl. Zeile 13-16 in Algorithmus 4). Solche Übergänge, bei welchen nicht gleich mehrere Konfigurationen miteinander vertauscht werden, gewährleisten, dass sich die neue Lösung in der Nachbarschaft befindet. Wie im vorhergehenden Abschnitt 6.2.3.1 bereits diskutiert, führt dies zu einer besseren Konvergenz des Algorithmus hin zum globalen Optimum.

Die Entscheidung, ob die neue Lösung akzeptiert werden soll, wird mithilfe des klassischen Vorgehens der Simulierten Abkühlung getroffen. Diese bezieht sich auf die *lokal beste Lösung* mit maximalen Artefaktdeckungen innerhalb eines Konfigurationssets mit fester Größe. Hierfür wird die Qualität der neuen Lösung C_{new} anhand deren Energiewert E_{new} bewertet (vgl. Zeile 18 in Algorithmus 4). Ist dieser kleiner als der Energiewert der vorherigen lokalen Lösung, wird C_{new} akzeptiert (vgl. Zeile 20-23 in Algorithmus 4). Ist E_{new} dagegen größer als die vorherige Lösung wird die Akzeptanz von C_{new} mittels der Wahrscheinlichkeitsfunktion P ermittelt (vgl. Formel 6.7). Dabei wird für den Fall, dass P größer als eine zufällig gewählte Zahl zwischen 0 und 1 ist, C_{new} akzeptiert (vgl. Zeile 20-23 in Algorithmus 4), ansonsten verworfen und die alte Lösung beibehalten. Dieses lokale Entscheidungsprinzip ist im linken Teil der Abbildung 39 schematisch verdeutlicht.

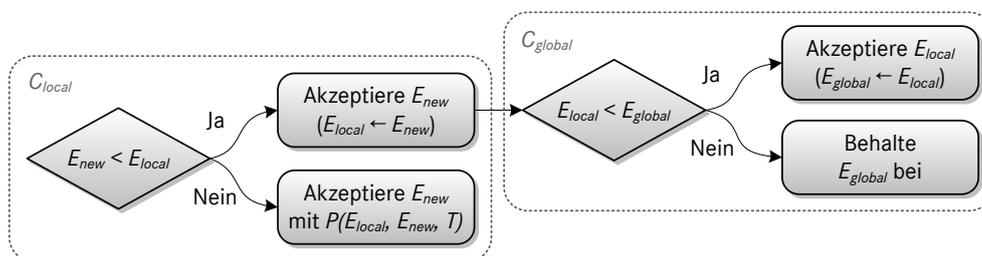


Abbildung 39: Entscheidungsprinzip der Simulierten Abkühlung für die lokal sowie global beste Lösung zur optimierten Anforderungs- und Merkmalabdeckung

Ergänzend zu dem klassischen Vorgehen der Simulierten Abkühlung wird die *global beste Lösung* C_{global} mit der entsprechenden Energie E_{global} gespeichert. Dies ist im rechten Teil der Abbildung 39 gezeigt. Im Gegensatz zu der lokalen Lösung C_{local} , welche für eine feste Konfigurationssetgröße gilt, entspricht C_{global} der besten innerhalb aller Konfigurationssets ermittelten Lösung aus den parallel durchgeführten Simulierten Abkühlungen. Immer dann, wenn die lokale Lösung durch eine neue verbessert wird (vgl. Zeile 21 in Algorithmus 4), wird geprüft, ob die Energie der neuen Lösung E_{new} kleiner ist, als die aus allen anderen zuvor ermittelten Lösungen E_{global} . Ist dies der Fall, wird E_{local} als neue beste Lösung E_{global} mit C_{global} gespeichert (vgl. Zeile 26-27 in Algorithmus 4). Die Bestimmung von C_{global} wird durch den letzten Term in der Energiefunktion ermöglicht.

Algorithmus 6 C_{sel} -Feature Filling

```

function FILLING( $C_{sel}, F_{all}$ )
2    $F_{uncovered} \leftarrow F_{all}$ 
    $C_{sel, filled} \leftarrow C_{sel}$ 
4   for all  $c_s \in C_{sel, filled}$  do
       while  $\exists f_y \in F_{all}, \text{cov}(c_s, f_y) = \emptyset$  do
6         if  $f_y \in F_{uncovered}$  then
            $\text{cov}(c_s, f_y) \leftarrow 1$ 
8         else
            $\text{sumCov}(f_y) \leftarrow \sum_{c_x \in C_{sel, filled}} \text{cov}(c_x, f_y)$ 
10           $\text{sumCov}(\neg f_y) \leftarrow \sum_{c_x \in C_{sel, filled}} \text{cov}(c_x, \neg f_y)$ 
           if  $\text{sumCov}(f_y) < \text{sumCov}(\neg f_y)$  then
12              $\text{cov}(c_s, f_y) \leftarrow 1$ 
           else if  $\text{sumCov}(f_y) > \text{sumCov}(\neg f_y)$  then
14              $\text{cov}(c_s, f_y) \leftarrow 0$ 
           else
16              $\text{cov}(c_s, f_y) \leftarrow \text{randomSelect}\{0, 1\}$ 
           end if
18         end if
       end while
20    $F_{uncovered} \leftarrow F_{uncovered} \setminus F_{cov}(c_s)$ 
   end for
22   return  $C_{sel, filled}, F_{uncovered}$ 
end function

```

Der Algorithmus wiederholt das Vorgehen bis zu einer Temperatur $T_{min} = 0,001$ (vgl. Zeile 12 in Algorithmus 4). Da bei sehr kleinen Temperaturen die Wahrscheinlichkeit zur Verbesserung der Lösung gering ist, ist es sinnvoll eine Temperaturgrenze als *Abbruchkriterium* zu verwenden.

Selektierte Konfigurationen können gegebenenfalls unbestimmte Merkmaleinträge enthalten (vgl. Tabelle 11). Dies ist zum einen der Fall, wenn die initialen Sets mit zufällig gewählten Konfigurationen aus der Menge C_{unsel} generiert werden (vgl. Zeile 5 in Algorithmus 4) und zum anderen wenn c_{out} mit der Konfiguration $c_{in} \in C_{unsel}$ vertauscht wird (vgl. Zeile 13-16 in Algorithmus 4). Das *Befüllen* der offenen Einträge erfolgt analog zu dem Filling des Greedy-Algorithmus (vgl. Algorithmus 2) zugunsten einer maximierten sowie gleichmäßigen Merkmalabdeckung und ist anhand des Pseudocodes in Algorithmus 6 dargestellt.

Hierbei wird immer dann mit f_y befüllt, wenn das Merkmal von keiner bis dahin selektierten Konfiguration abgedeckt wurde (vgl. Zeile 6-7 in Algorithmus 6) oder wenn häufiger mit $\neg f_y$ als mit f_y befüllt wurde (vgl. Zeile 11-12 in Algorithmus 6). Dagegen wird mit $\neg f_y$ befüllt, falls die selektierten Konfigurationen häufiger mit f_y befüllt wurden (vgl. Zeile 13-14 in Algorithmus 6). Treten f_y und $\neg f_y$ gleich häufig auf, wird zufällig entschieden (vgl. Zeile 16 in Algorithmus 6). Der einzige Unterschied zu

dem in Algorithmus 2 beschriebenen Filling des Greedy-Algorithmus besteht darin, dass im Rahmen der Simulierten Abkühlung alle Konfigurationen des gesamten Sets befüllt werden. Aus Gründen einer effizienteren Implementierung, werden die Sets auch dann ganzheitlich neu befüllt, wenn nur eine Konfiguration ausgetauscht wird, d.h. also bei jedem Temperaturwert bzw. Iterationsschritt. Die Befüllung der in einem Set enthaltenen Konfigurationen erfolgt in einer zufälligen Reihenfolge und wird schließlich als $C_{sel, filled}$ gespeichert.

Komplexitätsanalyse der Simulierten Abkühlung

Die *Laufzeit* der Simulierten Abkühlung hängt im Wesentlichen von der Anzahl der durchgeführten Iterationsschritte ab. Diese werden zum einen durch die Anzahl der Konfigurationssets $x + y + 1$ bestimmt, in welchen nach dem Optimum gesucht wird. Zum anderen legt der Kühlprozess für ein Konfigurationsset fest, wie oft iteriert werden muss, um von T_{init} auf T_{min} herunter zu kühlen. Hierbei ist die Anzahl der Schritte gegeben mit $\beta^{-1} \cdot T_{min}^{-1} + \log(T_{init})$.

Darüber hinaus hat das Befüllen der offenen Merkmaleinträge mit $\mathcal{O}(\sum_{C_{sel,i}} |F_{filled, C_{sel,i}}|)$ einen Einfluss auf die Laufzeit, wobei $F_{filled, C_{sel,i}}$ der Menge aller Merkmale, welche innerhalb der selektierten Konfigurationen $C_{sel,i}$ befüllt werden müssen, entspricht. Die Anzahl der maximal selektierten Konfigurationen ist durch $|C_{sel}| = |R_{all}|$ begrenzt. Zudem müssen mithilfe des Filling nahezu alle Konfigurationen vervollständigt werden $|F_{filled, C_{sel,i}}| \approx |F_{opt, all}|$, was jedoch nur für optionale Merkmale möglich ist.

Die Simulierte Abkühlung besitzt somit eine maximale Laufzeit von:

$$\mathcal{O}((x + y + 1) \cdot (\beta^{-1} \cdot T_{min}^{-1} + \log(T_{init})) \cdot |R_{all}| \cdot |F_{opt, all}|). \quad (6.9)$$

Der logarithmierte initiale Temperaturwert ist vernachlässigbar klein, sodass der Abkühlprozess in erster Linie von $\beta^{-1} \cdot T_{min}^{-1}$ dominiert wird, wobei für $T_{min} \rightarrow 0$ die Laufzeit des Algorithmus divergiert. Die letzten beiden Faktoren von 6.9 fallen umso stärker ins Gewicht, je größer die Konfigurationssetgrößen und je mehr Merkmaleinträge befüllt werden müssen. Wie bereits bei der Laufzeit des Greedy-Algorithmus argumentiert, wäre $|C_{sel}| = |R_{all}|$ entgegen des Gedanken der Produktlinienentwicklung und stellt somit einen hypothetischen Fall dar (vgl. Abschnitt 6.2.2.2).

Im Normalfall hängt die Laufzeit der Simulierten Abkühlung also ab von:

$$\mathcal{O}((x + y + 1) \cdot \beta^{-1} \cdot T_{min}^{-1} \cdot \sum_{C_{sel,i}} |F_{filled, C_{sel,i}}|), \quad (6.10)$$

wobei die vom Annealing Schedule abhängigen Faktoren β^{-1} und T_{min}^{-1} in der Regel die Laufzeit dominieren. Die empirisch ermittelte Laufzeitabhängigkeit ist in Abbildung 56 sowie 57 dargestellt und wird im Rahmen der Vorstellung der Fallstudienenergebnisse in Abschnitt 7.4.1 diskutiert.

Die *Speicherkomplexität* der Simulierten Abkühlung wird (wie auch die des Greedy-Algorithmus) bestimmt von der anfänglichen Erstellung der Basis. Die Basis besteht aus Listen, welche jede Konfiguration zu jeder Anforderung sowie zu jedem Merkmal zuordnen, und dessen Speicherbedarf abhängt von $\mathcal{O}(|C_{\text{all}}| \cdot |R_{\text{all}}| + |C_{\text{all}}| \cdot |F_{\text{all}}|)$. Zusätzlich werden befüllte Merkmale für alle Konfigurationssets gespeichert mit $\mathcal{O}((x + y + 1) \cdot \sum_{C_{\text{sel},i}} |F_{\text{filled},C_{\text{sel},i}}|)$, wobei im schlimmsten Fall zum einen $|C_{\text{sel}}| = |R_{\text{all}}|$ sowie zum anderen $|F_{\text{filled},C_{\text{sel},i}}| \approx |F_{\text{opt},\text{all}}|$ gilt.

Die Simulierte Abkühlung besitzt somit im Regelfall einen Speicherbedarf von:

$$\mathcal{O}(|C_{\text{all}}| \cdot |R_{\text{all}}| + |C_{\text{all}}| \cdot |F_{\text{all}}| + (x + y + 1) \cdot \sum_{C_{\text{sel},i}} |F_{\text{filled},C_{\text{sel},i}}|). \quad (6.11)$$

Im Normalfall wird die Speicherkomplexität dominiert von $\mathcal{O}(|C_{\text{all}}| \cdot |R_{\text{all}}|)$, da wie bereits diskutiert gilt $|C_{\text{sel}}| \ll |C_{\text{all}}|$ sowie $|R_{\text{all}}| \gg |F_{\text{all}}| > |F_{\text{filled},C_{\text{sel},i}}|$ gilt. Dies ist auch für den maximalen Speicherbedarf der Fall, aufgrund der Tatsache, dass die Anzahl der Konfigurationssets niemals größer sein kann, als die Anzahl aller Konfigurationen $(x + y + 1) < C_{\text{all}}$.

6.3 Selektion von Testfällen

Um die mittels der optimierten Anforderungs- sowie Merkmalabdeckung selektierten Konfigurationen testen zu können, müssen konkrete Testfälle für jede Konfiguration ermittelt werden. Hierbei ist das Ziel der vorliegenden Arbeit eine flexible und schnelle Ausleitung der Testfälle aus der generischen Testspezifikation für jedes mögliche Produkt der entsprechenden Produktlinie sicherzustellen. Die Thematik einer effizienten Strategie zur Wiederverwendung von Testfällen steht dagegen nicht im Fokus und wird daher nur kurz behandelt.

Grundsätzlich kommen für die Testfallermittlung zwei Strategienrichtungen in Frage:

- Durchführung aller Testfälle für jede selektierte Konfiguration oder
- Inkrementelles Testen der selektierten Konfigurationen.

Im Rahmen der ersten Methode werden für jede selektierte Konfiguration alle passenden Testfälle aus der generischen Testspezifikation ausgeleitet und durchgeführt (vgl. Abbildung 40 oben rechts). Dies stellt das einfachste Vorgehen dar, welches im Bezug auf die Testfallselektion eine hohe Ähnlichkeit zum *Product by Product Testen* (vgl. Abschnitt 4.2.2) aufweist. Die Methode hat den Vorteil, dass jede Konfiguration umfassend getestet wird und somit eine höhere Produktqualität im Vergleich zu einem unvollständigen Test garantiert werden kann.

Im Normalfall gelten einige Testfälle für alle bzw. mehrere selektierte Konfigurationen, sodass durch das erste Vorgehen redundante Testumfänge entstehen können. Aus diesem Grund kann es sinnvoll sein, die jeweils noch nicht durchgeführten konfigurations-spezifischen Testfälle mithilfe des Inkrementellen Testens, welches in Abschnitt 4.2.3

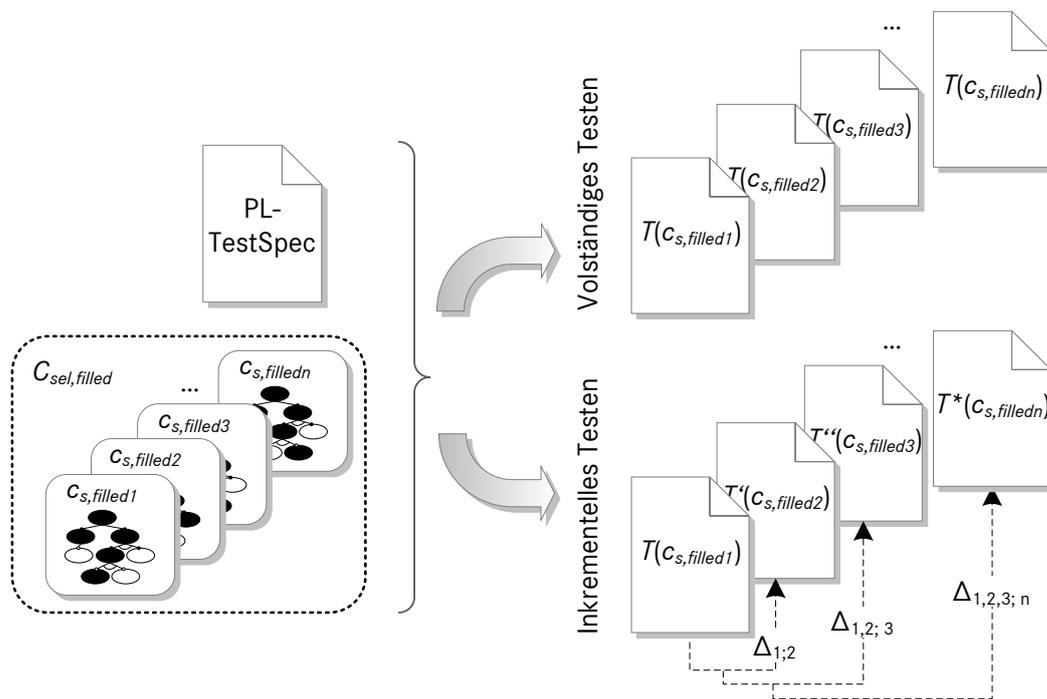


Abbildung 40: Strategien zur Bestimmung von Testfällen (aus der generischen Testspezifikation der Produktlinie) für die selektierten Konfigurationen

diskutiert wurde, zu ermitteln (vgl. Abbildung 40 unten rechts). Das Inkrementelle Testen hat den Vorteil, dass aufgrund der Wiederverwendung keine Testfälle unnötig durchgeführt werden und damit die Effizienz des Testprozesses gesteigert wird. Gleichzeitig ergibt sich ein Zusatzaufwand bei der Analyse der redundanten Testfälle. Dieser ist umso höher, je akkurater die Auswertung durchgeführt wird, um ausschließlich überflüssige Testfälle zu bestimmen. Zudem ist die Festlegung einer initialen Konfiguration, welche einem vollständigen Test unterzogen wird und von welcher ausgehend alle weiteren Testumfänge definiert werden, notwendig. Hierbei ist es sinnvoll Kriterien zur Bestimmung der initialen Konfiguration abzugrenzen.

Gleichzeitig ist zu berücksichtigen, dass das einzige Kriterium, anhand dessen die redundanten Testfälle beim Inkrementellen Testen bestimmt werden würden, den Merkmalen entspricht. An dieser Stelle kann es sinnvoll sein darüber hinaus gehende Kriterien hinzuzuziehen, um eine verlässlichere Aussage machen zu können, ob ein generischer Testfall tatsächlich nur in einer Konfiguration durchgeführt werden sollte. Hierfür kann etwa mithilfe der Regressionstestmethode von [80] eine Auswirkungsanalyse angewendet werden, in deren Rahmen festgestellt werden kann, ob sich die beteiligten Komponenten oder etwa die adressierten Testziele zwischen den Konfigurationen unterscheiden.

Die Umsetzung des Inkrementellen Testvorgehens wird in [66] vorgestellt. Darin werden Konfigurationen mithilfe der Pairwise-Methode selektiert und die entsprechenden Testfälle auf Basis eines produktübergreifenden Testmodells inkrementell generiert. Anhand einer Fallstudie des sogenannten Body Control Systems wird gezeigt, dass

ohne die Methode des Inkrementellen Testens pro selektierte Konfiguration im Mittel 64 Testfälle notwendig wären, wohingegen mit Wiederverwendung im Mittel nur neun Testfälle durchgeführt werden müssten [67]. Somit bewirkt das Inkrementelle Testen in der Fallstudie eine signifikante Reduktion des Testumfangs.

In der vorliegenden Arbeit werden mithilfe des Merkmalmappings die Variationspunkte in den Testfällen direkt abgebildet. Dadurch ist es möglich produktspezifische Umfänge aus der produktübergreifenden Testspezifikation für jede mögliche Konfiguration des entsprechenden Merkmalmodells abzuleiten. Damit ist sowohl die Strategieranwendung des vollständigen als auch die des Inkrementellen Testens bei der Auswahl der Testfälle für die mittels optimierter Anforderungs- und Merkmalabdeckung selektierten Konfigurationen denkbar. Der Einsatz beider Methoden wird im Rahmen von zwei Fallstudien in Abschnitt 7.4.3 näher diskutiert.

Schließlich können weitere Optimierungen erreicht werden, indem das vollständige und/oder das Inkrementelle Testen mit weiteren Methoden kombiniert werden. So kann sich die Festlegung einer Reihenfolge, in welcher die selektierten Konfigurationen zu testen sind, vorteilhaft auf den Testprozess auswirken. Hierbei ist es sinnvoll spezielle Konfigurationen, bei welchen beispielsweise aus Erfahrung eines Testers die meisten Fehler auftreten oder aufgedeckt werden, eine höhere Risikoeinstufung vorliegt, neue/geänderte Funktionen enthalten sind u. Ä., bevorzugt zu Beginn zu testen.

Zusammenfassung

Das vorliegende Kapitel stellt ein Vorgehen zur systematischen Bestimmung von zu testenden Konfigurationen auf Basis der im vorhergehenden Kapitel ausgearbeiteten Produktlinieninfrastruktur vor. Mittels der dabei angestrebten optimierten Anforderungs- sowie Merkmalabdeckung wird gewährleistet, dass zum einen jede in der produktübergreifenden Spezifikation enthaltene Anforderung sowie zum anderen jedes im Merkmalmodell definierte Merkmal in mindestens einer selektierten Konfiguration berücksichtigt und somit getestet wird. Aufgrund der Verwendung des Greedy-Algorithmus sowie der Simulierten Abkühlung, welche jeweils an das vorliegende Problem angepasst worden sind, wird eine bessere Aussage bezüglich der Güte der erzielten Ergebnisse sichergestellt. Darüber hinaus wird eine Vergleichbarkeit der Effizienz zwischen beiden Optimierungsalgorithmen im Rahmen der im folgenden Kapitel behandelten Fallstudien ermöglicht.

Schließlich werden Methoden zur Bestimmung der passenden Testfälle für die selektierten Konfigurationen diskutiert. Grundsätzlich ist es sinnvoll das Inkrementelle Testen zur effizienten Testfallwiederverwendung einzusetzen, jedoch sollte unterstützend eine Auswirkungsanalyse durchgeführt werden.

7 | UMSETZUNG UND VALIDIERUNG

Im vorliegenden Kapitel werden die zuvor ausgearbeitete Konsistenzprüfung sowie die Selektionsmethode innerhalb von zwei Fallstudien aus dem Automobilbereich angewendet und evaluiert. In Abschnitt 7.1 wird zunächst die Umsetzung der Methoden innerhalb eines Tools beschrieben. Darauffolgend werden die zur Evaluierung verwendeten Entwicklungsprojekte, nämlich *Thermischer Komfort* sowie *Ladesystem*, in Abschnitt 7.2 näher vorgestellt. Im Mittelpunkt des Kapitels stehen die Abschnitte 7.3 und 7.4, in welchen die bei der Konsistenzprüfung sowie die bei der Selektionsmethodik erzielten Ergebnisse dargestellt werden. Im Rahmen der Betrachtung werden insbesondere die Resultate des Greedy-Algorithmus mit denen der Simulierten Abkühlung verglichen. Abschließend findet in Abschnitt 7.5 eine umfassende Diskussion und Bewertung der Anwendbarkeit der Methoden sowie des aus den Fallstudien hervorgehenden Nutzens statt.

7.1 Werkzeugunterstützung

Die Konsistenzprüfung (vgl. Abschnitt 5.4.1) sowie Selektionsmethodik (vgl. Abschnitt 6.2) sind prototypisch als ein Plug-In innerhalb des Werkzeugs *pure::variants* in der Programmiersprache Java implementiert. Das Tool *pure::variants* von der Firma *pure-systems* wird dabei während der Produktlinienentwicklung zur Dokumentation und Verwaltung von Merkmalmodellen verwendet [92]. *pure::variants* stellt eine *Eclipse*-basierte Anwendung dar und ist innerhalb der sogenannten Variantenmanagement-Perspektive umgesetzt. Die Perspektive ermöglicht die Darstellung und Verwaltung des *feature model(s)* für die entsprechende Produktlinie als sogenannte **.xfm* Datei. Darin können unter anderem alle in Abschnitt 3.3.2 thematisierten Beziehungen zwischen Merkmalen abgebildet werden. Darüber hinaus werden die für die einzelnen Produkte geltenden Merkmale innerhalb von *variant model(s)* beschrieben und als **.vdm* Dateien gespeichert. Zusätzlich können Entwicklungsartefakte wie Anforderungen, Testmodelle, Code usw. innerhalb eines *family model(s)* als **.ccfm* Datei bearbeitet und an das *feature model* angeknüpft werden. Die Integration verschiedener Entwicklungswerkzeuge ist insbesondere im Hinblick auf ein universelles Variantenmanagement wichtig, da je nach Entwicklungsphase unterschiedliche Werkzeuge eingesetzt werden. So können Anforderungen aus DOORS sowie Modelle aus Mathworks Simulink [120] oder Sparx Systems Enterprise Architect [109] ins Variantenmanagement in *pure::variants* eingebunden werden.

Bei der Daimler AG wird *pure::variants* bei besonders umfangreichen Produktlinien mit komplexer Merkmalsemantik als Ergänzung zu dem in Abschnitt 2.3 bereits vorgestellten DOORS verwendet. Über eine Schnittstelle ist ein Import von den in DOORS dokumentierten Anforderungs-, Testspezifikationen (als **.ccfm* Dateien) sowie des

Variantenmoduls (als *.xfrm Datei) in die Entwicklungsumgebung von pure::variants möglich. Die entwickelten Methoden der vorliegenden Arbeit werden über das Plug-In in pure::variants aufgerufen und durchgeführt. Die erzielten Ergebnisse können über die genannte Schnittstelle anschließend zurück nach DOORS exportiert werden. Dies umfasst einerseits die mittels der Konsistenzprüfung identifizierten und umgesetzten Korrekturen in den Spezifikationsdokumenten sowie andererseits die mithilfe der Optimierungsalgorithmen selektierten Konfigurationen für den Test.

7.2 Aufbau der Fallstudien

Zur Bewertung der Ergebnisse sowie zum Nachweis der grundsätzlichen Praxistauglichkeit, werden die Konsistenzprüfung und Selektionsmethodik im Rahmen von zwei Entwicklungsprojekten aus dem Industriekontext evaluiert. Um eine verlässlichere Aussage bezüglich der Allgemeingültigkeit der Methoden treffen zu können, wurden Projekte, welche aus unterschiedlichen Fachbereichen stammen sowie eine gewisse Größe aufweisen, ausgewählt. Zusätzlich wurden ein Merkmalmodell, welches die Produkte der Produktlinie charakterisiert, sowie eine produktübergreifende System- und Testspezifikation vorausgesetzt. Diese Kriterien werden einerseits vom System *Thermischer Komfort* sowie andererseits vom *Ladesystem* erfüllt. Ein Überblick über den Umfang beider Projekte ist mithilfe der Kenngrößen in Tabelle 15 sowie 16 gegeben.

Artefakt	Quantität
Merkmale im Variantenmodul $ F_{all} $	37
Optionale Merkmalausprägungen $ F_{MC_i} $ (je Merkmalcluster)	5, 24
Alternative Merkmalausprägungen $ F_{MC_{ii}} $ (je Merkmalcluster)	2, 2, 4
Implikationen	4
Exklusionen	29
Konfigurationen	$\sim 10^6$
Konfigurationen $ C_{all} $ aus Merkmalmapping	1088
Anforderungen im Systemlastenheft $ R_{all} $	840
Testfälle in Testspezifikation $ T_{all} $	281
Verlinkungen vor Konsistenzprüfung L_{all}	5068
Verlinkungen nach Konsistenzprüfung	5075
Verweise auf andere Dokumente	343

Tabelle 15: Kenngrößen des Systems *Thermischer Komfort*

Das Projekt aus dem Bereich des *Thermischen Komforts* stellt eine Erweiterung des in der Arbeit durchgängig verwendeten Beispielsystems *Klimatisierung* dar. Das System

Artefakt	Quantität
Merkmale im Variantenmodul $ F_{all} $	20
Optionale Merkmalausprägungen $ F_{MC_i} $ (je Merkmalcluster)	2, 3, 4
Alternative Merkmalausprägungen $ F_{MC_{ii}} $ (je Merkmalcluster)	2, 2, 2, 5
Implikationen	6
Exklusionen	7
Konfigurationen	$\sim 10^3$
Konfigurationen $ C_{all} $ aus Merkmalmapping	600
Anforderungen im Systemlastenheft $ R_{all} $	563
Testfälle in Testspezifikation $ T_{all} $	139
Verlinkungen vor Konsistenzprüfung L_{all}	854
Verlinkungen nach Konsistenzprüfung	854
Verweise auf andere Dokumente	75

Tabelle 16: Kenngrößen des *Ladesystems*

Thermischer Komfort dient zur Konditionierung des Innenraumes eines Fahrzeugs bei entsprechend niedrigen oder hohen Außentemperaturen. Neben den Basisfunktionen Lüften, Kühlen und Heizen bspw. von Auflageflächen, umfasst es zusätzliche kundenerlebbare Komfortfunktionen, welche zur Verbesserung der Luftqualität im Fahrzeuginneren beitragen. Das System enthält eine insgesamt hohe Anzahl an Funktionen, welche als Sonderausstattungen über die Produktfahrzeuge variieren. Diese finden sich innerhalb eines Merkmalclusters mit 24 optionalen Merkmalen (vgl. Tabelle 15) wieder und wirken sich somit unmittelbar auf die Größe des Merkmalmodells aus. Das Merkmalmodell spannt mit insgesamt 37 Merkmalausprägungen einen umfangreichen Konfigurationsraum auf, welcher gleichzeitig von zahlreichen Constraints, insbesondere Exklusionen, wieder eingeschränkt wird. Insgesamt beschreibt das Merkmalmodell des *Thermischen Komforts* etwa 10^6 Konfigurationen, wobei 1088 über das Merkmalmapping in konjunktiver Normalform in den Anforderungs- und Testartefakten direkt abbildbar sind. Das Lastenheft enthält 840 Anforderungen, während die entsprechende Testspezifikation 281 Testfälle aufweist.

Das zweite zur Evaluierung der Methoden verwendete Projekt ist das *Ladesystem*. Das System beschreibt den Ladeprozess einer Hochvolt-Batterie bei Elektro- und Hybridfahrzeugen an einer Ladeinfrastruktur. Im Merkmalmodell der Produktlinie wird bspw. zwischen verschiedenen Lademodi oder länderspezifischen Ladedosen unterschieden. Grundsätzlich spannt das Merkmalmodell mit rund 10^3 möglichen Konfigurationen einen kleineren Konfigurationsraum auf (vgl. Tabelle 16). Dagegen ist die Anzahl der Artefakte in der Anforderungs- und Testspezifikation mit der des *Thermischen Komforts* vergleichbar. Die vergleichsweise kleine Anzahl an Testfällen

kommt dadurch zustande, dass sich die Erstellung des generischen Dokuments in der Entwicklung befindet.

Im Folgenden werden die ausgearbeiteten Methoden auf die beiden Projekte angewendet und die daraus resultierenden Ergebnisse ausführlich vorgestellt und diskutiert.

7.3 Fallstudien zur Konsistenzprüfung

Als erstes erfolgt die Evaluierung der in Abschnitt 5.4.1 entwickelten Konsistenzprüfung. Im Rahmen der Prüfung werden die Verlinkungen zwischen Anforderungen und Testfällen hinsichtlich der auf die Artefakte abgebildeten Variabilität analysiert. Das Ergebnis der Konsistenzprüfung für die Produktlinie des *Thermischen Komforts* ist in Abbildung 41 gezeigt. Darin ist die prozentuale Verteilung aller Verknüpfungen auf die vier möglichen Kategorien¹ – Konsistenz, Unvollständigkeit, Übervollständigkeit, Widerspruch – dargestellt. Die Angaben in/über den Balken spiegeln die absolute Anzahl der jeweils einsortierten Verlinkungen wider. Gleichzeitig werden die Verteilungen vor und nach der Umsetzung von Korrekturmaßnahmen gegenübergestellt.

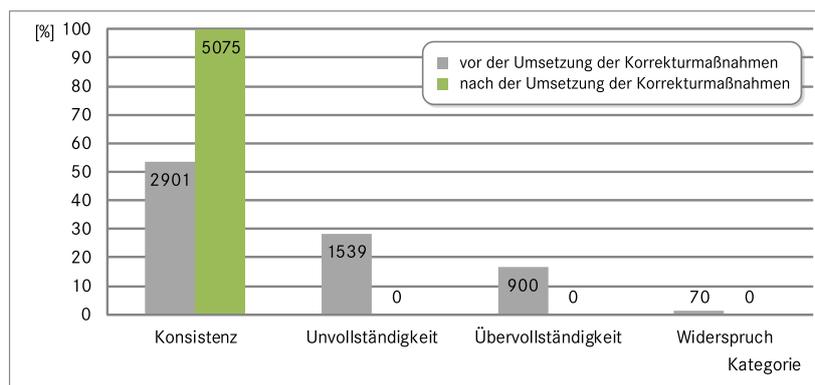


Abbildung 41: Ergebnis der Konsistenzprüfung für das System *Thermischer Komfort* mit der Verteilung der analysierten Verlinkungen auf die entsprechenden Kategorien vor und nach der Umsetzung von Korrekturen

Wie aus der Graphik entnommen werden kann, weist rund die Hälfte der Verknüpfungen ein konsistentes und damit korrektes Merkmalmapping auf. Die größte Gruppe der identifizierten Inkonsistenzen liegt in der Kategorie der Unvollständigkeit mit knapp 30% aller Verlinkungen vor, gefolgt von 17% übervollständiger und schließlich etwa 1% widersprüchlicher Links.

Nach der Durchführung der Prüfung erfolgt zunächst eine Untersuchung hinsichtlich der für die Entstehung der Inkonsistenzen verantwortlichen Ursachen. Hierfür werden

¹ Die Anzahl der in Kategorien-Klasse 4.2.4 einsortierten Verlinkungen wird sowohl zu der Kategorie Unvollständigkeit als auch Übervollständigkeit gezählt, da jeweils fehlende und überschüssige Merkmale korrigiert werden müssen.

die Ergebnisse den einzelnen Kategorien-Klassen zugeordnet (vgl. obere Graphik in Abbildung 42) und ausgewertet.

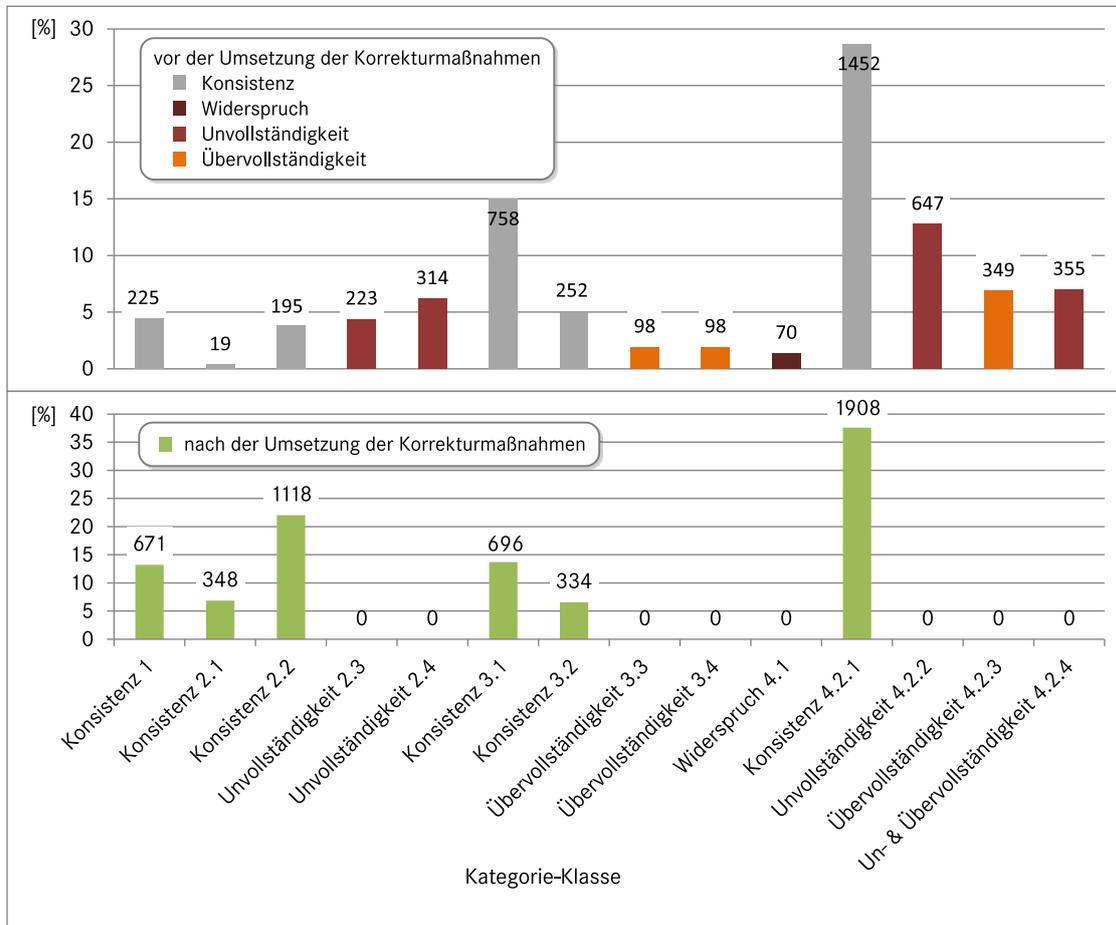


Abbildung 42: Ergebnis der Konsistenzprüfung für das System *Thermischer Komfort* mit der Verteilung der analysierten Verlinkungen auf die entsprechenden Kategorien-Klassen vor (oben) und nach (unten) der Umsetzung von Korrekturen

Grundsätzlich ist erkennbar, dass sich die meisten Verknüpfungen in der Kategorie 4 wiederfinden. Die Kategorie kennzeichnet dabei solche Fälle, in welchen ausgehend von einem Link sowohl die Merkmale weiterer Anforderungen als auch die weiterer Testfälle berücksichtigt werden müssen. Dies lässt auf besonders verflochtene Zusammenhänge zwischen den Anforderungen und Testfällen schließen und bestätigt somit die Komplexität der im Projekt vorliegenden $n : m$ -Verknüpfungen.

Die bei der Umsetzung der Korrekturmaßnahmen vorgenommene Analyse zeigt, dass weit über die Hälfte der gefundenen Fehler aufgrund von fehlenden oder überschüssig zugeordneten Merkmalen in den Artefakten entstehen. Entgegen der Annahme, dass sich die meisten Fehler auf die Testspezifikation beschränken, werden zahlreiche Unstimmigkeiten bezüglich der abgebildeten Merkmale ebenfalls in den Anforderungen identifiziert. Diese kommen dadurch zustande, dass bei einigen Anforderungen kein

Merkmalmapping vorgenommen wird und die entsprechenden Felder somit leere Einträge aufweisen. Solche Anforderungen gelten automatisch für alle Konfigurationen des Merkmalmodells und die entsprechenden Testfälle decken in der Regel nicht alle davon ab. Das fälschlicherweise „leere“ Merkmalmapping kommt in beiden Spezifikationen, jedoch häufiger im Lastenheft vor, wodurch zahlreiche als unvollständig kategorisierten Verlinkungen resultieren. Zusätzlich wird vermutet, dass die hohe Anzahl an Merkmalen sowie die teils lang gewählten Bezeichnungen der Merkmale zu einer erhöhten Fehleranfälligkeit beim Mapping führt.

Des Weiteren machen falsche Verlinkungen zwischen Artefakten etwa ein Drittel der Inkonsistenzen aus. Für den *Thermischen Komfort* werden mehr fehlende als überschüssige Verlinkungen festgestellt. Die Ursache dafür liegt einerseits in dem Umfang des Projekts und dem dadurch erschwerten Überblick über die hohe Anzahl an Verknüpfungen pro Artefakt. Andererseits unterliegen Spezifikationsdokumente der Entwicklung und daher einer Reihe von dauerhaften Änderungen. Insbesondere in der Testspezifikation werden regelmäßig neue Artefakte spezifiziert, alte angepasst oder gelöscht. Werden betroffene Verknüpfungen unzureichend angepasst, entsprechen sie nicht dem neuesten Stand und können dadurch Inkonsistenzen enthalten. Schließlich liegt die Vermutung nahe, dass das manuelle Verlinken, welches einen fehleranfälligen Prozess darstellt, zu einem Teil der Inkonsistenzen beiträgt.

Für das System *Thermischer Komfort* sind alle Korrekturmaßnahmen entsprechend der Vorschläge umgesetzt und schließlich die Konsistenz zwischen den Artefakten des Projektes hergestellt worden (vgl. untere Graphik in Abbildung 42). Dabei wirkt sich häufig eine Korrektur gleich auf mehrere Verknüpfungen und auch mehrere Kategorien aus, sodass nicht jede der 2509 identifizierten Inkonsistenzen separat analysiert und ausgebessert werden muss. Dieser Effekt ist ebenfalls daran erkennbar, dass sich die Anzahl der konsistenten Verknüpfungen der Kategorien-Klasse 3.1 nach dem Einpflegen der Korrekturen insgesamt verringerte. Demnach sind die entsprechenden Links aufgrund des Korrekturprozesses in eine andere Konsistenz-Klasse transferiert worden. Wegen der genannten Verkettung von Korrekturen, aber auch aus implementierungstechnischen Gründen ist es schwierig genauer zu quantifizieren, durch welche Modifikationen welche und wie viele Inkonsistenzen behoben werden.

Analog dazu ist die Konsistenzprüfung auf das zweite Projekt, die Produktlinie *Ladesystem*, angewendet worden. Das entsprechende Ergebnis ist in Abbildung 43 gezeigt. Hierbei werden knapp 30% der miteinander verknüpften Artefakte der Kategorie der Konsistenz zugeordnet, während 7% der Links als unvollständig und nur ein Link als widersprüchlich identifiziert sind. Besonders auffällig ist die Kategorie der Übervollständigkeit, welche über 65% aller Verknüpfungen ausmacht.

Ähnlich zu dem ersten Projekt wird die Zuordnung zu den Kategorien-Klassen für *Ladesystem* (vgl. Abbildung 44 oben) vorgenommen. Daraus ist ersichtlich, dass im Gegensatz zum *Thermischen Komfort* weitaus weniger komplexe $n : m$ -Verknüpfungen der Kategorie 4 analysiert worden sind. Insgesamt stellt das *Ladesystem* ein bezüglich des Evaluierungsumfangs eher kleineres Projekt dar. Dies ist hauptsächlich dadurch zu begründen, dass die Testspezifikation des *Ladesystems* einem sich in der Entwicklung befindlichen Dokument entspricht, und dadurch nicht nur eine vergleichsweise geringe Anzahl an Testartefakten, sondern auch an Verknüpfungen zum Spezifikationsdokument aufweist.

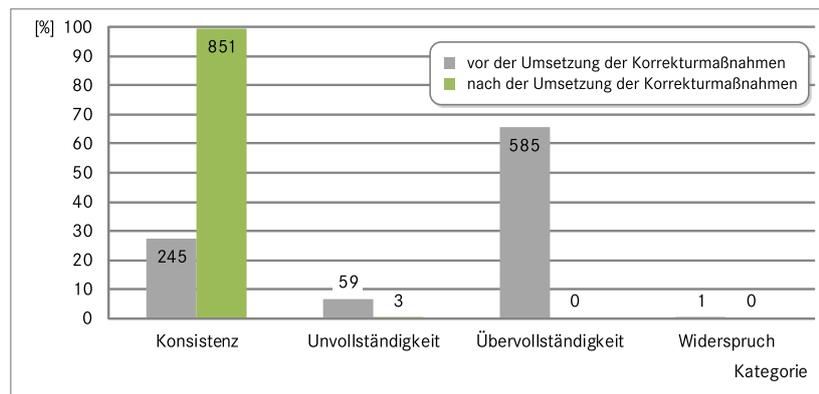


Abbildung 43: Ergebnis der Konsistenzprüfung für das *Ladesystem* mit der Verteilung der analysierten Verlinkungen auf die entsprechenden Kategorien vor und nach der Umsetzung von Korrekturen

Eine zentrale Fehlerursache der meisten entdeckten Inkonsistenzen ist die Tatsache, dass das Merkmalmapping der beiden analysierten Spezifikationen unterschiedlichen Ständen entspricht. Während in der Testspezifikation die Umfänge des induktiven Ladens bereits einigen Testfällen zugeordnet worden sind, sind die Umfänge in der Anforderungsspezifikation nicht berücksichtigt worden. Diese Inkonsistenzen spiegeln sich in dem auffälligen Balken der übervollständigen Verknüpfungen wider.

Die meisten unvollständigen Verknüpfungen resultieren aus fälschlicherweise „leeren“ Merkmaleinträgen innerhalb einiger Anforderungen. Ähnliche Fälle sind bereits im Rahmen der Analyse des *Thermischen Komforts* festgestellt und erläutert worden.

Die Korrekturvorschläge konnten für das *Ladesystem* nahezu vollständig umgesetzt werden. Da entsprechende Anforderungen im Lastenheft zum Zeitpunkt der Auswertung noch nicht ergänzt waren, sind drei inkonsistente Verknüpfungen geblieben.

Die Abhängigkeit der Laufzeit von der Anzahl der Verlinkungen sowie von der Anzahl der Konfigurationen wird im Rahmen einer empirischen Untersuchung betrachtet. Diesbezüglich wird jeweils eine (erwartete) annähernd lineare Abhängigkeit ermittelt, welche in Abbildung 45 dargestellt ist. Die Streuung der Datenpunkte um die Gerade könnte daher rühren, dass die Größen, welche die Rechenzeit beeinflussen, nicht unabhängig voneinander sind und bspw. beim Modifizieren von $|C_{all}|$ auch $|C_{def/exc}|$ verändert wird. Die stärkere Steigung der Laufzeit für den *Thermischen Komfort* hängt damit zusammen, dass das Projekt insgesamt umfangreicher ist. Dabei werden pro Verlinkung im Mittel mehr Konfigurationen miteinander verglichen und gleichzeitig umfangreichere Korrekturmaßnahmen berechnet. Der Schnitt mit der y-Achse erfolgt im linken Teil der Graphik 45 für beide Projekte bei $t > 0$, obwohl keine Links analysiert werden. Dies spiegelt die Zeit, welche zur initialen Berechnung der Basis für die Konsistenzprüfung notwendig ist sowie gegebenenfalls Tool-spezifische Initialisierungsabläufe wider. Für $|C_{all}| = 0$ ist die Durchführung der Konsistenzprüfung Tool-technisch nicht vorgesehen.

Insgesamt erfolgt die Prüfung vor der Umsetzung der Korrekturmaßnahmen für das System *Thermischer Komfort* im Mittel innerhalb von $t_{\emptyset} = 60,7 \pm 2,3$ s, während

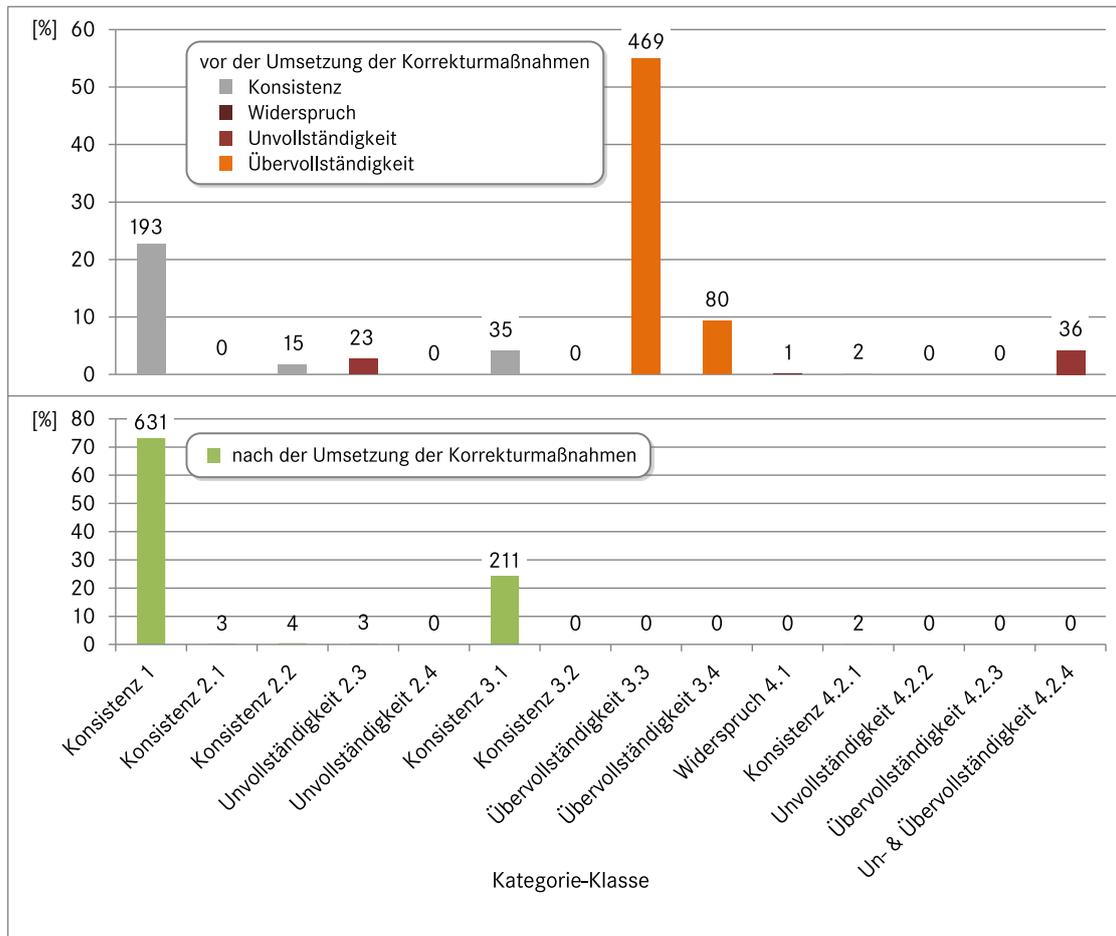


Abbildung 44: Ergebnis der Konsistenzprüfung für das *Ladesystem* mit der Verteilung der analysierten Verlinkungen auf die entsprechenden Kategorien-Klassen vor (oben) und nach (unten) der Umsetzung von Korrekturen

die Laufzeit für das *Ladesystem* im Schnitt $t_{\ominus} = 10 \pm 0,8s$ beträgt². Die Angaben repräsentieren jeweils den Mittelwert aus 10 Durchläufen mit der entsprechenden Standardabweichung. Nach der Umsetzung der Korrekturmaßnahmen erfolgt die Konsistenzprüfung innerhalb von $t_{\ominus} = 56,7 \pm 2s$ für das System *Thermischer Komfort* sowie von $t_{\ominus} = 7,4 \pm 0,7s$ für das *Ladesystem*. Die Tatsache, dass sich die Laufzeit nur unwesentlich verkürzt, deutet darauf hin, dass der zweite Term von 5.34 keinen starken Einfluss auf die Laufzeit hat.

² Alle Fallstudien sind auf einem 2,4 GHz Rechner mit zwei Prozessorkernen und 4 GB RAM durchgeführt worden.

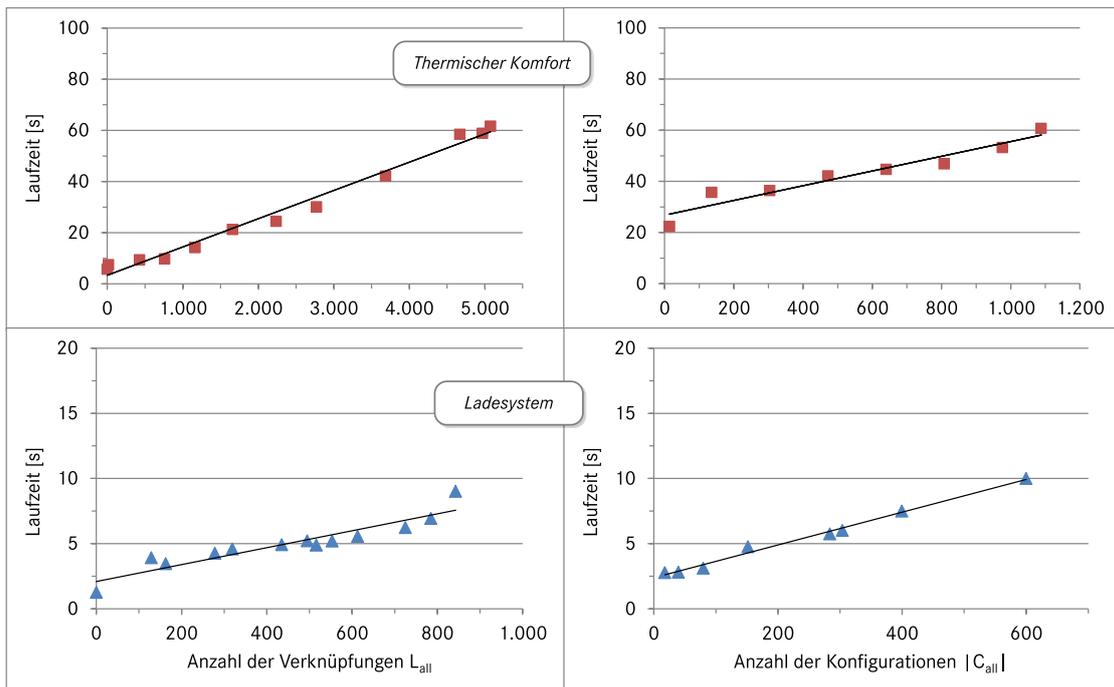


Abbildung 45: Abhängigkeit der Laufzeit der Konsistenzprüfung von der Anzahl der zu analysierenden Verlinkungen zwischen Anforderungen und Testfällen sowie von der Anzahl der Konfigurationen jeweils mit linearer Trendlinie (schwarz)

7.4 Fallstudien zur Selektionsmethodik

Durch die konsistent spezifizierten Anforderungen und Testfälle wird die Basis für die anschließende Selektion der optimalen Menge der zu testenden Konfigurationen gelegt. In den folgenden beiden Abschnitten werden die bei der Selektionsmethodik erzielten Ergebnisse mittels des Greedy-Algorithmus und der Simulierten Abkühlung vorgestellt und miteinander verglichen.

7.4.1 Fallstudien zur Selektion mittels Greedy-Algorithmus

Zunächst erfolgt die Evaluierung der auf Basis des Greedy-Algorithmus durchgeführten Selektion. Hierbei sucht das in Abschnitt 6.2.2.2 ausgearbeitete Optimierungsverfahren innerhalb einer Menge valider Konfigurationen, welche aus dem kartesischen Produkt der Merkmalcluster resultieren, nach einer möglichst kleinen Konfigurationsmenge, die alle Anforderungen und Merkmale mindestens einmal abdeckt.

Das Ergebnis für das System *Thermischer Komfort* ist in Abbildung 46 dargestellt. Die Graphik zeigt, dass mithilfe des Greedy-Algorithmus acht Konfigurationen, welche eine 100%ige Anforderungs- sowie Merkmalabdeckung (kumuliert) gewährleisten, selektiert werden. Im linken Teil der Abbildung ist erkennbar, dass mit den ersten vier

Konfigurationen bereits über 90% aller Anforderungen sowie Merkmale abgedeckt werden. Daraus kann geschlussfolgert werden, dass die entsprechende Anforderungsspezifikation eine Vielzahl generischer Artefakte enthält und daher der Ansatz einer optimierten Selektion insgesamt als sinnvoll zu erachten ist.

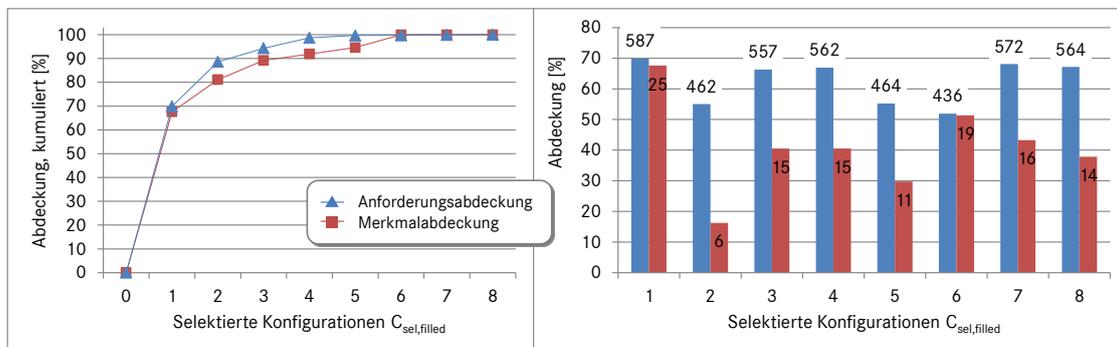


Abbildung 46: Ergebnis des Greedy-Algorithmus für das System *Thermischer Komfort*

Im rechten Teil der Graphik 46 sind die pro Konfiguration jeweils prozentual sowie (über/in den Balken) absolut abgedeckten Anforderungen und Merkmale abgebildet. Jede Konfiguration deckt im Mittel 62,6% der Anforderungen ab, wobei die jeweiligen Anforderungsabdeckungen mit einer Standardabweichung von 7,2% um den Mittelwert streuen. Von den Merkmalen werden pro Konfiguration im Mittel 40,9% mit einer Standardabweichung von 15% abgedeckt. Die stärkere Streuung der Merkmalabdeckung für den *Thermischen Komfort* resultiert aus dem Befüllen der undefinierten Merkmaleinträge. Dabei wird innerhalb der ersten selektierten Konfiguration das umfangreiche optionale Merkmalcluster der in Abschnitt 7.2 bereits diskutierten Sonderausstattungen zugunsten einer maximalen Merkmalabdeckung mit f_y befüllt, sodass dadurch 25 der 37 möglichen Merkmale abgedeckt werden. Dagegen werden in der danach selektierten, zweiten Konfiguration dieselben Merkmale zugunsten einer gleichmäßigen Abdeckung mit $\neg f_y$ befüllt, weshalb dadurch nur sechs Merkmale abgedeckt werden. Ab der dritten Konfiguration werden die offenen Merkmale nach dem Zufallsprinzip befüllt, wodurch die Abdeckungen nicht mehr so stark um den Mittelwert streuen.

Zusätzlich liegt ab der sechsten Konfiguration bereits eine 100%ige Merkmalabdeckung vor. Aus diesem Grund entscheidet der Greedy-Algorithmus ab diesem Punkt nur noch auf Basis der maximierten Anforderungsabdeckung. Dies bestätigt die anfängliche Annahme, dass mit einer Anforderungsabdeckung gleichzeitig die Merkmalabdeckung erreicht werden kann. Demzufolge ist es sinnvoll die Abdeckung der Anforderungen bei der Greedy-Selektion höher zu priorisieren (und in der Energiefunktion der Simulierten Abkühlung deutlich stärker zu gewichten) als die der Merkmale.

Ebenfalls interessant ist die Tatsache, dass sowohl innerhalb der Fallstudie des *Thermischen Komforts* als auch der des *Ladesystems* keine Anforderungen vorhanden sind, welche von genau einer Konfiguration abgedeckt werden. Die in Algorithmus 1 beschriebene Initialisierungsphase wurde somit in beiden Fällen nicht angewendet.

Obwohl das in Abschnitt 6.2.2.2 beschriebene Optimierungsvorgehen nicht deterministisch ist, da auch zufällige Entscheidungen enthalten sind (vgl. Zeile 18 und 21 in

Algorithmus 1), liefert der Greedy-Algorithmus immer dieselbe Anzahl von acht selektierten Konfigurationen. Dies deutet darauf hin, dass die Selektionsentscheidungen entweder immer eindeutig sind oder dass sie keine Auswirkung auf die Anzahl der selektierten Konfigurationen haben. Diese Tatsache trifft ebenfalls auf das im Folgenden vorgestellte Projekt *Ladesystem* zu.

Zur Bewertung des mittels des Greedy-Algorithmus erzielten Ergebnisses wird zusätzlich die in Abschnitt diskutierte 4.2.6 Pairwise-Methode mithilfe des Tools *MoSo-PoLiTe* [83] auf das Merkmalmodell der Produktlinie *Thermischer Komfort* angewendet. Das Vorgehen selektiert 32 Konfigurationen, durch welche jedes Merkmalpaar mindestens einmal berücksichtigt ist. Im Vergleich dazu stellen die acht identifizierten Konfigurationen auf Basis einer optimierten Anforderungs- und Merkmalabdeckung eine deutliche Reduktion des Testumfangs und damit eine Verbesserung dar. Eine Analyse ergibt, dass durch die 32 Konfigurationen ebenfalls eine 100%ige Anforderungsabdeckung der Spezifikation erreicht wird. Dieser positive Nebeneffekt ist jedoch zufällig und kann nicht für jedes Projekt garantiert werden, da die Anforderungen durch das Pairwise-Vorgehen in die Selektion nicht einbezogen werden.

Das mithilfe des Greedy-Algorithmus ermittelte Selektionsergebnis für das *Ladesystem* ist in Abbildung 47 gezeigt. Darin ist erkennbar, dass für das *Ladesystem* fünf Konfigurationen zur vollständigen Anforderungs- und Merkmalabdeckung selektiert werden. Im Gegensatz zum *Thermischen Komfort* werden mit der ersten selektierten Konfiguration bereits rund 98% aller Anforderungen abgedeckt (vgl. linken Teil der Graphik 47). Die Ursache dafür ist, dass aufgrund des innovativen Charakters des *Ladesystems* dessen Spezifikationen sich in der Entwicklung befindliche Dokumente darstellen und darin einige Inhalte, wie das induktive Laden, wesentlich umfangreicher beschrieben sind als bspw. das induktive Laden. Aus diesem Grund werden mit der ersten Konfiguration die detailliert spezifizierten und daher meisten Umfänge, dagegen mit den nachfolgend selektierten Konfigurationen die noch nicht ausführlich spezifizierten und somit geringeren Anforderungsumfänge abgedeckt (vgl. rechten Teil der Graphik 47).

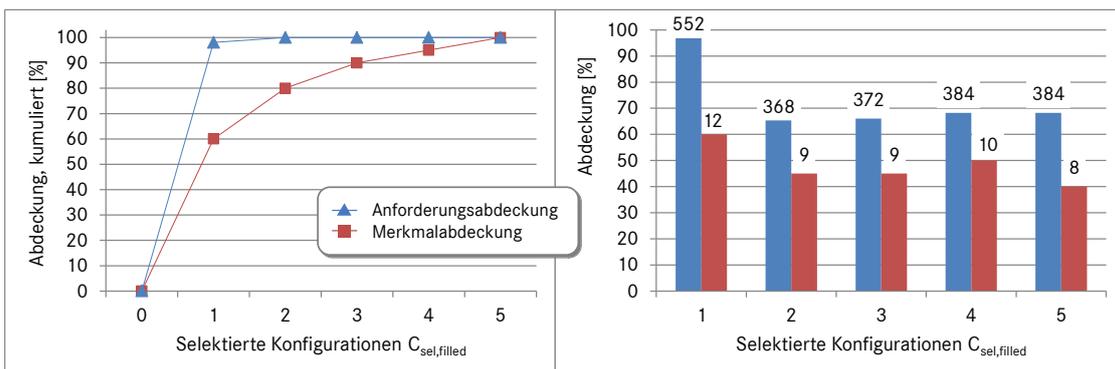


Abbildung 47: Ergebnis des Greedy-Algorithmus für das *Ladesystem*

Dies spiegelt sich ebenso in der stärkeren Streuung der einzelnen Anforderungsabdeckungen mit einer Standardabweichung von 14% um den Mittelwert von 73,2% wider. Dagegen liegt die durchschnittliche Merkmalabdeckung bei 48% mit einer Standardabweichung von 7,6%. Da keine große Anzahl an undefinierten Merkmaleinträgen befüllt

werden muss, wie dies für den *Thermischen Komfort* der Fall ist, ist die Merkmalabdeckung für jede Konfiguration ähnlich hoch. Ein weiterer Unterschied zum *Thermischen Komfort* besteht darin, dass ab der zweiten selektierten Konfiguration eine 100%ige Anforderungsabdeckung bereits erreicht ist. Dies liegt an dem stark generischen Charakter der Anforderungsspezifikation sowie den, wie bereits zuvor diskutiert, wenigen konfigurationsspezifischen Umfängen. Somit entscheidet der Greedy-Algorithmus ab diesem Punkt nur noch anhand einer zu maximierenden Merkmalabdeckung.

Durch das zusätzliche Berücksichtigen der Kenngrößen des *Ladesystems* aus Tabelle 16 ist festgestellt worden, dass das größte alternative Merkmalcluster des Variabilitätsmodells fünf Merkmalausprägungen aufweist. Da sich die alternativen Merkmalausprägungen gegenseitig ausschließen, sind mindestens fünf Konfigurationen notwendig, um diese abzudecken. Demzufolge kann keine kleinere Anzahl an Konfigurationen für das *Ladesystem* selektiert werden und das Ergebnis des Greedy-Algorithmus stellt in diesem Fall das gesuchte globale Optimum dar.

Im Vergleich dazu werden für die Pairwise-Abdeckung 19 Konfigurationen aus dem Merkmalmodell der Produktlinie *Ladesystem* selektiert. Dies stellt einen zum *Thermischen Komfort* insofern ähnlichen Wert dar, als dass dieser ebenso rund viermal höher ist, als der mittels der optimierten Anforderungs- und Merkmalabdeckung ermittelte Wert. Gleichzeitig liefern die zur Abdeckung der Merkmalpaare bestimmten Konfigurationen eine 100% Anforderungsabdeckung. Jedoch ist diese Tatsache, wie bereits diskutiert, nicht immer garantiert, sondern projektspezifisch.

Die in Abschnitt 6.2.2.2 erläuterte Laufzeitkomplexität des Greedy-Algorithmus wird für beide Projekte im Rahmen einer empirischen Untersuchung betrachtet. Die dabei erfasste Abhängigkeit der Laufzeit ist in Abbildung 48 präsentiert. Die beiden im Normalfall dominierenden Größen, welche einen Einfluss auf die Laufzeit des Algorithmus haben, sind zum einen die Anzahl der Anforderungen und zum anderen die Anzahl der Konfigurationen. Dabei entspricht die zuletzt genannte Größe den aufgrund des Merkmalmapping in konjunktiver Normalform resultierenden Konfigurationen und drückt im linken Teil der Graphik 48 mit $|C_{all}|$ die anfängliche Berechnung der Priority Queues aus. Für beide Einflussfaktoren wird eine (erwartete) annähernd lineare Abhängigkeit der Rechenzeit ermittelt. Die sowohl im rechten als auch linken Teil der Graphik abgebildete größere Steigung der Laufzeit für den *Thermischen Komfort* resultiert daher, dass einerseits die Priority Queues bei jedem Suchdurchlauf aktualisiert werden müssen und dieser Vorgang aufgrund des größeren Konfigurationsraumes aufwändiger ist sowie andererseits pro selektierte Konfiguration deutlich mehr Merkmale befüllt werden müssen. Die Durchführung der Suche ist sowohl für $|R_{all}| = 0$ als auch $|C_{all}| = 0$ Tool-technisch nicht vorgesehen. Die Streuung der Datenpunkte um die Geraden, ist mit den von einander abhängenden Größen, welche die Rechenzeit beeinflussen, erklärbar.

Insgesamt erfolgt die Selektion mithilfe des Greedy-Algorithmus beim *Thermischen Komfort* innerhalb von $t_{\circ} = 30,8 \pm 1$ s und für das *Ladesystem* innerhalb von $t_{\circ} = 10,3 \pm 0,6$ s. Die Angaben repräsentieren jeweils den Mittelwert aus 10 Durchläufen mit der entsprechenden Standardabweichung.

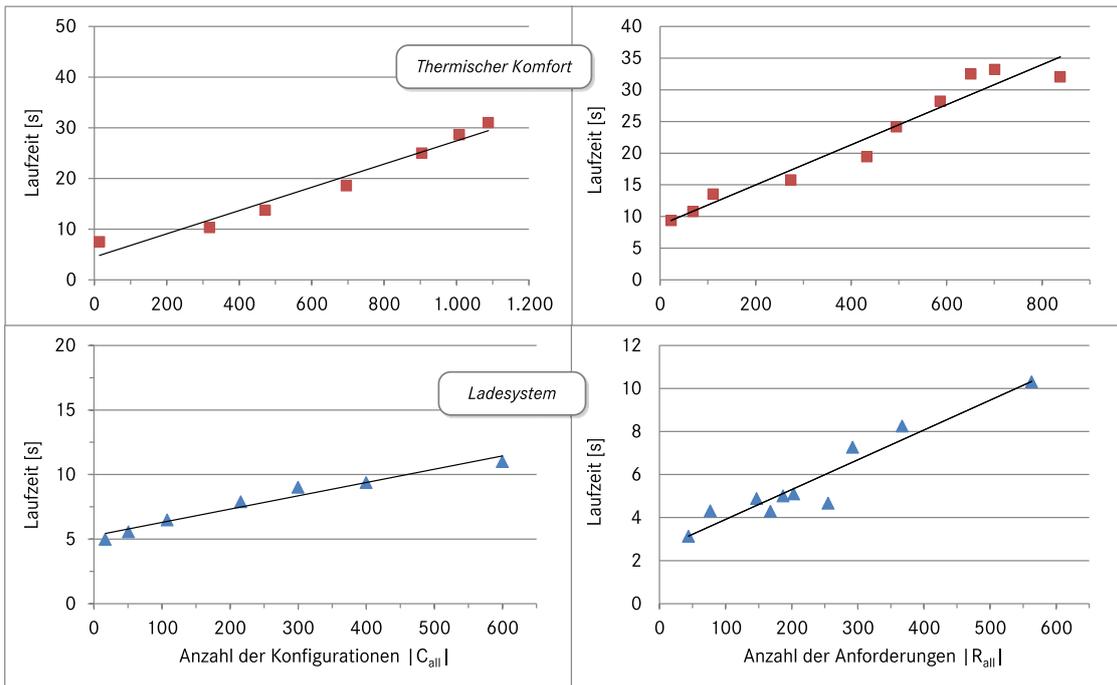


Abbildung 48: Abhängigkeit der Laufzeit des Greedy-Algorithmus von der Anzahl der Konfigurationen (links) sowie der Anzahl der Anforderungen (rechts) jeweils mit linearer Trendlinie (schwarz)

7.4.2 Fallstudien zur Selektion mittels Simulierter Abkühlung

Die mithilfe der Simulierten Abkühlung erzielten Selektionsergebnisse lassen sich in lokale und globale Anteile gruppieren. Die lokalen Ergebnisse sind in Abbildung 49 und 50 dargestellt und spiegeln das klassische Suchverfahren der Simulierten Abkühlung innerhalb einer festen Konfigurationssetgröße wider.

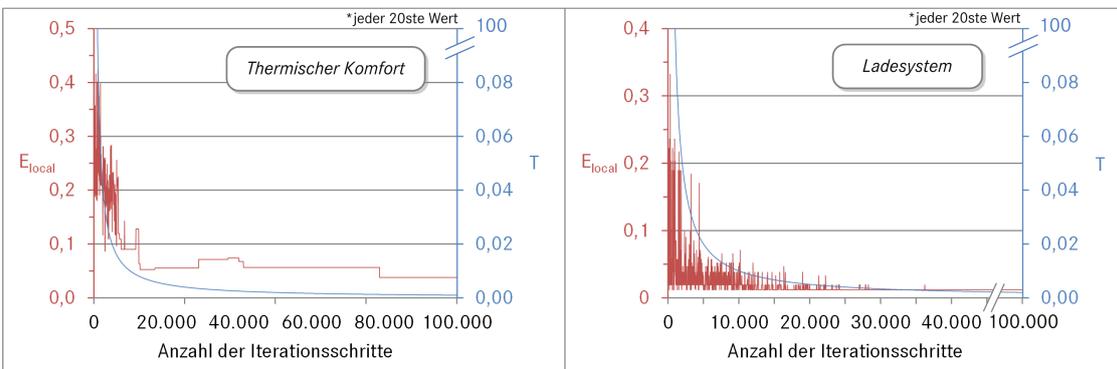


Abbildung 49: Verlauf der lokalen Energie sowie der Temperatur bei der Optimierung mittels Simulierter Abkühlung mit $T_{init} = 100$, $T_{min} = 0,001$, $\beta = 0,01$ und $N_{TK} = G_{TK} = 8$, $N_{LS} = G_{LS} = 5$

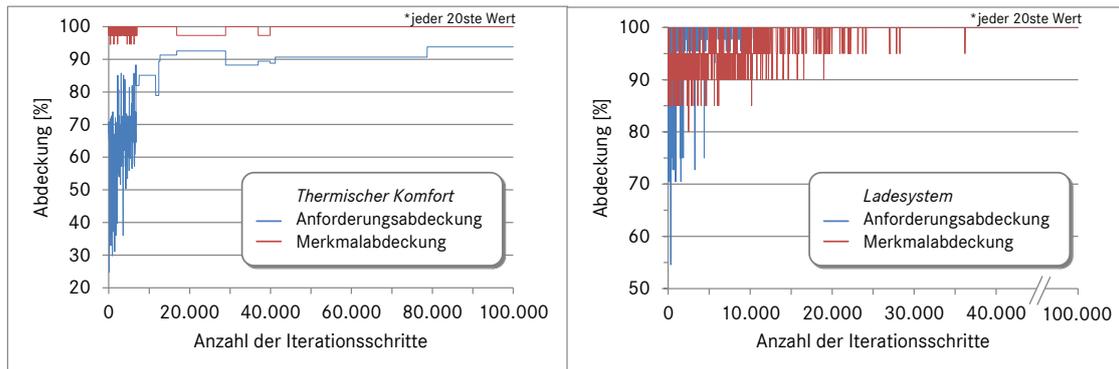


Abbildung 50: Von der Simulierten Abkühlung lokal erreichte Abdeckungen mit $T_{init} = 100$, $T_{min} = 0,001$, $\beta = 0,01$, $N_{TK} = G_{TK} = 8$ und $N_{LS} = G_{LS} = 5$

Darin nähert sich die Heuristik anhand der Energiefunktion E_{local} und gegebenenfalls der Wahrscheinlichkeit $P(E_{local}, E_{new}, T)$ einem Optimum an (vgl. Abschnitt 6.2.3). In Abbildung 49 ist für die Produktlinie *Thermischer Komfort* (links) sowie *Ladesystem* (rechts) jeweils ein Konfigurationsset N , welcher der Größe der vom Greedy-Algorithmus ermittelten Lösung G entspricht, gezeigt. Diese beträgt $G_{TK} = 8$ für den *Thermischen Komfort* und $G_{LS} = 5$ für das *Ladesystem* (vgl. Abschnitt 7.4.1). Ergebnisse für andere Konfigurationssetgrößen, welche in der Nähe der Greedy-Lösungsgröße liegen (bspw. $G \pm 5$), weisen ähnliche Verläufe auf. Weitere Parameter wie T_{init} , T_{end} und β sind empirisch durch Abwägen des Aufwandes (Rechenzeit) gegen den Nutzen (gewünschte Lösungsqualität) ermittelt worden. Aufgrund der hohen Anzahl an Datenpunkten ist zwecks besserer Übersicht jeweils nur jeder 20ste Wert aufgetragen.

In Graphik 49 ist die lokale Energie bei jedem Iterationsschritt abgebildet. Gleichzeitig ist der Temperaturverlauf jeweils auf der rechten Achse der Abbildung aufgetragen. Die Temperaturkurve zeigt einen sehr schnellen Abfall bei hohen Werten und eine anschließend langsame Konvergenz zum Nullpunkt. Sowohl für den *Thermischen Komfort* als auch das *Ladesystem* werden zu Beginn der Suche eher hohe und somit schlechte Energiewerte erreicht. Zusätzlich schwankt E_{local} am Anfang verhältnismäßig stärker zwischen besseren und schlechteren Ergebnissen. Das hängt damit zusammen, dass aufgrund der hohen Temperatur höhere Energiewerte wahrscheinlicher akzeptiert werden als dies bei niedrigeren Temperaturen der Fall ist. Je weiter die Temperatur abklingt, desto unwahrscheinlicher wird es, dass noch schlechtere Zustände angenommen werden, bis die Optimierung schließlich in eine lokale Suche übergeht. Dies spiegelt sich für den *Thermischen Komfort* in den flachen Verläufen ab dem ~ 40.000 sten Iterationsschritt wider. Dagegen verbessert sich die Lösung beim *Ladesystem* bereits nach dem ~ 37.000 sten Schritt nicht mehr, weshalb der Bereich danach ausgeblendet ist. Das Ergebnis für das *Ladesystem* stellt das tatsächlich gesuchte globale Optimum dar, wie in Abschnitt 7.4.1 bereits diskutiert.

In Abbildung 50 sind die bei jedem Iterationsschritt erzielten Anforderungs- und Merkmalabdeckungen für den *Thermischen Komfort* (links) sowie *Ladesystem* (rechts) präsentiert. Die Verläufe entsprechen den beiden zu optimierenden Größen innerhalb der lokalen Energie aus Abbildung 49, während die Konfigurationssetgröße jeweils

konstant ist. Aus der Abbildung ist ersichtlich, dass für den *Thermischen Komfort* bereits zu Beginn der Suche gute Lösungen hinsichtlich der Merkmalabdeckung erreicht und nach dem ~ 40.000 sten Schritt die Merkmale vollständig abgedeckt werden. Die schnelle Merkmalabdeckung ist ebenfalls bei der Selektion mithilfe des Greedy-Algorithmus festgestellt worden (vgl. Abschnitt 7.4.1). Demgegenüber beginnt die Simulierte Abkühlung mit wesentlich schlechteren Lösungen im Bezug auf die Anforderungsabdeckung und nähert sich der vollständigen Abdeckung langsam an. Für das Konfigurationsset mit $N_{TK} = 8$ wird keine 100%ige Anforderungsabdeckung erreicht. Dagegen werden beim *Ladesystem* die Anforderungen und Merkmale bei ähnlichem Temperaturwert vollständig abgedeckt.

Die entsprechenden globalen Ergebnisse der Simulierten Abkühlung sind in Abbildung 51 sowie 52 dargestellt. Diese spiegeln die beste Lösung aus allen betrachteten Konfigurationssetgrößen wider. In Abbildung 51 ist die Größe des global besten Konfigurationssets auf der linken sowie die entsprechende globale Energie auf der rechten Achse für den *Thermischen Komfort* (links) und *Ladesystem* (rechts) jeweils aufgetragen. Die x-Achse zeigt den jeweiligen Iterationsschritt, bei welchem der entsprechende Wert aktualisiert wird. Da hierbei das global beste Ergebnis verfolgt wird, werden ausschließlich bessere Lösungen akzeptiert und E_{global} kann somit im Gegensatz zu E_{local} mit den Iterationsschritten nur abnehmen. Die optimale Größe des Konfigurationssets wird für beide Systeme innerhalb von $G_{TK,LS} \pm 3$ gesucht, sodass in Abbildung 51 $|C_{global}|$ innerhalb der gesetzten Grenzen variiert. Die Tatsache, dass E_{global} auch bei wachsenden Werten von $|C_{global}|$ verbessert wird, deutet auf die höhere Priorisierung der zu optimierenden Anforderungs- sowie Merkmalabdeckung, welche mithilfe von Gewichtungsfaktoren in der Energiefunktion ausgedrückt ist, hin.

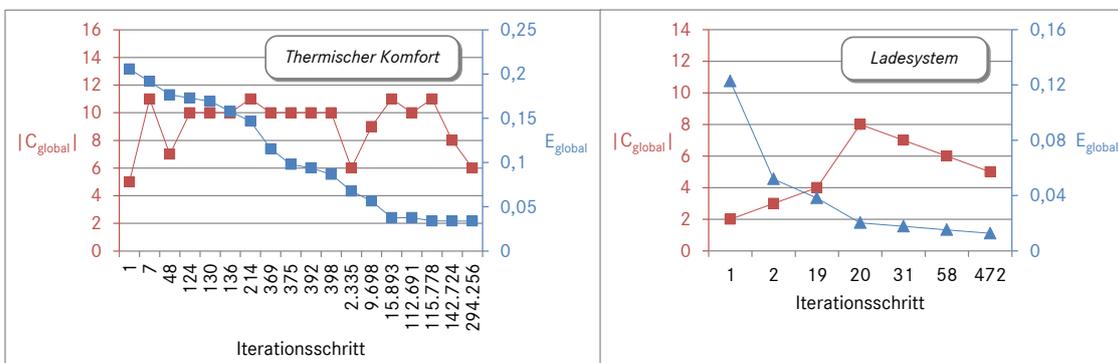


Abbildung 51: Von der Simulierten Abkühlung global selektierte Konfigurationen sowie der Verlauf der globalen Energie mit $T_{init} = 100$, $T_{min} = 0,001$, $\beta = 0,01$, $(G_{TK} - 3) \leq N_{TK} \leq (G_{TK} + 3)$ und $(G_{LS} - 3) \leq N_{LS} \leq (G_{LS} + 3)$

Abbildung 52 zeigt die jeweiligen Abdeckungen, welche vom entsprechenden C_{global} aus Abbildung 51 erreicht werden. Für den *Thermischen Komfort* wird das vom Greedy-Algorithmus ermittelte Ergebnis einer schnellen Merkmalabdeckung bestätigt. Die Anforderungsabdeckung nähert sich dagegen langsam dem Optimum an, erreicht jedoch nicht die 100%. Beim *Ladesystem* weisen die letzten vier Werte keine Verbesserung der

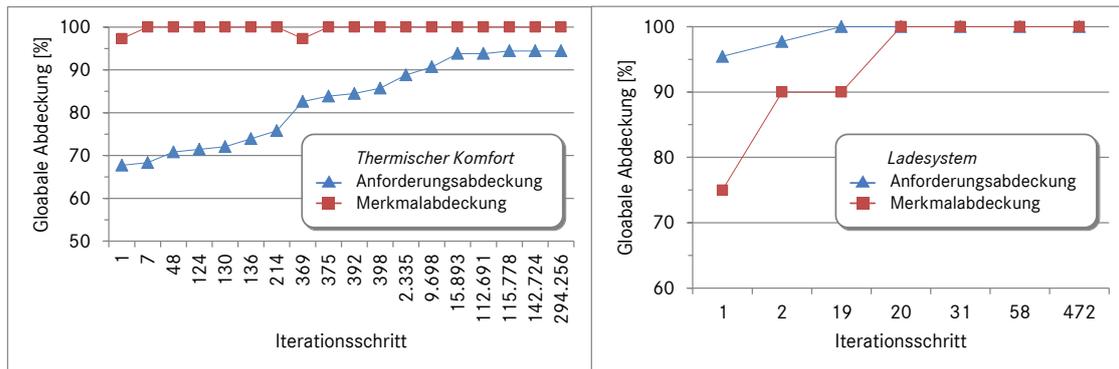


Abbildung 52: Von der Simulierten Abkühlung global erreichte Abdeckungen mit $T_{\text{init}} = 100$, $T_{\text{min}} = 0,001$, $\beta = 0,01$ sowie darüber hinaus $(G_{\text{TK}} - 3) \leq N_{\text{TK}} \leq (G_{\text{TK}} + 3)$, $(G_{\text{LS}} - 3) \leq N_{\text{LS}} \leq (G_{\text{LS}} + 3)$

Abdeckungen auf. Dies hängt damit zusammen, dass bei der Aktualisierung $|C_{\text{global}}|$ jeweils kleiner wird (vgl. Abbildung 51 rechts), bis das Optimum mit $|C_{\text{global}}| = G_{\text{LS}}$ erreicht ist.

Es besteht ein deutlicher Unterschied zwischen beiden Produktlinien im Bezug auf die globalen Energieverläufe. Einerseits wird für den *Thermischen Komfort* die globale Energie häufiger und andererseits nach einer größeren Anzahl von Iterationsschritten noch immer aktualisiert. Im Gegensatz dazu findet für das *Ladesystem* in dem gezeigten Fall nach dem 472sten Iterationsschritt keine Verbesserung von E_{global} mehr statt. Dabei werden für beide Systeme insgesamt gleich viele Iterationsschritte (700.000) durchgeführt. Die Häufigkeit, mit welcher eine Aktualisierung von E_{global} stattfindet, wird in Abbildung 53 näher untersucht. Darin wird analysiert, wie sich die genannte Größe in Abhängigkeit von verschiedenen Kühlraten β verhält. Über/in den Balken ist der jeweilige Mittelwert aus fünf Durchläufen mit der entsprechenden Standardabweichung aufgetragen.

Aus Abbildung 53 ist ersichtlich, dass β beim *Ladesystem* keine nennenswerte Rolle für die Aktualisierungshäufigkeit von E_{global} spielt, da nach vergleichsweise wenigen Iterationsschritten das gesuchte Optimum gefunden wird. Der eingeschränktere Suchraum, welcher im Gegensatz zu $\beta = 0,1$ bei $\beta = 0,01$ sowie $\beta = 0,001$ nur noch eine Suche in zwei Konfigurationssetgrößen oberhalb von G_{LS} erlaubt, scheint ebenfalls keine Auswirkungen auf die Häufigkeit der Lösungsverbesserung zu haben. Beim *Thermischen Komfort* ist ein leichter Anstieg der Häufigkeit bei der kleinsten Kühlrate feststellbar. Dies könnte damit zusammenhängen, dass einerseits bei einem niedrigeren β die Simulierte Abkühlung durch das häufigere Iterieren statistisch gesehen mehr Versuche hat die Lösung zu verbessern. Zudem werden niedrigere Temperaturwerte erst bei höheren Iterationsschritten erreicht und es ist anzunehmen, dass der Übergang in die lokale Suche erst zu einem späteren Zeitpunkt stattfindet und daher mehr Verbesserungen erreicht werden können.

Ebenso wird eine Analyse der mithilfe der Simulierten Abkühlung global erreichten Anforderungs- sowie Merkmalabdeckungen in Abhängigkeit von β in Abbildung 54

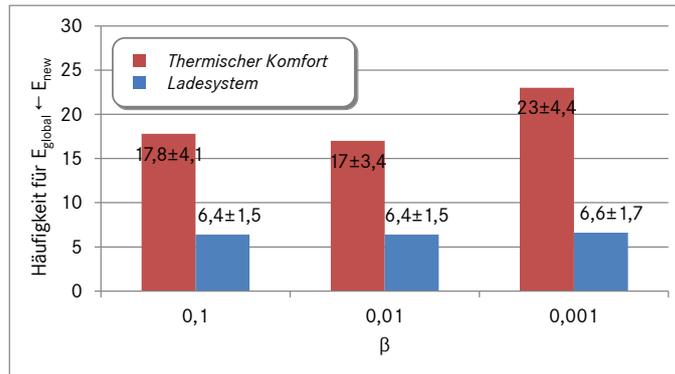


Abbildung 53: Häufigkeit, mit welcher die globale Energie erneuert wird, in Abhängigkeit von verschiedenen Kühlraten β mit $T_{\text{init}} = 100$, $T_{\text{min}} = 0,001$ (jeweils gemittelt über 5 Durchläufe). Bei

$$(G_{\text{TK}} - 1) \leq N_{\text{TK}} \leq (G_{\text{TK}} + 5), (G_{\text{LS}} - 2) \leq N_{\text{LS}} \leq (G_{\text{LS}} + 4) \text{ für } \beta = 0,1,$$

$$(G_{\text{TK}} - 2) \leq N_{\text{TK}} \leq (G_{\text{TK}} + 4), (G_{\text{LS}} - 2) \leq N_{\text{LS}} \leq (G_{\text{LS}} + 2) \text{ für } \beta = 0,01,$$

$$(G_{\text{TK}} - 3) \leq N_{\text{TK}} \leq (G_{\text{TK}} + 3), (G_{\text{LS}} - 2) \leq N_{\text{LS}} \leq (G_{\text{LS}} + 2) \text{ für } \beta = 0,001.$$

vorgenommen. Für das *Ladesystem* werden mithilfe der Heuristik vollständige Abdeckungen erreicht. Im Gegensatz dazu hängen die Abdeckungen beim *Thermischen Komfort* von der Kühlrate ab. Je häufiger iteriert wird, desto größer werden die Abdeckungen und somit besser die Lösung. Jedoch erreicht die Simulierte Abkühlung auch beim kleinsten β nicht die 100%ige Abdeckung der Anforderungen. Selbst nach rund 18 Stunden bei $\beta = 0,0001$ konnte keine vollständige Abdeckung hergestellt werden. Dies deutet darauf hin, dass aufgrund der zufälligen Suche die Heuristik zwar gute Ergebnisse bei generischen Anforderungsspezifikationen, wie beim *Ladesystem*, liefert. Liegen jedoch einige Anforderungen vor, welche nur von wenigen Konfigurationen abgedeckt werden können, wie dies für den *Thermischen Komfort* der Fall ist, hat das Verfahren Schwierigkeiten – selbst bei langer Rechenzeit – genau diese für ein Konfigurationsset zu selektieren.

Weitere wichtige Größen, anhand derer die Qualität der Simulierten Abkühlung evaluiert und mit der des Greedy-Algorithmus verglichen wird, ist die Anzahl der selektierten Konfigurationen und die Laufzeit. Die Ergebnisse sind für verschiedene Kühlraten in Abbildung 55 gezeigt. Die Anzahl der Konfigurationen innerhalb des global selektierten Sets kann für das *Ladesystem* mithilfe der Simulierten Abkühlung nicht weiter reduziert werden, da $G_{\text{LS}} = 5$ bereits das gesuchte Optimum darstellt. Bei allen Versuchen werden durch die Simulierte Abkühlung fünf Konfigurationen selektiert. Dies ist im linken Teil der Abbildung 55 dargestellt. Beim *Thermischen Komfort* verbessert sich die Anzahl der selektierten Konfigurationen für $\beta = 0,001$ nur unwesentlich. Auch ist der Wert höher und damit schlechter als der mithilfe des Greedy-Algorithmus ermittelte von $G_{\text{TK}} = 8$. Gleichzeitig wird dabei, wie schon zuvor diskutiert, keine vollständige Anforderungsabdeckung erreicht.

Die Laufzeit der Simulierten Abkühlung hängt stark von der Kühlrate ab, was im rechten Teil der Abbildung 55 verdeutlicht ist. Je häufiger iteriert wird, desto länger benötigt der Algorithmus bis T_{min} erreicht wird. Dies liegt bei beiden Systemen vor,

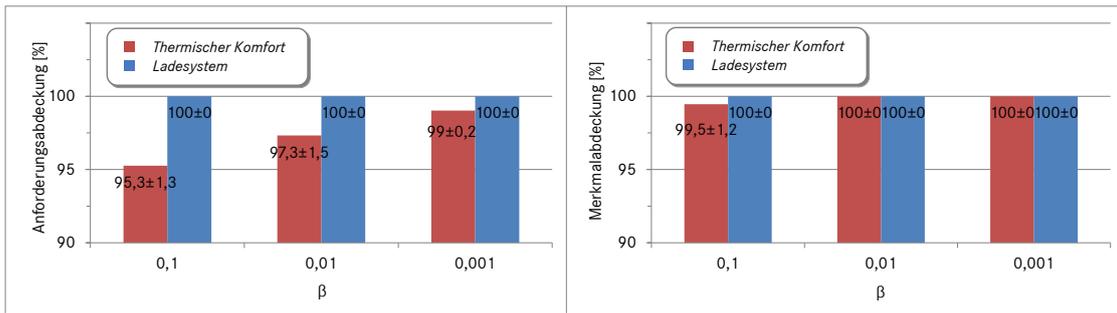


Abbildung 54: Von der Simulierten Abkühlung erreichte globale Abdeckungen in Abhängigkeit von verschiedenen Kühlraten β mit $T_{init} = 100$, $T_{min} = 0,001$ (jeweils gemittelt über 5 Durchläufe). Bei $(G_{TK} - 1) \leq N_{TK} \leq (G_{TK} + 5)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 4)$ für $\beta = 0,1$, $(G_{TK} - 2) \leq N_{TK} \leq (G_{TK} + 4)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 2)$ für $\beta = 0,01$, $(G_{TK} - 3) \leq N_{TK} \leq (G_{TK} + 3)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 2)$ für $\beta = 0,001$.

wobei die Suche für den *Thermischen Komfort* jeweils deutlich länger dauert. Hierfür ist der größere Umfang des Projektes verantwortlich.

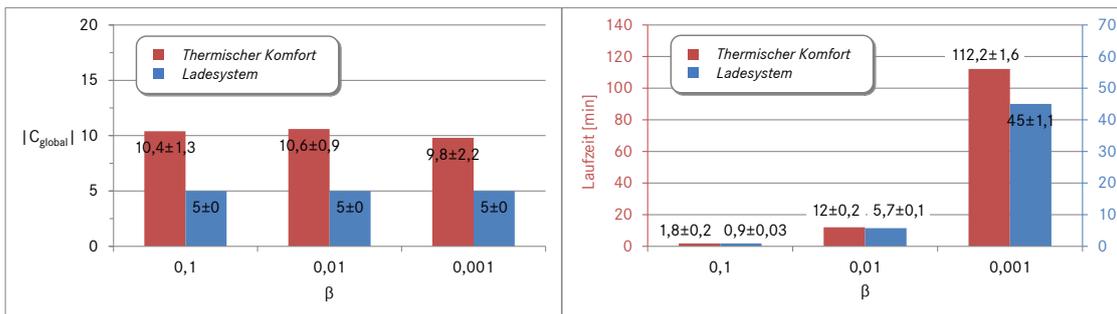


Abbildung 55: Größe des global selektierten Konfigurationssets sowie die Laufzeit der Simulierten Abkühlung in Abhängigkeit von verschiedenen Kühlraten β bei $T_{init} = 100$, $T_{min} = 0,001$ (jeweils gemittelt über 5 Durchläufe). Bei $(G_{TK} - 1) \leq N_{TK} \leq (G_{TK} + 5)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 4)$ für $\beta = 0,1$, $(G_{TK} - 2) \leq N_{TK} \leq (G_{TK} + 4)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 2)$ für $\beta = 0,01$, $(G_{TK} - 3) \leq N_{TK} \leq (G_{TK} + 3)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 2)$ für $\beta = 0,001$.

Die in Abschnitt 6.2.3.2 erläuterte Laufzeitkomplexität der Simulierten Abkühlung wird für verschiedene Größen empirisch untersucht. Dabei wird zum einen die Abhängigkeit der Rechenzeit von der Anzahl der Iterationsschritte in Abbildung 56 sowie von der Anzahl der zu befüllenden Merkmaleinträge in Abbildung 57 betrachtet.

Die Anzahl der Iterationsschritte wird einerseits mithilfe der Kühlrate β im linken Teil der Abbildung 56 und andererseits anhand der Anzahl der Konfigurationssetgrößen $x + y + 1$, in welchen nach dem Optimum gesucht wird, im rechten Teil der

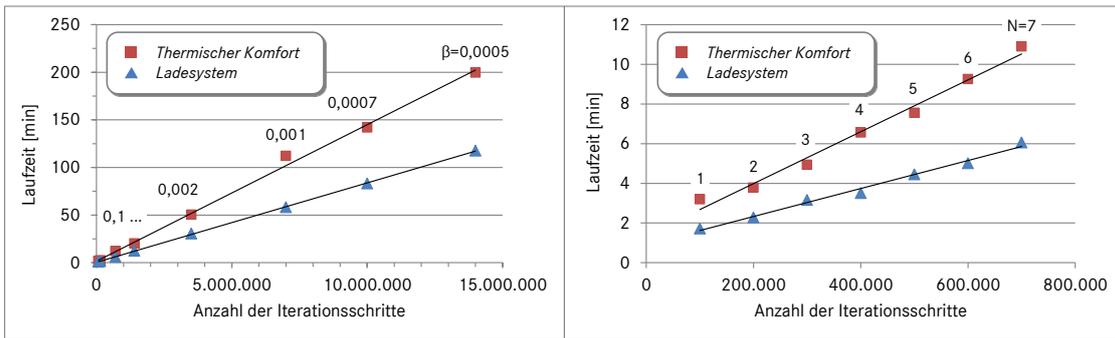


Abbildung 56: Abhängigkeit der Laufzeit der Simulierten Abkühlung von der Anzahl der Iterationsschritte variiert durch die Kühlrate β (links) sowie durch die Anzahl der Konfigurationssets (rechts) jeweils mit schwarzer Trendlinie (schwarz) für $T_{init} = 100$, $T_{min} = 0,001$, $N = 3$ (links) und $\beta = 0,001$ (rechts)

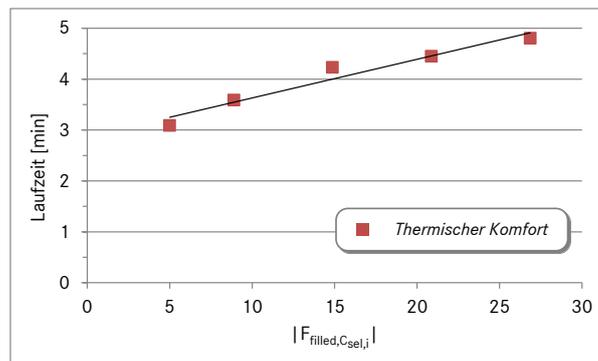


Abbildung 57: Abhängigkeit der Laufzeit der Simulierten Abkühlung von der Anzahl der unbestimmten Merkmaleinträge, welche im Rahmen des Feature Fillings befüllt werden, mit linearer Trendlinie (schwarz) für $T_{init} = 100$, $T_{min} = 0,001$, $\beta = 0,001$ und $N = 3$

Abbildung 56 variiert. Da der Kühlprozess sowie insbesondere die Anzahl der Iterationsschritte im Wesentlichen mithilfe der Kühlrate modifiziert werden können, bleiben T_{init} und T_{min} bei allen Experimenten konstant und es werden daher keine verschiedenen Temperaturparameter in die Betrachtung einbezogen. In beiden Graphiken wird für beide Produktlinien eine (erwartete) annähernd lineare Abhängigkeit festgestellt. Die jeweils größere Steigung für das System des *Thermischen Komforts* resultiert daher, dass im Mittel mehr Konfigurationen innerhalb der Sets vorliegen, da beim *Thermischen Komfort* um $G_{TK} = 8$, dagegen beim *Ladesystem* um $G_{LS} = 5$ gesucht wird. Gleichzeitig werden für den *Thermischen Komfort* pro Iteration und Konfiguration aufgrund des umfangreichen optionalen Merkmalclusters (vgl. Abschnitt 7.2) im Mittel mehr Merkmale befüllt $|F_{filled,C_{sel,i}}|$. Die Suche für $N = 0$ ist Tool-technisch nicht vorgesehen.

Abbildung 57 veranschaulicht den Zusammenhang zwischen dem Filling (vgl. Algorithmus 6) und der Laufzeit der Simulierten Abkühlung. Hierbei wird keine Analyse für

das *Ladesystem* vorgestellt, da in dem Projekt nicht genügend offene Merkmaleinträge vorliegen, um eine sinnvolle Datenreihe erstellen zu können. Für den *Thermischen Komfort* wird eine (erwartete) annähernd lineare Abhängigkeit der Rechenzeit ermittelt. Die Anzahl der befüllten Merkmale stellt jeweils einen Mittelwert aus den offenen Merkmaleinträgen *aller* Konfigurationen (vgl. Tabelle 11 der Basis) dar, da aufgrund der zufälligen Suche nicht bekannt ist, welche Konfigurationen selektiert und somit wie viele Merkmale genau befüllt wurden. Aufgrund von vorliegenden Constraints zwischen Merkmalen im Variabilitätsmodell kann die Anzahl der zu befüllenden Merkmale nicht weiter gesenkt werden.

7.4.3 Ausleitung von Testfällen

Die beiden in Abschnitt 6.3 diskutierten Strategien des vollständigen und inkrementellen Testens werden anhand der beiden Produktlinien untersucht. Dabei ist im Rahmen der vollständigen Testfallselektion die Ausleitung aller passenden Testfälle für die ermittelten Konfigurationen vorgesehen. Wird eine solche vollständige Testfallselektion für die mittels des Greedy-Algorithmus bestimmten Konfigurationen durchgeführt, so werden für den *Thermischen Komfort* im Mittel $124,6 \pm 14,5$ und für das *Ladesystem* im Mittel $122,2 \pm 9,6$ Testfälle pro Konfiguration identifiziert. Die Durchführung dieser Testfälle garantiert eine umfassende Testabdeckung jeder Konfiguration.

Dagegen sind bei der Strategie des inkrementellen Testens jeweils solche Testfälle relevant, welche durch keine vorherige(n) Konfiguration(en) berücksichtigt wurden. Bei der Anwendung der Methode ist festgestellt worden, dass bereits nach wenigen Konfigurationen eine vollständige, kumulierte Abdeckung der Testfälle vorliegt. Dies hätte zur Folge, dass für die restlichen Konfigurationen gar keine Testfälle vorliegen würden. An dieser Stelle sind weiterführende Konzepte notwendig, anhand derer erkennbar ist, wann ein Testfall tatsächlich redundant ist. Dies kann bspw. mithilfe von Regressionstestmethoden weiter untersucht werden. Die Strategie des inkrementellen Testens ist daher nicht ohne eingehendere Analyse auf die Projekte anwendbar.

Da der Fokus der vorliegenden Arbeit auf der Selektion von zu testenden Konfigurationen liegt, und lediglich das Ermöglichen einer Testfallselektion zu den in Abschnitt 1.2 definierten Zielen der Arbeit gehört, wird auf die Thematik nicht näher eingegangen.

7.5 Evaluierung und Zusammenfassung der Ergebnisse

Die bei der Konsistenzprüfung erzielten Fallstudienresultate sind in Tabelle 17 zusammengefasst. Innerhalb der Entwicklungsdokumente sind für die Produktlinie des *Thermischen Komforts* über 50% und für das *Ladesystem* sogar über 70% inkonsistente Verknüpfungen aufgedeckt worden. Die für die Inkonsistenzen am häufigsten verantwortliche Ursache liegt im falschen Merkmalmapping. Dabei fehlt das Mapping entweder gänzlich, wird nach Modifikationen der Entwicklungsartefakte nicht angepasst oder resultiert aus einem fehleranfälligen Mapping, welches von zahlreichen

Merkmalen mit gegebenenfalls ungünstig langen Bezeichnungen rührt. Weiterhin entstehen häufig Inkonsistenzen durch falsches Verlinken der Anforderungen und Testfälle. Diese sind auf große Projektumfänge mit komplexen $n : m$ -Verknüpfungen, welche es für den Autor der Spezifikationsdokumente schwer machen, einen Überblick zu behalten, zurückzuführen. Zudem ist die Weiterentwicklung und somit Veränderung sowie die anschließend fehlende Anpassung der Links als weitere Fehlerursache zu nennen.

	<i>Thermischer Komfort</i>		<i>Ladesystem</i>	
	vorher	nachher	vorher	nachher
Konsistente Links [%]	53,6	100	27,5	99,6
Widersprüchliche Links [%]	1,3	0	0,1	0
Unvollständige Links [%]	28,4	0	6,6	0,4
Übervollständige Links [%]	16,6	0	65,7	0
Laufzeit [s]	60,7±2,3	56,7±2	10±0,8	7,4±0,7

Tabelle 17: Zusammenfassung der Ergebnisse der Konsistenzprüfung vor und nach der Umsetzung der Korrekturvorschläge. Es sind Abweichungen von 100% aufgrund von Rundungsfehlern möglich. Im Bezug auf die Laufzeit sind Mittelwerte aus 10 Durchläufen angegeben.

Die bei der Konsistenzprüfung unterbreiteten Korrekturvorschläge sind als sinnvoll bewertet und sowohl für den *Thermischen Komfort* als auch das *Ladesystem* umgesetzt worden. Die Beseitigung von Inkonsistenzen ist einerseits wichtig, da ein falsches Merkmalmapping zur Durchführung von Testfällen in falschen Konfigurationen und letztendlich zu widersprüchlichen Testergebnissen sowie deren Fortpflanzung in weitere Entwicklungsphasen führen kann. Andererseits ist das korrekte Verlinken zwischen Artefakten für die Durchgängigkeit eines Entwicklungsprojektes erforderlich, um bspw. Rückschlüsse von gefundenen Fehlern auf die entsprechenden Anforderungen ziehen zu können. Demzufolge steigert die Behebung der mithilfe der Prüfung ermittelten Inkonsistenzen erheblich die Qualität der Spezifikationsdokumente und verhindert deren potentiell schwerwiegende Folgen im Verlauf des Entwicklungsprojektes. Des Weiteren bieten die widerspruchsfreien, produktübergreifenden Artefakte eine sinnvolle Basis zur Selektion einer repräsentativen Konfigurationsmenge für den Test. Hierbei unterstützt die Konsistenzprüfung insbesondere innerhalb der Testspezifikation dabei, ein Anwachsen der übertestständigen und damit überschüssigen Testfälle zu verhindern und das redundante Ausführen solcher Testumfänge zu vermeiden. Hiermit kann das erste in Abschnitt 1.2 gesetzte Ziel an die Arbeit, welches die Entwicklung eines konsistenten Variantenmanagements zwischen Spezifikationen vorsieht, als erfüllt betrachtet werden.

Die mithilfe der Selektionsmethodik ermittelten Ergebnisse sind in Tabelle 18 für den *Thermischen Komfort* sowie in Tabelle 19 für das *Ladesystem* zusammengefasst. Darin werden die vom Greedy-Algorithmus und die von Simulierter Abkühlung erreichten Resultate jeweils gegenübergestellt. In der letzten Spalte findet zudem ein Vergleich

mit den Ergebnissen der Pairwise-Methode statt. Letztere dient als Bezugspunkt für die in der vorliegenden Arbeit entwickelte Selektionsmethode.

	Greedy-Alg.	Simulierte Abkühlung			Pairwise
		$\beta=0,1$	$\beta=0,01$	$\beta=0,001$	
Selektierte Konfigurationen	8 ± 0	$10,4\pm 1,3$	$10,6\pm 0,9$	$9,8\pm 2,2$	32
Anforderungsabdeckung [%]	100 ± 0	$95,3\pm 1,3$	$97,3\pm 1,5$	$99\pm 0,2$	100
Merkmalabdeckung [%]	100 ± 0	$99,5\pm 1,2$	100 ± 0	100 ± 0	100
Laufzeit [s]	$30,8\pm 1$	108 ± 9	721 ± 9	6730 ± 97	1

Tabelle 18: Zusammenfassung der Selektionsergebnisse für die Produktlinie *Thermischer Komfort*. Die Ergebnisse des Greedy-Algorithmus stellen Mittelwerte aus 10 Durchläufen dar, während die der Simulierten Abkühlung jeweils Mittelwerten aus 5 Durchläufen entsprechen.

	Greedy-Alg.	Simulierte Abkühlung			Pairwise
		$\beta=0,1$	$\beta=0,01$	$\beta=0,001$	
Selektierte Konfigurationen	5 ± 0	5 ± 0	5 ± 0	5 ± 0	19
Anforderungsabdeckung [%]	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100
Merkmalabdeckung [%]	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100
Laufzeit [s]	$10,3\pm 0,6$	54 ± 2	344 ± 5	2701 ± 68	1

Tabelle 19: Zusammenfassung der Selektionsergebnisse für die Produktlinie *Ladesystem*. Die Ergebnisse des Greedy-Algorithmus stellen Mittelwerte aus 10 Durchläufen dar, während die der Simulierten Abkühlung jeweils Mittelwerten aus 5 Durchläufen entsprechen.

Obwohl der Greedy-Algorithmus im Gegensatz zur Simulierten Abkühlung lokale Optima nicht wieder verlassen kann, geht aus den Fallstudien dennoch klar hervor, dass die Selektion mithilfe des Greedy-Algorithmus bessere Ergebnisse liefert. Dies spiegelt sich für den *Thermischen Komfort* in der Qualität der Ergebnisse, aufgrund der erzielten vollständigen Anforderungs- sowie Merkmalabdeckungen und der kleineren Anzahl an selektierten Konfigurationen, wider. Darüber hinaus liegt die benötigte Rechenzeit für beide Projekte deutlich unter der der Simulierten Abkühlung, wobei letztere zusätzlich stark von der Parametrierung abhängt. Hierbei ist jedoch nicht auszuschließen, dass die Struktur beider Projekte für die bessere Performance des Greedy-Algorithmus verantwortlich ist und für ein Projekt mit bedeutend mehr Anforderungen das Optimierungsverfahren der Simulierten Abkühlung überlegen wäre. Da der Fokus der vorliegenden Arbeit darauf gelegt ist, die Anwendbarkeit der Methoden im realen Industrieumfeld zu demonstrieren, wird auf eine Untersuchung der Algorithmen im Rahmen von theoretischen Projekten verzichtet. Die Plausibilität der Ergebnisse ist jedoch anhand von kleineren manuell generierten Projekten geprüft und bestätigt

worden. Damit kann die zweite in Abschnitt 1.2 definierte Zielsetzung, welche eine effiziente Selektionsmethodik von zu testenden Konfigurationen anstrebt, als erfüllt bewertet werden.

Ein Vergleich mit dem Ergebnis der Pairwise-Methode zeigt, dass mithilfe des Greedy-Algorithmus eine etwa viermal kleinere Anzahl an Konfigurationen selektiert wird. Dies stellt ein effizienteres Ergebnis dar, da für jede weitere zu testende Konfiguration entsprechend mehr Zeit und Aufwand benötigt wird. Berücksichtigt man die Anzahl der Testfälle, welche pro Konfiguration im Schnitt durchgeführt werden müssen, so wären dies rund 1000 Testfälle für die mittels anforderungs- und merkmalsbasierter Selektion ermittelten sowie 4000 Testfälle für die mittels der Pairwise-Methode bestimmten Konfigurationen für das Projekt *Thermischer Komfort*. Beim *Ladesystem* wären 600 Testfälle im Falle der anforderungs- und merkmalsbasierten Selektionsmethode und 2300 im Falle der Pairwise-Methode notwendig. Durch die Anforderungs- und Merkmalabdeckung ist zudem sichergestellt, dass die Eigenschaften der gesamten Produktlinie berücksichtigt sind. Hierbei kann die *zusätzliche* Abdeckung aller Merkmalpaare insbesondere bei sicherheitsrelevanten Testobjekten dennoch sinnvoll sein. Die schnellere Rechenzeit der Pairwise-Methode ist darauf zurückzuführen, dass lediglich das Variabilitätsmodell in Betracht gezogen wird, die Anforderungen jedoch nicht berücksichtigt werden.

Der Einsatz *eines* universellen Variabilitätsmodell für die Anforderungen und Testfälle erlaubt es für die auf Basis einer optimierten Anforderungsabdeckung selektierten Konfigurationen passende Testumfänge abzuleiten. Hierdurch wird das dritte und letzte Ziel der Arbeit (vgl. Abschnitt 1.2), welches die schnelle Identifikation der Testfälle für jede mögliche aus dem entsprechenden Merkmalmodell ableitbare Konfiguration fordert, adressiert.

Zur abschließenden Evaluierung der in der vorliegenden Arbeit entwickelten Methoden eignen sich die einheitlich definierten Kriterien zur Bewertung von Produktlinientestmethoden aus Abschnitt 4.3. Hierfür wird Tabelle 5 durch die im Folgenden diskutierte Bewertung der vorliegenden Dissertation ergänzt und in Tabelle 20 vorgestellt.

Die Automatisierbarkeit der Konsistenzprüfung und der Selektionsmethodik ist aufgrund der prototypischen Unterstützung mithilfe des Plug-Ins innerhalb der Umgebung `pure::variants` als erfüllt zu bewerten (+). Da die innerhalb eines Variabilitätsmodells und des Merkmal mappings auf Artefakte realisierte Produktlinieninfrastruktur zum gesetzten Standard innerhalb des Konzerns gehören, ist der Zusatzaufwand bei der Bereitstellung der Basis als durchschnittlich (o) bis gering (+) zu beurteilen. Der aus der Methodik hervorgehende Nutzen ist im Rahmen von praktischen Fallstudien sowie im Vergleich zu weiteren Selektionsmethoden nachgewiesen (+) worden. Hierbei ist ein direkter Vergleich lediglich mit der Pairwise-Methode möglich. Dies resultiert zum einen aus den zur Evaluierung nicht vorliegenden Werkzeugen. Zum anderen ist aufgrund der Heterogenität der Methoden, welche teils andere Grundlagen oder andere auf die Projekte nicht direkt übertragbare Selektionskriterien verwenden, die Kompatibilität zur erforderlichen Produktlinieninfrastruktur nicht immer gegeben. Schließlich wird die Ausleitung von passenden Testfällen aufgrund der mithilfe der Konsistenzprüfung generierten widerspruchsfreien Basis ermöglicht (+). Der Fokus der Methoden liegt auf Systemebene. Das Konzept der Konsistenzprüfung und der Selektionsmethodik

	Manicke	Burgdorf	Oster	Cohen	Scheidemann	Cmyrev
Fokus (Testobjekttyp)	Funktion	System	alle	alle	System	System
Automatisierbarkeit (Werkzeug)	+	+	+	k.A.	k.A.	+
Geringer Aufwand (Produktlinieninfrastruktur)	o	-	o	+	o	o
Nutzen (Fallstudien)	o	k.A.	o	k.A.	o	+
Wiederverwendung (Testfälle)	+	-	+	-	-	+

Tabelle 20: Vergleich der verwandten Arbeiten (aus den Abschnitten 4.2.5, 4.2.6, 4.2.7) mit der in der vorliegenden Dissertation ausgearbeiteten Selektionsmethodik (siehe letzte Spalte).

Legende: (+) erfüllt das Kriterium, (o) erfüllt das Kriterium teils, (-) erfüllt das Kriterium nicht.

ist jedoch nicht ausschließlich auf Systeme beschränkt, sondern auf jedes Testobjekt erweiterbar, welchem die entsprechende Produktlinieninfrastruktur zugrunde liegt.

Als Fazit lassen sich die Konsistenzprüfung und die Selektionsmethodik als erfolgreich ausgearbeitete und umgesetzte Entwicklungen beurteilen. Dabei ist durch das Konsistenzmanagement eine qualitative Verbesserung der produktübergreifenden Spezifikationsdokumente und damit der Basis von Entwicklungsprojekten zu sehen. Zudem liefert die darauf aufbauende systematische Konfigurationsselektion eine effiziente Reduktion des Testaufwandes. Schließlich ermöglichen die als prototypische Werkzeuge umgesetzte Methoden deren sinnvolle Anwendung in der Praxis.

Teil IV

ABSCHLUSS

8

ZUSAMMENFASSUNG UND AUSBLICK

Im vorliegenden Kapitel erfolgt eine abschließende Zusammenfassung der ausgearbeiteten und umgesetzten Methoden. Diesbezüglich werden im ersten Abschnitt 8.1 die Hauptbeiträge der Arbeit herausgestellt sowie die Grenzen und Einschränkungen der dabei erzielten Ergebnisse kritisch diskutiert. Davon ausgehend wird die Arbeit mit einem Ausblick auf mögliche weiterführende Arbeiten in Abschnitt 8.2 abgeschlossen.

8.1 Fazit

Eine effiziente und effektive Methode beim Produktlinientest wird innerhalb des Entwicklungsprozesses von Fahrzeugen aufgrund des wachsenden Variantenreichtums sowie der damit einhergehende Komplexität der Automobile immer wichtiger. Eine Analyse zeigt, dass existierende Verfahren nicht allen an eine solche Testmethodik gestellten Anforderungen genügen (vgl. Abschnitt 4.3). Als zentrale Punkte sind fehlende empirische Untersuchungen und Evaluierungen im Rahmen von realen Industrieprojekten, hohe Aufwände bei der Bereitstellung der grundsätzlichen Voraussetzungen sowie stark eingeschränkte Randbedingungen, welche eine Verallgemeinerung der Ergebnisse verhindern, zu nennen. Zur Adressierung der identifizierten Potentialen ist einerseits die einheitliche Produktlinieninfrastruktur, deren Durchgängigkeit und Konsistenz zwischen den Entwicklungsphasen mithilfe der Konsistenzprüfung unterstützt wird, ausgearbeitet worden. Andererseits wurde auf Basis dieser einheitlichen Grundlage die Methodik zur Selektion von zu testenden Konfigurationen entwickelt. Schließlich ist die Anwendbarkeit beider Konzepte im Rahmen von zwei aktuellen Projekten der Daimler AG evaluiert und demonstriert worden.

Das der Konsistenzprüfung zugrunde liegende Konzept stellt den ersten großen Beitrag der vorliegenden Arbeit dar (vgl. Abschnitt 5.4.1). Hierfür ist basierend auf einem Variabilitätsmodell sowie einem einheitlichen Merkmalmapping ein systematischer Vergleich zwischen Artefakten definiert worden, welcher mögliche Inkonsistenzen zwischen Entwicklungsdokumenten aufdeckt. Darauf aufbauend sind Korrekturmaßnahmen abgeleitet worden, deren Umsetzung die Konsistenz herstellt. Das Vorgehen wurde auf Anforderungen und Testfälle angewandt. Das Konzept lässt sich darüber hinaus zwischen weiteren Entwicklungsphasen einsetzen und die Durchgängigkeit dadurch sicherzustellen.

Das zuvor analysierte produktübergreifende Anforderungsdokument sowie das zugehörige Merkmalmodell dienen als Basis für die Selektion von zu testenden Konfigurationen. Die Idee beruht darauf, eine möglichst kleine Menge an Konfigurationen zu bestimmen, welche eine Anforderungs- und Merkmalabdeckung gewährleistet. Zu diesem Zweck sind zwei existierende Heuristiken ausgewählt und an das problem-spezifische Umfeld, insbesondere im Hinblick auf die verwendete Assoziationslogik

beim Merkmalmapping, adaptiert worden. Das Konzept entspricht dem zweiten Hauptbeitrag der Arbeit (vgl. Abschnitt 6.2). Die Ergebnisse des Greedy-Algorithmus und der Simulierten Abkühlung sind im Rahmen der Fallstudien eingehend miteinander verglichen worden. Dabei wurde festgestellt, dass trotz des Nachteils lokale Optima nicht wieder verlassen zu können, das Suchvorgehen mittels Greedy-Algorithmus bessere Ergebnisse liefert.

Insgesamt ließ sich der Mehrwert der Methoden innerhalb von zwei Industrieprojekten nachweisen (vgl. Abschnitt 7.3 und 7.4). So sind mithilfe der Konsistenzprüfung je Projekt über 50% aller Verknüpfungen als inkonsistent bewertet und korrigiert worden. Die Behebung dieser Inkonsistenzen steigert die Qualität der Spezifikationsdokumente und gewährleistet demzufolge die Ausleitung von widerspruchsfreien produktspezifischen Artefakten. Im Bezug auf die Selektionsmethodik konnte jeweils eine systematische und effiziente Reduzierung des zu testenden Konfigurationsraumes (für den *Thermischen Komfort* von $\sim 10^6$ auf acht Konfigurationen und für das *Ladesystem* von $\sim 10^3$ auf fünf Konfigurationen) bei geforderten vollständigen Anforderungs- und Merkmalabdeckungen erreicht werden. Zudem ist die Ausleitung der für die Konfigurationen passenden Testfälle ermöglicht worden (vgl. Abschnitt 7.4.3).

Schließlich stellt die Umsetzung innerhalb eines Werkzeugs sowie die damit ermöglichte Automatisierung der Methoden den dritten großen Beitrag der Arbeit dar (vgl. Abschnitt 7.1). Die Integration in die Toolkette bei der Daimler AG erlaubt eine sinnvolle Anwendung der Konzepte in der entsprechenden Phase im Entwicklungsprozess.

Grenzen

Die vorgestellten Methoden weisen Grenzen auf bzw. unterliegen limitierenden Faktoren. Der wichtigste Faktor, welcher sich insbesondere auf die Ergebnisse der Selektionsmethode negativ auswirken kann, ist durch eine schlechte Qualität der Produktlinieninfrastruktur gekennzeichnet. Genügt ein Lastenheft nicht den Anforderungen einer qualitativ hochwertigen Spezifikation oder sind darin Merkmale den Artefakten falsch zugeordnet, so ist die Grundvoraussetzung für die Selektion nicht erfüllt. Dazu zählen ebenso Spezifikationen, welche unverhältnismäßig viele konfigurationsspezifische Anforderungen enthalten und dadurch keine sinnvolle Optimierung ermöglichen.

Des Weiteren legt die Struktur des Variantenmoduls sowie die des Merkmalmappings in konjunktiver Normalform Einschränkungen auf. Diese limitieren zwar nicht das Grundkonzept, jedoch sind bei deren Anwendung für abweichende Strukturen Anpassungen notwendig. Für die Konsistenzprüfung wird eine solche Adaptierung an ein allgemeingültiges Merkmalmodell mit einer beim Merkmalmapping disjunktiven Assoziationslogik (vgl. Abschnitt 3.3.2) in Abschnitt A vorgestellt. Für die Selektion wird keine solche Modifikation behandelt, da dies den Rahmen dieser Arbeit sprengen würde. Die Erweiterung zählt somit zu den möglichen, zukünftig zu betrachtenden Potentialen.

Schließlich sind die Methoden anhand von zwei Projekten evaluiert worden. Weitere Anwendungen waren aufgrund von fehlenden Voraussetzungen in der Produktlinieninfrastruktur nicht möglich, wären jedoch zwecks besserer statistischer Aussagekraft wünschenswert gewesen.

8.2 Ausblick

Ausgehend von den diskutierten Ergebnissen und den dabei identifizierten Grenzen sind weiterführende Arbeiten denkbar.

Einbindung der Pairwise-Methode in die anforderungs- und merkmalsbasierte Selektion

Aus empirischen Untersuchungen ist bekannt, dass die Pairwise-Methode hohe Quoten an Fehleraufdeckungen liefert. Gleichzeitig erreicht sie eine weniger effiziente Reduktion des kombinatorischen Testumfangs als die Selektionsmethodik der vorliegenden Arbeit. Eine Analyse bezüglich des Zusammenwirkens beider Methoden könnte somit einen wesentlichen Mehrwert generieren. Hierfür könnte bei der Optimierung der Anforderungsabdeckung der Freiheitsgrad offener Merkmale während des Fillings dazu genutzt werden, die Paarabdeckung der Merkmale zu maximieren. Das Ergebnis wäre ein repräsentativere Menge an Konfigurationen, welche die spezifizierten Eigenschaften der Produktlinie (mittels Anforderungsabdeckung) sowie deren Schnittstellen (mittels Paarabdeckung) effizient berücksichtigt. Durch die Absicherung der selektierten Konfigurationen ist eine verlässlichere Aussage bezüglich der Testabdeckung möglich.

Effiziente Testfallselektion

In Abschnitt 6.3 sind Möglichkeiten zur Bestimmung von Testfällen für die selektierten Konfigurationen diskutiert worden. Dabei wurde gezeigt, dass mithilfe des Inkrementellen Testens eine effizientere Testfallselektion erfolgen kann, als mithilfe des „vollständigen“ Testens. Derzeit ist in den präsentierten Fallstudien die inkrementelle Ausleitung von Testfällen aus der produktübergreifenden Testspezifikation ohne weitergehende Analyse nicht möglich. Mithilfe von Regressionstestmethoden kann jedoch eine solche Testfallbestimmung unterstützt und umgesetzt werden. Hierfür sind Untersuchungen weiterer Kriterien notwendig, anhand derer die Entscheidung getroffen werden kann, für welche Konfiguration(en) ein Testfall sinnvoll und für welche dieser redundant ist. Darüber hinaus ist eine umfassende Analyse von weiteren geeigneten Methoden zur Wiederverwendung von Testfällen, welche sich mit der anforderungs- und merkmalsbasierten Selektionsmethodik kombinieren ließen, sinnvoll.

Verbesserung des Werkzeuges

Das Werkzeug, welches zur Anwendung der ausgearbeiteten Methoden implementiert wurde, stellt eine prototypische Umsetzung dar. Um jedoch das Potential des Tools in vollem Umfang nutzen und die Anwendung benutzerfreundlicher gestalten zu können, sind Funktionserweiterungen denkbar.

So ist es beispielsweise sinnvoll, einem Nutzer die Möglichkeit zu geben, gewisse Merkmalkombinationen oder vollständige Konfigurationen vorzudefinieren, von denen ausgehend die optimierte Selektion beginnt. Dabei kann es sich um solche Konfigurationen handeln, welche basierend auf Erfahrungen des Testers die meisten potentiellen

Fehler aufdecken oder an welchen kritische Änderungen vorgenommen wurden, und die somit beim Test zwingend berücksichtigt werden sollten.

Bei der Konsistenzprüfung liegen die Potentiale des Tools in der Unterstützung bei den Korrekturmaßnahmen. Diese können zwar nicht voll-automatisiert durchgeführt werden, da der Inhalt der Artefakte verstanden werden muss. Jedoch lässt sich eine große Vereinfachung erreichen, wenn die Möglichkeit besteht, die Korrekturen per Knopfdruck zu übernehmen, sobald vom Anwender die entsprechende Stelle und konkrete Maßnahme identifiziert wurde. Darüber hinaus ist es denkbar, die Konsistenz bereits während des Spezifikationsprozesses zu prüfen und auf Inkongruenzen gegebenenfalls hinzuweisen. Dies stellt jedoch eine größere implementierungstechnische Herausforderung an die verwendete Tool-Landschaft und setzt zudem voraus, dass die Artefakte der untersuchten Dokumente bereits vor dem Merkmalmapping vollständig miteinander verlinkt sind.

Alternative Optimierungsverfahren

Die Fallstudien haben gezeigt, dass die vom Greedy-Algorithmus ermittelten Lösungen, sehr gute Ergebnisse repräsentieren. Bei Spezifikationsdokumenten mit wesentlich größeren Artefaktmengen ist es jedoch denkbar, dass das Suchverfahren schlechtere Resultate erzielt. Gleichzeitig war die Gewährleistung vollständiger Abdeckungen für die Simulierte Abkühlung aufgrund der zufälligen Suche schwierig. Demzufolge kann die Optimierung der Simulierten Abkühlung möglicherweise durch weitere Variationen des Mutationsoperators verbessert werden. Schließlich können andere Optimierungsalgorithmen, welche eine gerichtete Suche ermöglichen bzw. solche, bei welchen die 100%igen Abdeckungen als hartes Constraint definiert wird, geprüft werden.

Teil V

ANHANG

A KONSISTENZMANAGEMENT

A.1 Disjunktive Assoziationssemantik

Die einem Artefakt zugeordneten Merkmale können alternativ zur konjunktiven Normalform mithilfe einer nicht-ausschließenden Disjunktion, d.h. einem ODER-Operator miteinander verknüpft werden. Folglich ist ein Artefakt immer dann für eine Konfiguration gültig, sobald ein dem Artefakt zugeordnetes Merkmal in dieser Konfiguration enthalten ist. Um alle möglichen Konfigurationen mithilfe der disjunktiven Verknüpfung zwischen den Merkmalen auf die Artefakte abbilden zu können, müssen Anpassungen innerhalb des Merkmalmodells vorgenommen werden (nicht im Merkmalmapping). So müsste ein negiertes Merkmal $\neg f_y$ im Variabilitätsmodell ergänzt werden, um ausdrücken zu können, dass das entsprechende Merkmal f_y niemals für ein Artefakt gültig ist.

Das beschriebene Variantenmanagement bietet viele Vorteile. Zum einen entfällt die Überprüfung der Konsistenz zweiter Kategorie (vgl. Abschnitt 5.4), da die im Variabilitätsmodell definierten Abhängigkeiten sowie Constraint-Beziehungen nicht verletzt und somit beim Merkmalmapping vernachlässigt werden können. Demzufolge wird die Komplexität aus den Spezifikationsdokumenten in das Variabilitätsmodell verlagert. Ein weiterer Mehrwert liegt in dem intuitiveren Merkmalmapping für den Nutzer aufgrund der einheitlich disjunktiv miteinander verknüpften Merkmale. Schließlich ist das Konzept allgemeingültig bei den Merkmalmodellen einsetzbar.

Zum Zeitpunkt der Erstellung der vorliegenden Arbeit wird innerhalb der Anforderungs- und Testspezifikationen das Variantenmanagement mit einem Merkmalmapping in der konjunktiven Normalform eingesetzt. Für die Zukunft wird ein Merkmalmapping mit disjunktiver Assoziationslogik erwogen.

A.2 Konsistenzprüfung

Da beim Mapping mithilfe einer disjunktiven Assoziationslogik die im Variabilitätsmodell definierten Abhängigkeiten sowie Constraint-Beziehungen zwischen Merkmalen nicht relevant sind, müssen bei der Konsistenzprüfung keine Konfigurationsausdrücke c_x gebildet und verglichen werden. Das in Abschnitt 5.4.1 ausgearbeitete Konzept vereinfacht sich somit vom Vergleich der Konfigurationsmengen C auf einen Vergleich der Merkmalismengen F . Ebenso können die Differenzmengen direkt im Bezug auf die fehlenden bzw. überschüssigen Merkmale ermittelt werden. Dies ist in Tabelle 21 zusammengefasst und wird in [24] ausführlich behandelt.

Klasse	Formalismus	Verlinkte Artefakte	Kategorie	Differenz der Merkmalmengen
1	$F(r_i) = F(t_j)$	$r_i \leftrightarrow t_j$	konsistent	-
2	$F(r_i) \supset F(t_j)$:			
2.1	$F(r_i) = F(T_{r_i})$	$\forall t \in T_{r_i} [r_i \leftrightarrow t]$	konsistent	-
2.2	$F(r_i) \subset F(T_{r_i})$	$r_i \leftrightarrow t_j$	konsistent	-
2.3	$F(r_i) \supset F(T_{r_i})$	$\forall t \in T_{r_i} [r_i \leftrightarrow t]$	unvollständig	$F_{def}(T_{r_i}) = F(r_i) \setminus F(T_{r_i})$
2.4	$[F(r_i) \not\subseteq F(T_{r_i})] \wedge [F(r_i) \not\subseteq F(T_{r_i})]$	$r_i \leftrightarrow t_j$	unvollständig	$F_{def}(T_{r_i}) = F(r_i) \setminus F(T_{r_i})$
3	$F(r_i) \subset F(t_j)$:			
3.1	$F(R_{t_j}) = F(t_j)$	$\forall r \in R_{t_j} [r \leftrightarrow t_j]$	konsistent	-
3.2	$F(R_{t_j}) \supset F(t_j)$	$r_i \leftrightarrow t_j$	konsistent	-
3.3	$F(R_{t_j}) \subset F(t_j)$	$\forall r \in R_{t_j} [r \leftrightarrow t_j]$	überevollständig	$F_{exc}(t_j) = F(t_j) \setminus F(R_{t_j})$
3.4	$[F(R_{t_j}) \not\subseteq F(t_j)] \wedge [F(R_{t_j}) \not\subseteq F(t_j)]$	$r_i \leftrightarrow t_j$	überevollständig	$F_{exc}(t_j) = F(t_j) \setminus F(R_{t_j})$
4	$[F(r_i) \not\subseteq F(t_j)] \wedge [F(r_i) \not\subseteq F(t_j)]$:			
4.1	$F(r_i) \cap F(t_j) = \emptyset$	$r_i \leftrightarrow t_j$	widersprüchlich	-
4.2	$F(r_i) \cap F(t_j) \neq \emptyset$:			
4.2.1	$[F(r_i) \subset F(T_{r_i})] \wedge [F(R_{t_j}) \supset F(t_j)]$	$r_i \leftrightarrow t_j$	konsistent	-
4.2.2	$[F(r_i) \not\subseteq F(T_{r_i})] \wedge [F(R_{t_j}) \supset F(t_j)]$	$r_i \leftrightarrow t_j$	unvollständig	$F_{def}(T_{r_i}) = F(r_i) \setminus F(T_{r_i})$
4.2.3	$[F(r_i) \subset F(T_{r_i})] \wedge [F(R_{t_j}) \not\subseteq F(t_j)]$	$r_i \leftrightarrow t_j$	überevollständig	$F_{exc}(t_j) = F(t_j) \setminus F(R_{t_j})$
4.2.4	$[F(r_i) \not\subseteq F(T_{r_i})] \wedge [F(R_{t_j}) \not\subseteq F(t_j)]$	$r_i \leftrightarrow t_j$	unvollständig und überevollständig	$F_{def}(T_{r_i}) = F(r_i) \setminus F(T_{r_i}) \wedge$ $F_{exc}(t_j) = F(t_j) \setminus F(R_{t_j})$

Tabelle 21: Konzept der Konsistenzprüfung über den Vergleich der Merkmalmengen F zwischen verlinkten Anforderungen r und Testfällen t bei einer disjunktiven Assoziationslogik

Da das disjunktive Merkmalmapping allgemeingültiger ist, soll es bei Merkmalmodellen, welche über die in Abschnitt 5.2 beschriebene Struktur eines Variantenmoduls hinausgehen, eingesetzt werden. Dies hat zur Folge, dass ein Merkmalmodell nicht mehr von einer zu berücksichtigenden Merkmalebene (Merkmalausprägungen), sondern aus gegebenenfalls mehreren besteht. Werden Merkmale aus verschiedenen Ebenen abgebildet, muss sichergestellt werden, dass dies bei der Konsistenzprüfung berücksichtigt ist. Hierfür wird das Merkmalmapping vor der eigentlichen Prüfung innerhalb einer Basis (analog zu 5.4.1.1) vereinheitlicht. Alle Fälle, welche einer solchen Normierung bedürfen, sind im Aktivitätsdiagramm 58 dargestellt. Darin gilt $a \in A$ wobei $A \stackrel{\text{def}}{=} R \cup T$, sodass a als ein Artefakt aus Anforderungen bzw. Testfällen zu verstehen ist.

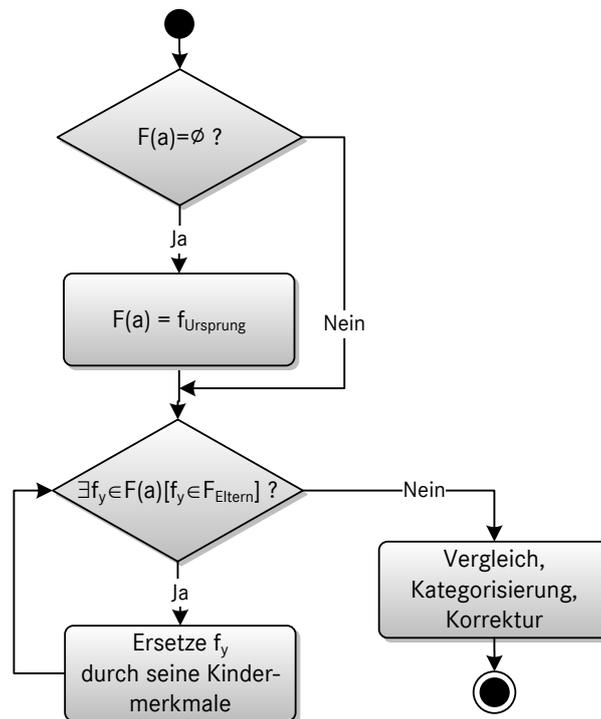


Abbildung 58: Vereinheitlichung der Merkmalmengen vor der Durchführung der Konsistenzprüfung

Für die Vereinheitlichung wird jedes zugeordnete Merkmal daraufhin geprüft, ob es zu der Menge der Elternmerkmale F_{Eltern} gehört und daher Kindermerkmale besitzt. Ist dies der Fall, wird das entsprechende Merkmal durch alle seine Kindermerkmale ersetzt. Dies wird solange fortgesetzt, bis keine zugeordneten Merkmale mehr Kindermerkmale besitzen. Dadurch wird erreicht, dass die Merkmale aus der untersten Ebene im Merkmalmodell miteinander verglichen werden. Da ein leeres Merkmalmapping ein generisches Artefakt kennzeichnet, welches für alle Konfigurationen gültig ist, wird dessen Merkmalmapping mit dem Ursprungsmerkmal f_{Ursprung} gleichgesetzt. Für den Vergleich wird ausgehend von f_{Ursprung} ebenfalls die Ersetzung durch die Kindermerkmale vorgenommen.

Der Vergleich, die Kategorisierung sowie die anschließende Ermittlung möglicher Korrekturmaßnahmen geschieht auf Basis der Merkmalmengen und wird analog zu der in Abschnitt 5.4.1 beschriebenen Konsistenzprüfung durchgeführt.

B ALGORITHMEN ABKÜRZUNGSVERZEICHNIS

Abkürzung	Beschreibung
$C_{cov}(r_{min})$	Menge der Konfigurationen, welche r_{min} abdecken (besitzt die Kardinalität 1)
C_{fmax}	Menge der Konfigurationen, welche die meisten Merkmale abdecken und in der Menge C_{max} enthalten sind
C_{global}	Menge der selektierten Konfigurationen über die Konfigurationssets hinweg als global beste Lösung
c_{in}	eine aus C_{unsel} zufällig gewählte Konfiguration
C_{max}	Menge der Konfigurationen, welche die meisten Anforderungen abdecken
C_{new}	neu generierte Lösung durch Vertauschen der Konfigurationen c_{in} und c_{out} innerhalb einer festen Konfigurationssetgröße
$cov(c_{x,filled}, f_y)$	Abdeckung des Merkmals f_y durch $c_{x,filled}$
$cov(c_{x,filled}, \neg f_y)$	Abdeckung des Merkmals $\neg f_y$ durch $c_{x,filled}$
c_{out}	eine aus C_{sel} zufällig gewählte Konfiguration
c_s	eine selektierte Konfiguration enthalten in der Menge C_{sel}
C_{sel}	Menge der selektierten Konfigurationen
$C_{sel,filled}$	Menge der selektierten Konfigurationen ohne offene Merkmaleinträge
$c_{s,filled}$	eine selektierte Konfiguration ohne offene Merkmaleinträge enthalten in der Menge $C_{sel,filled}$
C_{unsel}	Menge der nicht selektierten Konfigurationen
E_{global}	beste, d.h. kleinste, globale Energie aus allen Iterationsschritten (über die Konfigurationssets hinweg)
E_{init}	initiale Energie vor Ausführung der Simulierten Abkühlung $E_{init} = 1$
E_{local}	Energie der aktuellen Lösung C_{local} innerhalb einer festen Konfigurationssetgröße
E_{new}	Energie der neuen Lösung C_{new} für eine feste Konfigurationssetgröße

Abkürzung	Beschreibung
F_{all}	Menge aller im Variantenmodul modellierten Merkmalausprägungen
$F_{cov}(c_{s,filled})$	Menge der durch $c_{s,filled}$ abgedeckten Merkmale
$F_{filled,C_{sel,i}}$	Menge aller in der selektierten Konfiguration $C_{sel,i}$ mittels Filling befüllten Merkmale
$F_{uncovered}$	Menge der noch nicht abgedeckten Merkmale
f_y	Merkmal(-ausprägung) aus der Menge F_{all}
$\neg f_y$	negierte Merkmal(-ausprägung)
$ GreedySolution = G$	Größe des Konfigurationsets der vom Greedy-Algorithmus ermittelten Lösung
N	fixe Größe eines Konfigurationsets
R_{all}	Menge aller in der generischen Anforderungsspezifikation beschriebenen Anforderungen
R_{min}	Menge der Anforderungen, welche von exakt einer Konfiguration abgedeckt werden
r_{min}	Anforderung, welche von exakt einer Konfiguration abgedeckt wird und in der Menge R_{min} enthalten ist
$R_{uncovered}$	Menge der noch nicht abgedeckten Anforderungen
$sumCov(f_y)$	Anzahl, wie oft das Merkmal f_y durch bereits selektierte Konfigurationen abgedeckt ist
$sumCov(\neg f_y)$	Anzahl, wie oft das Merkmal $\neg f_y$ durch bereits selektierte Konfigurationen abgedeckt ist
w_c	Gewichtungsfaktor des Terms, welcher das Verhältnis der selektierten zu allen Konfigurationen innerhalb der Energiefunktion beschreibt
w_r	Gewichtungsfaktor des Terms, welcher die prozentual noch abzudeckenden Anforderungen innerhalb der Energiefunktion beschreibt
w_f	Gewichtungsfaktor des Terms, welcher die prozentual noch abzudeckenden Merkmale innerhalb der Energiefunktion beschreibt

ABBILDUNGSVERZEICHNIS

Abbildung 1	Überblick über den Aufbau der Arbeit	8
Abbildung 2	Schematische Darstellung eines eingebetteten Systems nach [12]	11
Abbildung 3	Übersicht über den sequenziellen Entwicklungsprozess nach dem V-Modell im Automobilssektor	13
Abbildung 4	Übersicht über die aufeinanderfolgenden Phasen des Testprozesses sowie die vorhergehende Spezifikationsphase von Anforderungen nach [107]	19
Abbildung 5	Übersicht über die Prozessschritte der Testkonzeption nach [81]	20
Abbildung 6	Übersicht über die Prozessschritte der Testspezifikation nach [79]	21
Abbildung 7	Qualitative Gegenüberstellung von Entwicklungskosten (links) sowie Time-to-Market (rechts) mit und ohne Produktlinienentwicklung nach [4]	24
Abbildung 8	Übersicht über den Referenzprozess für die Produktlinienentwicklung von Software [90]	26
Abbildung 9	Merkmalmodell für das beispielhafte System <i>Klimatisierung</i> . .	30
Abbildung 20	Orthogonales Variabilitätsmodell für das beispielhafte System <i>Klimatisierung</i> aus Abbildung 9	31
Abbildung 11	Abstrahierte Darstellung einer beispielhaften <i>Aufteilung nach Verantwortlichkeiten</i>	37
Abbildung 12	Abstrahierte Darstellung des <i>Product by Product Testens</i>	38
Abbildung 13	Abstrahierte Darstellung des <i>Inkrementellen Testens</i>	40
Abbildung 14	Abstrahierte Darstellung des <i>Modellbasierten Produktlinientests</i> .	42
Abbildung 15	Abstrahierte Übersicht über die <i>Methode der Priorisierung</i> von Manicke	44
Abbildung 16	Abstrahierte Übersicht über die <i>Methode der Priorisierung</i> von Burgdorf	46
Abbildung 17	Abstrahierte Übersicht über das <i>Kombinatorische Testvorgehen</i> von Oster	48
Abbildung 18	Abstrahierte Übersicht über das <i>kombinatorische Testvorgehen</i> von Cohen	50
Abbildung 19	Abstrahierte Übersicht über die <i>Anforderungsbasierte Selektion</i> von Scheidemann	51
Abbildung 20	Zusammenhang zwischen dem Variabilitätsmodell (VM), Anforderungen (R_i) und Testfällen (T_j) sowie eine Übersicht über die jeweils diskutierten Elemente der Produktlinieninfrastruktur innerhalb der entsprechenden Abschnitte des vorliegenden Kapitels	57

Abbildung 21	Merkmalmodell mit der zugrundeliegenden Struktur eines Variantenmoduls für das beispielhafte System <i>Klimatisierung</i>	59
Abbildung 22	Überblick über die aufeinanderfolgenden Schritte der Konsistenzprüfung unter Angabe der Abschnitte, in denen entsprechende Elemente diskutiert werden	65
Abbildung 23	Beispiel für eine konsistente Verlinkung (grün hervorgehoben) der Klasse 1 (vgl. Tabelle 10). Weitere (grau angedeutete) Verlinkungen sind für die Analyse und Kategorisierung von $r_1 \leftrightarrow t_1$ irrelevant.	70
Abbildung 24	Beispiel für konsistente Verlinkungen (grün hervorgehoben) der Klasse 2.1 (vgl. Tabelle 10). Weitere (grau angedeutete) Verlinkungen sind für die Analyse und Kategorisierung von $r_2 \leftrightarrow T_{r_2}$, mit $T_{r_2} = \{t_2, t_3\}$ irrelevant.	70
Abbildung 25	Beispiel für eine konsistente Verlinkung (grün hervorgehoben) der Klasse 3.2 (vgl. Tabelle 10). Schwarz markierte Verlinkung wird berücksichtigt, während weitere (grau angedeutete) Verknüpfungen bei der Analyse und Kategorisierung von $r_3 \leftrightarrow t_4$ irrelevant sind.	71
Abbildung 26	Beispiel für eine konsistente Verlinkung (grün hervorgehoben) der Klasse 4.2.1 (vgl. Tabelle 10). Schwarz markierte Verlinkungen werden berücksichtigt, während weitere (grau angedeutete) Verknüpfungen bei der Analyse und Kategorisierung von $r_5 \leftrightarrow t_5$ irrelevant sind.	72
Abbildung 27	Beispiel für eine widersprüchliche Verlinkung (rot hervorgehoben) der Klasse 4.1 (vgl. Tabelle 10). Weitere (grau angedeutete) Verlinkungen sind bei der Analyse und Kategorisierung von $r_7 \leftrightarrow t_7$ irrelevant.	73
Abbildung 28	Beispiel für unvollständige Verlinkungen (rot hervorgehoben) der Klasse 2.3 (vgl. Tabelle 10). Weitere (grau angedeutete) Verlinkungen sind bei der Analyse und Kategorisierung von $r_8 \leftrightarrow C(T_{r_8})$ mit $T_{r_8} = \{t_8, t_9\}$ irrelevant.	74
Abbildung 29	Beispiel für eine unvollständige Verlinkung (rot hervorgehoben) der Klasse 4.2.2 (vgl. Tabelle 10). Schwarz markierte Verlinkungen werden bei der Analyse und Kategorisierung von $r_9 \leftrightarrow t_{10}$ berücksichtigt. Auf die Darstellung von möglichen Verlinkungen zwischen r_{10} , r_{11} und weiteren Testfällen sowie zwischen t_{11} und weiteren Anforderungen wird zwecks Übersichtlichkeit verzichtet. Diese wären für die Analyse des diskutierten Falls irrelevant.	75
Abbildung 30	Beispiel für eine übervollständige Verlinkung (rot hervorgehoben) der Klasse 3.4 (vgl. Tabelle 10). Schwarz markierte Verlinkungen werden berücksichtigt, während weitere (grau angedeutete) Verknüpfungen bei der Analyse und Kategorisierung von $r_{12} \leftrightarrow t_{12}$ irrelevant sind.	76

Abbildung 31	Beispiel für eine unvollständige und gleichzeitig übervollständige Verlinkung (rot hervorgehoben) der Klasse 4.2.4 (vgl. Tabelle 10). Schwarz markierte Verlinkungen werden berücksichtigt, während weitere (grau angedeutete) Verknüpfungen bei der Analyse und Kategorisierung von $r_{15} \leftrightarrow t_{13}$ irrelevant sind.	77
Abbildung 32	Überblick über mögliche Inkonsistenzen (identifizierbar mithilfe der im vorhergehenden Abschnitt 5.4.1.2 diskutierten Prüfmethodik), der Kritikalitätsgrad sowie die Zuordnung zu den in Abbildung 33 dargestellten Fällen (a)–(e).	78
Abbildung 33	Verhältnisse der Konfigurationsräume zwischen Anforderungen und Testfällen (ausgehend von Tabelle 10) sowie die damit einhergehenden und zur Behebung von Inkonsistenzen erforderlichen Differenzmengen C_{def} , C_{exc}	80
Abbildung 34	Übersicht über die Produktlinieninfrastruktur sowie die darauf aufbauende Selektionsmethodik mit den jeweils diskutierten Elementen innerhalb der entsprechenden Abschnitte des vorliegenden Kapitels	85
Abbildung 35	Überblick über die aufeinanderfolgenden Schritte der Selektionsmethodik unter Angabe der Abschnitte, in denen entsprechende Elemente diskutiert werden	87
Abbildung 36	Abstrahiertes Vorgehen des Greedy-Algorithmus zur optimierten Anforderungs- und Merkmalabdeckung	93
Abbildung 37	Entscheidungsprinzip der Simulierten Abkühlung für die Akzeptanz einer Lösung	98
Abbildung 38	Überblick über das Vorgehen der parallel durchgeführten Simulierten Abkühlungen zur optimierten Anforderungs- und Merkmalabdeckung. Die Lösungen mit festen Konfigurationssetgrößen (links) werden durch das Austauschen der Konfigurationen aus den selektierten mit denen aus den nicht selektierten Konfigurationsmengen (rechts) ermittelt.	100
Abbildung 39	Entscheidungsprinzip der Simulierten Abkühlung für die lokal sowie global beste Lösung zur optimierten Anforderungs- und Merkmalabdeckung	103
Abbildung 40	Strategien zur Bestimmung von Testfällen (aus der generischen Testspezifikation der Produktlinie) für die selektierten Konfigurationen	107
Abbildung 41	Ergebnis der Konsistenzprüfung für das System <i>Thermischer Komfort</i> mit der Verteilung der analysierten Verlinkungen auf die entsprechenden Kategorien vor und nach der Umsetzung von Korrekturen	112
Abbildung 42	Ergebnis der Konsistenzprüfung für das System <i>Thermischer Komfort</i> mit der Verteilung der analysierten Verlinkungen auf die entsprechenden Kategorien-Klassen vor (oben) und nach (unten) der Umsetzung von Korrekturen	113
Abbildung 43	Ergebnis der Konsistenzprüfung für das <i>Ladesystem</i> mit der Verteilung der analysierten Verlinkungen auf die entsprechenden Kategorien vor und nach der Umsetzung von Korrekturen	115

Abbildung 44	Ergebnis der Konsistenzprüfung für das <i>Ladesystem</i> mit der Verteilung der analysierten Verlinkungen auf die entsprechenden Kategorien-Klassen vor (oben) und nach (unten) der Umsetzung von Korrekturen	116
Abbildung 45	Abhängigkeit der Laufzeit der Konsistenzprüfung von der Anzahl der zu analysierenden Verlinkungen zwischen Anforderungen und Testfällen sowie von der Anzahl der Konfigurationen jeweils mit linearer Trendlinie (schwarz)	117
Abbildung 46	Ergebnis des Greedy-Algorithmus für das System <i>Thermischer Komfort</i>	118
Abbildung 47	Ergebnis des Greedy-Algorithmus für das <i>Ladesystem</i>	119
Abbildung 48	Abhängigkeit der Laufzeit des Greedy-Algorithmus von der Anzahl der Konfigurationen (links) sowie der Anzahl der Anforderungen (rechts) jeweils mit linearer Trendlinie (schwarz) .	121
Abbildung 49	Verlauf der lokalen Energie sowie der Temperatur bei der Optimierung mittels Simulierter Abkühlung mit $T_{init} = 100$, $T_{min} = 0,001$, $\beta = 0,01$ und $N_{TK} = G_{TK} = 8$, $N_{LS} = G_{LS} = 5$	121
Abbildung 50	Von der Simulierten Abkühlung lokal erreichte Abdeckungen mit $T_{init} = 100$, $T_{min} = 0,001$, $\beta = 0,01$, $N_{TK} = G_{TK} = 8$ und $N_{LS} = G_{LS} = 5$	122
Abbildung 51	Von der Simulierten Abkühlung global selektierte Konfigurationen sowie der Verlauf der globalen Energie mit $T_{init} = 100$, $T_{min} = 0,001$, $\beta = 0,01$, $(G_{TK} - 3) \leq N_{TK} \leq (G_{TK} + 3)$ und $(G_{LS} - 3) \leq N_{LS} \leq (G_{LS} + 3)$	123
Abbildung 52	Von der Simulierten Abkühlung global erreichte Abdeckungen mit $T_{init} = 100$, $T_{min} = 0,001$, $\beta = 0,01$ sowie darüber hinaus $(G_{TK} - 3) \leq N_{TK} \leq (G_{TK} + 3)$, $(G_{LS} - 3) \leq N_{LS} \leq (G_{LS} + 3)$.	124
Abbildung 53	Häufigkeit, mit welcher die globale Energie erneuert wird, in Abhängigkeit von verschiedenen Kühlraten β mit $T_{init} = 100$, $T_{min} = 0,001$ (jeweils gemittelt über 5 Durchläufe). Bei $(G_{TK} - 1) \leq N_{TK} \leq (G_{TK} + 5)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 4)$ für $\beta = 0,1$, $(G_{TK} - 2) \leq N_{TK} \leq (G_{TK} + 4)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 2)$ für $\beta = 0,01$, $(G_{TK} - 3) \leq N_{TK} \leq (G_{TK} + 3)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 2)$ für $\beta = 0,001$	125
Abbildung 54	Von der Simulierten Abkühlung erreichte globale Abdeckungen in Abhängigkeit von verschiedenen Kühlraten β mit $T_{init} = 100$, $T_{min} = 0,001$ (jeweils gemittelt über 5 Durchläufe). Bei $(G_{TK} - 1) \leq N_{TK} \leq (G_{TK} + 5)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 4)$ für $\beta = 0,1$, $(G_{TK} - 2) \leq N_{TK} \leq (G_{TK} + 4)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 2)$ für $\beta = 0,01$, $(G_{TK} - 3) \leq N_{TK} \leq (G_{TK} + 3)$, $(G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 2)$ für $\beta = 0,001$	126

Abbildung 55	<p>Größe des global selektierten Konfigurationssets sowie die Laufzeit der Simulierten Abkühlung in Abhängigkeit von verschiedenen Kühlraten β bei $T_{init} = 100, T_{min} = 0,001$ (jeweils gemittelt über 5 Durchläufe).</p> <p>Bei $(G_{TK} - 1) \leq N_{TK} \leq (G_{TK} + 5), (G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 4)$ für $\beta = 0,1,$</p> <p>$(G_{TK} - 2) \leq N_{TK} \leq (G_{TK} + 4), (G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 2)$ für $\beta = 0,01,$</p> <p>$(G_{TK} - 3) \leq N_{TK} \leq (G_{TK} + 3), (G_{LS} - 2) \leq N_{LS} \leq (G_{LS} + 2)$ für $\beta = 0,001.$ 126</p>
Abbildung 56	<p>Abhängigkeit der Laufzeit der Simulierten Abkühlung von der Anzahl der Iterationsschritte variiert durch die Kühlrate β (links) sowie durch die Anzahl der Konfigurationssets (rechts) jeweils mit schwarzer Trendlinie (schwarz) für $T_{init} = 100, T_{min} = 0,001, N = 3$ (links) und $\beta = 0,001$ (rechts) 127</p>
Abbildung 57	<p>Abhängigkeit der Laufzeit der Simulierten Abkühlung von der Anzahl der unbestimmten Merkmaleinträge, welche im Rahmen des Feature Fillings befüllt werden, mit linearer Trendlinie (schwarz) für $T_{init} = 100, T_{min} = 0,001, \beta = 0,001$ und $N = 3.$ 127</p>
Abbildung 58	<p>Vereinheitlichung der Merkmalmengen vor der Durchführung der Konsistenzprüfung 143</p>

TABELLENVERZEICHNIS

Tabelle 1	Darstellung der Dokumentationsbestandteile eines beispielhaften Testfalls	22
Tabelle 2	In einem Merkmalmodell verwendete Merkmalstypen zur Abbildung von Abhängigkeiten zwischen Merkmalen nach [32] . . .	29
Tabelle 3	In einem Merkmalmodell verwendete Constraint-Beziehungen zwischen Merkmalen	29
Tabelle 4	Vereinfachtes Entscheidungsmodell angelehnt an das beispielhafte System <i>Klimatisierung</i> aus Abbildung 9	32
Tabelle 5	Vergleich sowie Evaluierung der verwandten Arbeiten, behandelt in den Abschnitten 4.2.5, 4.2.6 sowie 4.2.7. Legende: (+) erfüllt das Kriterium, (o) erfüllt das Kriterium teils, (-) erfüllt das Kriterium nicht.	53
Tabelle 6	Logische Definition der Constraint-Beziehungen und Abhängigkeiten eines Variantenmoduls anhand des Systems <i>Klimatisierung</i> aus Abbildung 21. Für die Darstellung der logischen Kontravalenz (XOR) wird das Symbol ($\dot{\vee}$) verwendet.	60
Tabelle 7	Prinzip des Merkmal mappings innerhalb eines Auszugs aus einer beispielhaften Anforderungsspezifikation	62
Tabelle 8	Prinzip des Merkmal mappings innerhalb eines Auszugs aus einer beispielhaften Testspezifikation (Aus Gründen der Übersichtlichkeit wird in der Darstellung auf die Elemente „Aktion“ und „Nachbedingung“ der Testspezifikation verzichtet).	63
Tabelle 9	Abbildbare Konfigurationen c_x für das System <i>Klimatisierung</i> durch das Merkmal mapping in konjunktiver Normalform (gemäß 5.5). Die Berechnung erfolgt mittels des kartesischen Produktes von F_{MC_2}	66
Tabelle 10	Konzept der Konsistenzprüfung über den Vergleich der Konfigurationsmengen C zwischen verlinkten Anforderungen r und Testfällen t bei einer Assoziationslogik in konjunktiver Normalform	69
Tabelle 11	Beim Merkmal mapping in konjunktiver Normalform abbildbare Konfigurationen c_x (gemäß 5.5) für das System <i>Klimatisierung</i> sowie die Anzahl $ F_{cov}(c_x) $ wie viele Merkmale von jeweils einer Konfiguration abgedeckt werden.	88
Tabelle 12	Merkmal mapping in der beispielhaften Anforderungsspezifikation für das System <i>Klimatisierung</i>	89

Tabelle 13	Abbildung welche Konfigurationen für welche Anforderungen gelten $\text{cov}(c_x, r_i)$, wie viele Anforderungen von jeweils einer Konfiguration abgedeckt werden $ R_{\text{cov}}(c_x) $ sowie wie viele Konfigurationen jeweils eine Anforderung abdecken $ C_{\text{cov}}(r_i) $	90
Tabelle 14	Durchläufe sowie das Ergebnis des Greedy-Vorgehens für das System <i>Klimatisierung</i>	95
Tabelle 15	Kenngrößen des Systems <i>Thermischer Komfort</i>	110
Tabelle 16	Kenngrößen des <i>Ladesystems</i>	111
Tabelle 17	Zusammenfassung der Ergebnisse der Konsistenzprüfung vor und nach der Umsetzung der Korrekturvorschläge. Es sind Abweichungen von 100% aufgrund von Rundungsfehlern möglich. Im Bezug auf die Laufzeit sind Mittelwerte aus 10 Durchläufen angegeben.	129
Tabelle 18	Zusammenfassung der Selektionsergebnisse für die Produktlinie <i>Thermischer Komfort</i> . Die Ergebnisse des Greedy-Algorithmus stellen Mittelwerte aus 10 Durchläufen dar, während die der Simulierten Abkühlung jeweils Mittelwerten aus 5 Durchläufen entsprechen.	130
Tabelle 19	Zusammenfassung der Selektionsergebnisse für die Produktlinie <i>Ladesystem</i> . Die Ergebnisse des Greedy-Algorithmus stellen Mittelwerte aus 10 Durchläufen dar, während die der Simulierten Abkühlung jeweils Mittelwerten aus 5 Durchläufen entsprechen.	130
Tabelle 20	Vergleich der verwandten Arbeiten (aus den Abschnitten 4.2.5, 4.2.6, 4.2.7) mit der in der vorliegenden Dissertation ausgearbeiteten Selektionsmethodik (siehe letzte Spalte). Legende: (+) erfüllt das Kriterium, (o) erfüllt das Kriterium teils, (-) erfüllt das Kriterium nicht.	132
Tabelle 21	Konzept der Konsistenzprüfung über den Vergleich der Merkmalmengen F zwischen verlinkten Anforderungen r und Testfällen t bei einer disjunktiven Assoziationslogik	142

LITERATURVERZEICHNIS

- [1] J. Al-Dallal und P. Sorenson (2008). *Testing Software Assets of Framework-Based Product Families During Application Engineering Stage*. *Journal of Software*, 3(5), 11–25. (Zitiert auf Seite 37.)
- [2] D. Batory (2005). *Feature models, grammars, and propositional formulas*. In *Proceedings of the 9th International Software Product Line Conference (SPLC)*, Seiten 7–20. (Zitiert auf Seite 31.)
- [3] D. Batory, D. Benavides und A. Ruiz-Cortes (2006). *Automated analysis of feature models: challenges ahead*. *Communications of the ACM*, 49(12), 45–47. (Zitiert auf Seite 31.)
- [4] G. Böckle, P. Knauber, K. Pohl und K. Schmid (2004). *Software-Produktlinien: Methoden, Einführung und Praxis*. dpunkt.verlag. (Zitiert auf den Seiten 24, 30 und 147.)
- [5] D. Benavides, D. Segura und A. Ruiz-Cortés (2010). *Automated analysis of feature models 20 years later: A literature review*. *Information Systems*, 35(6), 615–636. (Zitiert auf den Seiten 30 und 64.)
- [6] K. Berg, J. Bishop und D. Muthig (2005). *Tracing software product line variability: from problem to solution space*. In *Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries (SAICSIT)*, Seiten 182–191. (Zitiert auf den Seiten 5, 57 und 64.)
- [7] Berner & Mattner Systemtechnik GmbH (2013). *Die Klassifikationsbaum-Methode*. URL <http://www.cte-xl-professional.com/de/cte-xl-professional/klassifikationsbaum-methode/index.html>. (Zitiert auf Seite 47.)
- [8] K. Berns, B. Schürmann und M. Trapp (2010). *Eingebettete Systeme - Systemgrundlagen und Entwicklung eingebetteter Software*. Vieweg. (Zitiert auf Seite 12.)
- [9] D. Beuche, H. Papajewski und W. Schröder-Preikschat (2004). *Variability management with feature models*. *Science of Computer Programming*, 53(3), 333–352. (Zitiert auf Seite 30.)
- [10] E. Boutkova (2010). *Herausforderungen für Variantenmanagement in Anforderungsdokumenten*. In *Proceedings der Software Engineering - Fachtagung des GI-Fachbereichs Softwaretechnik (Workshops)*, Band 160, Seiten 329–338. (Zitiert auf Seite 15.)

- [11] E. Boutkova (2011). *Experience with Variability Management in Requirement Specifications*. In Proceedings of the 15th International Software Product Line Conference (SPLC), Seiten 303–312. (Zitiert auf den Seiten 58 und 61.)
- [12] B. Broekman und E. Notenboom (2002). *Testing Embedded Software*. Addison-Wesley. (Zitiert auf den Seiten 11 und 147.)
- [13] R. C. Bryce und C. J. Colbourn (2006). *Prioritized interaction testing for pair-wise coverage with seeding and constraints*. Information and Software Technology Journal, **48**(10), 960–970. (Zitiert auf Seite 47.)
- [14] R. C. Bryce, Y. Lei, D. R. Kuhn und R. Kacker (2010). *Handbook of Research on Software Engineering and Productivity Technologies: Implications of Globalization*, Kapitel 14 Combinatorial Testing, Seiten 196–208. IGI. (Zitiert auf Seite 47.)
- [15] F. Burgdorf (2010). *Eine kunden- und lebenszyklusorientierte Produktfamilienabsicherung für die Automobilindustrie*. Dissertation, Karlsruher Institut für Technologie. (Zitiert auf den Seiten 5, 43 und 45.)
- [16] I. Burnstein (2010). *Practical Software Testing: A Process-Oriented Approach*. Springer Verlag. (Zitiert auf den Seiten 13, 17 und 18.)
- [17] G. H. Campbell Jr., S. R. Faulk und D. M. Weiss (1990). Introduction to synthesis. Technischer Bericht, Software Productivity Consortium. (Zitiert auf Seite 32.)
- [18] I. do Carmo Machado, J. D. McGregor und E. S. de Almeida (2012). *Strategies for testing products in software product lines*. SIGSOFT Software Engineering Notes, **37**(6), 1–8. (Zitiert auf den Seiten 35 und 52.)
- [19] Carnegie Mellon Software Engineering Institute (2013). *Catalog of Software Product Lines - Case Studies*. URL <http://www.sei.cmu.edu/productlines/casestudies/catalog/index.cfm>. (Zitiert auf Seite 24.)
- [20] L. Chen und M. Ali Babar (2011). *A systematic review of evaluation of variability management approaches in software product lines*. Information and Software Technology, **53**(4), 344–362. (Zitiert auf den Seiten 28 und 32.)
- [21] L. Chen, M. Ali Babar und N. Ali (2009). *Variability Management in Software Product Lines: A Systematic Review*. In Proceedings of the 13th International Software Product Line Conference (SPLC), Seiten 81–90. (Zitiert auf den Seiten 28 und 32.)
- [22] H. Cichos, S. Oster, M. Lochau und A. Schürr (2011). *Model-based Coverage-Driven Test Suite Generation for Software Product Lines*. In Proceedings of the 14th International Conference on Model Driven Engineering Languages and Systems (MODELS), Seiten 425–439. (Zitiert auf Seite 48.)
- [23] P. Clements und L. Northrop (2007). *Software Product Lines: Practices and Patterns*. Addison-Wesley. (Zitiert auf den Seiten 4, 23 und 24.)
- [24] A. Cmyrev, R. Noerenberg, D. Hopp und R. Reissing (2012). *Consistency Checking of Feature Mapping between Requirements and Test Artefacts*. In Proceedings of the 19th ISPE International Conference on Concurrent Engineering (CE), Seiten 121–132. (Zitiert auf Seite 141.)

- [25] M. B. Cohen (2004). *Designing test suites for software interaction testing*. Dissertation, The University of Auckland. (Zitiert auf den Seiten 5, 47 und 49.)
- [26] M. B. Cohen, M. B. Dwyer und J. Shi (2006). *Coverage and adequacy in software product line testing*. In Proceedings of the ISSTA Workshop on Role of Software Architecture for Testing and Analysis (ROSATEA), Seiten 53–63. (Zitiert auf Seite 49.)
- [27] M. B. Cohen, M. B. Dwyer und J. Shi (2007). *Interaction Testing of Highly-Configurable Systems in the Presence of Constraints*. In Proceedings of the International Symposium on Software Testing and Analysis (ISSTA), Seiten 129–139. (Zitiert auf Seite 49.)
- [28] M. B. Cohen, M. B. Dwyer und J. Shi (2008). *Constructing Interaction Test Suites for Highly-Configurable Systems in the Presence of Constraints: A Greedy Approach*. IEEE Transactions on Software Engineering, **34**(5), 633–650. (Zitiert auf Seite 49.)
- [29] S. Cohen (2003). Predicting when product line investment pays. Technical note, Carnegie Mellon University, Software Engineering Institute. (Zitiert auf Seite 24.)
- [30] T. H. Cormen, C. Stein, R. L. Rivest und C. E. Leiserson (2009). *Introduction to Algorithms*. The MIT Press. (Zitiert auf den Seiten 95 und 96.)
- [31] K. Czarnecki und U. W. Eisenecker (2000). *Generative Programming: Methods, Tools, and Applications*. ACM Press/Addison-Wesley Publishing Co. (Zitiert auf Seite 28.)
- [32] K. Czarnecki und C. H. P. Kim (2005). *Cardinality-Based Feature Modeling and Constraints: A Progress Report*. In Proceedings of the International Workshop on Software Factories At OOPSLA. (Zitiert auf den Seiten 29, 31 und 152.)
- [33] K. Czarnecki, S. Helsen und U. W. Eisenecker (2004). *Staged configuration using feature models*. In Proceedings of the 3rd International Software Product Line Conference (SPLC), Seiten 266–283. (Zitiert auf den Seiten 30 und 31.)
- [34] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid und A. Waśowski (2012). *Cool features and tough decisions: a comparison of variability modeling approaches*. In Proceedings of the 6th International Workshop on Variability Modeling of Software-Intensive Systems (VaMoS), Seiten 173–182. (Zitiert auf Seite 28.)
- [35] Daimler AG (2013). *Mercedes-Benz Konfigurator*. URL http://www.mercedes-benz.de/content/germany/mpc/mpc_germany_website/de/home_mpc/passengercars/home/new_cars/model_overview_configurator.flash.html. (Zitiert auf Seite 23.)
- [36] I. S. Dunietz, W. K. Ehrlich, B. D. Szablak, C. L. Mallows und A. Iannino (1997). *Applying Design of Experiments to Software Testing*. In Proceedings of the 19th International Conference on Software Engineering (ICSE), Seiten 205–215. (Zitiert auf Seite 47.)
- [37] E. Engström (2010). *Regression Test Selection and Product Line System Testing*. Dissertation, Lund University. (Zitiert auf Seite 40.)

- [38] E. Engström und P. Runeson (2011). *Software product line testing - A systematic mapping study*. Information and Software Technology, **53**, 2–13. (Zitiert auf den Seiten 6, 35 und 54.)
- [39] D. Ganesan, J. Knodel, R. Kolb, U. Haury und G. Meier (2007). *Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG*. In Proceedings of the 11th International Software Product Line Conference (SPLC), Seiten 74–83. (Zitiert auf Seite 39.)
- [40] K. Grimm (1995). *Systematisches Testen von Software - Eine neue Methode und eine effektive Teststrategie*. Dissertation, Technische Universität Berlin. (Zitiert auf Seite 47.)
- [41] M. Große-Rhode, P. Manhart, R. Mauersberger, S. Schröck, M. Schulze und T. Weyer (2013). *Anforderungen von Leitbranchen der deutschen Industrie an Variantenmanagement und Wiederverwendung und daraus resultierende Forschungsfragestellungen*. In Tagungsband des Dritten Workshops zur Zukunft der Entwicklungssoftwareintensiver eingebetteter Systeme (ENVISION2020). (Zitiert auf Seite 61.)
- [42] C. Hafner (2013). *Skript zur Vorlesung „Optimization Methods for Engineers“*. <http://alphard.ethz.ch/hafner/Vorles/lect.htm>. (Zitiert auf Seite 97.)
- [43] M. Hamburg (2013). *ISTQB®/GTB Standardglossar der Testbegriffe*. Version 2.2, German Testing Board e.V. (Zitiert auf den Seiten 13, 16, 17 und 21.)
- [44] D. Henderson und A. W. Johnson (2003). *Handbook of Metaheuristics*, Kapitel 10 The Theory and Practice of Simulated Annealing, Seiten 287–319. Springer. (Zitiert auf Seite 99.)
- [45] J. Hromkovic und W. M. Oliva (2002). *Algorithmics for Hard Problems*. Springer. (Zitiert auf Seite 91.)
- [46] IBM (2013). *Rational DOORS®*. URL <http://www-03.ibm.com/software/products/de/de/ratidoor/>. (Zitiert auf Seite 16.)
- [47] Industrieanlagen-Betriebsgesellschaft mbH (IABG) (2013). *V-Modell®XT*. URL <http://www.v-modell.iabg.de/>. (Zitiert auf Seite 12.)
- [48] International Organization for Standardization (2011). *ISO 26262 Road vehicles - Functional safety*. (Zitiert auf den Seiten 19, 20 und 87.)
- [49] A. Jaaksi (2002). *Developing Mobile Browsers in a Product Line*. IEEE Software, **19**(4), 73–80. (Zitiert auf den Seiten 3, 38 und 39.)
- [50] L. Jin-hua, L. Qiong und L. Jing (2008). *The W-Model for Testing Software Product Lines*. In Proceedings of the International Symposium on Computer Science and Computational Technology (ISCSCT), Seiten 690–693. (Zitiert auf den Seiten 37 und 38.)
- [51] M. F. Johansen, O. Haugen und F. Fleurey (2011). *A Survey of Empirics of Strategies for Software Product Line Testing*. In Proceedings of the 4th International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Seiten 266–269. IEEE Computer Society. (Zitiert auf Seite 39.)

- [52] F. Kammüller, A. Rein und M.-O. Reiser (2010). *Feature link propagation across variability representations with Isabelle/HOL*. In Proceedings of the 2010 ICSE Workshop on Product Line Approaches in Software Engineering (PLEASE), Seiten 48–53. (Zitiert auf Seite 64.)
- [53] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak und A. S. Peterson (1990). Feature-oriented domain analysis (FODA) feasibility study. Technical report, Carnegie-Mellon University, Software Engineering Institute. (Zitiert auf den Seiten 24 und 30.)
- [54] P. Knauber, J. Bermejo, G. Böckle, J. C. S. do Prado Leite, F. van der Linden, L. Northrop, M. Stark und D. M. Weiss (2001). *Quantifying Product Line Benefits*. In Proceedings of the 4th International Workshop on Software Product-Family Engineering, Seiten 155–163. (Zitiert auf Seite 24.)
- [55] R. Kolb (2003). *A Risk-Driven Approach for Efficiently Testing Software Product Lines*. In Proceedings of the 5th GPCE Young Researches Workshop. (Zitiert auf Seite 43.)
- [56] T. Koomen und M. Pol (1999). *Test Process Improvement: A Practical Step-by-step Guide to Structured Testing*. Addison-Wesley. (Zitiert auf Seite 17.)
- [57] D. R. Kuhn, D. R. Wallace und A. M. Gallo Jr. (2004). *Software Fault Interactions and Implications for Software Testing*. IEEE Transactions on Software Engineering, **30**(6), 418–421. (Zitiert auf Seite 47.)
- [58] J. Lahtinen, P. Myllymäki, T. Silander und H. Tirri (1996). *Empirical Comparison of Stochastic Algorithms*. In Proceedings of the Second Nordic Workshop on Genetic Algorithms and their Applications. (Zitiert auf Seite 97.)
- [59] B. P. Lamanha, M. P. Usaola und M. P. Velthius (2009). *Software Product Line Testing - A Systematic Review*. In Proceedings of the 4th International Conference on Software and Data Technologies (ICSOFTE), Seiten 23–30. (Zitiert auf Seite 35.)
- [60] K. Lauenroth und K. Pohl (2008). *Dynamic Consistency Checking of Domain Requirements in Product Line Engineering*. In Proceedings of the 16th IEEE International Requirements Engineering Conference. (Zitiert auf Seite 64.)
- [61] K. Lauenroth und K. Pohl (2009). *Variabilität als eine eigenständige Sicht auf Produktlinien*. In Proceedings der Software Engineering - Fachtagung des GI-Fachbereichs Softwaretechnik (Workshops), Band 150, Seiten 119–124. (Zitiert auf Seite 31.)
- [62] W. E. Lewis (2008). *Software Testing and Continuous Quality Improvement*. Auerbach Publishers. (Zitiert auf Seite 17.)
- [63] P. Liggesmeyer (2009). *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. Spektrum Akademischer Verlag. (Zitiert auf den Seiten 4, 16, 17 und 18.)
- [64] P. Liggesmeyer und D. Rombach, Herausgeber (2005). *Software-Engineering eingebetteter Systeme: Grundlagen - Methodik - Anwendungen*. Spektrum Akademischer Verlag. (Zitiert auf Seite 11.)

- [65] F. Lindlar (2012). *Modellbasierter evolutionärer Funktionstest*. Dissertation, Technische Universität Berlin. (Zitiert auf Seite 97.)
- [66] S. Lity, M. Lochau, I. Schaefer und U. Goltz (2012). *Delta-Oriented Model-Based SPL Regression Testing*. In Proceedings of the 3rd International Workshop on Product Line Approaches In Software Engineering (PLEASE), Seiten 53–56. (Zitiert auf den Seiten 40 und 107.)
- [67] M. Lochau, I. Schaefer, J. Kamischke und S. Lity (2012). *Incremental Model-based Testing of Delta-oriented Software Product Lines*. In Proceedings of the 6th International Conference on Tests and Proofs (TAP), Seiten 67–82. (Zitiert auf Seite 108.)
- [68] M. Lundy und A. Mees (1986). *Convergence of an Annealing Algorithm*. Mathematical Programming, **34**(1), 111–124. (Zitiert auf Seite 102.)
- [69] O. Manicke (2011). *Durchgängiges Variantenmanagement zur Komplexitätsbeherrschung im Entwicklungsprozess mechatronischer Fahrzeugfunktionen*. Dissertation, Technische Universität Dresden. (Zitiert auf den Seiten 5 und 43.)
- [70] O. Manicke und D. R. Dorn (2008). *Methoden zur Entwicklung eines Variantenmanagements zur Optimierung von EE-Funktionstests vernetzter Steuergeräte*. In Tagungsband zur 2. AutoTest Fachkonferenz "Test von Hard- und Software in der Automobilentwicklung". (Zitiert auf Seite 43.)
- [71] T. W. Manikas und J. T. Cain (1996). Genetic algorithms vs. simulated annealing: a comparison of approaches for solving the circuit partitioning problem. Technical report, SMU Computer Science and Engineering Research. (Zitiert auf Seite 97.)
- [72] S. Mann und G. Rock (2011). *Control variant-rich models by variability measures*. In Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems (VaMoS), Seiten 29–38. (Zitiert auf Seite 63.)
- [73] J. D. McGregor (2001). Testing a software product line. Technical report, Carnegie Mellon University, Software Engineering Institute. (Zitiert auf Seite 47.)
- [74] J. D. McGregor (2008). *Toward a Fault Model for Software Product Lines*. In Proceedings of the 5th Software Product Line Testing Workshop (SPLiT), Seiten 157–162. (Zitiert auf Seite 43.)
- [75] A. Metzger, P. Heymans, K. Pohl, P.-Y. Schobbens und G. Saval (2007). *Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis*. In Proceedings of the 15th IEEE International Requirements Engineering Conference (RE), Seiten 243–253. (Zitiert auf Seite 31.)
- [76] H. E. Mößmer, M. Schedlbauer und W. A. Günthner (2007). *Neue Wege in der Automobillogistik: Die Vision der Supra-Adaptivität*, Kapitel 1.1 Die automobile Welt im Umbruch, Seiten 3–15. Springer. (Zitiert auf Seite 3.)

- [77] P. A. da Mota Silveira Neto, I. do Carmo Machado, J. D. McGregor, E. S. de Almeida und S. R. de Lemos Meira (2011). *A systematic mapping study of software product lines testing*. Information and Software Technology, 53(5), 407–423. (Zitiert auf den Seiten 6, 35 und 54.)
- [78] C. Nebut, Y. L. Traon und J.-M. Jézéquel (2006). System testing of product lines: From requirements to test cases. In *Software Product Lines*, Kapitel 12, Seiten 447–477. Springer Verlag. (Zitiert auf Seite 37.)
- [79] R. Nörenberg (2012). *Effizienter Regressionstest von E/E-Systemen nach ISO 26262*. Dissertation, Karlsruher Institut für Technologie. (Zitiert auf den Seiten 15, 17, 18, 21 und 147.)
- [80] R. Nörenberg, A. Cmyrev, R. Reißing und K. D. Müller-Glaser (2011). *An Efficient Specification-Based Regression Test Selection Technique for E/E-Systems*. In Tagungsband zur 41. GI Jahrestagung, 9. Workshop Automotive Software Engineering. (Zitiert auf den Seiten 40, 41 und 107.)
- [81] R. Nörenberg, A. Cmyrev, R. Reißing und K. D. Müller-Glaser (2011). *Efficient verification planning for ISO-conformant functional testing of automotive applications*. In Proceedings des 4. Symposiums Testen im System- und Software-Life-Cycle, Technische Akademie Esslingen. (Zitiert auf den Seiten 20 und 147.)
- [82] E. M. Olimpiew (2008). *Model-based Testing for Software Product Lines*. Dissertation, George Mason University. (Zitiert auf Seite 41.)
- [83] S. Oster (2011). *Feature Model-based Software Product Line Testing*. Dissertation, Technische Universität Darmstadt. (Zitiert auf den Seiten 3, 5, 31, 47, 60 und 119.)
- [84] S. Oster, F. Markert und P. Ritter (2010). *Automated Incremental Pairwise Testing of Software Product Lines*. In Proceedings of the 14th International Software Product Line Conference (SPLC), Seiten 196–210. (Zitiert auf Seite 48.)
- [85] S. Oster, P. Ritter und A. Schürr (2010). *Featuremodellbasiertes und kombinatorisches Testen von Software-Produktlinien*. In Proceedings der Software Engineering - Fachtagung des GI-Fachbereichs Softwaretechnik, Seiten 177–188. (Zitiert auf Seite 37.)
- [86] S. Oster, A. Wübbeke, G. Engels und A. Schürr (2011). Model-based software product lines testing survey. In J. Zander, I. Schieferdecker, und P. J. Mosterman, Herausgeber, *Model-based Testing for Embedded Systems*. CRC Press. (Zitiert auf den Seiten 5, 35, 42 und 43.)
- [87] S. Oster, I. Zoricic, F. Markert und M. Lochau (2011). *MoSo-PoLiTe: Tool Support for Pairwise and Model-Based Software Product Line Testing*. In Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems (VaMoS), Seiten 79–82. (Zitiert auf Seite 48.)
- [88] G. Perrouin, S. Sen, J. Klein, B. Baudry und Y. le Traon (2010). *Automated and Scalable T-wise Test Case Generation Strategies for Software Product Lines*. In Proceedings of the 3rd International Conference on Software Testing, Verification and Validation (ICST), Seiten 459–468. (Zitiert auf Seite 47.)

- [89] G. Perrouin, S. Oster, S. Sen, J. Klein, B. Baudry und Y. le Traon (2011). *Pairwise Testing for Software Product Lines: A Comparison of Two Approaches*. *Software Quality Journal*, **20**(3–4), 605–643. (Zitiert auf Seite 47.)
- [90] K. Pohl, G. Böckle und F. J. van der Linden (2005). *Software Product Line Engineering - Foundations, Principles, and Techniques*. Springer Berlin Heidelberg. (Zitiert auf den Seiten 3, 4, 23, 24, 25, 26, 27, 28, 31, 38 und 147.)
- [91] A. Pols, K. Fuest und C. Krys (2007). *Zukunft digitale Wirtschaft*. Gemeinsame Studie des BITKOM e.V. und der Roland Berger Strategy Consultants. (Zitiert auf Seite 3.)
- [92] pure-systems GmbH (2013). *pure::variants*. URL http://www.pure-systems.com/pure_variants.49.0.html. (Zitiert auf Seite 109.)
- [93] R. Rabiser und D. Benavides (2011). *Variability Modeling of Diverse Systems*. In Survey on Diversity Awareness and Management, Deliverable D1.2. (Zitiert auf den Seiten 28, 30 und 32.)
- [94] M. Recknagel (2010). *Serienproduktion von Spezifikationen - Neue Wege bei der Unterstützung des Spezifikationsprozesses*. Vortrag bei den Requirements Days, München. (Zitiert auf Seite 15.)
- [95] M.-O. Reiser und M. Weber (2007). *Multi-level feature trees: A pragmatic approach to managing highly complex product families*. *Requirements Engineering*, **12**(2), 57–75. (Zitiert auf Seite 28.)
- [96] A. Reuys, E. Kamsties, K. Pohl und S. Reis (2005). *Model-Based System Testing of Software Product Families*. In Proceedings of the 17th International Conference on Advanced Information Systems Engineering (CAiSE), Seiten 519–534. (Zitiert auf den Seiten 5 und 41.)
- [97] F. Roos-Frantz, D. Benavides und A. Ruiz-Cortés (2009). *Feature Model to Orthogonal Variability Model Transformations. A First Step*. In Actas del VI Taller sobre Desarrollo de Software Dirigido por Modelos, Band 3, Seiten 81–90. (Zitiert auf Seite 31.)
- [98] C. Rupp und die SOPHISTen (2009). *Requirements-Engineering und -Management: Professionelle, iterative Anforderungsanalyse für die Praxis*. Hanser. (Zitiert auf Seite 14.)
- [99] K. D. Scheidemann (2006). *Optimizing the Selection of Representative Configurations in Verification of Evolving Product Lines of Distributed Embedded Systems*. In Proceedings of the 10th International Software Product Line Conference (SPLC), Seiten 75–84. (Zitiert auf Seite 51.)
- [100] K. D. Scheidemann (2007). *Verifying Families of System Configurations*. Dissertation, Technische Universität München. (Zitiert auf den Seiten 3, 5 und 50.)
- [101] K. Schmid und I. John (2004). *A customizable approach to full lifecycle variability management*. *Science of Computer Programming*, **53**(3), 259–284. (Zitiert auf den Seiten 5 und 64.)

- [102] K. Schmid, R. Rabiser und P. Grünbacher (2011). *A comparison of decision modeling approaches in product lines*. In Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems (VaMoS), Seiten 119–126. (Zitiert auf Seite 32.)
- [103] F. Schmidt (2012). *Ein effizienter Testprozess für sicherheitskritische und variantenreiche Systeme*. In Tagungsband zur 4. AutoTest Fachkonferenz "Test von Hard- und Software in der Automobilentwicklung". (Zitiert auf Seite 43.)
- [104] G. Schouten (2007). *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*, Kapitel 15 Philips Medical Systems, Seiten 233–248. Springer. (Zitiert auf den Seiten 4, 24 und 39.)
- [105] M. Sinnema und S. Deelstra (2007). *Classifying variability modeling techniques*. Information and Software Technology, **49**(7), 717–739. (Zitiert auf Seite 28.)
- [106] S. S. Skiena (2008). *The Algorithm Design Manual*. Springer. (Zitiert auf den Seiten 97, 98 und 99.)
- [107] H. M. Sneed, M. Baumgartner und R. Seidl (2007). *Der Systemtest: Anforderungsbasiertes Testen von Software-Systemen*. Hanser. (Zitiert auf den Seiten 19 und 147.)
- [108] SOPHIST GROUP und C. Rupp (2009). *Systemanalyse kompakt*. Spektrum Akademischer Verlag. (Zitiert auf Seite 14.)
- [109] SparxSystems (2013). *Enterprise Architect*. URL <http://www.sparxsystems.eu/>. (Zitiert auf Seite 109.)
- [110] SPES XT 2020 (2013). *Software Plattform Embedded Systems XT*. URL http://spes2020.informatik.tu-muenchen.de/spes_xt-home.html. (Zitiert auf Seite 61.)
- [111] A. Spillner und T. Linz (2011). *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester - Foundation Level nach ISTQB-Standard*. dpunkt.verlag. (Zitiert auf den Seiten 4, 16, 17, 18 und 35.)
- [112] A. Spillner, T. Roßner, M. Winter und T. Linz (2012). *Praxiswissen Softwaretest - Testmanagement, Aus- und Weiterbildung zum Certified Tester - Advanced Level nach ISTQB-Standard*. dpunkt.verlag. (Zitiert auf Seite 19.)
- [113] M. Staples und D. Hill (2004). *Experiences Adopting Software Product Line Development without a Product Line Architecture*. In Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC), Seiten 176–183. (Zitiert auf Seite 39.)
- [114] M. Stier (2009). *Automatische Testfallgenerierung mittels selbstlernender Methoden für die Coverageanalyse von Automotive Embedded Software*. Diplomarbeit, Universität Stuttgart. (Zitiert auf Seite 97.)
- [115] N. Stoyanova (2013). *Verbesserung der Qualität natürlich-sprachlicher Spezifikationen*. Dissertation, Universität Stuttgart. (Zitiert auf den Seiten 14 und 15.)

- [116] V. Stricker, A. Metzger und K. Pohl (2010). *Avoiding redundant testing in application engineering*. In Proceedings of the 14th International Software Product Line Conference (SPLC), Seiten 226–240. (Zitiert auf Seite 37.)
- [117] B. Suman (2013). *Encyclopedia of Operations Research and Management Science*, Kapitel Simulated Annealing, Seiten 1395–1404. Springer. (Zitiert auf den Seiten 97, 98 und 99.)
- [118] A. Tevanlinna, J. Taina und R. Kauppinen (2004). *Product family testing: a survey*. SIGSOFT Software Engineering Notes, **29**(2), 1–12. (Zitiert auf den Seiten 37, 38 und 40.)
- [119] The Institute of Electrical and Eletronics Engineers (1990). *Standard Glossary of Software Engineering Terminology*. IEEE Standard 610.12-1990, Seiten 1–84. (Zitiert auf den Seiten 12, 14, 16, 17 und 21.)
- [120] The MathWorks (2013). *Simulink®*. URL <http://www.mathworks.de/products/simulink/>. (Zitiert auf Seite 109.)
- [121] S. Thiel, S. Ferber, T. Fischer, A. Hein und M. Schlick (2001). *A Case Study in Applying a Product Line Approach for Car Periphery Supervision Systems*. In Proceedings of In-Vehicle Software, Seiten 43–55. (Zitiert auf Seite 24.)
- [122] W. T. Tsai, X. Zhou, R. A. Paul, Y. Chen und X. Bai (2007). *A Coverage Relationship Model for Test Case Selection and Ranking for Multi-version Software*. In Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium (HASE), Seiten 105–112. (Zitiert auf Seite 41.)
- [123] E. Uzuncaova, S. Khurshid und D. Batory (2010). *Incremental Test Generation for Software Product Lines*. IEEE Transactions on Software Engineering, **36**(3), 309–322. (Zitiert auf Seite 40.)
- [124] V. V. Vazirani (2001). *Approximation Algorithms*. Springer. (Zitiert auf Seite 91.)
- [125] A. Wübbecke (2010). *Variabilitätsmanagement in Anforderungs- und Testfallspezifikation für Software-Produktlinien*. Dissertation, Universität Paderborn. (Zitiert auf Seite 64.)
- [126] H. Zeng, W. Zhang und D. Rine (2004). *Analysis of testing effort by using core assets in software product line testing*. In Proceedings of the 3rd Software Product Line Testing Workshop (SPLiT), Seiten 1–6. (Zitiert auf Seite 43.)
- [127] S. Ziegler (2010). *Eingebettete Systeme - ein strategisches Wachstumsfeld für Deutschland: Anwendungsbeispiele, Zahlen und Trends*. BITKOM - Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. (Zitiert auf den Seiten 3 und 12.)

EIDESSTATTLICHE ERKLÄRUNG

Ich versichere wahrheitsgemäß, die Dissertation bis auf die dort angegebene Hilfe selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer und eigenen Veröffentlichungen unverändert oder mit Änderungen entnommen wurde.

Berlin, 2014

Anastasia Cmyrev