Institut für Informatik
der Technischen Universität München

Quantifizierung und Reduktion von testinduzierten Qualitätskosten

**Praxisorientierte Optimierungen
für die Entwicklung eingebetteter Systeme
in der Automobilindustrie**

Matthias Wiemann

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. B. Radig
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. Dr. h.c. M. Broy
2.	Univ.-Prof. Dr. A. Pretschner
	Technische Universität Kaiserslautern

Die Dissertation wurde am 15.04.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 03.08.2010 angenommen.

Kurzfassung

Der Qualitäts- und Kostendruck in der Softwareentwicklung von eingebetteten Systemen veranlasst die Automobilindustrie, stetig nach Optimierungen des Entwicklungsprozesses zu suchen. Vielfach befassen sich Optimierungsansätze mit einer Intensivierung der konstruktiven Qualitätssicherung. Jedoch kann selbst eine optimale konstruktive Qualitätssicherung die dynamische Qualitätssicherung nicht ersetzen. Analysen weisen zudem den Test als Kostentreiber im Entwicklungsprozess aus. In der vorliegenden Arbeit wird daher ein Optimierungsprozess entwickelt, mit dem effektive, kostenreduzierende Teststrategien für die dynamische Qualitätssicherung aufgezeigt werden können.

Der Optimierung geht eine Kosten- und Strukturanalyse der Entwicklungspraxis von automobilen eingebetteten Systemen voran. Untersucht werden acht Entwicklungsprojekte eines großen Fahrzeugteilezulieferers. Laut Fehlerstromanalysen entstanden Fehlerursachen größtenteils in der Softwareimplementierung und wurden zu maßgeblichen Anteilen erst im Systemtest oder durch den Kunden identifiziert. Kostenschwerpunkte im Testprozess bildeten dementsprechend die Durchführung des Systemtests sowie die Fehlerbeseitigung. Die Fehlerbeseitigungskosten werden in den untersuchten Projekten durch die Testphase, in der ein Fehler identifiziert wird, sowie durch den Systemtyp und die Fehlerpriorität beeinflusst. Lediglich die Testphase, in der ein Fehler identifiziert wird, ist durch Teststrategien beeinflussbar und steht somit im Fokus der später beschriebenen Optimierungen. Zur Reduktion von Testdurchführungskosten können Testautomatisierungen eingesetzt werden. Um die Effektivität der in den Projekten eingesetzten Automatisierungen bewerten zu können, wird ein Algorithmus zur Abschätzung des Break-Even-Points von Testautomatisierungen aufgestellt. In den zwei analysierten Projekten wird der Break-Even-Point nicht innerhalb der Projektlaufzeit erreicht. Dies ist auf unterschiedliche, projektspezifische Kostenfaktoren, wie z. B. hohe Lizenzkosten für Testautomatisierungswerkzeuge, zurückzuführen.

Der entwickelte Optimierungsprozess umfasst mehrere Bearbeitungsschritte. Zunächst wird an einem Referenzprojekt das durch frühzeitige Fehleridentifikation maximal zu erwartende Kosteneinsparpotential abgeschätzt. Anschließend werden die Fehlerursachen auf Basis der Fehlereigenschaften mittels eines erarbeiteten Klassifikationsschemas gruppiert. Den Fehlerklassen sind Testverfahren zugewiesen, die Fehlerursachen mit den entsprechenden Eigenschaften frühzeitig identifizieren. Eine Gewichtung der Fehlerklassen anhand der Testziele führt zu einer optimierten Auswahl von Testverfahren. Weiterhin wird der Kosteneinfluss von möglichen Testautomatisierungen und von Veränderungen kostentreibender Faktoren bewertet. Die Kostenwirksamkeit aller Strategieoptimierungen wird mit Hilfe eines hergeleiteten Kostenmodells für testinduzierte Qualitätskosten abgeschätzt. Vorteile gegenüber

vorliegenden Qualitätskostenmodellen liegen darin, dass die Struktur des Testprozesses nachgebildet wird, die für Strategieoptimierungen relevanten testspezifischen Kostenelemente berücksichtigt und dabei die Kostenstrukturen feingliedrig aufgefächert werden. So werden Kostenschwerpunkte sichtbar, die Ansatzpunkte für die Optimierungsschritte sein können. Das Kostenmodell ermöglicht eine an die Datenqualität angepasste Anwendung. Kosten können sowohl retrospektiv präzise als auch prospektiv abgeschätzt ermittelt werden.

Detaillierte Daten zu Fehlereigenschaften und -ursachen liegen nur für ein Projekt vor. Die Analyse der Fehlerklassifikation ergibt, dass die Identifikation speziell von solchen Fehlerursachen zu intensivieren ist, die auf fehlerhafte oder unvollständige Spezifikationen sowie auf fehlerhafte bzw. fehlende Realisierungen von modulübergreifenden Anforderungen zurückzuführen sind. Auf Grund dieses Ergebnisses wird für die Strategieoptimierung der Einsatz eines verhaltensmodellbasierten Testverfahrens angenommen. Simulierte Kosten und Effizienz des Einsatzes werden diskutiert. Der durchgeführte Optimierungsprozess umfasst die Anwendung einer verhaltensmodellbasierten Testmethodik, die Reduktion eines projektspezifischen Kostenschwerpunktes und weitgehende Testautomatisierungen. Die Kostensimulationen beziffern erhebliche Einsparungen.

Zusammenfassend besteht der wissenschaftliche Beitrag der vorliegenden Arbeit in der detaillierten Darstellung von Kostenzusammenhängen in Testprozessen und Teststrategien bei Entwicklungen eingebetteter Systeme der Automobilindustrie. Vergleichbar umfassende Kostenzusammenhänge wurden bisher weder aufgezeigt noch empirisch belegt. Auf Basis der durchgeführten Struktur- und Kostenanalysen wird ein Optimierungsprozess, der Teststrategien hinsichtlich Kosten und Effektivität optimiert, samt unterstützenden Verfahren erarbeitet und erprobt.

Danksagung

Diese Dissertation wurde an der Technischen Universität München am Lehrstuhl IV für *Software & Systems Engineering* unter der Leitung von Herrn Prof. Dr. Dr. h.c. Broy in Kooperation mit der Siemens AG angefertigt und dort von Herrn Gericke betreut. In der Forschungsabteilung *Corporate Research and Technologies* der Siemens AG hatte ich die Möglichkeit, zahlreiche Softwareprojekte von eingebetteten Systemen in verschiedenen Sparten, so auch der Automobilindustrie, zu begleiten.

Mein erster Dank geht an Herrn Prof. Dr. Dr. h.c. Broy sowie an meinen Zweitgutachter Herrn Prof. Dr. Pretschner für deren Betreuung und Anleitung in den letzten Jahren. Deren konstruktive Hinweise sowie Meinungen waren mir stets wertvolle Hilfe. Auch beim Lehrstuhl möchte ich mich für die gemeinsame Zeit und deren Unterstützung besonders bedanken.

Ebenso essentiell zur Anfertigung der Arbeit war die Betreuung seitens der Siemens AG durch Herrn Gericke. Neben zahlreichen, fachlichen Diskussionen ermöglichte Herr Gericke mir die Integration in verschiedene Testabteilungen unterschiedlicher Unternehmensbereiche. Dadurch erlangte ich projekt-, team- und branchenübergreifende Erfahrungen sowie Zugang zu den für diese Arbeit wichtigen Praxisdaten.

Mein ganz besonderer Dank gilt letztlich meiner Frau Aude sowie meinen Eltern für ihre fortwährende liebevolle Unterstützung, ihr Verständnis und ihre Geduld. Sie gaben mir den notwendigen familiären Rückhalt sowie den Freiraum zur Erstellung dieser Arbeit.

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

Automobile eingebettete Systeme: *Eingebettete Systeme* sind derzeit die technische Umsetzung für *Steuergeräte* [43]. Der Begriff Steuergerät umfasst auch Regelungen (Steuerung mit Rückkopplung). Diese Software-Hardware-Kombinationen sind als Bestandteile der Elektronik in größere Gesamtsysteme integriert und übernehmen in heutigen Automodellen ein äußerst breites Spektrum an Steuerungs- und Regelungsaufgaben für Komfortfunktionen [17, 177].

Softwarekomplexität und -volumen: Bis vor zirka 40 Jahren wurden Automobile ausschließlich durch Mechanik und Elektronik ohne jeglichen Einsatz von Software betrieben [43]. Seit 1970 steigt die Anzahl der Steuergeräte, der Funktionen und der Funktionen je Steuergerät exponentiell [17]. Inzwischen werden 80-90% der Innovationen in Automobilen softwaregestützt mit Hilfe von eingebetteten Systemen realisiert [17, 66]. Der Umfang der Softwaresysteme in Fahrzeugen war bereits im Jahre 2002 vergleichbar mit denen gängiger Desktopapplikationen [177]. Nach damaligen Prognosen wird die Menge der Software in Automobilen auch weiterhin exponentiell anwachsen [67]. Analysen aus dem Jahre 2006 sagen für 2010 mehr als 1 GB Maschinencode voraus [38, 205]. Mit dem Anstieg des integrierten Softwarevolumens steigt gleichzeitig auch die Komplexität der Software.

Bedarf nach Steigerung der Softwarequalität: Die Qualität der Software ist als ein maßgeblich wettbewerbsbestimmender Faktor anzusehen [17, 161, 231], da eine geringe Pannenhäufigkeit einer der wichtigsten Gründe für eine Kaufentscheidung darstellt [17, 167]. Die Qualitätssicherungsmaßnahmen der Entwicklungsprozesse können derzeit nicht immer mit der rapide ansteigenden Komplexität der Softwaresysteme Schritt halten und das angestrebte hohe Qualitätsniveau erreichen [17, 83, 103, 108]. Unter Beibehaltung heutiger Softwareentwicklungsverfahren und eines ungeminderten Wachstums des Software- und Elektronikvolumens wird ein Anstieg der softwarebedingten Pannen von heute etwa ein Drittel bis 2013 auf zwei Drittel prognostiziert [83, 108]. Die Softwareerstellung verlangt somit nach einer systematischen Qualitätssteigerung [17, 38, 46, 66, 67, 205]. Hierbei können sowohl konstruktive Qualitätssicherungsmaßnahmen als auch analytische Qualitätssicherungsmaßnahmen beitragen [176].

Dynamische analytische Qualitätssicherung: Auch bei intensiver konstruktiver und statisch analytischer Qualitätssicherung in frühen Projektphasen erübrigt sich die dynamische analytische Qualitätssicherung nicht [176]. In derzeitiger Praxis hat die dynamische analytische Qualitätssicherung mit umfangreichem Testprozess im Allg. noch eine große, praktische Bedeutung [8, 15, 197, 199, 214, 215, 262].

1.2. Zielsetzung und Beitrag der Arbeit

Betrachteter Entwicklungsbereich und Zielsetzung: Betrachtet wird der Testprozess in der Entwicklung von eingebetteten Systemen in der Automobilindustrie. Der Testprozess umfasst eine Abfolge von Testphasen, in denen Testteams die Software testfallgesteuert ausführen. In einer Teststrategie werden die einzusetzenden Testverfahren, die Art und der Umfang der Testfallerstellung und die Art der Testdurchführung festgelegt. Die Auswahl der Testverfahren beeinflusst Entwicklungszeit, Entwicklungskosten und Produktqualität. Es wird angestrebt, Wege zu einer effektivitätssteigernden und kostenreduzierenden Optimierung des Testprozesses aufzuzeigen. Die für die Optimierung erforderlichen Informationen zum Testablauf sowie zu dessen Kosten und Strukturen werden aus Entwicklungsprojekten abgeleitet.

Eingliederung des Testprozesses: Der Testprozess zählt zur analytischen Qualitätssicherung und ergänzt konstruktive und statische analytische Qualitätssicherungsmaßnahmen. Die Effektivität vorangegangener Qualitätssicherungsmaßnahmen kann auf die Ausrichtung der Testziele sowie die erforderliche Intensität des Testprozesses und damit auf die Kosten einwirken [176].

Begründung der Zentrierung auf den Testprozess: Eine intensive konstruktive Qualitätssicherung lässt eine profunde Qualitätssteigerung und gleichzeitig rückläufige Fehlerbeseitigungskosten für das Gesamtprojekt erwarten [28, 186]. Die Wissenschaft hat zahlreiche Methoden zur konstruktiven Qualitätssicherung entwickelt [28, 149, 176, 199], die in der Praxis nur zögerlich Einzug halten [176, 178, 197, 214]. Nach Abschluss der Implementierung ist die Software trotz konstruktiver Qualitätssicherung noch nicht als verifiziert anzusehen und es ist ein Testprozess anzuschließen [176]. Das Erarbeiten von Optimierungsmöglichkeiten für derzeitige Testprozesse stellt einen umgehend realisierbaren Beitrag für Qualitätssteigerung in automobilen Entwicklungen dar.

Optimierung von Effektivität und Kosten: In der Entwicklung der Software von eingebetteten Systemen der Automobilindustrie bildet der Testprozess zusammen mit der sich anschließenden Fehlerbeseitigung einen Kostenschwerpunkt [8, 15, 34, 86, 145, 199, 215]. Eine Senkung dieser Qualitätskosten ist erstrebenswert. Andererseits muss die Software bei Markteinführung ein gefordertes Qualitätsniveau aufweisen. Folglich gilt es, die Intensität des Testprozesses unter Berücksichtigung der Kosten zu bestimmen. Angestrebt wird, die Kosten des Testprozesses durch Fehlerfrüherkennung und Testautomatisierung zu senken und hierbei durch eine optimierte Auswahl der Testverfahren die Effizienz der Fehleridentifikation zu steigern. Weiterhin sollen projektspezifische Kostenschwerpunkte sichtbar gemacht werden.

Bearbeitung der Thematik: Es sollen Kostenzusammenhänge von Testprozessen isoliert und daraus Wege zu optimierten Teststrategien abgeleitet sowie Hilfsmittel

zur Verfügung gestellt werden, um projektspezifische Lösungen zu ermitteln. Die Formulierung des Ziels bringt zum Ausdruck, dass nicht davon ausgegangen wird, dass *eine* Teststrategie für alle Entwicklungen von eingebetteten Systemen in der Automobilindustrie optimal ist bzw. im Zuge der Weiterentwicklung bleiben wird. Es wird eine *projektspezifische Effizienz* angestrebt, die ein optimiertes Verhältnis von Qualitätskosten und Softwarequalität aus einem vergleichbaren, abgeschlossenen Projekt für ein zukünftiges Projekt ableitet. Als grundlegende Voraussetzung ergibt sich die Notwendigkeit, Qualitätskosten prospektiv zu quantifizieren. Betrachtet wird der Ausschnitt der Qualitätskosten, der durch den Testprozess und die resultierenden Fehlerbeseitigungen bestimmt wird. Diese Kosten werden im Folgenden begrifflich als *testinduzierte Qualitätskosten* zusammengefasst. Die testinduzierten Qualitätskosten müssen prospektiv durch u. U. feingranulare Kostenveränderungen variiert werden können, um Strategievarianten zur Bestimmung der optimalen Strategie simulieren zu können. Daher werden Strukturen, Kostenelemente und Abhängigkeiten der testinduzierten Qualitätskosten für die betrachteten Entwicklungen herausgearbeitet. Hierzu werden Arbeitsabläufe von Projekten im Bereich der Fahrzeugteileentwicklung eines Großkonzerns untersucht. Die Kosten und die Struktur des testfallbasierten Testprozesses im V-Modell werden anhand der vorliegenden Entwicklungsprojekte sowie anhand der Literatur analysiert. Zusätzlich wird überprüft, ob im Falle von fehlenden Entwicklungsdaten gezielt Erkenntnisse aus der Literatur die nicht ermittelbaren Informationen substituieren können, z. B. Daten zum Anstieg der Fehlerbeseitigungskosten über Testphasen hinweg.

Auf der Basis der Projekt- und Literaturanalysen wird ein Kostenmodell zur Ermittlung der testinduzierten Qualitätskosten entwickelt. Weitere, aus dem Kostenmodell abgeleitete Verfahrensentwicklungen zielen auf die Unterstützung des Optimierungsprozesses, wie z. B. durch Bestimmung von Kosteneinsparpotentialen, durch Klassifizierung von unzureichend erkannten Fehlerursachen und durch objektivierte Auswahl von Testverfahren für priorisierte Testziele. Es wird ein Vorgehensschema für den Optimierungsprozess formuliert. Dessen Funktion und das Zusammenspiel der entwickelten Verfahren werden an einem Projekt eines großen Fahrzeugteileherstellers demonstriert. Für dieses Projekt werden die testinduzierten Qualitätskosten verschiedener, optimierter Teststrategien simuliert. Es wird geprüft, ob die Strategievarianten sowohl die Softwarequalität steigern als auch die Qualitätskosten positiv beeinflussen.

Auf der Grundlage der empirischen Ergebnisse zur Optimierung werden Folgerungen für eine kostenoptimierte Qualitätssicherung im Testprozess für künftige Entwicklungen abgeleitet und die Verallgemeinerbarkeit der Ergebnisse diskutiert.

Entwicklungsdaten: Für den betrachteten Entwicklungsbereich liegen vergleichsweise wenig publizierte Untersuchungen vor, die auf detaillierten Entwicklungsdaten beruhen. Die Daten werden von den Unternehmen oft als sehr sensibel eingestuft und nur zögerlich und eingeschränkt zur Verfügung gestellt. Da hier jedoch bewusst Daten aus dem Entwicklungsprozess, also nicht durch künstliche oder experimentelle Situationen verfälscht, analysiert werden sollen, müssen Einschränkungen bezüglich der Untersuchungsmöglichkeiten, z. B. wegen unvollständiger Daten, in Kauf genommen werden.

1.3. Verwandte Arbeiten

Qualitätskosten, -modelle: In der Literatur existieren zahlreiche Ansätze [31, 44, 65, 69, 70, 104, 143, 154, 157, 228, 256, 261, 265], Qualitätskosten zu analysieren und in Qualitätskostenmodelle zu überführen. Diese Qualitätskostenmodelle betrachten vorrangig das beratende Risiko- und Qualitätsmanagement in Produktionsprozessen und weniger den entwicklungsbegleitenden Softwaretest. Branchen- und testspezifische Kostenelemente, wie z. B. Testfallautomatisierungskosten, sind in den publizierten Qualitätskostenmodellen zum Teil nicht bzw. nicht differenziert enthalten. In einigen Arbeiten [195, 257, 263] haben Testkostenmodelle zwar den Vergleich von Testverfahren zum Ziel, aber wichtige kostenrelevante Aspekte, wie projektspezifische Ausführungsumgebungen und Werkzeuge für Tests oder Testautomatisierungen, werden unberücksichtigt gelassen. Weiterhin sind die Gegebenheiten von iterativ durchgeführten, phasenorientierten Vorgehensmodellen in den Kostenmodellen häufig nicht ausreichend [241] berücksichtig. Wiederum andere Modelle [97, 166, 226, 234] untersuchen Fehlervermeidungsmethoden und nicht die hier betrachteten Testverfahren oder dienen der Optimierung von Prüflingen [234], Testprozessen [225] oder der Wahl von zu testenden Systemkomponenten [50, 175].

Arbeiten zu speziellen Themenbereichen werden in den jeweiligen Kapiteln bei der Diskussion der Ergebnisse berücksichtigt.

1.4. Aufbau der Arbeit

Die Arbeit beginnt mit der Analyse der derzeitigen Entwicklungspraxis bei einem großen Fahrzeugteilezulieferer und der Darstellung des Erkenntnisstandes zur Thematik. Darauf aufbauend werden methodische Entwicklungen vorgenommen, um testinduzierte Qualitätskosten und Optimierungen quantifizieren zu können. Quantifizierungen auf der Basis von Daten aus der Entwicklungspraxis, ergänzt um Ergebnisse der Literatur, werden diskutiert, mit verwandten Untersuchungen verglichen und, soweit möglich, verallgemeinert. Im Einzelnen gliedert sich die Arbeit im Weiteren wie folgt:

Kapitel 2: Die Praxis der Entwicklung von eingebetteten Systemen im automobilen Umfeld wird analysiert und strukturiert dargestellt (Kapitel 2.1 und 2.2). Die Terminologien der Begriffe Fehler und Test werden eingeführt (Kapitel 2.2 und 2.3). Bedeutung und Eigenschaften des Testprozesses sowie dessen Einbettung in die Systementwicklung werden verdeutlicht (Kapitel 2.4). Kostenstrukturen im Testprozess, insbesondere die der Fehlerbeseitigung, werden aufgezeigt (Kapitel 2.5).

Kapitel 3: Der Erkenntnisstand zu Qualitätskosten und zur Fehleranalyse wird aufgearbeitet. In der Literatur beschriebene Qualitätskostenstrukturen und Qualitätskostenmodelle werden dargestellt (Kapitel 3.1) und im Hinblick auf den Beitrag für diese Arbeit bewertet. Da testspezifische Aspekte bisher unzureichend berücksichtigt sind, wird der Bedarf zur Entwicklung eines strukturabbildenden

Kostenmodells für testinduzierte Qualitätskosten verdeutlicht. Den Modellentwurf und spätere Optimierungen vorbereitend, werden Erfahrungsdaten zu Fehlerströmen, Fehlerbeseitigungskosten und -analysen sowie Fehlerklassen durch Literaturrecherchen ermittelt (Kapitel 3.2 bis 3.4).

Kapitel 4: Dieses Kapitel leitet Hilfsmittel her, die zur Optimierung von Strategien hinsichtlich Qualität und Kosten verwendet werden. Dazu werden zunächst die zu optimierenden, testinduzierten Qualitätskosten abgegrenzt (Kapitel 4.1). Anschließend wird der Optimierungsprozess beschrieben (Kapitel 4.2). Dieser fokussiert Kosteneinsparungen durch Fehlerfrüherkennung und Testautomatisierungen. Zur Ermittlung der Ist-Kostensituation und zur Simulation der Kostenreduktionen durch Strategieoptimierungen wird ein Kostenmodell für testinduzierte Qualitätskosten aufgestellt (Kapitel 4.3). Die Kosteneinsparungen durch Fehlerfrüherkennung sollen durch eine optimierte Auswahl von Testverfahren umgesetzt werden, wofür eine fehlerklassenorientierte Testverfahrensauswahl aufgestellt wird (Kapitel 4.4). Die somit möglichen und zu erwartenden Kosteneinsparungen sollen durch einen hergeleiteten Berechnungsalgorithmus abgeschätzt werden (Kapitel 4.5). Ein Verfahren zur Bewertung der Wirtschaftlichkeit von Testautomatisierungen wird in Kapitel 4.6 hergeleitet.

Kapitel 5: Zur Vorbereitung der Anwendung der Verfahren aus Kapitel 4 werden Daten von Entwicklungsprojekten charakterisiert (Kapitel 5.1) und Kostenschwerpunkte analysiert (Kapitel 5.2). Die Realitätstreue von geschätzten Kosten zur Fehlerbeseitigung als Eingangswerte für Kostensimulationen wird untersucht (Kapitel 5.3). Kostenmerkmale (Kapitel 5.5) und Fehlerverteilungen (Kapitel 5.6) werden ür spätere Quantifizierungen von Qualitätskosten erhoben und Einflussfaktoren auf das Kostenverhalten isoliert (Kapitel 5.4).

Kapitel 6: In diesem Kapitel werden die Verfahren zur Optimierung von Teststrategien aus Kapitel 4 mit den Erkenntnissen aus Kapitel 5 angewendet. Zunächst werden mögliche Kosteneinsparpotentiale durch Fehlerfrüherkennung quantifiziert (Kapitel 6.1). Am Beispiel eines Projektes werden Testverfahren aufgezeigt, die diese Einsparung realisieren (Kapitel 6.2). Die Effizienzoptimierung für ein mögliches Testverfahren (Kapitel 6.3) und die Wirtschaftlichkeit der in zwei Projekten durchgeführten Testautomatisierungen (Kapitel 6.4) werden analysiert.

Kapitel 7: Auf den Ergebnissen von Kapitel 6 aufbauend, wird mit Hilfe des in Kapitel 4.3 entwickelten Kostenmodells für testinduzierte Qualitätskosten und des vorgestellten Optimierungsprozesses (Kapitel 4.2) die Teststrategie eines Referenzprojektes optimiert (Kapitel 7.1). Dabei werden zunächst die Ist-Kosten quantifiziert, dann die Teststrategie durch die Testverfahrensauswahl, durch Reduktion von Lizenzkosten (als projektspezifischer Kostenschwerpunkt) und durch Testautomatisierungen optimiert. Anschließend werden die Kostensimulationen exemplarisch für Literaturwerte durchgeführt (Kapitel 7.2).

Kapitel 8: Abschließend werden der Gültigkeitsbereich der Untersuchungsergebnisse und die Allgemeingültigkeit der entwickelten Verfahren diskutiert sowie Konsequen-

zen für die Softwareentwicklung gezogen.

Kapitel 9: Im Ausblick wird weiterer Forschungsbedarf, der sich aus dieser Arbeit ergibt, aufgezeigt. Neben Vorschlägen zur Verbesserung der Datenarchivierung werden zwei zum thematischen Umfeld gehörende Problematiken, die Qualität von Testfällen und die Steigerung der Spezifikationsqualität durch ein halbformales Modell, mit Lösungsansätzen skizziert.

Kapitel 10: Die Arbeit schließt mit einer Zusammenfassung.

Anhang A: Detaillierte, formelbasierte Beschreibungen für Passagen der Qualitätskostenberechnung ergänzen den Hauptgedanken in Kapitel 4.3.

Anhang B: Formelbasierte Beschreibungen für Passagen der testzielorientierten Fehlerklassengewichtung ergänzen den Hauptgedanken in Kapitel 4.4.2.

Anhang C: Formelbasierte Beschreibungen werden für Passagen der Bestimmung von Testautomatisierungskosten in Kapitel 4.6.2 ergänzt.

Anhang D: Analysen zur Aufwandsverteilung auf einzelne Schritte der Fehlerbeseitigung werden für Kapitel 5.2.2 angefügt.

Anhang E: Ergebnisse der Analyse über die Realitätstreue von prospektiven Aufwandsschätzungen von Testern werden ergänzend zu Kapitel 5.3 dokumentiert.

Anhang F: Das Verfahren zur Berechnung der Kosteneinsparpotentiale (Kapitel 6.1) wird an einem Beispiel erläutert.

Anhang G: Das Verfahren zur Berechnung der testzielorientierten Fehlerklassengewichtung (Kapitel 6.2) wird an einem Beispiel demonstriert.

Anhang H: Die Ableitungen der Quantifizierung der Kostenelemente, die für die Ermittlung der testinduzierten Qualitätskosten für das Body Controller-Projekt BC1 sowie die Simulationen in Kapitel 7 erforderlich sind, werden beschrieben.

Anhang I: Tabellen mit detaillierten Untersuchungsergebnissen sind zur Einsicht hinterlegt.

2. Softwaretest von eingebetteten Systemen

In diesem Kapitel werden die Ergebnisse der Analyse des Softwaretests von eingebetteten Systemen dargestellt. Betrachtet wurde schwerpunktmäßig die derzeitige Entwicklungspraxis eines großen Fahrzeugteilezulieferers der Automobilindustrie. Die Entwicklungspraxis wird, aufgrund von Vergleichen mit Publikationen zu Einzelaspekten (z. B. [230], [49]), als branchenüblich angesehen.

Im Anschluss an eine Charakterisierung der eingebetteten Systeme in der Automobilindustrie wird die Struktur der Entwicklung abstrahiert dargestellt, die verwendete Terminologie zum Begriff *Fehler* erläutert und die verwendeten Begriffe zur Fehlerverarbeitung definiert. Es folgt die Darstellung von Aspekten der Qualitätssicherung im Testprozess. Eingangs wird die Einbettung der konstruktiven und analytischen Qualitätssicherung in die Softwareentwicklung dargelegt, dann werden Eigenschaften der Testfallerstellung und Testautomatisierung zusammengestellt. Den Abschluss bilden die Ergebnisse der Analyse zur Kostenstruktur mit Beschreibung von Kostenelementen und Kostenabhängigkeiten.

2.1. Eingebettete Systeme im Automobilbereich

Aufgaben: Eingebettete Systeme übernehmen meist Überwachungs-, Steuerungs- und Regelungsaufgaben [43] und verrichten dabei klar abgegrenzte Dienste [90, 177], die oft im sicherheitskritischen Kontext und in Echtzeit auszuführen sind [93].

Body Controller und Infotainment-Systeme: Von besonderer Bedeutung für diese Arbeit sind Body Controller und Infotainment-Systeme, da diese aufgrund des stetigen Innovations- und Optimierungsbedarfs einen bedeutenden Stellenwert in den automobilen Entwicklungen haben. Body Controller umfassen für gewöhnlich mehrere in Software implementierte Reglerfunktionen in einer Hardwarekomponente, z. B. die Steuerung von Blinker, Innenlicht, Außenlicht, Scheibenwischer, Fensterheber, Zentralverriegelung, Tempomaten, Abstandswarnsysteme. Infotainment-Systeme dagegen realisieren komplexe Unterhaltungs- und Informationsdienste in Automobilen, z. B. Navigations-, Kommunikations-, Multimedia- und Audio-Dienste.

Systemumgebung: *Eingebettete Systeme* sind Software-Hardware-Einheiten [7, 17, 83, 126], die in Gesamtsysteme integriert sind, über diese angesteuert werden [93] und somit oft keine eigene Nutzerschnittstelle aufweisen [93]. Damit sind sie typischerweise dem Nutzer nicht sichtbar [43]. Ausgaben erfolgen über Aktuatoren und führen zu Aktionen in der meist sehr komplexen Umwelt [93]. Als Umwelt

sind i. Allg. die mit dem betrachteten System interagierenden System(teil)e und in gewissen Fällen auch die Systemumgebungen anzusehen. Über Sensoren werden Umweltdaten gesammelt.

Produktionscharakteristika: Die Produktion der in dieser Arbeit betrachteten eingebetteten Systeme zeichnen sich durch folgende Charakteristika aus: hohe Stückzahlen, Variantenvielfalt, große Spezifikationsmenge und Fremdentwicklungen. Für detaillierte Analysen dieser Charakteristika sei auf die Untersuchungen von Fleischmann [92] verwiesen.

Produktentwicklung: Die Produktentwicklung in der Automobilindustrie ist nicht zuletzt aufgrund des Konkurrenzdruckes gekennzeichnet durch hohen Innovationsdruck, kurze Entwicklungszeiten und -zyklen, hohen Preisdruck und sehr hohe Qualitätsanforderungen [40, 43, 167, 215]. Die Entwicklung eingebetteter Systeme geschieht interdisziplinär, gemeinsam mit Hard- und Softwareentwicklern [105]. Die Produktentwicklung und insbesondere die Softwareentwicklung wurden auf diese besonderen Anforderungen angepasst [17, 177]. Seit den 1980er Jahren konnte die Systementwicklung von Projektstart bis zum Produktionsstart von etwas mehr als vier Jahre auf etwa drei Jahre verkürzt werden [230, 231]. Die Verkürzung wurde u. a. durch hochgradig parallele Arbeitsprozesse innerbetrieblich als auch zwischen den beteiligten Unternehmen erreicht [43, 48, 125].

2.2. Struktur der Systementwicklung

2.2.1. V-Modell

Vorgehensmodelle in der Softwareentwicklung: Softwareentwicklungsprozesse lassen sich durch Vorgehensmodelle beschreiben, strukturieren und steuern. Der Entwicklungsprozess wird durch Vorgehensmodelle in planbare, zeitlich und inhaltlich begrenzte Phasen unterteilt. Vorgehensmodelle unterstützen das Projektmanagement und können als Vertragsgrundlage, Arbeitsanleitung und Kommunikationsbasis dienen und sind somit letztlich der Qualitätssteigerung förderlich.

Charakteristika verschiedener Vorgehensmodelle: Im Laufe der letzten Jahrzehnte wurden verschiedene Vorgehensmodelle (weiter-)entwickelt, die sich in bestimmten Charakteristika unterschieden. Die Abarbeitung der Phasen und Arbeitsschritte kann z. B. in Abhängigkeit vom Vorgehensmodell sowohl sequentiell (z. B. Wasserfallmodell [107]) als auch ohne strikte zeitlich definierte Abfolge (z. B. V-Modell XT [219]) erfolgen. Weiterhin kann der Entwicklungszyklus einmal (z. B. Wasserfallmodell [107]) oder auch mehrmals (z. B. Spiralmodell [29]) durchlaufen werden. Sind mehrere vollständige Durchläufe möglich, können die Systemkomponenten in jedem Durchlauf verfeinert, optimiert oder um Funktionen erweitert werden (z. B. V-Modell XT [219]). Vorgehensmodelle sind in der Regel branchenunabhängig. In der Automobilindustrie wird vorrangig auf der Basis von V-Modellen entwickelt.

V-Modelle: Das V-Modell wurde seit der Einführung im Jahre 1986 immer wieder bedarfsorientiert weiterentwickelt [17] und liegt heute in verschiedenen Versionen vor [74, 220]. Die in dieser Arbeit untersuchten Automobilprojekte verwenden das V-Modell.

V-Modell XT: Die derzeit aktuelle Version ist das *V-Modell XT* (*Extreme Tailoring*) [219], welches sich durch höhere Flexibilität und Anpassbarkeit an individuelle Bedürfnisse auszeichnet. So können die Aktivitätsvorgaben für den Auftrag*nehmer* nun auch um Aktivitätsvorgaben für den Auftrag*geber* erweitert und dieser so in den Prozess einbezogen werden (z. B. für Spezifikation, Musterabgaben und Wartung [135]). Prozessdokumente durchlaufen einen Lebenszyklus und werden von Personen mit zugewiesenen Rollen bearbeitet. Eine zeitliche Abfolge der Arbeitsschritte ist nicht zwingend vorgeschrieben, so dass Tests frühzeitig vor Abschluss der Implementierung beginnen können. Das V-Modell XT sieht Produktzustandsmodelle zur analytischen Qualitätssicherung vor [39]. Sie definieren verschiedene Bearbeitungszustände für die Erstellung der Produkte (s. Abbildung 2.1). Die Entwicklung erfolgt im Bearbeitungszustand „*in Bearbeitung*“. Daran schließt sich ggf. eine eigenständige Qualitätssicherung durch ein Testteam an (Bearbeitungszustand: „*vorgelegt*“) [39]. Das Produkt wird erst nach Sicherstellung der Qualität mit dem Bearbeitungszustand „*fertig gestellt*“gekennzeichnet [39]. Das Produktzustandsmodell stellt somit sicher, dass Aussagen zur erreichten Korrektheit von Produkten möglich sind, ohne dass die Reihenfolge der Fertigstellung vorgeschrieben ist.

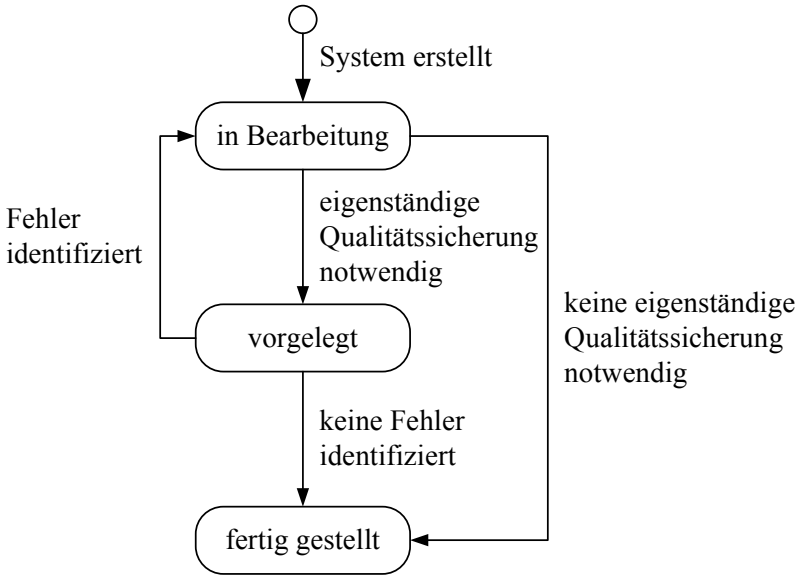


Abbildung 2.1.: Bearbeitungszustände von (Zwischen-)Produkten im V-Modell XT [39]

Abstraktionsebenen zur Softwareerstellung (linke Seite des V-Modells): Das Durchlaufen des V-Modells beinhaltet Aktionen, die wie in Abbildung 2.2 dargestellt

werden können. Es werden in verschiedenen Abstraktionsebenen Anforderungen an das Fahrzeug, das Fahrzeugsubsystem, das Steuergerät, den Mikrocontroller bis hinunter zur Software definiert. Hierbei werden die Anforderungen an die zu erstellende Software top-down in der Abfolge der Abstraktionsebenen zunehmend verfeinert bis zur Implementierung. Anders betrachtet, folgt auf den logischen und technischen Entwurf letztendlich die Softwareentwicklung. Von dieser ist die Implementierung die letzte Aktivität. Die Ergebnisse der hier genannten Aktivitäten der Softwareentwicklung (Anforderungsdefinition, Entwurf und Implementierung) sind die im V-Modell definierten (Zwischen-)Produkte: Spezifikation, Entwurf und Quelltext. Das V-Modell beschreibt auf der linken Seite der V-förmigen Darstellung also den Prozess der Softwareerstellung, der auch von Entwicklern durchgeführte, fehlervermeidende (konstruktive) und fehleridentifizierende (analysierende) Qualitätssicherungsmaßnahmen umfasst. Zu den analysierenden Qualitätssicherungsmaßnahmen gehören nach Fagan auch der von Entwicklern ausgeführte Unit-Test sowie Quelltextinspektionen [86].

Abstraktionsebenen zum Softwaretestprozess (rechte Seite des V-Modells): Der bottom-up integrierende Testprozess befindet sich auf der rechten, aufsteigenden Seite des V-Modells. Er definiert Testaktivitäten für links vorgenommene Abstraktionsstufen. Den Abstraktionsstufen werden also ebenenweise Testverfahren zugeordnet. Tester erstellen Testfälle zu den Spezifikationen der jeweiligen Abstraktionsebene (s. Abbildung 2.2) und spezifisch für die ausgewählten Testverfahren. Beispielsweise werden Testfälle für den Fahrzeugintegrationstest nach Abbildung 2.2 auf der Basis des *Entwurfs der logischen Systemarchitektur* erstellt. Die Ausführung der Testfälle geschieht meist nach dem Bottom-Up-Prinzip [162,230], bei dem zunächst die Einzelfunktionen der untersten Ebene und anschließend schrittweise aggregierte Funktionen bis hin zum Gesamtsystem getestet werden. So nimmt der Detaillierungsgrad der Sicht auf das zu testende System analog zum Detaillierungsgrad von Spezifikationen und Entwürfen ab.

2.2.2. Musterphasen

Entwicklung in Musterphasen: Der Entwicklungsprozess für eingebettete Systeme im Automobilbereich untergliedert sich in mehrere *Musterphasen*, wie *Konzeptentwicklung, Serienentwicklung* und *Serienvorbereitung* [43, 230] (s. Abbildung 2.2). Der Fahrzeugteilezulieferer übergibt am Ende jeder der drei Musterphasen dem Fahrzeughersteller ein *Muster* des entwickelten Produkts im entsprechenden Entwicklungsstand, ein A-, B- bzw. C-Muster [17,230]. Der Fahrzeughersteller nimmt das Muster durch einen Akzeptanztest ab.

Konzeptentwicklung: In der ersten Musterphase nach Projektstart, der *Konzeptentwicklung*, werden die gewünschten funktionalen Eigenschaften erarbeitet und deren Machbarkeit geprüft [43, 125, 167]. Ziel der Musterphase ist die Fertigung eines *Vorentwicklungsprototypen* (*A-Muster*), das den funktionalen Anforderungen des Fahrzeugherstellers entspricht.

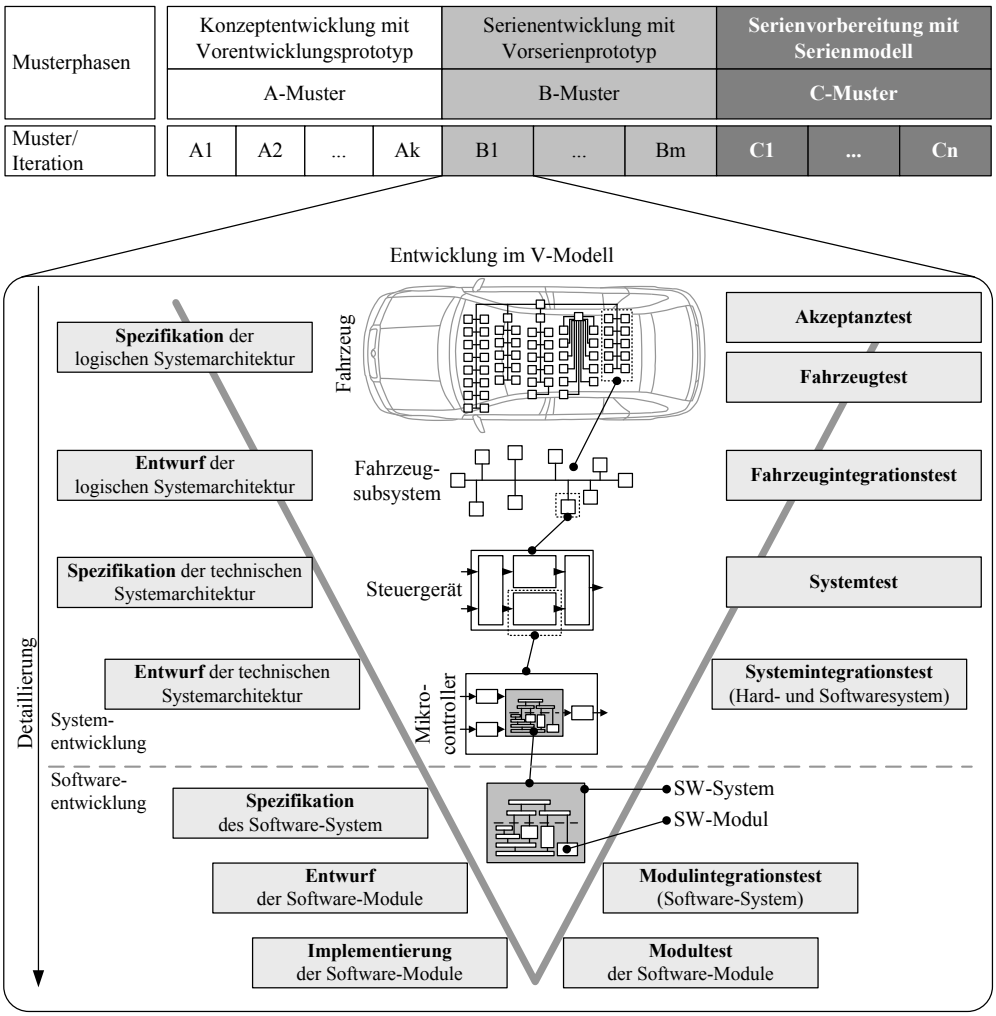


Abbildung 2.2.: Iterativer Entwicklungsprozess zur Entwicklung von elektronischen Geräten und Software in der Automobilindustrie unter Verwendung des V-Modells mit Betrachtung von Abstraktionsebenen (unter Einbeziehung von [17])

Serienentwicklung: In der zweiten Musterphase, der *Serienentwicklung*, liegt der Fokus auf der Optimierung der Qualitätseigenschaften wie Geräteabmessungen und Produktqualität sowie Produktionskosten [43, 125, 167]. Das Ergebnis eines kompletten Durchlaufs durch die Musterphase ist ein oft handgefertigter *Vorserienprototyp* (*B-Muster*) (s. Abbildung 2.2), der bereits die Qualitätseigenschaften des späteren Serienprodukts aufweist [167].

Serienvorbereitung: Die *Serienvorbereitung* beinhaltet eine sogenannte *Applikationsphase*, in der die Fahrzeugteilezulieferer die weitreichend konfigurierbaren Steuergeräte (sog. *Plattformen* [230]) auf das gewünschte Funktionsspektrum der

Fahrzeughersteller anpassen [17, 43, 125, 216, 236, 272]. Beispielsweise können erweiternde Zusatzfunktionen als Sonderwünsche hinzugefügt werden. Fahrzeughersteller können aber auch, z. B. aus Kostengründen, auf den vollen Funktionsumfang verzichten und implementierte Funktionen deaktivieren lassen. Vor allem bei Nutzfahrzeugen ist gelegentlich eine Anpassung des fertigen Produkts durch Parametrisierung an Normen, Gesetze oder Kundenwünsche notwendig, beispielsweise das Einstellen der Höchstgeschwindigkeit bei für den Export bestimmten Nutzfahrzeugen [43]. Das Ergebnis dieser Musterphase ist ein *Serienmodell* (*C-Muster*), dessen Serienproduktion anschließend startet (Start of Produktion, SOP) [43, 125, 167].

2.2.3. Iterationen

Entwicklung in Iterationen: Jede Musterphase (Konzept, Serienentwicklung, Serienvorbereitung) wird in der Automobilindustrie in der Regel mehrmals durchlaufen [162, 230]. Das Durchlaufen wird als *Iteration* bezeichnet (A1 bis Ak, B1 bis Bm und C1 bis Cn in Abbildung 2.2). In den Iterationen wird das System auf Basis der Erfahrungen aus vorangegangenen Iterationen weiterentwickelt. Mit jeder Iteration entsteht ein Produktentwurf, z. B. ein A1-, A2-Muster. Mit diesem validiert der Auftraggeber seine Produktwünsche und korrigiert, verfeinert und ergänzt ggf. seine Anforderungen [43, 125, 167]. Es wird ein Lösungsentwurf entwickelt, implementiert und anschließend getestet. In den betrachteten Entwicklungen wird das Funktionsspektrum zudem in der Regel von Iteration zu Iteration sukzessive erweitert. Die iterative Anwendung lässt Parallelisierungen zu, so kann beispielsweise parallel zur Implementierung einer Iteration bereits der Entwurf der nächsten Iteration erstellt werden. In den Iterationen müssen jedoch nicht zwingend alle Abstraktionsebenen durchlaufen werden. So wird z. B. ein Akzeptanztest zu jeder Musterabnahme, aber nicht in jeder Iteration durchgeführt.

Inkrementelle Entwicklungen: Eine inkrementelle Entwicklung, bei der z. B. verschiedene Teile des Systems unabhängig voneinander, parallel entwickelt und direkt bei Fertigstellung ins Gesamtsystem integriert werden [74], findet bei den untersuchten automobilen Entwicklungen nicht statt.

2.2.4. Testphasen

Testphasen: Den Phasen der Softwareerstellung schließen sich auf dem rechten, aufsteigenden Zweig des V-Modells die Testphasen an, welche im Zentrum der Betrachtung in dieser Arbeit stehen (s. Abbildung 2.3). In diesen Testphasen wird die zu testende Software von Testteams durch dynamische Qualitätssicherung auf Systemanomalien überprüft. Es werden folgende, für diese Arbeit wichtige Testphasen unterschieden [176]: *Modultest*[1]*, Modulintegrationstest, Systemintegrationstest, Systemtest, Fahrzeugintegrationstest, Fahrzeugtest* und *Akzeptanztest*. In den Testphasen werden entsprechend der erreichten Systemabstraktion unterschiedliche Testziele adressiert,

[1] Häufig auch Funktionstest genannt, z. B. in [86].

die jeweils auf ausgewählte Eigenschaften der entwickelten Software fokussieren und dementsprechend auf unterschiedlichen Verfahren basieren:

- *Modultest*: Als Modultest wird in dieser Arbeit das Ausführen von strukturorientierten Verfahren verstanden, die durch funktionsorientierte Verfahren ergänzt werden können. Für die Testdurchführung werden Treiber und Platzhalter eingesetzt [149, 176].
- *Modulintegrationstest*: Das Zusammenspiel von Softwaremodulen wird sichergestellt. Für die Testdurchführung werden Treiber und Platzhalter eingesetzt [149, 176].
- *Systemintegrationstest*: (Hardware-/Software-)Module werden in der Automobilindustrie bottom-up [162, 230] zu einem Gesamtsystem zusammengefasst. Das spezifikationsgemäße Zusammenwirken von Hard- und Software sowie von Modulen untereinander an den Schnittstellen wird sichergestellt [149, 176]. Für die Testdurchführung werden Treiber und Platzhalter eingesetzt [149, 176].
- *Systemtest*: Das Gesamtsystem wird i. Allg. funktional getestet, z. B. anhand des Systemverhaltens. Das Hauptziel ist es, Fehlerursachen aus Systemanalyse und Spezifikation zu finden [149, 176]. Weiterhin wird die Einhaltung nichtfunktionaler Anforderungen überprüft [149, 176]. Dazu gehören beispielsweise Zeitanforderungen, Speicherauslastung, Robustheit oder auch Stromverbrauch. Systemtests umfassen somit funktionsorientierte Tests und nicht-funktionale wie Stress-, Last- und Performanztests, Konfigurationstests und Sicherheitstests [15].
- *Fahrzeugintegrationstest*: Das Steuergerät (System) wird in das Fahrzeug eingebaut [230]. Der Fahrzeugintegrationstest wird analog zum Systemintegrationstest ausgeführt [230]. Das Kommunikationsverhalten wird geprüft, z. B. an einem mit anderen Steuergeräten gemeinsam genutzten Bus.
- *Fahrzeugtest*: Insbesondere physikalische Eigenschaften wie Größe des Steuergerätes oder Positionen von Funkantennen sowie das exemplarische Zusammenspiel aller im Automobil verbauten Komponenten werden im Fahrzeugtest überprüft [230].
- *Akzeptanztest:* Der Kunde (Fahrzeughersteller) testet, ob das entwickelte System mit seinen spezifizierten Anforderungen übereinstimmt (Validation) [149, 176]. Der Akzeptanztest entspricht der Abnahme des entwickelten Systems [149,176].

Zuständigkeit und Kostenzuordnung für Testphasen: Vom Modultest bis Systemtest ist von einer Durchführung beim Hersteller auszugehen. Fahrzeugintegrationstest und Fahrzeugtest finden beim Hersteller oder beim Kunden statt, der Akzeptanztest beim Kunden (im Falle von eingebetteten Systemen in der Automobilindustrie also beim Fahrzeughersteller). Für die beim Kunden durchgeführten Tests liegen Vorgehen und Testkosten – sofern nicht vertraglich anders geregelt – in der Zuständigkeit des Kunden. Da dem Hersteller lediglich das identifizierte Fehlverhalten gemeldet wird, ist der Testvorgang beim Kunden nicht Gegenstand der Untersuchungen dieser Arbeit.

2.2.5. Feldeinsatz

An die Musterphasen (s. Kapitel 2.2.2) schließt sich mit Produkteinführung als gewissermaßen letzte Projektphase der sog. *Feldeinsatz* an. In der Regel werden nicht alle Systemfehler durch den Testprozess identifiziert und vor Produkteinführung beseitigt. Infolgedessen kann ein Fehlverhalten nach der Produkteinführung spontan als sog. *Feldfehler* bei der Produktnutzung auftreten. Feldfehler werden dem Hersteller gemeldet und sind dann zu beseitigen. Zudem bestehen Wechselwirkungen zwischen Testkosten und Fehlerbeseitigungskosten von Feldfehlern. Daher wird später in der Arbeit der Feldeinsatz strukturell als Testphase aufgefasst, wie auch in Abbildung 2.3 dargestellt, und die Fehlerbeseitigungskosten werden als den Qualitätskosten des Testprozesses zugehörig betrachtet.

2.2.6. Testprozess

Testphasen im Testprozess: Der Testprozess umfasst hier zunächst die Testphasen *Modultest **[109]**, Modulintegrationstest, Systemintegrationstest, Systemtest, Fahrzeugintegrationstest, Fahrzeugtest* und *Akzeptanztest*.

Feldeinsatz als weitere Testphase: Wie in Kapitel 2.2.5 ausgeführt, werden auch nach Produkteinführung Fehlverhalten identifiziert. Im Hinblick auf Analysen zu der Fehlerbeseitigung wird der Feldeinsatz hier als weitere Testphase aufgefasst, wie auch Abbildung 2.3 aufzeigt.

2.3. Softwarefehler

2.3.1. Terminologie Fehler

Im Softwaretest wird ein Abweichen des Systemzustands oder des Verhaltens vom Erwarteten als Anomalie bezeichnet [138]. Eine Anomalie kann aufgrund der Ursachenvielfalt facettenreich ausgeprägt sein [147, 169, 170]. Dies ist wohl der Grund, weshalb sich in der Literatur unterschiedliche Definitionen und Begriffe im Bereich Softwarefehler finden lassen [35]. In der deutschen Sprache wird der Begriff *Fehler* häufig allumfassend und somit unpräzise verwendet:

Ein Entwickler macht einen (1) Fehler und hinterlässt somit einen (2) Fehler im Programmquelltext, der zu einem (3) Fehler im Programmzustand führen kann, der wiederum einen (4) Fehler im Systemverhalten oder in den berechneten Ergebnissen verursachen kann.

Das Beispiel verdeutlicht, dass ein differenzierter Fehlerbegriff zu empfehlen ist.

Irrtum, Fehlerursache, Fehlerzustand, Versagen/Ausfall: Auf Basis der englischen, bei Lapie [169], Lee et al. [170] und Johnson [147] verwendeten Begriffe *mistake, fault, error* und *failure* ergibt sich mit den deutschen Übersetzungen aus [169]

folgende Unterscheidung:

Ein Entwickler begeht einen (1) Irrtum (engl.: mistake) und hinterlässt somit eine (2) Fehlerursache (engl.: fault), die zu einem (3) Fehlerzustand (engl.: error) führen kann, der wiederum ein(en) (4) Versagen/Ausfall (engl.: failure) des Systems verursachen kann.

Der Begriff *failure* wird im Deutschen differenziert betrachtet. Es werden zeitweise Fehlfunktionen (*Versagen*) und dauerhaftes Nicht-Funktionieren (*Ausfall*) unterschieden. In dieser Arbeit wird vorrangig das Softwaresystem betrachtet, weshalb der häufiger eintretende Fall des *Versagens* stellvertretend für beide Ausprägungen verwendet wird.

Ursache-Wirkungs-Kette: Aus dem Beispiel sowie der Literatur [147,169] wird deutlich, dass die Begriffe Irrtum, Fehlerursache, Fehlerzustand und Versagen in einem kausalen Zusammenhang, einer Ursache-Wirkungskette, stehen:

- Ein Irrtum führt zu einer Fehlerursache,
- eine Fehlerursache zu einem Fehlerzustand,
- ein Fehlerzustand zu einem Versagen.

Die Ursache-Wirkungskette kann, muss aber nicht zwingend vollständig durchlaufen werden [147, 169] und kann jederzeit durchbrochen werden. So kann ein Programmierer eine Fehlerursache im Programmquelltext hinterlassen haben, ohne dass die Fehlerursache durch eine Programmausführung aktiviert wird (z. B. Fehler in selten oder niemals ausgeführten Programmstellen).

Fehlverhalten: Im Hinblick auf die intendierte Optimierung von Teststrategien ist im Allg. die Unterscheidung von Fehlerzustand und Versagen nicht erforderlich. Entscheidend ist, dass eine Fehlerursache zu einer Anomalie geführt hat. Daher werden Fehlerzustand und Versagen zum Begriff *Fehlverhalten* zusammengefasst.

Fehlerfortpflanzung: Ein Fehlverhalten kann die Fehlerursache für einen Folgefehler sein, diese Eigenschaft wird *Fehlerfortpflanzung* genannt [26]. So ist häufig nicht immer eindeutig der Anfang einer Ursache-Wirkungskette zu identifizieren.

2.3.2. Fehlerentstehung

Phasen der Fehlerentstehung: Die softwarebezogenen Irrtümer und Fehlerursachen entstehen im V-Modell hauptsächlich während der Softwareentwicklung [2], also auf dem linken, absteigenden Zweig (s. Abbildung 2.2 und 2.3). In dieser Arbeit werden drei Projektphasen der Softwareentwicklung unterschieden:

[2] Eine Ausnahme bilden Fehlerursachen, die durch Fehlerbeseitigungen entstehen. Nach Analysen von Basili [11] entstehen 6% der Fehlerursachen durch Fehlerbeseitigungen.

- *Anforderungsdefinition*: Anforderungen beschreiben zu realisierende Eigenschaften des zu entwickelnden Systems, die in Spezifikationen dokumentiert werden. Das Ergebnis der Anforderungsdefinition soll eine den Qualitätskriterien in [221] genügende Spezifikation sein.
- *Entwurf*: Das zu implementierende System wird vom Groben ins Feine modelliert. Hierzu gehören u. a. Architektur, Schnittstellen, Verhalten [230].
- *Implementierung*: Das spezifizierte und entworfene System wird realisiert, z. B. durch das Erstellen von Programmquelltext.

Diese Projektphasen werden hier als *Phasen der Fehlerentstehung* bezeichnet.

Fehlerentstehung bei Fehlerkorrekturen: Prinzipiell können erneut Irrtümer und Fehlerursachen im Zuge der Fehlerkorrektur, somit nach der Implementierungsphase, entstehen. Diese Irrtümer und Fehlerursachen werden primär durch Korrekturtests und Regressionstests [176] identifiziert und sind daher nicht im Fokus der hier betrachteten Fehlerentstehung.

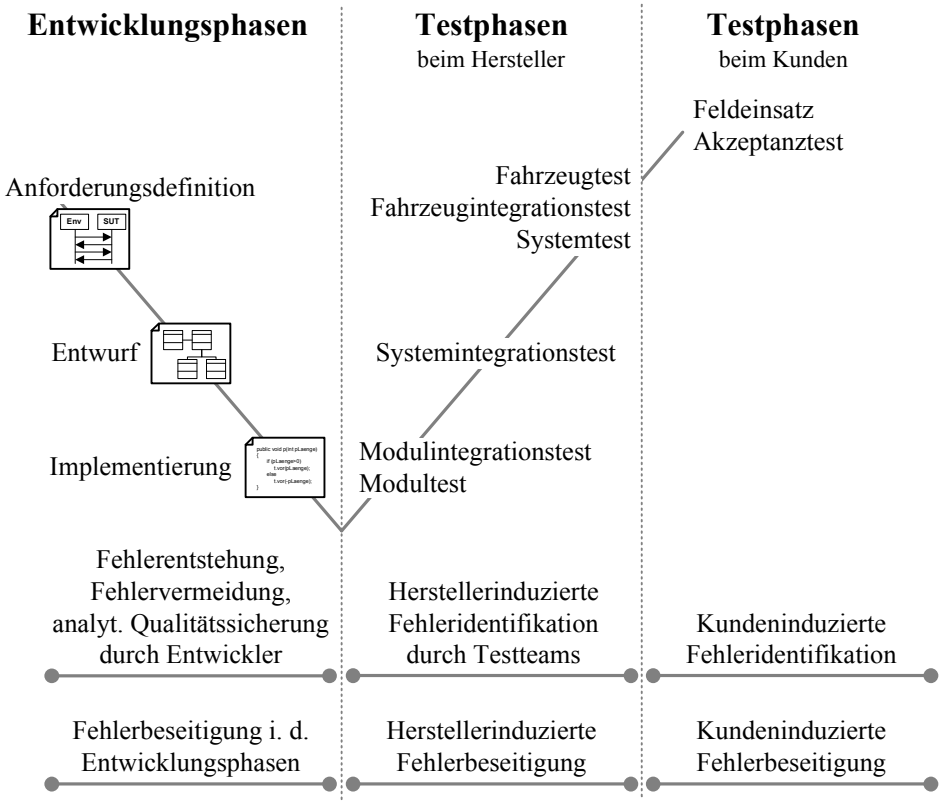


Abbildung 2.3.: Einordnung von Fehlerentstehung, Fehleridentifikation und Fehlerbeseitigung in das V-Modell

2.3.3. Fehleridentifikation

Das Feststellen des Abweichens von den erwarteten Ergebnissen wird hier als *Fehleridentifikation* bezeichnet.

(Test-)Phasen der Fehleridentifikation: Im Testprozess kann die Durchführung der Testverfahren Fehlverhalten auslösen. Ebenso können nach Produkteinführung bei Produktnutzung Fehlverhalten ausgelöst werden. In beiden Fällen wird beim Hersteller ein Fehlerbeseitigungsprozess angestoßen. Dies hat in Kapitel 2.2.5 bereits dazu geführt, den Feldeinsatz als Testphase zu interpretieren. Demzufolge werden die Testphasen von Modultest bis Akzeptanztest ergänzt um den Feldeinsatz als *Phasen der Fehleridentifikation* bezeichnet.

2.3.4. Fehlermanagement

Formalisierter Umgang mit Fehlermeldungen: Ein Fehlverhalten kann während der gesamten Projektlaufzeit sowie während des gesamten Lebenszyklus der Software identifiziert werden [39]. Wenn das betrachtete Produkt einen bestimmten Fertigstellungsgrad erreicht hat [39], wird ein formalisierter Umgang mit den Fehlermeldungen notwendig [39]. Das erfolgt i. Allg. dann, wenn die Verantwortlichkeit der Qualitätssicherung von den Entwicklern auf die Tester übergeht.

Fehlerdokumentation: Mit einsetzendem Fehlermanagement werden Fehlermeldungen zentral gesammelt, bewertet und der Fehlerbeseitigungsprozess dokumentiert [39]. Auf der Basis der vorliegenden Informationen wird über das weitere Vorgehen entschieden [39]. Die Qualität der Beschreibung des Fehlverhaltens beeinflusst den Vorgang der Fehlerbeseitigung. Es gibt zahlreiche Vorschläge [271] sowie auch die IEEE-Norm 1044-1993 [138], die festlegen, welche Informationen unbedingt zu erfassen sind. In der untersuchten Praxis ist die Erfassungsrealität jedoch von diesen Vorgaben deutlich entfernt.

Fehlerdatenbanken: Grundsätzlich können Eintragungen zu Fehlerursachen und Fehlverhalten von allen an der Entwicklung beteiligten Personen vorgenommen werden [39]. Das Fehlermanagement wird häufig in (Fehler-)Datenbanken protokolliert. Die Fehlerdatenbanken sind also als Realisierung des Fehlermanagements anzusehen. Fehlerdatenbanken erlauben aufgrund ihrer Strukturierung einen schnellen Zugriff auf die zur Fehlerbearbeitung erforderlichen Informationen wie Angaben zum Testfortschritt oder zu Aufwandsabschätzungen [24].

2.3.5. Fehlerbeseitigung

Fehlerbeseitigung in Abhängigkeit von Fehleranalyse: Für ein identifiziertes Fehlverhalten wird eine Fehlermeldung erstellt (Fehlermanagement s. Kapitel 2.3.4) und eine Fehleranalyse eingeleitet. Die Fehleranalyse kann eine Kosten-Nutzen-Analyse umfassen, in Abhängigkeit derer eine Fehlerkorrektur mit anschließendem Korrekturtest initiiert wird oder die Fehlerursache im Sinne der Opportunität im

System belassen werden kann [271].

Fehlerbeseitigung bei Identifizierung: Die Analyse des Entwicklungsprozesses bei einem großen Fahrzeugteilezulieferer hat ergeben, dass die Fehlerbearbeitung fast ausschließlich unmittelbar nach der Fehleridentifikation aufgenommen wird. In der Literatur wurden keine Hinweise gefunden, die auf generell anderes Vorgehen schließen lassen. Somit wird im Rahmen der Arbeit durchgehend die Fehlerbeseitigung in der Testphase der Fehleridentifikation angenommen.

Analyse Fehlerbearbeitungsprozess, Beschreibung Lebenszyklus einer Fehlermeldung: In der Literatur existieren verschiedene Lebenszyklen [140, 196, 245, 271] für Fehlermeldungen, die allesamt ihre Existenz durch die Optimierung auf bestimmte Aspekte begründen [271]. Für die intendierte Qualitätskostenquantifizierung der betrachteten Fehlerbeseitigung sind diese nicht optimal. So wird auf der Basis der Abläufe des Fehlerbeseitigungsprozesses und der Fehlerdokumentation in den Datenbanken der untersuchten Projekte sowie existierenden Fehlerdatenbankkonzepten nach Literatur [140, 196, 241, 245] ein optimierter Lebenszyklus einer Fehlermeldung entworfen und, wie in Abbildung 2.4 gezeigt, auf einen Zustandsraum abgebildet.

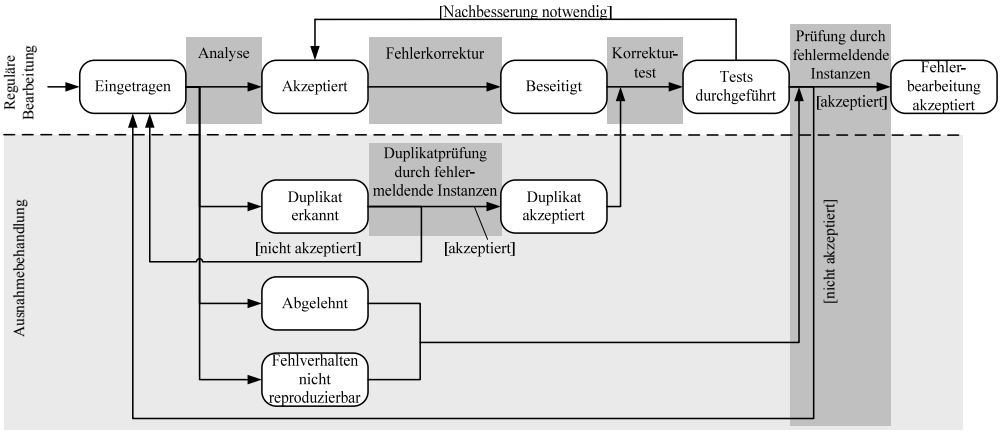


Abbildung 2.4.: Lebenszyklus einer Fehlermeldung – Entwurf eines Zustandsraums, optimiert auf Qualitätskostenquantifizierung für automobile Softwareentwicklungsprojekte

Nach der Fehleridentifikation wird eine Fehlermeldung in die Fehlerdatenbank eingetragen. Die Fehlermeldung befindet sich dadurch in dem einzigen initialen Zustand *Eingetragen*. In der ersten Bearbeitung soll das bemängelte Verhalten des eingebetteten Systems reproduziert und analysiert werden. Die vier möglichen Analyseergebnisse bestimmen den Übergang in einen der vier Zustände:

- Konnte das Fehlverhalten reproduziert werden und wird das Fehlverhalten erstmals beschrieben, so wird die Fehlermeldung als *Akzeptiert* gekennzeichnet, so-

fern kein Sonderfall (siehe letzter Aufzählungspunkt) vorliegt, der eine Ablehnung bewirkt.
- Konnte das Fehlverhalten reproduziert werden und die analysierte Fehlermeldung beschreibt ein Fehlverhalten, das bereits durch eine andere Fehlermeldung beschrieben wurde, so liegt ein Duplikat vor. Die Fehlermeldung wird durch den Zustand *Duplikat erkannt* als Duplikat gekennzeichnet.

Zur Bearbeitungslogik von Duplikaten gehört: Gegenseitige Referenzen zwischen Duplikat und bereits eingetragener Fehlermeldung werden erstellt. Als nächste Bearbeitung erfolgt eine Duplikatsprüfung durch die fehlermeldende Instanz. Akzeptiert diese die Diagnose, so folgt der Zustand *Duplikat akzeptiert*, andernfalls wird die Fehlermeldung mit neuen, zusätzlichen Fehlerinformationen wieder in den Zustand *Eingetragen* zurückversetzt. Für akzeptierte Duplikate wird keine Fehlerbeseitigung durchgeführt. Allerdings werden die zu den Duplikaten gehörenden Testfälle nach der Fehlerbeseitigung ausgeführt, um das spezifikationsgemäße Verhalten für diese Testfälle zu prüfen. Nach dem Test erreichen die Duplikate dadurch den Zustand *Tests durchgeführt*. Im verbleibenden Lebenszyklus wird das Duplikat wie eine reguläre Fehlermeldung behandelt.

- Ist das beschriebene Fehlverhalten nicht reproduzierbar, so wird in den Zustand *Fehlverhalten nicht reproduzierbar* übergegangen. Dies dokumentiert einen Kommunikationsbedarf mit der fehlermeldenden Instanz, z. B. dem Automobilhersteller, zum Zwecke der Problemlösung. Nach erfolgter Rücksprache wird die Fehlermeldung entweder nicht weiter bearbeitet (Zustand: *Fehlerbearbeitung akzeptiert*) oder mit zusätzlichen Informationen erneut analysiert (Zustand: *Eingetragen*).
- Ist das vermeintliche Fehlverhalten (a) nicht auf das betrachtete System zurückzuführen oder (b) spezifikationsgemäß oder (c) ergibt eine Kosten-Nutzen-Analyse, dass keine Fehlerbeseitigung eingeleitet werden soll, so wird die Fehlermeldung als *Abgelehnt* gekennzeichnet und nach Absprache mit der fehlermeldenden Instanz als *Fehlerbearbeitung akzeptiert* markiert. Ein Beispiel für ein Fehlverhalten, das nicht durch das Entwicklungssystem bedingt ist, kann das Fehlverhalten aufgrund einer nicht vorschriftsgemäßen Systemkonfiguration (z. B. Konfigurationsdatei) sein. Die Systemkonfiguration ist als externe Fehlerursache anzusehen, weshalb die Implementierung des Systems an sich nicht geändert werden muss.

Für eine als *Akzeptiert* markierte Fehlermeldung wird eine Fehlerkorrektur eingeleitet. Sie umfasst zunächst eine Fehlerlokalisierung [3] und anschließend eine Systemänderung (z. B. Implementierung) zur Herstellung einer Konformität mit der Spezifikation. Nach der Fehlerbeseitigung wird der Fehlermeldung der Zustand *Beseitigt* zugewiesen.

Anschließend wird das spezifikationsgemäße Funktionieren durch ausgewählte Korrekturtests überprüft: Ausgeführt werden die fehleridentifizierenden Testfälle der

[3] Oft auch als Debuggen bezeichnet.

Fehlermeldung sowie die aller ggf. vorhandenen Duplikate. Zusätzlich sollten Regressionstests durchgeführt werden. Nach erfolgreicher Ausführung beider Testarten geht die Fehlermeldung in den Zustand *Tests durchgeführt* über. Wird Fehlverhalten in einem der Tests beobachtet, so muss eine erneute Fehlerbeseitigung stattfinden, wofür die Fehlermeldung ggf. mit zusätzlichen Fehlerinformationen in den Zustand *Akzeptiert* zurückgesetzt wird.

Der abschließende Schritt der Fehlerbeseitigung ist die Freigabe der Fehlermeldung und deren Duplikate durch die fehlermeldenden Instanzen, die bestätigen, dass das Fehlverhalten nicht mehr beobachtet wurde. Es wird in den finalen Zustand *Fehlerbearbeitung akzeptiert* übergegangen.

2.4. Qualitätssicherung

2.4.1. Konstruktive und analytische Qualitätssicherung

Aufgaben der Qualitätssicherung: In den Projektphasen Anforderungsdefinition, Entwurf und Implementierung können aufgrund von Irrtümern Fehlerursachen entstehen (s. Kapitel 2.3.2), die es zu vermeiden bzw. zu beseitigen gilt. Eine Qualitätssicherung soll sicherstellen, dass das für die Softwaresysteme angestrebte Qualitätsniveau erreicht wird [8]. Hierzu tragen Fehlervermeidung (konstruktive Qualitätssicherung) und Fehlerbeseitigung in Folge der analytischen Qualitätssicherung bei [8].

Konstruktive Qualitätssicherung: Die konstruktive Qualitätssicherung kann bereits frühzeitig mit Beginn der Projektplanung einsetzen und basiert auf der Auswahl von Konstruktionsprinzipien, dem Einsatz von formalen Verfahren, unterstützenden Werkzeugen und Vorgehensmodellen [8]. Die konstruktive Qualitätssicherung wird durch die analytische Qualitätssicherung ergänzt [176].

Analytische Qualitätssicherung: Die analytische Qualitätssicherung strebt die Identifikation entstandener Fehlerursachen an. Aufgrund des Vorgehens werden statische und dynamische Prüfungen unterschieden [176]. Statische Prüfungen bedingen nicht die Ausführung des Systems und können daher auf alle Entwicklungsprodukte angewendet werden, insbesondere auch auf Spezifikationen und Entwurfsdokumente [176]. Dynamische Prüfungen, auch *dynamische Qualitätssicherung* oder *Tests* genannt, verlangen hingegen die Ausführung des Systems, wobei geprüft wird, ob sich das zu testende System gemäß seiner Spezifikation verhält [176].

Dynamische Tests: In der Prüfung eingebetteter Systeme gelten Tests als Stand der Praxis. Ein Grund ist deren universelle und vergleichsweise einfache Anwendung [176]. Jedoch können Tests aufgrund der kombinatorischen Vielfalt der Systemeingaben i. Allg. das System nur stichprobenartig überprüfen und somit keinen Beweis für die Fehlerfreiheit liefern [71,176]. Vielmehr dienen Tests der Identifikation von Fehlverhalten [71], von dem auf die Fehlerursachen geschlossen werden muss. Die Testverfahren können einerseits anhand der Abstraktionsebene (s. Kapitel 2.3.2)

als auch anhand der Testfallherleitung klassifiziert werden [15, 151, 199]. Die Abstraktionsebene differenziert entsprechend der Sicht auf das System und unterscheidet somit beispielsweise Modul- und Systemtest. Das Vorgehen zur Testfallherleitung unterscheidet im Wesentlichen funktionsorientierte und strukturorientierte Testverfahren. Funktionsorientierte Testverfahren leiten Testfälle aus der Spezifikation ab, strukturorientierte Testverfahren zusätzlich aus der Struktur des Programmquelltextes. Strukturorientierte Testverfahren dienen somit als Ergänzung zu funktionsorientierten Testverfahren, da sie unerwartete Konstrukte im Programmquelltext, wie nicht erreichbare Programmteile, komplexe Abfragelogiken oder Sonderbehandlungen spezieller Eingabewerte, testen [176].

Zeitliche Abfolge: Aus den genannten Eigenschaften der konstruktiven und analytischen Qualitätssicherung ist zu erkennen, dass die konstruktive Qualitätssicherung frühzeitig, nämlich zeitgleich zur Abarbeitung der Arbeitsschritte in den Projektphasen Anforderungsdefinition, Entwurf und Implementierung, angewendet werden kann [37]. Die analytische Qualitätssicherung wird auf die (Teil-)Produkte der genannten Arbeitsschritte angewendet und ist daher der konstruktiven Qualitätssicherung zeitlich nachgelagert [8].

Abhängigkeit: In der automobilen Praxis wird die konstruktive Qualitätssicherung erst ansatzweise umgesetzt. So kommt der analytischen Qualitätssicherung eine große Bedeutung zu und sie bildet einen Kostenschwerpunkt [8, 15, 34, 86, 145, 199, 215]. Da die Kosten für die Fehlerbeseitigung i. d. R. mit Projektfortschritt ansteigen, ist eine frühe Fehlerbeseitigung und vor allem eine Fehlervermeidung entschieden anzustreben [28]. Zwar erübrigt eine intensive, konstruktive Qualitätssicherung die analytische Qualitätssicherung nicht [176]. Jedoch können deren Umfang und Zielsetzung verändert werden [241]. Durch Fehlervermeidung und frühe Fehlerbeseitigung ergibt sich primär eine Verlagerung der Kosten in die frühen Projektphasen. Es wird jedoch erwartet, dass eine Qualitätssteigerung und insgesamt auch eine Reduktion der Qualitätskosten erreichbar ist [28]. Weitere Kostenabhängigkeiten werden in Kapitel 2.5.3 dargestellt.

2.4.2. Testfallbasiertes Testen

Testphasen mit dynamischer Qualitätssicherung: Eine dynamische Qualitätssicherung findet sich in den Testphasen Modultest, Modulintegrationstest, Systemintegrationstest, Systemtest, Fahrzeugintegrationstest, Fahrzeugtest und Akzeptanztest [230].

Tätigkeitenkatalog: Zum testfallbasierten Test gehören jeweils das Planen, Erstellen und Ausführen der Testfälle sowie die Dokumentation und Auswertung der Ergebnisse [176, 241]. Diese Tätigkeiten sind erforderlich, unabhängig davon, ob ein Fehlverhalten gefunden wird. Ein zusätzlicher Aufwand entsteht bei einer Fehleridentifikation. Hier ist das Fehlverhalten zu dokumentieren und ggf. eine Fehlerbeseitigung einzuleiten. Jeder Aufwand, der erst ab Identifikation eines Fehlverhaltens auftritt, wird in dieser Arbeit dem Bereich *Fehlerbeseitigung* zugeordnet, um deutlich die

kausale Kostenzugehörigkeit zu dokumentieren.

Methodische Ableitung von Testfällen: Die Herleitung von Testfällen ist verfahrensabhängig. Funktionsorientierte Testfälle basieren auf Spezifikations- bzw. Entwurfsdokumenten [150, 176]. Spezifikationen können Spezifikationsmodelle (z. B. Sequenzdiagramme [144] und Automaten [241]) umfassen, aus denen Testfälle abgeleitet werden können [207, 208]. Sind derartige Modelle nicht existent, können Testmodelle zur ausschließlichen Ableitung von Testfällen aufgestellt oder aus früheren Projekten adaptiert werden [214]. Folglich bestimmt der Entwicklungsprozess in Anforderungs- und Entwurfsphase, welcher Aufwand zur Testfallableitung aufzuwenden ist.

Unsystematische Testfallerstellung: Eine größtenteils unsystematische und nicht reproduzierbare Testfall- und Testdatenherleitung, die auf der Erfahrung der Tester basiert, wird in dieser Arbeit als *unsystematische Testfallerstellung* bezeichnet. Die unsystematische Testfallerstellung wurde überwiegend in den untersuchten Projekten angewendet und wird daher hier als entwicklungstypisch bezeichnet. Effektivität und Kosten der unsystematischen Testfallerstellung sind hier die Bezugsbasis für die durchgeführten Optimierungen.

Vorteile frühzeitiger Testfallerstellung: Die Spezifikations- und Entwurfsdokumente, aus denen Testfälle abgeleitet werden können, liegen aufgrund der sequentiellen Abarbeitung der Entwicklungsphasen (Anforderungsdefinition, Entwurf und Implementierung) bereits vor Abschluss der Implementierung vor [39]. Folglich kann das Ableiten von Testfällen aus diesen Dokumenten z. B. bereits parallel zur Implementierung erfolgen [39]. So kann die Testbarkeit der Spezifikationen frühzeitig sichergestellt werden. Dieses Vorgehen wird in der Literatur zum Teil empfohlen [150], da es Testern ermöglicht, die Testbarkeit von Spezifikationen und Entwürfen durch frühzeitige Überarbeitung zu erhöhen [150]. Dies soll zu einer erhöhten Spezifikations- oder Entwurfsqualität führen, potentielle Fehlerursachen in der Implementierung vermeiden und die Fehleridentifikationswahrscheinlichkeit im Test erhöhen [150]. Idealerweise werden im Zuge dieser frühzeitigen Testfallgenerierung also bereits Spezifikationsfehler erkannt [229].

Testfallqualität: Testfälle sollen Fehlverhalten erzeugen. Folglich gilt die fehleridentifizierende Wirkung, die Fehlersensitivität, als ein Qualitätsmerkmal. Sollen die Ausführungskosten durch Reduktion der Testfallanzahl minimiert werden, kann durch die Auswahl fehlersensitiverer Testfälle die Qualität ggf. gehalten und somit insgesamt die Teststrategie optimiert werden [178]. Die Qualität von Testfällen kann immer nur in Abhängigkeit von dem gewählten Testziel bewertet werden. Das Problem der Testfallqualität wurde bislang wenig diskutiert [13, 218] und wird im Ausblick (s. Kapitel 9.2) kurz dargestellt.

2.4.3. Automatisierung im Test

Begriffsbestimmung zur Automatisierung: Unter Automatisierung wird die selbsttätige Abarbeitung von sonst manuellen Arbeitsschritten verstanden. Im Kontext des Testens sind dies zum einen die automatisierte Erzeugung von Testfällen (automatisierte Testfall- und Testdatenerstellung) sowie die Testautomatisierung [25], unter der hier das Vorbereiten von Testfällen für eine automatisierte Ausführung (die Test*fall*automatisierung) sowie die automatisierte Testfallausführung selbst zu verstehen sind. Mit der automatisierten Testausführung geht ein automatisierter Ergebnisvergleich einher.

Automatisierte Testfallerstellung: Spezifikations- und Entwurfsdokumente können in formaler und maschinenlesbarer Form vorliegen [6, 55, 218, 229], so z. B. als Modelle. Die maschinenlesbaren Dokumente können zur automatisierten Generierung von Testfällen verwendet werden [76]. Die automatisierte Testfallerstellung soll den Aufwand reduzieren und somit zu einer Kostenersparnis führen [76].

Testautomatisierung: Die erstellten Testfälle, die während des Tests automatisiert ausgeführt werden sollen, sind für die automatisierte Testfallausführung vorzubereiten [76]. Dieser Schritt verursacht zunächst Mehrkosten. Die automatisierte Testfallausführung kann jedoch erhebliche Zeit in der Ausführung der Testfälle einsparen [176], wodurch Personalkosten deutlich reduziert werden können [76]. Diese Einsparung soll die Mehrkosten der Testfallautomatisierung amortisieren und in Summe Kosten einsparen. Die einmaligen[4] Investitionskosten der Testfallautomatisierung werden durch Einsparungen in der mehrfachen Testfallausführung besonders effektiv amortisiert. Mehrfache Ausführungen ergeben sich z. B. durch die Wiederholung von Tests im Rahmen von Regressionstests in verschiedenen Entwicklungsiterationen und nach erfolgter Fehlerbeseitigung [176]. Regressionstests werden also notwendig, wenn sich das Produkt aufgrund einer Weiterentwicklung (z. B. iterative Entwicklung) oder Fehlerbeseitigung verändert hat [176].

Mögliche Nutzenziehung: Durch die Verringerung der Ausführungszeit der Testfälle ist es möglich, innerhalb des gegebenen Zeitrahmens eine höhere Testfallanzahl auszuführen. Dies kann zu höheren Überdeckungen und so zu einer Qualitätssteigerung führen [212]. Unter Beibehaltung der Testfallanzahl wird eine Kosten- und Zeiteinsparung erwartet.

2.5. Kostenstruktur

Vorgehen: Grundlegende Voraussetzung für eine Kostenoptimierung des Testprozesses ist eine differenzierte Kenntnis der Kostenelemente und deren strukturelle Ordnung [70]. Für die intendierte Optimierung von Teststrategien werden die Aktivitäten des Testprozesses in der Abfolge der Systementwicklung analysiert und in Kapitel 2.5.1 mit den identifizierten, monetär messbaren Kostenelementen

[4] Sofern Testfälle nicht infolge von Spezifikationsänderungen anzupassen sind.

beschrieben. Auf die hier eingeführten Bezeichnungen der Kostenelemente und deren maßgeblichen Strukturen wird in einer späteren Kostenmodellbildung Bezug genommen.

Quellen: Die Identifikation der kostenrelevanten Einflussgrößen und deren strukturelle Verknüpfung basiert auf

- allgemeinen Analysen der Aktivitäten im V-Modell (Projektphasen und deren Abhängigkeiten, Tätigkeitsabfolgen),
- Analysen von Qualitätskostenmodellen des industriellen Produktionsprozesses aus der Literatur [31, 44, 65, 69, 70, 104, 143, 154, 157, 228, 256, 261, 265],
- Analysen von automobilen Entwicklungsprojekten (Fehlerdatenbanken, Entwicklungsprozesse, Projektbudgetplänen und deren Kostenelementen),
- Expertengesprächen mit Projektmanagern, Testmanagern und Testern.

2.5.1. Kostenelemente

Herstellerinduzierte und Kundeninduzierte Qualitätskosten: Testvorgänge finden im betrachteten Testprozess sowohl beim Hersteller als auch beim Kunden statt. Die für diese Tätigkeiten anfallenden Qualitätskosten werden nachfolgend als *Herstellerinduzierte Qualitätskosten* bzw. als *Kundeninduzierte Qualitätskosten* bezeichnet. Den kundeninduzierten Qualitätskosten werden hier auch die Kosten für die im Feldeinsatz identifizierten Fehler zugerechnet. Der Feldeinsatz ist zwar keine Test- sondern die letzte Projektphase, wird hier aber aufgrund der fehleridentifizierenden Eigenschaften einer Testphase beim Kunden gleichgesetzt.

Hauptkostengruppen der Herstellerinduzierten Qualitätskosten: Die *Herstellerinduzierten Qualitätskosten* umfassen Kosten für das Testmanagement, den Testvorgang und die Beseitigung der in den Tests identifizierten Fehler. Diese Kosten werden inhaltlich-strukturell anhand des chronologischen Ablaufs eines Projekts in folgende Hauptkostengruppen gegliedert: *Testmanagement, Testaufbau, Testvorbereitung, Testdurchführung* und *Herstellerinduzierte Fehlerbeseitigung*. Die Hauptkostengruppen untergliedern sich in der Regel in mehrere Kostengruppen, die nachfolgend mit den zugehörigen Kostenelementen dargestellt sind.

Testmanagement: Für das Testmanagement lagen in den untersuchten Projekten keine Informationen für eine Analyse vor. Die Planung der Testaktivitäten sowie die flankierenden Verwaltungstätigkeiten (Kommunikation mit dem Kunden etc.) können jedoch großen Einfluss auf Effektivität und Effizienz des Tests haben [24]. Daher werden *Testplanung* und *Verwaltung* als eigenständige Kostengruppen definiert. Deren Verfeinerungen können projekt- bzw. firmenspezifisch geprägt sein, sind jedoch weder Ziel noch im maßgeblichen Wirkungsbereich der intendierten Optimierung von Teststrategien. Folglich wird auf eine Verfeinerung der Kostengruppen *Testplanung* und *Verwaltung* verzichtet. Beide Kostengruppen können jedoch leicht durch projekt- bzw. firmenspezifische Kostenelemente erweitert werden.

Testaufbau: Zum Testaufbau zählt die Kostengruppe *Projektspezifische Ausführungsumgebung* mit den aus den vorliegenden Budgetplänen abgeleiteten Kostenelementen *Prüfstand* [230], *Prüflinge* [230] sowie *Treiber und Platzhalter* [176]. Die Kostengruppe *Werkzeugkosten* wird hierbei weit gefasst und umfasst z. B. alle Kosten für projektübergreifend einsetzbare Software oder sonstige Hilfsmittel die den Testprozess unterstützen. Im Wesentlichen sind dies Beschaffungskosten, wie Lizenzen, aber es können z. B. auch Schulungskosten in Betracht kommen. Die Werkzeugkosten umfassen Kostenelemente für die Aktivitäten *Testmanagement*, *Testvorbereitung*, *Testdurchführung* und *Fehlerbeseitigung*. Die Kostenelemente bzw. Kostengruppen des Testaufbaus haben folgende maßgebenden Eigenschaften:

- Die Testumgebung besteht für eingebettete Systeme der Automobilindustrie vorwiegend aus dem für den Testvorgang notwendigen Hard-/Software-System, in der Praxis meist Prüfstand (engl.: *Test Bench*) genannt. Es simuliert die Umgebung an den Schnittstellen zur Umwelt [241]. Die Kosten für die Testumgebung setzen sich aus den Anschaffungskosten für die Komponenten des Prüfstandes zusammen. Personalkosten für Aufbau und Wartung des Prüfstandes sind in den Kostenabrechnungen als Paketpreis bei der Anschaffung enthalten. Die Kosten für den Prüfstand sind in der Regel als einmalige Investitionskosten anzusehen. Somit sind die Kosten im Projekt konstant.
- Weiterhin sind *Prüflinge*, sog. *Systeme unter Test* (engl.:*systems under test*), zu erstellen, die die noch in Entwicklung befindlichen Systeme im Testvorgang ersetzen. Prüflinge werden häufig für bestimmte Entwicklungsiterationen der manuellen Vorserienproduktion erstellt und können nicht unerhebliche Kosten erzeugen.
- *Treiber* (engl.: *Driver*) simulieren aktive Systemteile, die die zu testenden Systemteile aufrufen bzw. triggern [176]. *Platzhalter* (engl.: *Dummy, Stubs*) bilden das Gegenstück zu Treibern und simulieren reaktive Systemteile, die vom zu testenden System gesteuert und kontrolliert werden, jedoch noch nicht implementiert sind [176]. Treiber und Platzhalter bezeichnen hier die notwendige Software, die zur Ausführung von Testfällen benötigt wird. Sowohl Treiber als auch Platzhalter stellen für Testzwecke ein bestimmtes Ein-/Ausgabeverhalten an den Schnittstellen des zu testenden Systems sicher [176]. In jeder Iteration der Entwicklung werden neue Funktionen implementiert, die je nach Testverfahren durch Treiber und Platzhalter ausführbar gemacht werden müssen. Der Personalaufwand zur Entwicklung der Treiber und Platzhalter ist als Teil der Testkosten zu betrachten, da weder Treiber noch Platzhalter Teil des fertigen Systems sind und nur für den Test benötigt werden.
- Werkzeugkosten fallen für unterstützende Software an, beispielsweise als Lizenzkosten zur Unterstützung von
  - Testmanagement (z. B. Projektplanungswerkzeuge, kaufmännische Systeme),
  - Testvorbereitung (z. B. Modellierungswerkzeuge, Testfallgeneratoren, Testfallverwaltungssysteme),
  - Testdurchführung (z. B. Testautomatisierungswerkzeuge, Simulatoren) und

  - Hersteller-/kundeninduzierte Fehlerbeseitigung (z. B. Fehlerdatenbanken, Entwicklungsumgebungen).

Lizenzkosten werden für die Dauer des Projekts fällig, folglich hängen die Kosten i. Allg. von Dauer und Anzahl der durchzuführenden Iterationen und Testphasen ab. Es sind aber auch fixe Werkzeugkosten möglich. Die Auswahl der Testverfahren kann die zu verwendenden Werkzeuge und somit die entstehenden Kosten bestimmen, beispielsweise wenn Testfallgeneratoren einzusetzen sind.

Testvorbereitung: Die Testvorbereitung umfasst die Bereitstellung von Testfällen, unterteilbar in die Kostengruppen *Modellbasierter Test*, *Testfallgenerierung* und *Testfallautomatisierung*. Die Aktivitäten der Testvorbereitung weisen folgende maßgebenden Eigenschaften auf:

- Modellbasierte Tests können auf Spezifikations- oder Testmodellen basieren [214], die z. B. in Form von Zustandsautomaten das Systemverhalten beschreiben. Spezifikationsmodelle können verwendet werden, um Testfälle abzuleiten. Liegen keine Spezifikationsmodelle vor, können mitunter bestehende Spezifikationsmodelle von anderen Projekten nach erfolgter Anpassung verwendet werden oder es sind Testmodelle zu erstellen, die ausschließlich dem Zweck der Testfallherleitung dienen. Erstellung und Anpassung von Modellen erzeugen Kosten, die hier als Kostenelement *Modellerstellung* bezeichnet werden. Weiterhin sind die Kosten der Testfallgenerierung für modellbasierte Tests wie nachfolgend ausgeführt zu veranschlagen.
- Die Testfallgenerierung kann manuell oder automatisiert, z. B. aus Modellen, erfolgen. Für beide Vorgehensweisen fallen Personalkosten an, die hier den Kostenelementen *Manuelle Testfallerstellung* bzw. *Automatisierte Testfallerstellung* zugeordnet werden. Der Zeitaufwand für die automatisierte Testfallerstellung fällt i. Allg. geringer aus.
- In iterativen Entwicklungsprozessen können Testfälle aus vorangegangenen Iterationen wiederverwendet werden. Wenn sich das System durch Weiterentwicklungen testrelevant verändert hat, sind die Testfälle anzupassen. Zum Beispiel könnten neue Signale oder Aktionen zu berücksichtigen sein. Der Aufwand für die Anpassung bereits existierender Testfälle wird als Kostenelement *Testfallanpassung* vermerkt.
- Die erstellten Testfälle sind im Falle, dass eine automatisierte Testfallausführung erfolgen soll, für diese vorzubereiten. Diese Testfallautomatisierung wird hier als eine gesonderte Kostengruppe betrachtet. Die entstehenden Personalkosten werden als Kostenelement *Vorbereitung zur automatisierten Ausführung* berücksichtigt.
- Wird die automatisierte Testdurchführung in Iterationen wiederholt, kann eine Anpassung von bereits vorbereiteten Testfällen erforderlich werden. Diese Kosten zur Anpassung der Automatisierung sind in einem eigenen Kostenelement *Anpassung der Automatisierung* erfasst.

Testdurchführung: Die Testdurchführung umfasst die Kostengruppen *Testfallausführung* und *Fehleridentifikation* mit folgenden maßgebenden Eigenschaften:

- Für die Testfallausführung werden die aufgewendeten Personalkosten berücksichtigt. Die Personalkosten der *Automatisierten Testfallausführung* sind i. Allg. deutlich geringer als die der *Manuellen Testfallausführung* und werden daher getrennt betrachtet.
- Die Fehleridentifikation umfasst die Kosten des Abgleichs der Testergebnisse mit den erwarteten Ergebnissen. In diesem Schritt wird bei Abweichungen ein Fehlverhalten identifiziert. Die Testauswertung erfolgt analog zur Testfallausführung automatisiert bzw. nicht automatisiert, als *Automatisierte Testauswertung* bzw.*Manuelle Testauswertung*. Das Fehlverhalten wird wie nachfolgend ausgeführt im Fehlermanagement dokumentiert.

Hersteller-/Kundeninduzierte Fehlerbeseitigung: Wird Fehlverhalten durch Tests des Herstellers in den entwicklungsbegleitenden Testphasen (Modultest, Modulintegrationstests, Systemtest und Integrationstests) erzeugt, werden die Fehlerbeseitigungskosten nachfolgend als *Herstellerinduzierte Fehlerbeseitigungskosten* bezeichnet. Beseitigungskosten für beim Kunden (Fahrzeugtest, Akzeptanztest und Feldeinsatz) identifizierte Fehler werden als *Kundeninduzierte Fehlerbeseitigungskosten* bezeichnet. Beide Arten der Fehlerbeseitigung verursachen hauptsächlich Personalkosten und durchlaufen vergleichbare Fehlerbeseitigungsprozesse. Daher können die gleichen Kostengruppen und Kostenelemente betrachtet werden:

- Identifiziertes Fehlverhalten sowie die Beseitigung der Fehlerursachen sind zu dokumentieren, i. Allg. in Fehlerdatenbanken. Kosten für die *Datenpflege* der Fehlermeldungen gehören zu den Kosten der Kostengruppe *Fehlermanagement*.
- Fehlerursachen können durch eine *Fehlerbearbeitung* (Kostengruppe) behoben werden. Diese beginnt mit einer *Fehleranalyse* (Kostenelement), die mit einer Reproduktion des Fehlverhaltens beginnt und die Ursachen des beobachteten Fehlverhaltens identifiziert. Der bei kundeninduzierten Fehlerbeseitigungen zur Reproduktion notwendige Systemablauf kann einem fehleridentifizierenden Testfall gleichgesetzt werden. Weiterhin ist zu analysieren, ob für diese Fehlermeldung eine Fehlerkorrektur einzuleiten ist. Das ist beispielsweise nicht der Fall, wenn das Fehlverhalten nicht durch die Software verursacht ist oder wenn sich bereits die identische Fehlerursache in der Korrektur befindet (Duplikatsfehlermeldung). Von einer Fehlerkorrektur kann auch im Falle hoher Fehlerbeseitigungskosten abgesehen werden. Diese Opportunitätsentscheidung ist häufig das Ergebnis einer Kosten-Nutzen-Analyse, die die Kosten der Fehlerbeseitigung dem erwarteten Nutzen (Vermeidung von Fehlerfolgekosten) gegenüber stellt.
- Die Beseitigung einer Fehlerursache wird unter dem Kostenelement *Fehlerkorrektur* erfasst und quantifiziert den Aufwand der Fehlerlokalisierung und der Systemänderung (in Abhängigkeit von der Fehlerursache Spezifikations-, Entwurfs-, Implementierungs-, Konfigurations-, Dokumentationsänderungen).
- Abschließend findet eine Überprüfung der Fehlerkorrektur statt (Kostengruppe *Korrekturtest*). Diese untergliedert sich im Kostenmodell in *Verifikation der Fehlerbeseitigung* (Ausführung aller die ehemalige Fehlerursache identifizierenden Testfälle) und Durchführung von *Regressionstests*.

Hauptkostengruppen der kundeninduzierten Qualitätskosten: Für Testphasen, die bei den Kunden stattfinden, entstehen dem Hersteller im Allg. lediglich Kosten für die Beseitigung der beim Kunden identifizierten Fehler. Die Kosten für Testvorbereitung und Testvorgang werden in der Regel nicht im Projekt des Herstellers budgetiert. Eine Fehleridentifikation nach Markteinführung des Produktes kann – neben den Kosten für die Fehlerbeseitigung – weitere Kosten nach sich ziehen, z. B. durch Rückrufaktionen und Imageschäden [28]. Quantifizierende und differenzierende Analysen finden sich in der Literatur [44, 45, 63, 65, 69, 70, 104, 143, 154, 184, 228, 256, 261, 265]. Diese Kosten werden hier als Hauptkostengruppe *Fehlerfolgekosten* bezeichnet und sind aufgrund der kaum prospektiv abschätzbaren Kosten nicht detailliert betrachtet. Die Hauptkostengruppe dient somit als Platzhalter.

2.5.2. Strukturelle Ordnung

Determinanten der strukturellen Ordnung: Die strukturelle Ordnung der Kostenelemente ist durch den Ablauf von Tests sowie durch die Abwicklung des Testprozesses in einem iterativen, phasenorientierten Vorgehensmodell (z. B. iterativ durchlaufenes V-Modell) bestimmt.

Verknüpfung der Kostenelemente: Alle Kostenelemente des Testprozesses sind additiv verknüpft.

Testmanagement: Die Kosten des *Testmanagements* ergeben sich aus den Kosten für *Testplanung* und *Verwaltung* (s. Kapitel 2.5.1) und sind von den zu optimierenden Hauptkostengruppen *Testaufbau, Testvorbereitung, Testdurchführung* und *Herstellerinduzierte Fehlerbeseitigung* entkoppelt.

Testaufbau: Die Kosten der Kostengruppen des *Testaufbaus* (*Prüfstand, Prüflinge, Treiber, Platzhalter* sowie *Werkzeuge*) fallen iterationsabhängig an, innerhalb einer Iteration in Abhängigkeit von den Testphasen und den eingesetzten Testverfahren. Beispielsweise fallen Werkzeugkosten (wie Lizenzkosten der Software) in Abhängigkeit vom Einsatz von Testautomatisierungen an und hängen in der Regel von der Projektdauer ab.

Testvorbereitung, Testdurchführung und Fehlerbeseitigung: Die Kostenhöhe der Kostenelemente in den Hauptkostengruppen *Testvorbereitung*, *Testdurchführung* und *(Herstellerinduzierte bzw. Kundeninduzierte) Fehlerbeseitigung* sind durch komplexe Abhängigkeiten zwischen den Kostenelementen bestimmt. Zum Beispiel können durch eine Testfallautomatisierung Lizenzkosten, Personalkosten zur Automatisierung (Kostenelement: *Vorbereitung zur automatisierten Ausführung*) sowie Kosten zur Anpassung automatisierter Testfälle (Kostenelement: *Anpassung der Automatisierung*) zusätzlich zu den Personalkosten zur Testfallausführung (Kostenelemente: *Manuelle Testfallausführung, Automatisierte Testfallausführung*) anfallen. Die Abhängigkeiten zwischen den Kostenelementen werden anhand des Diagramms in Abbildung 2.5 erläutert:

Ein zu testendes eingebettetes System besitzt eine Anzahl *Fehlerursachen*. Jede Fehlerursache ist anhand charakteristischer Eigenschaften bestimmten *Fehlerklassen* zuordenbar. Fehlerursachen bzw. resultierendes *Fehlverhalten* können unbemerkt bleiben. Fehlverhalten soll jedoch in den *Iterationen* der Systementwicklung durch Tests identifiziert werden. Eine Systementwicklung untergliedert sich in der Regel in mehrere *Iterationen*, die jeweils aus mehreren *Testphasen* bestehen können. Die Testphasen bestimmen, welche *Testverfahren* eingesetzt werden können. Beispielsweise kann eine Zweigüberdeckung des Quelltextes [176] häufig nur in der Testphase *Modultest* sichergestellt werden [176]. Ein Testverfahren kann mehrere *Testreihen* zur Sicherstellung aller Überdeckungskriterien benötigen. Jede Testreihe besteht aus einer Anzahl von *Testfällen*, welche Kosten für deren Erstellung, eventuelle Automatisierung und ggf. mehrfache Ausführung (z. B. infolge der iterativen Systementwicklung) verursachen. Die ausgeführten Testfälle können Fehlverhalten produzieren. *Analyse* und *Korrektur* der Fehlerursache sowie *Verifikation der Fehlerbeseitigung* führen zu testphasenabhängigen Kosten. Die Verifikation der Fehlerbeseitigung umfasst auch die Ausführung von *Regressionstests*, also die Ausführung der Testreihe des fehleridentifizierenden Testfalls. Unter bestimmten Voraussetzungen kann eine Systemänderung im Zuge der Fehlerkorrektur eine *Anpassung der Testreihen* erfordern. Der Regressionstest kann wiederum ein Fehlverhalten identifizieren, das zu einer Fehlerbeseitigung führt. Hierdurch ist die Kostenfunktion als rekursiv anzusehen. In Abbildung 2.5 ist diese Rekursion zwischen *Fehlerbeseitigung* und *Testverfahren* erkennbar.

Jede Fehlerbearbeitung wird durch eine Fehlermeldung angestoßen. Bei der Darstellung der Abhängigkeiten wurde zwar die resultierende Fehlerbeseitigung, nicht jedoch die Fehlermeldungen selbst berücksichtigt. Auf deren Darstellung wurde verzichtet, da die Abarbeitung von Fehlermeldungen nur marginale Verflechtungen in den dargestellten Abhängigkeiten aufweist. Die Abarbeitung von Duplikatsfehlermeldungen ist jedoch in Abbildung 2.5 durch die dargestellte M:N-Beziehung zwischen *Testfallausführung* und *Fehlverhalten* angedeutet.

Das beim Kunden identifizierte Fehlverhalten kann eine *Fehlerbeseitigung* induzieren, die Kosten verursacht, die u. a. von der Testphase der Fehleridentifikation beeinflusst sind. Hierbei wird der Feldeinsatz als eine fehleridentifizierende Testphase angesehen. Wird die kundeninduzierte Fehleridentifikation als Pseudo-Testverfahren *kundeninduzierter Test* aufgefasst und die Fehlermeldungen mit allen Angaben samt Fehlerreproduktion als Testfall angesehen, gliedert sich die Bearbeitung strukturell ein.

Abbildung 2.5 verdeutlicht sowohl die Struktur der Systementwicklung als auch die Abhängigkeiten der Kostenelemente. Die Werte aller Kostenelemente von *Testvorbereitung*, *Testdurchführung* und *Fehlerbeseitigung* bestimmen sich in Abhängigkeit von den Iterationen, in diesen von den Testphasen und je Testphase von den Testverfahren. Innerhalb eines Testverfahrens ergeben sich wiederum weitere Abhängigkeiten. Ein Testverfahren identifiziert vorrangig Fehlverhalten bestimmter Fehlerklassen. Dem Testverfahren sind Testreihen, bestehend aus Testfällen, zugeordnet. Das von einer Testreihe identifizierte Fehlverhalten kann akzeptiert und Fehlerklassen zugeordnet werden (indirekte Beziehung zwischen Testreihe und Fehlverhalten in

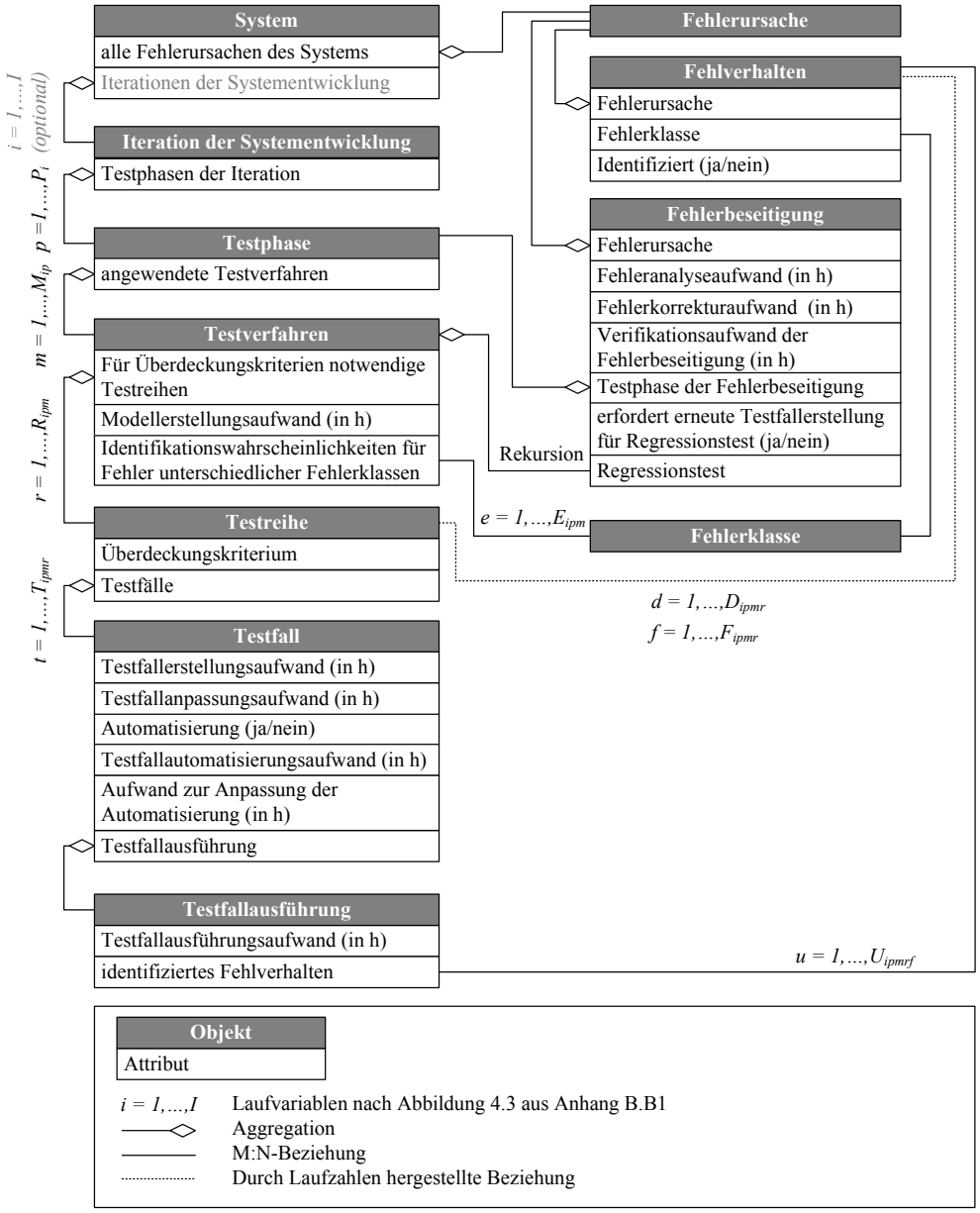


Abbildung 2.5.: Kostenbestimmende Abhängigkeiten in der Struktur der Systementwicklung

Abbildung 2.5). Neben dem originären, fehleridentifizierenden Testfall können zu Duplikatsfehlermeldungen gehörige Testfälle das gleiche Fehlverhalten bzw. dieselbe Fehlerursache identifizieren. Bei kundenidentifiziertem Fehlverhalten können Testverfahren unsichtbar bleiben und die Testfälle im engeren Sinne wie erwähnt durch eine zur Fehlerreproduktion erforderliche Systemabfolge ersetzt werden.

2.5.3. Kostenabhängigkeiten zwischen Test und Fehlerbeseitigung

Fehlervermeidung und Fehlerbeseitigung vor dem Test: Konstruktive und analytische Qualitätssicherung durch die Entwickler verursachen in den Phasen der Softwareentwicklung (Anforderungsdefinition, Entwurf und Implementierung), also vor dem Testprozess, Qualitätskosten, die auch Kosten für ausgelöste Fehlerbeseitigungen enthalten (s. Abbildung 2.6). Hierbei bewirkt ein erhöhter Aufwand zur analytischen Qualitätssicherung tendenziell auch eine erhöhte Fehleridentifikation, die zu einer Erhöhung der Fehlerbeseitigungskosten vor den Testphasen führt. Da Fehlervermeidung, ausgewählte Maßnahmen zur analytischen Qualitätssicherung und Fehlerbeseitigungen vor den eigentlichen Testphasen entwicklungsbegleitend von den Entwicklern selbst durchgeführt werden, ist der Aufwand zwar gegeben, jedoch in der Regel nicht explizit dokumentiert und daher selten quantifizierbar. Die Beseitigungskosten für eine einzelne Fehlerursache steigen exponentiell mit Entwicklungsfortschritt an [28]. Je später eine Fehlerursache entdeckt wird, desto breiter kann sich der Einfluss der Fehlerursache fortgesetzt haben (Folgefehler) [28, 257] und umso mehr Korrekturtests sind zur Überprüfung der durchgeführten Korrektur zu wiederholen [241]. Je nach Projektfortschritt kann die Fehlerbeseitigung außerdem vielfältige, flankierende Maßnahmen erforderlich machen, beginnend bei der Änderung von Dokumentationen bis hin zu Rückrufaktionen bei Fehlverhalten während des Feldeinsatzes. Weiterhin steigen die Fehlerbeseitigungskosten durch die mit Testfortschritt abstrahierte Systemsicht, die im Mittel eine aufwändigere Lokalisierung der Fehlerursachen und einen umfangreicheren Testaufwand zur Verifizierung der Softwarekorrektur bzw. für Regressionstests bedingt [24]. Somit reduziert eine frühzeitige, also auch eine vor dem Testprozess liegende Fehlerbeseitigung zweifelsfrei die Projektkosten [28], da Kosten aufgrund später Fehlerbeseitigungen vermieden werden. Somit wirkt sich die frühzeitige Fehlerbeseitigung vor den Testphasen in der Regel kostensenkend auf die Kosten der testinduzierten Fehlerbeseitigung aus [85].

Fehlerbeseitigungskosten im Testprozess: Die Anzahl der im Testprozess zu bearbeitenden Fehlerursachen bestimmt zusammen mit der Testphase der Fehleridentifikation maßgeblich die Kosten der testinduzierten Fehlerbeseitigung. Die Anzahl identifizierter Fehler ist zwar wesentlich von der Effektivität der Testverfahren beeinflusst [96, 134], aber selbstverständlich auch von der Anzahl der im System befindlichen Fehlerursachen bzw. der nach den Maßnahmen der Qualitätssicherung durch die Entwickler verbliebenen Anzahl Fehlerursachen [257]. Die Identifikationsrate wird entweder durch verstärktes Testen mit erhöhter Testfallanzahl [25], einhergehend mit steigenden Testkosten, oder durch veränderte Verfahrenszusammenstellung [23] erhöht. Zu beachten ist, dass nach dem Marginalprinzip die Anzahl der

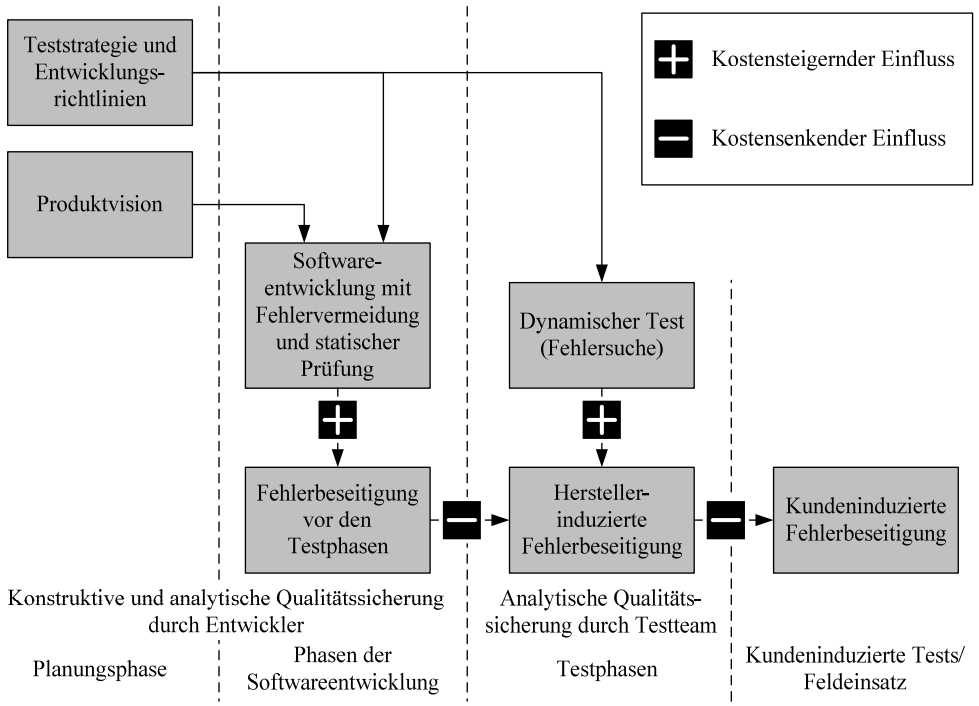


Abbildung 2.6.: Kostendeterminanten im Testprozess

Fehleridentifikationen durch Erhöhung des Testaufwands nicht linear erhöht werden kann [228]. Die Teststrategie, also die Anzahl und die Auswahl der Testverfahren zusammen mit der Anzahl der durchzuführenden Testfälle, reguliert die Kosten des Testprozesses in den Testphasen [23, 99].

Kundeninduzierte Fehlerbeseitigungskosten: Nach Abschluss der Testphasen beim Hersteller wird in dem System in der Regel noch Fehlverhalten in den abschließenden Tests bei den Kunden oder im Feldeinsatz diagnostiziert. Dieses wird als kundeninduzierte Fehlerbeseitigung zu bearbeiten sein [28]. Durch einen intensiveren herstellerseitigen Test mit Fehlerbeseitigung können die kundeninduzierten Fehlerbeseitigungskosten gesenkt werden. Die kundeninduzierten Fehlerbeseitigungskosten sind die kostenintensivsten. Bei Feldfehlern können zudem Folgekosten wie Rückrufaktionen usw. hinzukommen [28].

Kostenbalance: Die in Abbildung 2.7 dargestellten Kostenabhängigkeiten verdeutlichen das Grundproblem für die Erstellung einer Teststrategie: Es gilt eine Auswahl an Testverfahren und Testfällen zu finden, die Entwicklungszeit, Kosten und Qualität optimal kombiniert [23]. Optimal bedeutet, dass die Verfahren, inkl. Bestimmung des Testfallumfangs, so kombiniert werden, dass sie die Fehlerursachen effizient und frühzeitig identifizieren, so dass die angestrebte Produktqualität (Testendekriterium) innerhalb der Zeit- und Kostenbeschränkungen erreicht wird.

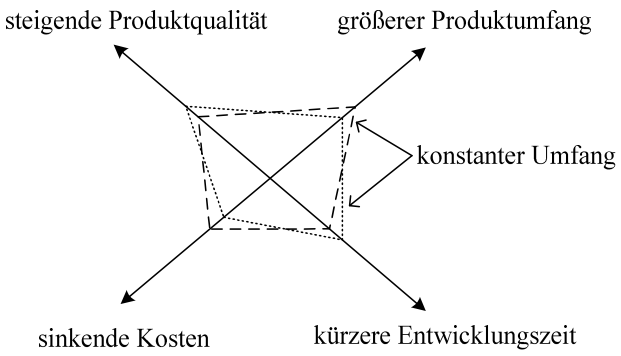


Abbildung 2.7.: Teufelsviereck der Optimierung der Systementwicklung nach Sneed [238]

3. Qualitätskosten und Fehleranalysen in der Literatur

In diesem Kapitel wird aufgezeigt, inwiefern publizierte Qualitätskostenmodelle den Anforderungen für die Zielsetzung dieser Arbeit genügen. Weiterhin werden Fehlerströme und das testphasenabhängige Kostenwachstum von Fehlerbeseitigungskosten in der Literatur erarbeitet. Ferner werden Fehlerklassenanalysen skizziert, auf die im Späteren Bezug genommen wird.

3.1. Qualitätskostenmodelle

Qualitätskostenmodelle bilden die Struktur der Kosten ab und machen damit die Qualitätskosten transparent und nachvollziehbar [256]. Eine Untersuchung von Geiger und Kotte [104] ergab, dass in der Literatur überwiegend Qualitätskostenmodelle existieren, die produktunabhängig formuliert sind und den gesamten industriellen Produktionsprozess bewerten.

Dreigeteilte Qualitätskosten nach DIN 55350: Die traditionelle Betrachtung von Qualitätskosten zwischen 1950 und 1980 [256] findet sich in der Festlegung der DIN 55350: Die DIN 55350 [142] bildet insbesondere für die Automobilindustrie [143] die Grundlage vieler Qualitätskostenbetrachtungen und definiert dreigeteilte Qualitätskosten wie folgt: „*Qualitätsbezogene Kosten sind im Rahmen des Qualitätsmanagements entstehende Fehlerverhütungs-, Prüf- und Fehlerkosten*“ [142]. Diese Dreiteilung von Feigenbaum [89] aus dem Jahre 1956 wird als PAF-Schema (engl.: *Prevention, Appraisal & Failure Costs*) bezeichnet und ist in der ISO 10014 [143] international standardisiert. *Fehlerverhütungskosten* steigern die Qualität eines Produktes durch Analyse und Vermeidung sowie Beseitigung von Fehlerursachen während der Entwicklung und Produktion [69], z. B. durch Qualitätsmanagement, Prüfplanung oder Lieferantenbewertungen [104]. *Prüfkosten* sind planmäßige, die Entwicklung und Produktion begleitende Kosten zur Überprüfung der Qualität [142] und setzen sich z. B. aus Wareneingangsüberprüfungen, Fertigungsprüfungen und Prüfdokumentationen zusammen [104]. *Fehlerkosten* beziffern Folgekosten fehlerhafter Produkte. Entsprechend dem Ort der Fehleridentifikation werden Fehlerkosten in *interne und externe Fehlerkosten* unterteilt. Interne Fehlerkosten betreffen beim Hersteller (z. B. Fahrzeugteilezulieferer), externe Fehlerkosten beim Kunden (z. B. Fahrzeughersteller) identifiziertes Fehlverhalten [104]. Interne Fehlerkosten umfassen vornehmlich direkte Kosten, externe Fehlerkosten darüber hinaus indirekte Kosten wie Verdienstausfälle, Wertminderungen oder Imageschäden [142]. Eine detaillierte Auflistung von Kostenelementen der Qualitätskostenrechnung liefern Geiger und Kotte [104].

Modellerweiterungen ab 1980: Spätere Ansätze zwischen 1980 und 1995 erweitern die traditionelle PAF-Einteilung und führen die Unterteilung in *Konformitäts-* und *Nichtkonformitätskosten* ein [45, 63, 184]. Seit 1995 wird darauf aufbauend das *umfassende Qualitätsmanagement* (engl.: *Total Quality Management*, TQM) [118] entwickelt. Kostenmodelle, die auf dem umfassenden Qualitätsmanagement beruhen, unterscheiden zwischen positiven und negativen Kosten [117]. Als positiv werden Kosten bewertet, die die Produktion fehlerfreier Produkte sichern sollen, also der Qualitätssteigerung dienen. Negativ werden Fehlerfolgekosten bewertet, die für die Beseitigung von Fehlerursachen und deren Folgen aufzubringen sind und somit nicht unabwendbar zur Qualitätssteigerung erforderlich gewesen wären. Daher sind zuerst negativ, dann positiv zu bewertende Kosten zu reduzieren. Bei der Reduktion positiv bewerteter Kosten sollten die als negativ bewerteten Kosten nicht ansteigen. Auf diesen Grundgedanken basierend wurden verschiedene Kostenmodelle entwickelt [44, 65, 69, 70, 104, 143, 154, 228, 256, 261, 265]. Exemplarisch aufgrund der unterschiedlichen Ansätze werden die Kostenmodelle von Wildemann [265], Kamiske und Tomys [154] sowie von Wagner [257] kurz charakterisiert:

- Wildemann [265] fasst Fehlerverhütungskosten und interne Prüfkosten zu *Kosten der Übereinstimmung*, externe Prüfkosten und Fehlerkosten zu *Kosten der Abweichung* zusammen. Die Begriffe Übereinstimmung und Abweichung beziehen sich hierbei auf den Vergleich von Spezifikation und realisiertem System. Die Kosten der Übereinstimmung tragen zum Unternehmenserfolg bei und werden daher als sinnvoll erachtet. Kosten der Abweichung werden als *Verschwendung von Ressourcen* angesehen und sind daher zu minimieren.
- Kamiske und Tomys [154] unterscheiden primär zwischen *Qualitätskosten* und *Fehlerkosten*. Qualitätskosten setzen sich aus Fehlerverhütungskosten und Prüfkosten zusammen und dienen somit der Qualitätssteigerung. Interne und externe Fehlerkosten werden als zu vermeidende Kosten angesehen.
- Wagner [257] unterteilt Qualitätskosten ähnlich wie Wildemann [265] in *Kosten der Übereinstimmung* und *Kosten der Abweichung*. Die Kosten der Übereinstimmung beinhalten u. a. auch die in dieser Arbeit fokussierten Testkosten (Aufbau und Ausführung). Die Kosten der Abweichung werden in interne und externe Fehlerkosten unterteilt, die beide Fehlerbeseitigungskosten beinhalten. Externe Fehlerkosten haben zusätzliche Kosteneffekte, wie z. B. Imageschäden.

Eignungsbewertung der vorliegenden Modelle: Die analysierten Qualitätskostenmodelle [31, 44, 65, 69, 70, 104, 143, 154, 228, 256, 261, 265] betrachten vornehmlich die Produktion und nur untergeordnet das Testen. Häufig sind eine Beschreibung der Kostenelemente und deren Erfassungsvorschriften nicht ersichtlich. Die Kostenmodelle [195, 257, 263], die nur auf die Testdurchführung fokussiert sind, erweisen sich oft als zu speziell und sind daher nicht umfassend genug einsetzbar. Häufig fehlen auch für die vorliegende Thematik wichtige Kostenelemente (wie z. B. Prüfstand, Werkzeugkosten) sowie eine native Berücksichtigung der iterativen Anwendung von phasenorientierten Vorgehensmodellen (z. B. V-Modell). Andere Kostenmodelle wiederum betrachten bestimmte Teilaspekte, wie Fehlervermeidungsmaßnahmen [97, 166, 226, 234], Prüflinge [234], Testprozesse [225] oder die Wahl

von zu testenden Systemkomponenten [50, 175].

Eine generelle Problematik liegt in der Verfügbarkeit der Kostendaten. Sie werden häufig nicht bzw. nicht detailliert erfasst. So stehen im Betriebsrechnungswesen die erforderlichen Basisdaten oft nur implizit zur Verfügung [117]. Sie müssen daher aus vorliegenden Daten, also aus verschiedenen Kosten, Aufwendungen und Erlösschmälerungen extrahiert oder geschätzt werden. Letzteres führt zwangsläufig zu unpräzisen Werten [117].

In dieser Arbeit wird ein Modell zur Bestimmung der testinduzierten Qualitätskosten aufgestellt, das die Kosten des testfallbasierten Tests detailliert bewerten kann und die kostenrelevanten Strukturen des Testprozesses eingebetteter Systeme in der Automobilindustrie abbildet.

3.2. Quantifizierung von Fehlerströmen

Begriffliche Abgrenzungen für Fehlerstromdiagramme: Die Zeitpunkte der Entstehung der Fehlerursachen und der Fehleridentifikation/-beseitigung können in *Fehlerstromdiagrammen* dokumentiert werden [176]: Es werden Häufigkeitsverteilungen für die Phasen der Fehlerentstehung (Irrtum) und für die der Fehleridentifikation (s. Kapitel 2.3) gebildet. Jede identifizierte Fehlerursache wird sowohl in der Phase ihrer Entstehung als auch in der Phase ihrer Identifizierung gezählt. Die Anzahlen können absolut oder relativ wiedergegeben werden. Die Relativzahlen ermöglichen einen Vergleich von Projekten unabhängig von der Fehleranzahl.

Zuordnungsproblematik für die Phasen der Fehlerentstehung: Die Zuordnung der identifizierten Fehlerursachen zu der Phase ihrer Entstehung ist nicht immer unproblematisch bzw. eindeutig. Beabsichtigt ist, in einer Fehleranalyse die Entwicklungsphase zu identifizieren, in der ein Irrtum zur Fehlerursache führt, die die beobachtete Abweichung vom Sollverhalten bewirkt hat. In den Dokumenten dieser Entwicklungsphase wird häufig mit der Korrektur begonnen, um die Fehlerursache zu beseitigen. Beispielsweise ist ein Spezifikationsfehler demnach zunächst in der Anforderungsdefinition zu beheben, dann folgen die weiteren Entwicklungsphasen. Auch sind Entwurfs- und Implementierungsfehler nicht immer eindeutig unterscheidbar [53], da Fehlerursachen existieren, die wahlweise durch Änderungen in Entwurf *und* Implementierung als auch durch ausschließliche Änderungen in der Implementierung korrigiert werden können. Dennoch ist diese Klassifikation der Fehlerursache nach ihrer Entstehung in der Literatur weit verbreitet und wird auch daher hier verwendet.

Ziel der Fehlerstrombetrachtung: Fehlerströme werden in dieser Arbeit als Ausgangspunkt für Analysen zur Effektivität von Testverfahren und als Kenngrößen für Optimierungen der Fehleridentifikation verwendet. Es werden hierzu Fehlerströme aus Projektdaten erstellt. Aber auch publizierte Fehlerströme der Literatur dienen in dieser Arbeit als Grundlage für Berechnungen oder für Vergleiche.

Fehlerströme aus der Literatur: Die Fehlerströme aus der nachfolgend zitierten Literatur beziehen sich mit einer Ausnahme (L6 aus [98]) nicht auf den Kontext der Automobilindustrie. Eine Übertragbarkeit der Literaturwerte untereinander ist aufgrund der unterschiedlichen Projekteigenschaften nur begrenzt möglich. Dennoch soll die folgende Betrachtung zur Orientierung einen von einzelnen Projekteigenschaften unabhängigen Durchschnitt liefern. Die Literaturprojekte L1 bis L7 wurden wie folgt chronologisch publiziert:

- L1: Basili [11] analysierte ein System zur Planung des Lebenszyklus von Satelliten. Das System besteht aus etwa 370 Softwaremodulen und besitzt etwa 90.000 Quelltextzeilen, die größtenteils in Fortran erstellt wurden. Ein großer Teil der 370 Module wurde aus vorangegangenen Projekten entnommen und angepasst. Die von den nachfolgenden Untersuchungen abweichende, ungefähre Gleichverteilung der Fehlerentstehung auf die drei Phasen Anforderungsdefinition, Entwurf und Implementierung könnte dadurch bedingt sein, dass das System nicht neu entwickelt, sondern weiterentwickelt wurde.
- L2: Möller [193] aggregierte Fehlerströme einer unbekannten Anzahl von Projekten der Siemens AG. Die Projekte wurden über mehrere Systemversionen hinweg betrachtet. Die Entwicklungszeit betrug jeweils mehrere Jahre. Das Anwendungsgebiet der entwickelten Systeme ist nicht näher spezifiziert.
- L3: Die Untersuchungen von Ebert [78] basieren auf einer Softwareentwicklung für Telefonvermittlungsstellen bei Alcatel. Es handelt sich um ein verteiltes System, das von ca. 500 Programmierern *„sehr verteilt“* [78] entwickelt wurde. Der Quellcode umfasst ca. 2 Millionen Quelltextzeilen der Programmiersprachen Assembler, C und Chill.
- L4: Burghardt [47] charakterisiert die von ihm untersuchten Systeme nicht.
- L5: Freimut [99] untersuchte Inspektionstechniken in zwei Softwareprojekten der Allianz Life. Die untersuchten Softwaresysteme wurden angepasst, es fand also wie in L1 keine Neuentwicklung statt. Die im Vergleich zu anderen Projekten hohe Anzahl an Spezifikationsfehlern sowie die vergleichsweise frühe Fehleridentifikation könnte an der Fokussierung auf Inspektionstechniken zusammen mit der Konstellation *Anpassung statt Neuentwicklung* liegen.
- L6: Freimut [98] beschreibt den Fehlerstrom für ein automobiles Steuergerät der Motorsteuerung. Die Entwicklung wurde nach drei Entwicklungszyklen abgeschlossen. Das System besitzt ca. 200.000 Quelltextzeilen.
- L7: MethodPark [187] berichtet über einen *„führenden Anlagenbauer“* [187], der eingebettete Systeme herstellt.

Fehlerentstehung: Die Fehlerströme in der referenzierten Literatur weisen eine Unterteilung in die drei Entwicklungsphasen Anforderungsdefinition, Entwurf und Implementierung auf, wie sie auch bei den für diese Arbeit betrachteten Daten des Fahrzeugteilezulieferers vorliegen.

Die relativen Fehleranteile in den Entwicklungsphasen weichen in den referenzierten Projekten zum Teil deutlich voneinander ab (s. Tabelle 3.1). Dies kann u. a. sowohl auf den oben skizzierten, unterschiedlichen Projektsituationen als auch auf den oben

erwähnten Zuordnungsproblematiken von Fehlermeldungen beruhen. Wegen der großen Varianz repräsentiert hier der Median (normiert [5]) die Fehleranteile besser als der Mittelwert. Die Mediane (bzw. Mittelwerte) der Fehleranteile steigen von der Anforderungsdefinition über den Entwurf zur Implementierung an. Die Phasen Entwurf und Implementierung sind mit relativen Fehleranteilen von 39% (37%) und 41% (38%) die beiden Phasen, in denen die meisten Irrtümer und somit Fehlerursachen entstehen. Spezifikationsfehler aus der Anforderungsdefinitionsphase sind mit 20% (25%) etwa halb so häufig.

Bezeichnung	Quelle	Projektphase der Fehlerentstehung		
		Anforderungs-definition	Entwurf	Implementierung
L1	Basili, 1984 [11]	34%	30%	36%
L2	Möller, 1993 [193]	10%	40%	50%
L3	Ebert, 1996 [78]	20%	38%	42%
L4	Burghardt, 1997 [47]	18%	44%	38%
L5	Freimut, 2000 [99]	57%	34%	9%
L6	Freimut, 2005 [98]	15%	34%	51%
L7	MethodPark, 2007 [187]	20%	40%	40%
Median		20%	38%	40%
Median, normiert (auf Zeilensumme 100)		20%	39%	41%
Mittelwert		25%	37%	38%

Tabelle 3.1.: Fehlerströme für die Fehlerentstehung aus der Literatur

Fehleridentifikation: Die Phasen der Fehleridentifikation umfassen in der Literatur sowohl die Phasen der konstruktiven und der statischen, analytischen Qualitätssicherung (s. Tabelle 3.2) als auch die des hier betrachteten Testprozesses mit Softwareausführung. Die relativen Anteile der in jeder Testphase identifizierten Fehlerursachen weichen zwischen den Untersuchungen deutlich ab (s. Tabelle 3.2). Bis zum Ende der Entwurfsphase entstanden etwa 60% der Fehlerursachen (s. Tabelle 3.1), von denen bis nach der Entwurfsinspektion etwa ein Drittel identifiziert war. In den Phasen der Fehleridentifikation durch Entwickler (Anforderungs-, Entwurfs- und Quelltextinspektion) wurden im Median (normiert) 44% (Mittelwert 46%) der Fehlerursachen identifiziert. In den sich anschließenden Testphasen (Modultest und Integrations-/Systemtest) wurden im Median (normiert) 48% (Mittelwert 47%) der Fehlerursachen identifiziert, weitere 8% (7%) erst durch den Kunden (z. B.

[5] Die Summe der Medianwerte der Projektphasen Anforderungsdefinition, Entwurf und Implementierung ergibt naturgemäß nicht zwingend 100%. Daher werden die Medianwerte auf die Summe 100% normiert.

Bezeichnung	Quelle	Projektphase der Fehleridentifikation					
		Inspektionen			Testprozess mit Softwareausführung		
		Anforderungs-inspektion	Entwurfs-inspektion	Quelltext-inspektion	Modultest (3)	Systemtest (3)	Feldeinsatz
L2	Möller, 1993 [193]	3%	5%	7%	25%	50%	10%
L3	Ebert, 1996 [78]	1%	2%	20%	30%	40%	7%
L4	Burghardt, 1997 [47]	10%	15%	40%	20%	10%	5%
L5	Freimut, 2000 [99]	49%	18%	0%	5%	21%	7%
L6	Freimut, 2005 [98]	2%	22%	39%	19%	14%	4%
Median		3%	15%	20%	20%	21%	7%
Median, normiert auf Zeilensumme 100		4%	17%	23%	23%	25%	8%
Mittelwert		13%	12%	21%	20%	27%	7%
L1	Basili, 1984 [11] (2)	(1)	(1)	(1)	(1)	(1)	(1)
L7	MethodPark, 2007 [187] (2)	(1)	(1)	5%	25%	50%	20%

Erläuterungen:
(1) Keine Angabe zur Fehleridentifikation
(2) Die Werte liegen unvollständig vor und können daher nicht in die Berechnung einbezogen werden.
(3) Inkl. Integrationstest

Tabelle 3.2.: Fehlerströme für die Fehleridentifikation aus der Literatur

Fahrzeughersteller, Endkunde) im Feldeinsatz. In Abbildung 3.1 sind die medianen Fehlerströme zur Fehlerentstehung und -identifikation je Projektphase grafisch dargestellt.

Zeitliche Entwicklung: Obwohl ein stetiges Bestreben zur Qualitätssteigerung mit frühestmöglicher Fehleridentifikation in der Produktentwicklung vorausgesetzt werden kann, lässt sich – Vergleichbarkeit der Projekte angenommen –, beurteilt an den Publikationsjahren, bestenfalls ein leichter Trend zur Fehleridentifikation in den frühen Phasen erkennen. Zu erwarten wären mit den Publikationsjahren ansteigende Anteile der identifizierten Fehlerursachen in den ersten Phasen sowie geringere Anteile in späteren Phasen.

3.3. Quantifizierung von Fehlerbeseitigungskosten

Kostenanstieg mit Projektfortschritt: Die Kosten der Fehlerbeseitigung steigen mit Projektfortschritt an (s. Kapitel 2.5.3). Tabelle 3.3 zeigt für verschiedene Literatur-

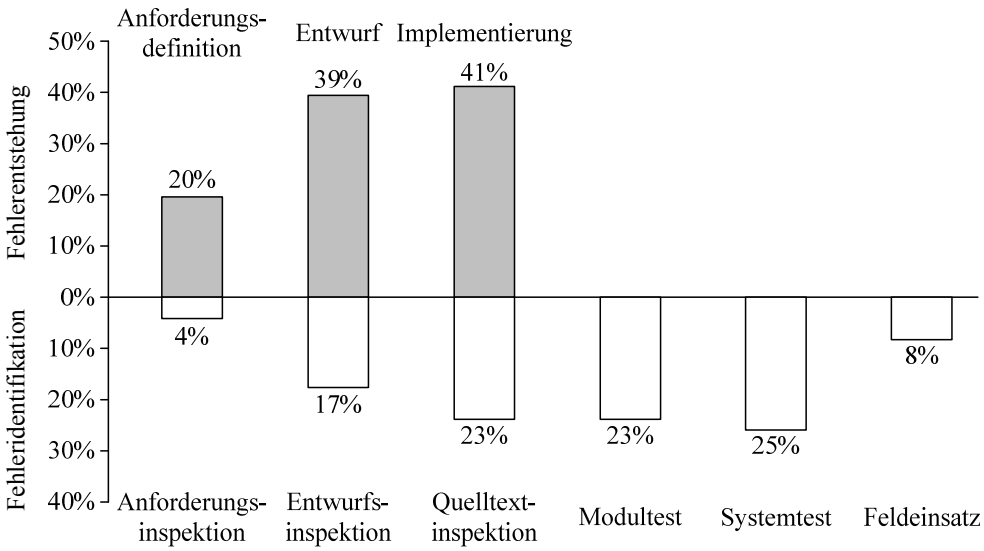


Abbildung 3.1.: Mediane Fehlerströme der Literatur zur Fehlerentstehung und -identifikation

quellen das relative Kostenwachstum über die Phasen normiert auf den Modultest, dem Beginn des dynamischen Tests. Beachtenswert ist der jeweils verstärkte Kostenanstieg im Systemtest und im Feldeinsatz im Vergleich zu den vorangegangenen Testphasen. Wird z. B. der Median über alle Literaturprojekte (Tabelle 3.3, letzte Tabellenzeile) betrachtet, ist die Beseitigung einer Fehlerursache im Systemtest zweimal so teuer wie im Modultest, im Feldeinsatz sogar 5,5-mal. Ebenso wird deutlich, dass Fehlerbeseitigungen noch während der Entwicklungsphasen, hier Phasen der Fehlerentstehung, deutlich kostengünstiger sind. So kostet die Beseitigung einer Fehlerursache in der Anforderungsinspektion nur etwa 20% der Beseitigung im Modultest.

Kostenunterschiede zwischen den Projekten: Die Differenz des Kostenanstiegs von Testphase zu Testphase unterscheidet sich deutlich zwischen den Quellen, wie aus Tabelle 3.3 ersichtlich ist. Der Unterschied wird besonders evident durch den Vergleich zweier häufig zitierter Untersuchungen:

- Boehm [28] identifizierte in großen, sicherheitskritischen Projekten ein exponentielles Anwachsen des Fehlerbeseitigungsaufwands: Koste demnach eine Fehlerbeseitigung im Anschluss an eine Anforderungsinspektion eine Währungseinheit, so würde die Fehlerbeseitigung eines Feldfehlers einhundert Währungseinheiten kosten. Dieses Ergebnis findet sich in der ersten Zeile in Tabelle 3.3.
- Die sogenannte Zehnerregel [242] geht sogar von einem Kostenwachstum um den Faktor zehn in jeder folgenden Testphase aus. Allerdings werden Anforderungs- und Entwurfsinspektion sowie Quelltextinspektion und Modultest jeweils zu einer Phase zusammengelegt. Die Beseitigung eines Spezifikati-

Quelle	Fehlerbeseitigungsaufwand je Phase – normiert auf den Modultest –					
	Inspektionen			Testprozess mit Softwareausführung		
	Anforderungs-inspektion	Entwurfs-inspektion	Quelltext-inspektion	Modultest [1]	Systemtest [1]	Feldeinsatz
Boehm, 1981 (große Projekte) [28]	0,06	0,17	0,33	1,0	2,2	5,6
Boehm, 1981 (kleine Projekte) [28]	0,42	0,52	0,70	1,0	1,5	1,8
Remus, 1983 [223]		0,05	1,00	1,0	1,0	4,1
Collofello, 1989 [58]	1,19		0,48			2,1
Jones, 1991 [148]	0,20	0,30	0,50	1,0	2,0	2,0
Kelly, 1992 [158]		1,20	0,48			
Möller, 1992 [192]	0,50	0,50	0,50	1,0	2,0	3,3
Möller, 1993 [193]		0,10	0,48		2,0	3,3
Kan, 1994 [155]		0,08	1,00	1,0	1,0	7,1
Spur, 1994 [242]	0,10	0,10	1,00	1,0	10,0	100,0
Möller, 1996 [191]	0,25	0,25	0,25	1,0	3,0	12,5
Ebert, 1996 [78]			1,00	1,0	30,0	100,0
Gilb, 1998 [109]	0,03	0,02	0,17	1,0		1,7
Willis, 1998 [268]	0,11	0,32	0,53	1,0	1,2	11,1
Dustin, 2001 [76] (min)	0,07	0,13	0,67	1,0	1,5	6,7
Dustin, 2001 [76] (max)	0,07	0,33	0,67	1,0	3,3	6,7
RTI, 2002 (Finanzwesen) [244]	0,24		0,48			3,1
RTI, 2002 (Transportwesen) [244]	0,83		0,48			5,5
MethodPark, 2007 (Anlagenbau) [187]			0,50	1,0	5,0	10,0
Wagner (Mittelwert), 2007 [257]	0,39	0,85	0,48			10,1
NIST, ohne Jahr [14]	0,83		1,00	1,0	1,7	5,5
Mittelwert (arithm.)	0,4	0,3	0,6	1,0	4,5	15,1
Median	0,2	0,3	0,5	1,0	2,0	5,5

[1] inkl. entsprechenden Integrationstests

Tabelle 3.3.: Faktor des Wachstums des Fehlerbeseitigungsaufwands in der Literatur nach Phase der Fehleridentifikation (=Fehlerbeseitigung), normiert auf den Modultest

onsfehlers im Feldeinsatz kostet hier weitaus mehr als nach der Schätzung von Boehm [28].

Abhängigkeit von Projekteigenschaften: Boehm [28, 30] beobachtete weiterhin ein deutlich geringeres Kostenwachstum für kleine, nicht-sicherheitskritische Projekte. Für große, sicherheitskritische Systeme schätzt Boehm [30] das Kostenwachstum der Fehlerbeseitigung zwischen Anforderungsinspektion und Feldeinsatz – wie soeben zitiert – auf 1:100, für kleine nicht-sicherheitskritische Systeme dagegen nur auf 1:5, wie aus der 2. Zeile in Tabelle 3.3 ersichtlich. Böhm belegt mit diesen Projektgruppierungen die Abhängigkeit des Kostenwachstums von Projekteigenschaften. Es bleibt offen, ob das Wachstum von der Projektgröße und/oder der Kritikalität bzw. von weiteren nicht genannten Faktoren bestimmt wird.

Abhängigkeit von der Verweildauer: Die vorgestellten Untersuchungen gehen von einer Abhängigkeit der Fehlerbeseitigungskosten von der Testphase der Fehleridentifikation aus. Schweiggert [233] hingegen argumentiert, dass die Verweildauer einer Fehlerursache im System ausschlaggebend für die Fehlerbeseitigungskosten ist (nicht die Testphase). Die Ansicht wird durch die Menge der zu wiederholenden Arbeitsschritte und zu ändernden Dokumente begründet. Dies kann somit als einer der Gründe für das Anwachsen der Fehlerbeseitigungskosten über die Testphasen herangezogen werden. In dieser Arbeit wird die eingangs beschriebene Abhängigkeit von der Testphase der Fehleridentifikation (=Fehlerbeseitigung) verwendet.

3.4. Fehlerklassenanalysen

Folgende publizierte Beobachtungen, in denen während der Entwicklung bzw. im Feld identifiziertes Fehlverhalten klassifiziert wurde, werden in Kapitel 5.6 für vergleichende Betrachtungen herangezogen:

- Basili und Perricone [11] analysierten ein System zur Planung von Satellitenprojekten (Planung des Lebenszyklus des Satelliten). Die Autoren ordneten Feldfehler aus diesem Projekt Fehlerklassen zu. Danach sind für 48% des beobachteten Fehlverhaltens fehlerhafte oder missinterpretierte Spezifikationen verantwortlich, was dafür spricht, dass Anforderungsinspektionen zu optimieren sind. 4% der Fehlerursachen betrafen den Systementwurf, 39% die Implementierung, wobei 12% Schreibfehler enthalten sind. 7% des Fehlverhaltens betraf das untrennbare Zusammenwirken von Spezifikation, Entwurf und Implementierung. Die restlichen Fehler konnten nicht eindeutig zugeordnet werden.
- Auch nach Endres [82] verursachen Spezifikationsfehler einen Großteil (46%) der Fehlverhalten. Gründe sind entweder Uneindeutigkeiten, Unvollständigkeiten, Widersprüche oder Änderungen der Spezifikationen während der Entwicklung.
- Lutz [180] verglich zwei sicherheitskritische NASA-Raumfahrtprojekte: Voyager und Galileo. Bei Voyager wirken sich 52%, bei Galileo 48% aller *funktionalen* Fehler auf das beobachtbare Systemverhalten aus. Fehlerhafte Bedingungen und Grenzwerte waren ursächlich für zusammen 73% des sicherheitskritischen

Fehlverhaltens. Weiterhin sind Integrationsfehler die Ursache für 35% (Voyager) bzw. 19% (Galileo) der sicherheitskritischen Fälle für Fehlverhalten.

- Nakajo und Kume [200] schlossen aus Studien, dass funktionale Fehlerursachen und Integrationsfehler 90% des gesamten Fehlverhaltens ausmachen. Spezifikationsfehler scheinen, sofern überhaupt betrachtet, eine untergeordnete Fehlermenge darzustellen.
- In den Fehlerströmen der Literatur [11, 47, 78, 98, 187, 193] weisen Spezifikationsfehler einen mittleren Anteil von 20% auf (s. auch Kapitel 3.2).

4. Entwicklungen zur Testoptimierung

In diesem Kapitel wird aufgezeigt, wie Teststrategien für die Entwicklung von automobilen eingebetteten Systemen effektivitätssteigernd und kostenreduzierend optimiert werden können. In Kapitel 4.1 wird der betrachtete, zu optimierende Kostenbereich als *testinduzierte Qualitätskosten* definiert. Auf Basis der in Kapitel 2 und 3 durchgeführten Untersuchungen wird in Kapitel 4.2 ein Optimierungsprozess für Teststrategien abgeleitet. Dieser erfordert u. a. die prospektive Abschätzung der testinduzierten Qualitätskosten des Testprozesses. Für diese Zielsetzung wird in Kapitel 4.3 ein *Kostenmodell für testinduzierte Qualitätskosten* entwickelt. In Kapitel 4.4 bis 4.6 werden weitere, die Optimierung unterstützende Verfahren dargestellt, die jeweils unter Einbeziehung von Algorithmen aus dem Kostenmodell erarbeitet werden.

4.1. Testinduzierte Qualitätskosten

Allgemeine Abgrenzung: Betrachtet wird die Entwicklung von automobilen eingebetteten Systemen. In diesem Kontext stellen *testinduzierte Qualitätskosten* einen Ausschnitt der beim Hersteller anfallenden Qualitätskosten dar und umfassen

- Kosten für analytische, dynamische Qualitätssicherungsmaßnahmen, die für Vorbereitung und Durchführung von Tests sowie das zugehörige Testmanagement aufgewendet werden,
- Beseitigungskosten für Fehler, die in der analytischen, dynamischen Qualitätssicherung beim Hersteller oder anschließend durch den Kunden identifiziert werden.

Fehlerbeseitigungskosten: Fehlerbeseitigungen bewirken unmittelbar die angestrebte Qualitätssteigerung und stellen somit nach allgemeiner Auffassung zwar Qualitätskosten, jedoch keine Testkosten dar (s. Kapitel 3.1). Eine Fehlerbeseitigung umfasst nach der Fehleranalyse i. Allg. die Durchführung einer Systemänderung (Fehlerlokalisierung mit anschließender Fehlerkorrektur) sowie Korrektur- und Regressionstests. Die Kosten der Fehlerkorrektur steigen, wie in Kapitel 3.3 ausgeführt, mit fortschreitenden Testphasen an. Eine optimale Wahl von Testverfahren kann über eine frühzeitige Fehleridentifikation mit frühzeitiger Fehlerbeseitigung eine Senkung der Projektkosten erreichen. Alle hersteller- und kundeninduzierten Fehlerbeseitigungskosten (s. Kapitel 2.5.3) werden also durch die Effektivität angewendeter Testverfahren unmittelbar beeinflusst und werden daher in die testinduzierten Qualitätskosten einbezogen.

4.2. Prozess der Testoptimierung

Maßgebliche Eigenschaften des Testprozesses: Der Testprozess ist gemäß den Untersuchungen in Kapitel 2.2.4 ein Prozess mit mehreren, nacheinander ablaufenden Testphasen, in denen Fehlverhalten durch dynamische Testverfahren identifiziert wird. Die Testverfahren identifizieren Fehlverhalten verschiedener Fehlerklassen unterschiedlich effektiv. Die Fehlerursachen werden nach einer Kosten-Nutzen-Abwägung i. Allg beseitigt. Die Kosten der Fehlerbeseitigung steigen mit den Testphasen an (s. Kapitel 3.3).

Ziel und Ansatzpunkte der Optimierung: Die Kosten im Testprozess sollen bei verbesserter Fehleridentifikation reduziert werden. Der im Rahmen der Arbeit vorgeschlagene Optimierungsprozess basiert auf drei Ansatzpunkten zur Optimierung:

- Kostenreduktion durch Fehlerfrüherkennung mit Einsatz von effizienten Testverfahren,
- Einsatz von Testautomatisierungen,
- Isolierung und Reduktion von projektspezifischen Kostenschwerpunkten.

Die ersten beiden Punkte stellen allgemein anerkannte Optimierungsansätze dar. Die Identifikation von Kostenschwerpunkten, hier als Isolierung bezeichnet, kann sich aus einer kritisch wertenden Analyse von detaillierten Kostendarstellungen ergeben.

Optimierungsprozess: Für die systematische Optimierung von Teststrategien wird ein Vorgehensschema definiert, mit dem schrittweise eine projektspezifisch effektive und kostenreduzierende Teststrategie aus der Teststrategie eines abgeschlossenen Referenzprojekts hergeleitet werden kann. Die neue Teststrategie ist optimiert in Bezug auf die des Referenzprojekts.

Unterstützende Verfahren: Zur Durchführung der einzelnen Optimierungsschritte sind Kosteneinsparpotentiale zu berechnen, Fehlerklassen zu bilden, Testverfahren und Automatisierungen auszuwählen und vor allem testinduzierte Qualitätskosten zu simulieren. Für letzteres wird in Kapitel 4.3 ein Qualitätskostenmodell entwickelt. Aus diesem werden weitere Verfahren abgeleitet. Insgesamt finden folgende Verfahren Anwendung (Kapitel der Entwicklung in Klammer):

- Kostenmodell für testinduzierte Qualitätskosten (Kapitel 4.3),
- Fehlerklassenorientierte Testverfahrensauswahl mit testfokusorientierter Fehlerklassifikation und testzielorientierter Fehlerklassengewichtung (Kapitel 4.4),
- Kosteneinsparpotentiale durch Früherkennung (Kapitel 4.5),
- Break-Even-Point von Testautomatisierungen (Kapitel 4.6).

Vorgehensschema im Optimierungsprozess: Der Optimierungsprozess umfasst eine Abfolge von mehreren Tätigkeitskomplexen (Hinweis auf die unterstützenden Verfahren in Klammern):

1. Quantifizierung der testinduzierten Ist-Qualitätskosten

Als Ausgangsbasis werden die testinduzierten Ist-Qualitätskosten des betrachteten Referenzprojekts bestimmt (Kostenmodell für testinduzierte Qualitätskosten). Es ergeben sich die Gesamtsumme der aufgewendeten Kosten und eine detaillierte Verteilung der Kosten auf Kostengruppen und -elemente. Anhand dessen können projektspezifische Testziele abgeleitet werden.

2. Aufzeigen von Einsparpotentialen durch Fehlerfrüherkennung

Es werden im Referenzprojekt fehlerklassenorientiert Kosteneinsparpotentiale durch Fehlerfrüherkennung unter Berücksichtigung der Verteilung der Fehler auf Fehlerklassen ermittelt (Einsparpotentiale durch Fehlerfrüherkennung). Existieren keine ökonomisch interessanten Kosteneinsparpotentiale, kann keine Optimierung vorgeschlagen werden und der Optimierungsschritt endet. Wahlweise kann zur ersten Orientierung der fehlerklassenorientierten Betrachtung von Einsparpotentialen eine weniger aufwändige, aber realitätsfernere Ermittlung von maximalen Einsparpotentialen voran gehen.

3. Kostenreduktion durch Fehlerfrüherkennung

Im ersten Optimierungsschritt sollen die Fehlerbeseitigungskosten durch Fehlerfrüherkennung über die Veränderung der Fehlerströme reduziert werden, d. h. es sollen Fehlerklassen mit Kosteneinsparpotential für Früherkennung durch Testverfahren adressiert werden. Die Ausgangsbasis bildet die Fehlerverteilung des Referenzprojekts, die im zweiten Schritt des Optimierungsprozesses erstellt wird. Sie kann mit einem vom Testmanagement definierten Testziel kombiniert werden. Das Testziel wird zur Gewichtung eingesetzt (testzielorientierte Fehlerklassengewichtung). Aus den Klassenverteilungen ergeben sich Kategorien von einzusetzenden Testverfahren (Auswahl von Testverfahren). Die testinduzierten Qualitätskosten aus dem Einsatz der so vorgeschlagenen Testverfahren werden simuliert (Kostenmodell für testinduzierte Qualitätskosten).

4. Kostenreduktion durch Testautomatisierung

Mögliche Testautomatisierungen sind durch Inspektion der eingesetzten Teststrategie zu identifizieren. Die Kostenwirkung der Automatisierungen ist durch Break-Even-Berechnungen oder durch Simulation der resultierenden testinduzierten Qualitätskosten zu prüfen (Break-Even-Point von Testautomatisierungen, Kostenmodell für testinduzierte Qualitätskosten).

5. Isolierung von Kostenschwerpunkten

Die detailliert dargestellten testinduzierten Ist-Qualitätskosten lassen ggf. kostenintensive, projektspezifische Kostenschwerpunkte erkennen, für die individuelle Lösungen angestrebt werden können. Die Kostenwirkungen lassen sich dann ebenfalls simulieren (Kostenmodell für testinduziertes Qualitätskosten). Die Isolierung

von Kostenschwerpunkten kann an beliebiger Stelle nach der Ist-Kostenerstellung erfolgen.

Diese Vorgehensweise bildet den allgemeinen Rahmen für die hier dargestellten, projektspezifischen Strategieoptimierungen.

Erforderliche Datenbasis: Grundlage der Optimierung bildet, wie erwähnt, die *Analyse der Ist-Kosten* eines abgeschlossenen Projekts. Die Ist-Kosten sind als Abbild der eingesetzten Teststrategie aufzufassen. Für die Ist-Kostenermittlung werden feingranulare Informationen zu den Kosten sowie zu den identifizierten Fehlern benötigt. Die Kosten sollten als detaillierte Budgetpläne vorliegen und die Plandaten um die tatsächlich entstandenen Kosten ergänzt sein. Für die identifizierten Fehler sollten Fehlercharakteristika zur Fehlerklassenbestimmung, Phasen der Fehleridentifikation und -beseitigung sowie Aufwandsangaben zur resultierenden Fehlerbeseitigung nach Fehlerlokalisierung, -korrektur, Retest vorliegen. Diese Angaben können in Datenbanken in Form von Fehlermeldungen gespeichert sein. In der beobachteten Projektpraxis ist die Datenarchivierung allerdings weniger differenziert oder unvollständig. In diesem Fall müssen die fehlenden Werte durch Abschätzungen substituiert werden.

4.3. Kostenmodell für testinduzierte Qualitätskosten

Ziel der Modellbildung: Ziel der Modellbildung ist es, die testinduzierten Qualitätskosten des betrachteten Testprozesses zu bestimmen. Das Modell zur Bestimmung der testinduzierten Qualitätskosten soll retrospektiv exakte Ergebnisse liefern sowie prospektive Abschätzungen ermöglichen. Die Modellstruktur soll dabei der Kostenstruktur des Testprozesses, wie in Kapitel 2.5 dargestellt, folgen und sich im Besonderen für den Einsatz bei Optimierungen von Teststrategien eignen.

4.3.1. Struktur des Testprozesses

Phasenorientiertes Vorgehensmodell: Die Optimierung von Teststrategien betrachtet vorrangig den Testprozess, also die Struktur auf der rechten Seite des in Kapitel 2.2.1 dargestellten V-Modells. Der Testprozess beinhaltet eine Abfolge von Testphasen, in denen jeweils mehrere Testverfahren mit unterschiedlichen Zielsetzungen durchgeführt werden können [241]. Der Testprozess kann in mehreren Iterationen durchlaufen werden, wobei nicht zwingend alle Testphasen wiederholt werden [241]. Die Struktur des Testprozesses orientiert sich also an einem phasenorientierten Vorgehensmodell, das iterativ durchlaufen werden kann.

Gruppierung der Testphasen: Der Testprozess beinhaltet in derzeitiger Praxis entsprechend Abbildung 2.2 in Kapitel 2.2.1 die Testphasen ab Modultest bis Akzeptanztest. Der Feldeinsatz stellt primär keine Testphase sondern die letzte Projektphase dar. Da jedoch, wie in Kapitel 2.5 begründet, die Fehlerbeseitigungskosten der Feldfehler in die testinduzierten Qualitätskosten eingehen, wird der Feldeinsatz hier

den Testphasen zugerechnet.

Testinduzierte Qualitätskosten entstehen mit der Abarbeitung der Testphasen. Einige Testphasen werden immer beim entwickelnden Unternehmen durchgeführt. In dieser Arbeit sind dies Modultest, Systemtest und die zugehörigen Integrationstests. Diese Phasen werden in der hier vorgenommenen Modellbildung als *Testphasen* TP_1 bezeichnet. Die Testphasen Akzeptanztest und Feldeinsatz finden in der Regel nicht beim Hersteller statt und werden hier als *Testphasen* TP_2 bezeichnet. Der Fahrzeugtest mit Fahrzeugintegrationstest kann projektabhängig beim Hersteller oder beim Kunden angesiedelt sein. In den untersuchten Projekten fand er beim Kunden statt und wird hier bei globaler Betrachtung den *Testphasen* TP_2 zugerechnet. Die Gruppierung der Testphasen zu TP_1 und TP_2 wird im Kostenmodell für testinduzierte Qualitätskosten strukturell realisiert.

4.3.2. Abbildung der Kostenstruktur

Qualitätskostenkategorien: Die in Kapitel 2.5 dargelegte Unterteilung in *Herstellerinduzierte Qualitätskosten* und *Kundeninduzierte Qualitätskosten* wird in dem Qualitätskostenmodell als oberste strukturelle Trennung als sog. *Qualitätskostenkategorie* realisiert, s. Abbildung 4.1. Dies entspricht einer Gliederung anhand der Testphasen TP_1 und TP_2, s. Kapitel 4.3.1.

Hauptkostengruppen: Die herstellerinduzierten Qualitätskosten werden in die Aktivitäten *Testmanagement, Testaufbau, Testvorbereitung* und *Testdurchführung* gegliedert (s. Abbildung 4.1). Hinzugefügt werden, wie in Kapitel 2.5.3 ausgeführt, die Kosten für die *Herstellerinduzierte Fehlerbeseitigung*. Den kundeninduzierten Qualitätskosten wird die *Kundeninduzierte Fehlerbeseitigung* untergeordnet. Somit sind alle Fehlerbeseitigungen bis inkl. Feldeinsatz inkludiert.

Fehlervermeidung als formale Hauptkostengruppe: Die dem Testprozess vorangegangene konstruktive und statische analytische Qualitätssicherung mit entsprechend frühzeitiger Fehlerbeseitigung kann die Qualitätskosten des Testprozesses, z. B. durch vorweggenommene Fehlerbeseitigungskosten, beeinflussen. Eine Optimierung der konstruktiven Qualitätssicherung ist hier jedoch nicht Ziel der Optimierungen und somit wird die konstruktive Qualitätssicherung zur wahlfreien Integration mit dem Platzhalter *Fehlervermeidung* in das Kostenmodell aufgenommen. Die Kosten der *Fehlervermeidung* werden formal als aggregierte Hauptkostengruppe entsprechend dem chronologischen Entwicklungsablauf zwischen Testmanagement und Testaufbau als Platzhalter eingefügt, ohne jedoch in der Kostensimulation berücksichtigt zu werden. Die aktuelle Qualität der frühzeitigen Qualitätssicherung geht indirekt durch die Bezugnahme auf die Fehlerklassen des Referenzprojekts ein.

Fehlerfolgekosten als formale Hauptkostengruppe: *Fehlerfolgekosten* (s. Kapitel 2.5) treten nur sporadisch und schwer vorhersehbar auf. Die monetären Auswirkungen sind somit nur bedingt vorhersagbar, liegen in der betrachteten Praxis nicht vor und eignen sich kaum für prospektive Abschätzungen. Fehlerfolgekosten werden

in der prospektiven Berechnung von testinduzierten Qualitätskosten daher in dieser Arbeit im Weiteren nur marginal betrachtet. Im Qualitätskostenmodell werden *Fehlerfolgekosten* als Platzhalter für eine Hauptkostengruppe den *Kundeninduzierten Qualitätskosten* angefügt, ohne jedoch in der Kostensimulation berücksichtigt zu werden. Somit bietet das Modell die Möglichkeit, die Kosten bei entsprechendem Kenntnisstand und Bedarf einzubeziehen.

Verfeinerung top-down: Die Hauptkostengruppen werden wie in Kapitel 2.5 beschrieben top-down über Kostengruppen zu Kostenelementen untergliedert. Es ergibt sich die Kostenstruktur in Abbildung 4.1.

Differenzierungstiefen: Bei der Untergliederung der Kosten werden die Differenzierungstiefen in den betrachteten Kostenbereichen zielbezogen bestimmt. Das bedeutet, dass bei Kostenelementen, die für die Optimierung des Testprozesses sehr wichtig sind, eine größere Detaillierung vorgenommen wird als bei weniger wichtigen. So wird z. B. in der Fehlerbeseitigung bis hinunter zu den Bearbeitungsschritten eines Einzelfehlers differenziert, die Kosten für einen Prüfstand jedoch nicht untergliedert.

Vorteile des sukzessiven Differenzierens: Das sukzessive Differenzieren der Kostenstruktur macht die einzelnen Schritte nachvollziehbar. Es unterstützt das Einordnen der Kosten und vermeidet am ehesten Inkonsistenzen und Doppelzählungen. Bei der Modellanwendung kann dann wiederum die Detaillierungstiefe für die Datenerfassung und Kostendarstellung anwendungsorientiert begrenzt werden. Das heißt, es ist zu entscheiden, ob bei bestimmten Teilkosten von der untersten Ebene auf eine höhere Ebene aggregiert wird, z. B. die Fehlerbeseitigungskosten in der Gesamtheit betrachtet werden, die Testfallerstellung aber differenziert wird. Diese Variationsmöglichkeiten tragen u. a. auch der Praxis der Datenarchivierung mit nicht immer vollständigen bzw. hinreichend detaillierten Daten Rechnung oder ermöglichen eine Steuerung des Rechnungsaufwands entsprechend den Ansatzpunkten für die aktuell intendierte Optimierung. Die schrittweise verfeinernde Abbildung der Kostenstruktur macht das Modell offen für Anpassungen, z. B. aufgrund veränderter Entwicklungsprozesse.

Qualitätseigenschaften der Abbildung: Die definierten Kostenelemente sind bedingt durch die sequentielle Top-Down-Verfeinerung und die strukturelle Gliederung nach Testphasen und Iterationen disjunkt. Die Menge der Kostenelemente bildet in den untersuchten Quellen die vorhandenen Kostenarten vollständig ab. Für das Testmanagement lagen keine Daten vor, hier wurden zwei Kostengruppen als Platzhalter für projektspezifische Verfeinerungen vorgegeben.

Fehlersuche, Fehleridentifikation und Fehlerbeseitigung: Der Test wird häufig in Fehlersuche (Provozieren von Fehlverhalten), Fehleridentifikation (Erkennen von Fehlverhalten) und Fehlerbeseitigung (Korrektur der Fehlerursache) unterteilt. Fehleridentifikation und Fehlerbeseitigung (herstellerinduziert sowie kundeninduziert) sind im Kostenmodell für testinduzierte Qualitätskosten direkt abgebildet (s. Abbildung 4.1). Die Dokumentation der Fehleridentifikation erfolgt in den Aktivitäten des Fehlermanagements, das hier strukturell (da nur im Fehlerfall) der Fehlerbeseitigung zugeordnet ist. Somit beinhalten Testaufbau, Testvorbereitung und Testdurchführung

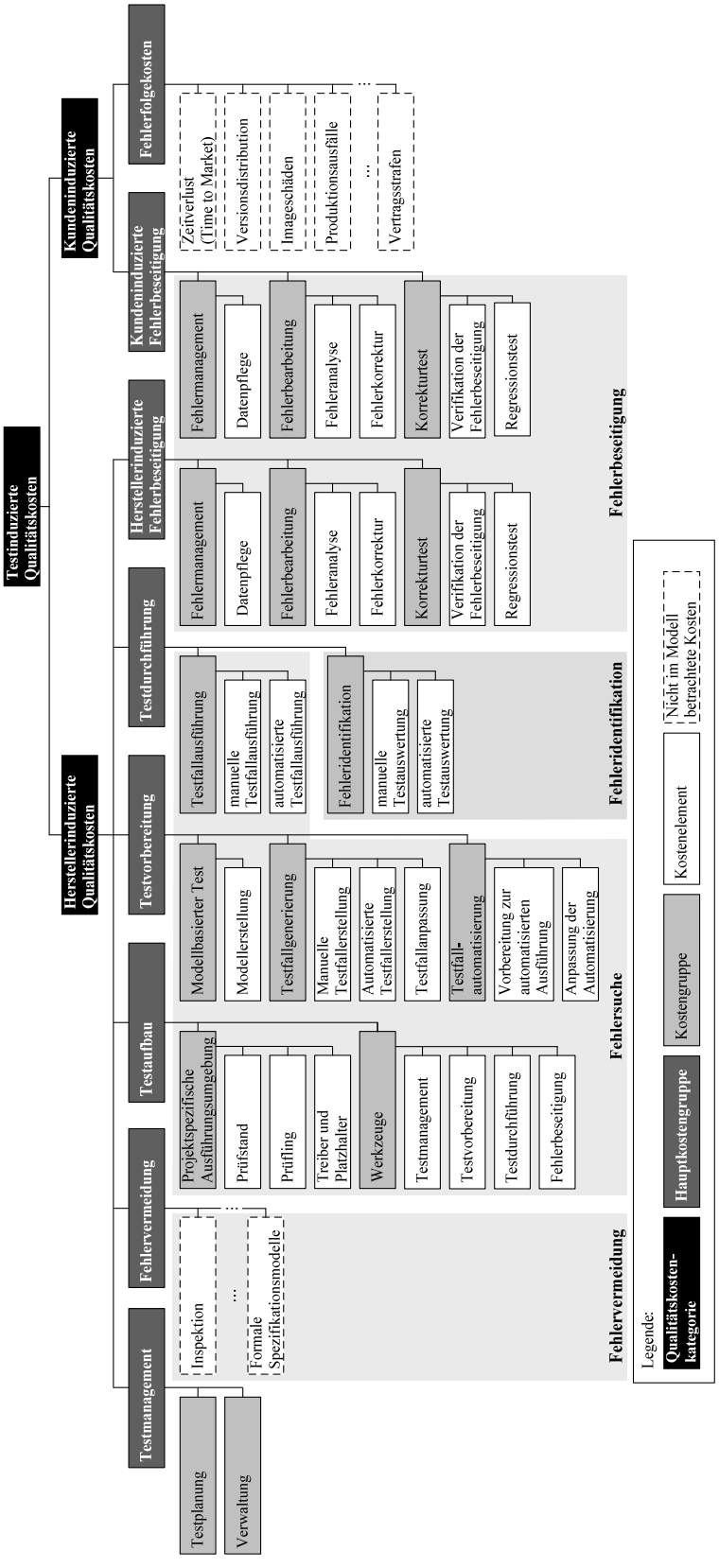


Abbildung 4.1.: Kostenelemente der testinduzierten Qualitätskosten

Kosten der Fehlersuche, die unabhängig von einer Fehleridentifikation anfallen. Fehlerbeseitigungskosten hingegen fallen nur bei einer Fehleridentifikation an.

4.3.3. Algorithmen

Indizierung der Kostenelemente: Spillner [241] empfiehlt für größere Projekte eine Kostenerfassung je Testphase und Iteration. Dieser Forderung kann in den Kostengruppen und -elementen nachgekommen werden, indem die Kostengruppen und - elemente des Modells, z. B. das Kostenelement *Manuelle Testausführung,* zwar für alle Testphasen in allen durchgeführten Iterationen gelten, aber wiederholt zu berücksichtigen sind. Daher ist für die algorithmische Unterscheidung der Kosten eine Indizierung der Kostenelemente eingeführt, s. Anhang A.1. Im Besonderen werden die Indizes i für iterative Wiederholungen des Testprozesses und p für die Testphasen verwendet. Das Qualitätskostenmodell schlüsselt weiterhin nach Testverfahren auf, die mit dem Index m bezeichnet werden.

Addition bottom-up: Die Berechnung der Qualitätskosten erfolgt durch schrittweise Addition der Kostenelemente bottom-up. Die schrittweise Addition ermöglicht eine Übersicht und Wertung der testinduzierten Qualitätskosten je Kostengruppe. Das Modell, s. Anhang A.2, wird allerdings stufenweise top-down verfeinernd von den Qualitätskostenkategorien zu den Kostenelementen dargestellt, beginnend mit

$$K^{\textit{Testinduzierte Qualitätskosten}} = K^{\textit{Herstellerinduzierte Qualitätskosten}} + K^{\textit{Kundeninduzierte Qualitätskosten}} \tag{4.1}$$

(s. auch Gleichung (A.1) in Anhang A.2)

Die Algorithmen sind dadurch gut überschaubar und lassen die logische Struktur der Berechnung und die zu betrachtenden Komponenten deutlich erkennen. Sind gewisse Kostenelemente nicht mit Werten belegt, z. B. weil keine Testautomatisierung durchgeführt wurde, sind diese Elemente mit 0 zu belegen und mathematisch als neutrale Elemente in die Kostenberechnung einzubeziehen.

Algorithmus für exakte Qualitätskostenberechnung: Im Anhang A.2 wird zunächst ein Algorithmus für die exakte Berechnung von testinduzierten Qualitätskosten entwickelt. Dieser Algorithmus stellt eine feingranulare Summe über alle Kosten dar. Feingranular bedeutet, dass die angefallenen Kosten für jeden Testfall und jeden identifizierten Fehler explizit eingehen, z. B. wird für jeden Testfall der spezifische Aufwand zur Erstellung herangezogen. Diese Aufwandsangaben werden dann im ersten Aggregationsschritt je Testverfahren aufsummiert.

Aufgrund der Anforderungen an die Detaillierung der Datenbasis ist der Algorithmus in der Praxis kaum für retrospektive Berechnungen denkbar und beinhaltet einen

erheblichen Berechnungsaufwand. Aber er kann als Ausgangs- und Bezugspunkt für die Ableitung weiterer Algorithmen dienen.

Vereinfachte Algorithmen für exakte Qualitätskostenberechnung:. Im Anhang A.3 werden auf dem exakten Algorithmus aufbauend vereinfachte Quantifizierungsalgorithmen hergeleitet. Der Begriff *vereinfacht* ist hier als Bezeichnung für Abstraktion bzw. als Synonym für *verallgemeinert* anzusehen. Die Berechnungen basieren hier auf stärker aggregierten Daten. Es wird z. B. je Testverfahren der Erstellungsaufwand für die Testfälle als ein Produkt aus Testfallanzahl und mittlerem Aufwand für die Erstellung eines Testfalls dargestellt. Der *mittlere Aufwand* steht hier und im Folgenden bei retrospektiven Betrachtungen stellvertretend für den arithmetischen Mittelwert. Die Anwendung des arithmetischen Mittelwertes ermöglicht ein exaktes Berechnungsergebnis. Exakte Mittelwerte können nur bei retrospektiven Betrachtungen zur Verfügung stehen. Liegen in der Praxis die für die retrospektiv vereinfachte Berechnung vorgesehenen Werte nicht vor, kann u. U. auf vorhandene Werte in höherer Aggregation übergegangen werden. Hieraus können ebenfalls exakte Ergebnisse erzielt werden.

Vereinfachte Algorithmen für retrospektiv geschätzte Qualitätskosten: Liegen in der Praxis keine Werte für bestimmte Bereiche des Testprozesses vor, können fehlende Werte abgeschätzt werden. Die Algorithmen für die retrospektiv geschätzte Qualitätskostenermittlung sind in Anhang A.3 dargestellt. Die ermittelten Qualitätskosten ergeben in diesem Fall keinen exakten sondern einen geschätzten Wert.

Vereinfachte Algorithmen für prospektiv geschätzte Qualitätskosten: Prospektive Qualitätskosten sind immer geschätzte Größen. Die Abschätzung erfolgt über die vereinfachten Algorithmen unter Verwendung von Abschätzungen. Bei prospektiven Abschätzungen können der *mittlere Aufwand*, Anzahlen und Anteile aufgrund von Verteilungsüberlegungen aus statistischen Kennzahlen (z. B. arithmetischer Mittelwert, Median) abgeleitet werden oder auf Basis von Erfahrungen geschätzt werden (zur Güte von Schätzwerten s. Kapitel 5.3). Die Ableitungen sind im Anhang A.3.1 dargestellt.

4.3.4. Anwendungen

Das Qualitätskostenmodell erlaubt sowohl die retrospektive als auch die prospektive Berechnung von testinduzierten Qualitätskosten. Die Quantifizierung von testinduzierten Qualitätskosten ist die grundlegende Voraussetzung für die *Optimierung der Teststrategien* [70]. Hierbei steht die prospektive Kostenabschätzung im Mittelpunkt der Betrachtungen.

Das Qualitätskostenmodell stellt die Grundlage für alle Qualitätskostenermittlungen für den betrachteten Optimierungsprozess (s. Kapitel 4.2) dar. Ab Kapitel 4.4 werden zur Unterstützung des Optimierungsprozesses weitere Verfahren gebildet, die auf Algorithmen aus dem Qualitätskostenmodell aufbauen. Somit ergeben sich insgesamt folgende Anwendungsmöglichkeiten für Algorithmen des Qualitätskostenmodells (in

Klammern steht ein Verweis auf das anzuwendende Verfahren und das Kapitel, in dem das Verfahren hergeleitet wird):

- retrospektive Berechung von testinduzierten Qualitätskosten (mittels Qualitätskostenmodell aus Kapitel 4.3.3),
- prospektive Simulation von testinduzierten Qualitätskosten (mittels Qualitätskostenmodell aus Kapitel 4.3.3),
- Optimierung der Auswahl von Testverfahren (mittels abgeleitetem Verfahren zur Fehlerfrüherkennung durch eine testzielorientierte Fehlerklassengewichtung aus Kapitel 4.4),
- Berechnung von Kosteneinsparpotentialen durch Fehlerfrüherkennung (mittels abgeleitetem Verfahren aus Kapitel 4.5),
- Ermittlung eines Break-Even-Point für Automatisierungen (mittels abgeleitetem Verfahren aus Kapitel 4.6),
- Identifikation von Kostenschwerpunkten aufgrund von projektspezifischen Konstellationen durch detaillierte Kostendarstellungen (mittels Qualitätskostenmodell aus Kapitel 4.3).

Die genannten Anwendungsmöglichkeiten werden in den Kapiteln 6 und 7 zu Kostenoptimierungen eingesetzt.

4.3.5. Aspekte zur Validierung

Beschränkungen des Kostenmodells für testinduzierte Qualitätskosten: Das Kostenmodell für testinduzierte Qualitätskosten ist wie die meisten Qualitätskostenmodelle in der Literatur nicht allumfassend konzipiert [157]. Es bildet einen Ausschnitt der Qualitätskosten zielgerichtet ab, der durch vier Beschränkungen abgegrenzt ist:

- testinduzierte Kosten,
- Entwicklungen eingebetteter Systeme,
- iteratives Durchlaufen eines phasenorientierten Vorgehensmodells,
- Differenzierung der Kosten im Hinblick auf das Ziel der Teststrategieoptimierung.

Die ersten beiden Beschränkungen wirken sich vor allem auf die Auswahl der zu betrachtenden Kostenelemente, die letzten beiden auf die Modellstruktur aus. Es werden u. a. die Kostenbereiche stärker detailliert, die ggf. durch Optimierungsmaßnahmen beeinflusst werden können, wie z. B. Testfallerstellungskosten. Durch Teststrategien nicht beeinflussbare Bereiche bleiben undifferenziert.

Das Hauptziel des Kostenmodells ist weder die reine Kostenprognose noch die Kostenkontrolle sondern die strukturierte Darstellung von Kosten eines abgeschlossenen Projekts, das als Referenzprojekt dient. An dessen testinduzierten Qualitätskosten werden Kostenschwerpunkte identifiziert und optimierende Teststrategien abgeleitet. Die optimierten Teststrategien dienen zur Entwicklung von künftigen, vergleichbaren Projekten.

Validierung von Kostenmodellen der Literatur: Publizierte Qualitätskostenmodelle wurden weitgehend generisch entwickelt, eine industrielle Validierung anhand von Projektdaten ist nur sehr vereinzelt zu finden [157]. Zuweilen durchgeführte konstruierte empirische Validierungen werden nicht als Beweis für ein valides Modell angesehen [157, 186].

Teilvalidierung des entwickelten Kostenmodells anhand eines Projekts: Das hier beschriebene Kostenmodells für testinduzierte Qualitätskosten konnte aufgrund der Datenlage an einem Praxisprojekt validiert werden. Das Kostenmodell wurde für die Bestimmung der testinduzierten Ist-Qualitätskosten eines Entwicklungsprojekts und für darauf aufbauende Kostensimulation für mehreren Strategievarianten eingesetzt und lieferte überprüfbare und nachvollziehbare, valide Ergebnisse mit der Besonderheit, dass die Datenbasis des Bezugsprojekts nicht vollständig war und Kostenelemente abgeschätzt werden mussten. Für eine weitergehende Validierung wäre eine Anwendung des testinduzierten Qualitätskostenmodells auf weitere Projekte mit detaillierten Budgetdaten notwendig. Wegen mangelnder Projektdaten bleibt die Frage einer projektübergreifenden Validität offen.

Teilvalidierung der Auswahl der Kostenelemente: Die Ist-Qualitätskosten-Ermittlung beruht entsprechend dem Cost-Accounting-Ansatz [157] auf einem Herleitungsprozess mit

- Abgrenzung des betrachteten Qualitätskostenbereiches (Testprozess im iterativ durchlaufenem, phasenorientierten Vorgehensmodell),
- Identifikation der beteiligten Organisationseinheiten (schwerpunktmäßig Testteams, punktuell ergänzt um Kunden- und Entwicklertätigkeiten),
- Identifikation der kostenrelevanten Aktivitäten (Testmanagement bis Fehlerbeseitigung, s. Kapitel 2.5),
- Detailanalyse des Test- und Fehlerbeseitigungsvorgangs (s. Kapitel 2.5.3) mit Ermittlung der Kostenelemente in unterschiedlichem Detaillierungsgrad.

Das Vorhandensein der wichtigsten Kostenelemente ist – wie in Kapitel 2.5 ausgeführt – durch die Analyse von industriellen Projekten in Vereinigung mit einem Abgleich mit existierenden Qualitätskostenmodellen (s. Kapitel 3.1) überprüft und damit ansatzweise validiert. Die Auswahl der übernommenen Kostenelemente wurde immer an den oben genannten Beschränkungen bemessen, also eher eingeschränkt auf den Testprozess und erweitert für die Entwicklung von eingebetteten Systemen (z. B. Einbeziehung von Prüfständen). Der Bezug auf die eingebetteten Systeme stellt per se eine Beschränkung auf bestimmte Entwicklungsverfahren dar, u. U. auch auf die Automobilindustrie.

Basis zur Qualitätskostenberechnung bilden immer alle relevanten Kostenarten der betrachten Entwicklungsphasen eines abgeschlossenen Projekts. Diese vorliegenden, testinduzierten Kosten gehen konzeptionsgemäß vollständig in die Betrachtung ein. Es mag wohl Interpretationsspielräume geben, welchen Kostengruppen vorliegende Kosten zuzurechnen sind, jedoch sollten sie vollständig erfasst sein.

Lassen sich testinduzierte Kosten eines Projekts nicht unmittelbar in Kostenberechung einbeziehen, so ist zu prüfen, ob die explizite Abbildung der Kosten für die intendierte Verwendung des Kostenmodells für testinduzierte Qualitätskosten relevant ist und weitere Kostengruppen bzw. -elemente in das Modell aufgenommen werden sollten. Aufgrund des Konzeptes der stufenweisen Verfeinerung bedeutet die Aufnahme weiterer Kostenelemente, dass in einer Detaillierungsebene feiner als bisher differenziert werden muss, also ein vorhandener Ast feingranularer aufgesplittet wird. Nicht durch die Optimierung des Testprozesses vorrangig beeinflussbare Kosten können in der Regel einem bereits existierenden Kostenelement als unbetrachtete Verfeinerung (da nicht zielrelevant) zugeordnet werden, so z. B. Stromkosten für den Prüfstand dem Kostenelement *Prüfstand*.

Qualität der verwendeten Daten: Wie bei allen Modellen, wird auch die Validität der Ergebnisse des Kostenmodells für testinduzierte Qualitätskosten entscheidend von der Qualität der Eingangsdaten beeinflusst. Diese müssen geeignet differenziert und exakt erfasst sein. Müssen fehlende Daten abgeschätzt werden, wird das Ergebnis von der Qualität der Schätzungen abhängig.

Ein valides Qualitätskostenmodell kann sowohl zutreffende als auch nicht zutreffende Prognosen für die Qualitätskosten künftiger Projekte liefern. Die Grenzen der Übertragbarkeit werden maßgeblich durch die Repräsentativität des Referenzprojekts bestimmt. Hierfür sind die Einflussfaktoren (z. B. Systemtyp, wie später gezeigt wird) auf die testinduzierten Qualitätskosten zu bestimmen.

4.3.6. Bewertung

Das in diesem Kapitel entwickelte Kostenmodell für testinduzierte Qualitätskosten beinhaltet alle in den untersuchten Quellen identifizierten und relevanten testinduzierten Kostenelemente und beschreibt deren Abhängigkeiten. Das Qualitätskostenmodell berücksichtigt die iterative Anwendung eines phasenorientierten Vorgehensmodells (z. B. des V-Modells).

Der Aufbau des Kostenmodells für testinduzierte Qualitätskosten basiert auf der Analyse des Testprozesses im V-Modell, der Analyse von Qualitätskostenmodellen in der Literatur, der Analyse von Projektbudgetplänen und Projektabläufen sowie Expertengesprächen mit Projektmanagern, Testmanagern und Testern. Daraus leiten sich die Hauptkostengruppen *Testaufbau, Testvorbereitung, Testdurchführung, Herstellerinduzierte Fehlerbeseitigung* und *Kundenindizierte Fehlerbeseitigung* ab. Diese werden durch das *Testmanagement* ergänzt. Die Kosten der Fehlerbeseitigung, speziell die der kundeninduzierten Fehlerbeseitigung, stehen in engem Zusammenhang mit der Testqualität und werden durch diese beeinflusst. Ein weiterer Zusammenhang besteht zur Fehlervermeidung und zu analytischen Qualitätssicherungsmaßnahmen in den Entwicklungsphasen. Da dieser Bereich der Qualitätssicherung aus dargelegten Gründen nicht Ziel der hier betrachtenden Optimierung ist, wird er nicht integriert, sondern nur zur wahlfreien Integration angedeutet. Ebenso stehen die Fehlerfolgekosten nicht im Zentrum der Betrachtungen, dennoch sind sie der Vollständigkeit

halber angedeutet. Wegen der additiven Verknüpfung der Kostenelemente kann die Zusammenstellung der Kostenelemente bei Änderung des Entwicklungsablaufs oder für die Übertragung auf andere Fachgebiete mit geringem Aufwand problemadäquat angepasst werden.

Das Kostenmodell für testinduzierte Qualitätskosten kann nur durch Berechnung der Ist-Kosten von Projekten und durch Kostensimulationen zu diesen Projekten validiert werden. Die Möglichkeiten zur Validierung und die Validierung in Teilaspekten werden diskutiert.

Das Qualitätskostenmodell bildet die Grundlage für die Entwicklungen in den nachfolgenden Kapiteln.

4.4. Fehlerklassenorientierte Testverfahrensauswahl

Die hier eingeführte *fehlerklassenorientierte Testverfahrensauswahl* umfasst eine *testfokusorientierte Fehlerklassifikation*, fakultativ eine *testzielorientierte Fehlerklassengewichtung* und eine *Auswahl der Testverfahren*. Ziel der fehlerklassenorientierten Testverfahrensauswahl ist es, die Teststrategie für ein künftiges Projekt bedarfsgerecht zu optimieren. Es werden Testverfahren erkannt, die fokussierte Fehlerklassen frühestmöglich, d. h. bei Erreichen der erforderlichen Systemabstraktionsebene, adressieren und damit das Testziel umsetzen.

Kern der Entwicklungen in diesem Kapitel ist eine testfokusorientierte Fehlerklassifikation (Kapitel 4.4.1). Sie gruppiert Fehlerursachen anhand des Erscheinungsbildes derart, dass den Fehlerklassen jeweils genau eine Testphase zugeordnet werden kann, in der die Fehlerursachen frühestmöglich durch Testverfahren adressiert werden können. Den Fehlerklassen werden Testverfahren zugewiesen, die das Fehlverhalten identifizieren. Die Fehlerfrüherkennung kann kombiniert werden mit einem Testziel, das festlegt, welche Fehlerklassen oder -eigenschaften vorrangig identifiziert werden sollen. Die *testzielorientierte Fehlerklassengewichtung* gewichtet hierzu die klassifizierten Fehlerursachen entsprechend dem Testziel (Kapitel 4.4.2). Anschließend erfolgt die *Auswahl der Testverfahren* fehlerklassenbasiert (Kapitel 4.3.3).

4.4.1. Testfokusorientierte Fehlerklassifikation

Fokus der Anwendung: Die testfokusorientierte Fehlerklassifikation strebt die Optimierung des Testprozesses durch Fehlerfrüherkennung an. Jede Fehlerklasse ist das Abbild von Testfoki der zugeordneten Testverfahren und der Testphase, die die zum Test erforderliche Systemabstraktionsebene angibt. Klassifiziert werden die Fehlerursachen anhand von Fehlereigenschaften, wie sie in Fehlermeldungen nach erfolgter Fehlerbeseitigung in Fehlerdatenbanken festgehalten sind. Aus der Klassifikation ergibt sich eine Fehlerverteilung über die definierten Klassen.

Die testfokusorientierte Fehlerklassifikation ist in dieser Arbeit die Grundlage für Verfahren zur Berechung von Kosteneinsparpotentialen (Kapitel 4.5), für die testzielorientierte Fehlerklassengewichtung (Kapitel 4.4.2) und für die Testverfahrensauswahl zur Nutzung der Kosteneinsparpotentiale (Kapitel 4.4.3).

Klassifikationsziel, Anforderungen: Die Klassenbildung soll nachfolgenden Anforderungen genügen:

- Den Fehlerklassen soll jeweils genau eine *Testphase* (von Modultest bis Fahrzeugtest) zugewiesen werden können, in der die zugeordneten Fehler frühestmöglich durch Qualitätssicherungsmaßnahmen, hier Testverfahren des Testprozesses, adressiert werden können.
- Den Fehlerklassen sollen *Kategorien von Testverfahren* zugeordnet werden können, die Fehlverhalten der entsprechenden Klasse in der zugeordneten Testphase identifizieren. Unter einer Kategorie von Testverfahren wird eine Gruppierung von Testverfahren mit gleichem Fokus auf die Fehlereigenschaften verstanden.
- Die Fehlerklassen sollen *alle potentiell möglichen Fehlertypen* berücksichtigen.
- Die Klassenbildung ist *disjunkt* zu formulieren.

Die Bezugnahme auf die Testphase ist von grundlegender Bedeutung, da die Testphasen Abstraktionsebenen darstellen und die Abstraktionsebene ein Klassifikationskriterium für Testverfahren ist (s. Kapitel 2.4.1). Im Falle der Ermittlung von Kosteneinsparpotentialen ist der Bezug auf die Testphase aufgrund der Abhängigkeit der Fehlerbeseitigungskosten von den Testphasen erforderlich, im Falle der Strategieoptimierung für die Aussage, in welcher Phase verstärkt zu testen ist.

Erkenntnisstand zu Fehlerklassifikationen: In der Literatur sind verschiedene Fehlerklassifizierungen beschrieben, so z. B. das *Orthogonal Defect Classification Scheme* (ODC) von IBM [53], das *Hewlett-Packard Scheme* [111], die *Classification for Software Anomalie* als IEEE-Standard der NIST [138] oder die Klassifikationen von Beizer [15] und Breitling [35]. Diese genannten Klassifikationen dienen hauptsächlich der Ursachenanalyse. Das heißt, Ursachen beschreibende Klassen, wie z. B. fehlerhafte Berechnung oder Schreibfehler, stellen das Endergebnis dar. Diese Klassifizierungen zielen damit nicht primär, wie hier gefordert, auf eine Zuordenbarkeit zu Kategorien von Testverfahren und -phasen. Verschiedene Arbeiten zeigen jedoch, dass Testverfahren Fehlverhalten bestimmter Fehlerklassen vorrangig identifizieren [12, 26, 98].

Ableitung aus ODC: Da die genannten Fehlerklassifikationen nicht den oben beschriebenen Anforderungen genügen, wird eine Klassifikation entwickelt, die aus zwei hier sinngebenden, orthogonalen Attributen der ODC von IBM abgeleitet wird und zwar den Attributen *Target* und *Defect Type*. Die übrigen sechs Attribute der ODC-Klassifikation geben keine zusätzliche, für die hier angestrebte Fehlerklassifikation verwertbare Differenzierung.

Target: *Target* beschreibt hier analog zu IBM das Dokument, das zur Fehlerbehebung verbessert werden muss (erste Spalte in Tabelle 4.1). IBM unterscheidet die als Objekte bezeichneten Dokumente *Specification, Design* und *Code*. Diesen Dokumenten können die Dokumente aus den drei Entwicklungsphasen Anforderungsdefinition, Entwurf und Implementierung, die in der vorliegenden Arbeit als *Spezifikation*, *Entwurf* und *Quelltext*, bezeichnet werden, gleichgesetzt werden.

Defect Type: Der *Defect Type* klassifiziert bei ODC das Fehlverhalten anhand der Fehlerursache. ODC deckt hier die Gesamtheit der möglichen Fehlertypen mit sieben Klassen ab (zweite Spalte in Tabelle 4.1). Die Klassenbildung beginnt mit Implementierungsfehlern in kleinen Programmeinheiten, z. B. (Daten-)Zuweisung, geht über algorithmische Fehler in einer Programmeinheit zu Fehlern im Zusammenspiel der Programmeinheiten, z. B. Synchronisationsfehlern. Zwei Klassen, *Function/Class/Object* und *Relationship*, kategorisieren Entwurfsfehler .

Testfokusorientierte Fehlerklassifikation: Wie im Klassifikationsziel eingangs formuliert, wird hier eine Klassifikation angestrebt, in der jede Fehlerklasse (Spalte 4 in Tabelle 4.1) mit einem Testfokus verbunden werden kann, den Verfahrenskategorien und frühestmögliche Testzeitpunkte zugewiesen werden können. Die nachfolgend beschriebene Klassenbildung wird im Rahmen der Arbeit als *testfokusorientierte Fehlerklassifikation* bezeichnet. Die Benennung der Fehlerklassen orientiert sich an den üblichen Bezeichnungen in den untersuchten Projekten und ist als eine Bezeichnung des Testfokus der Fehlerklasse anzusehen.

Abbildung des Klassifikationsziels auf das Attribut Target: In dem in dieser Arbeit betrachteten Testprozess wird das Dokument *Quelltext* in verschiedenen Stadien der Systemvollständigkeit (z. B. Module, Teil- oder Gesamtsystem) ausgeführt und jeweils die Ursache des beobachteten Fehlverhaltens beurteilt. Die Fehlerursachen können dabei sowohl im Dokument *Quelltext* als auch in den Dokumenten *Spezifikation* und *Entwurf* liegen.

Wird die testfokusorientierte Fehlerklassifikation zur Ermittlung von Kosteneinsparpotentialen eingesetzt, können diagnostizierte Spezifikations- und Entwurfsfehler den entsprechenden Target-Ausprägungen *Code* und *Specification* mit den jeweiligen frühestmöglichen Phasen der Identifikation zugeordnet werden. Liegt die Fehlerursache in der Spezifikation oder dem Entwurf, so ist der Quelltext zwar konform zu den Dokumenten *Spezifikation* und *Entwurf*, jedoch weicht das implementierte System vom gewünschten System ab, da die Dokumente *Spezifikation* und *Entwurf* fehlerhaft sind. Spezifikations- und Entwurfsfehler sollten idealerweise vor dem hier betrachteten Test des Quelltextes identifiziert werden. Qualitätssicherungsmaßnahmen sehen z. B. statische Inspektionen als mögliches Verfahren vor. Da diese Qualitätssicherungsmaßnahmen nicht im Fokus der Klassifikation liegen und Spezifikations- und Entwurfsinspektionen nicht Teil des hier betrachteten Testprozesses sind, finden sich in Tabelle 4.1 nur Hinweise auf die relevanten Methoden.

Wird die testfokusorientierte Fehlerklassifikation zur optimalen Auswahl von Testverfahren für den Testprozess eingesetzt, sind die im Quelltext verbliebenen Spezifikations- und Entwurfsfehler der *Target*-Ausprägung *Code* zuzuordnen und dort zu klassifizieren.

In der Praxis ist im Testprozess eine Entscheidung, ob Fehlerursachen dem Dokument *Entwurf* oder *Quelltext* zuzuordnen sind, oft nicht eindeutig möglich [53]. Ein Fehlverhalten, z. B. unzureichende Performanz, kann häufig sowohl durch Entwurfsänderungen als auch durch Implementierungsänderungen korrigiert werden. Da die Fehlerklassifikation nach Abschluss der Fehlerbeseitigung vorgenommen wird, wird die Fehlerursache in dieser Arbeit dem geänderten Dokument zugeordnet.

Abbildung des Klassifikationsziels auf das Attribut Defect Type: In der testfokusorientierten Fehlerklassifikation soll die Klassenbildung so vorgenommen werden, dass jeder Fehlerklasse mindestens eine Testverfahrenskategorie zugeordnet werden kann, die deren Fehlerursachen bei Erreichen der erforderlichen Systemvollständigkeit adressieren kann. Es sind die Klassen von *Defect Type* für die *Target*-Ausprägung *Code*, da nur dieses Dokument im Testprozess einer Prüfung unterliegt, zu betrachten. Es ist eine Zuordnungsvorschrift zu formulieren, die die Eigenschaften der ODC-Fehlerklassen anhand des Testfokus und ggf. der Abhängigkeit von der Systemvollständigkeit auf testphasenorientierte Fehlerklassen abbildet. Diese Zuordnungsvorschrift (Spalte 3 der Tabelle 4.1) ist bezogen auf die automobile Entwicklung formuliert und stellt gleichzeitig die Klassifikationsvorschrift für die testfokusorientierte Fehlerklassifikation dar. Im einfachsten Fall ergibt sich eine 1:1-Abbildung zwischen der Klasse von ODC und der testzielorientierten Fehlerklasse, wie bei der ODC-Klasse *Assignment/Initialization* und der testzielorientierten Fehlerklasse *Zuweisung.* In den meisten Fällen ist die ODC-Klasse aufgrund des jeweils gebotenen Testfokus zu differenzieren, wie z. B. bei *Checking* eine Trennung in *Bedingung* und *Grenzwert* erforderlich ist.

Mitunter erfordert der Testfokus aufgrund der erforderlichen Systemvollständigkeit eine testphasenbezogene Differenzierung, wie für *Integration* und *Performanz*. Der Testfokus ist bei der Klassenbenennung durch den identischen Namensstamm, z. B. *Integration* ausgedrückt, dem dann die differenzierende Testphase vorangestellt ist, z. B. *Modulintegration* und *Systemintegration*. Dies stellt quasi eine Unterklassenbildung bezogen auf den Testfokus dar. Differenzierungen aufgrund der zu berücksichtigenden Systemabstraktion finden sich in allen ODC-Klassen ab *Algorithm/Method*, im Besonderen natürlich in den integrationsbezogenen ODC-Klassen *Timing/Serialization* und *Interface/O-O Messages*.

Abbildung von nicht testbaren Fehlerursachen: Im Hinblick auf die Verknüpfung jeder Fehlerklasse mit dem Testfokus können nur Fehlerklassen gebildet werden, die im Testprozess durch Testverfahren adressierbar sind. In diese Klassen sind auch Fehlerursachen einzugliedern, die primär nicht im Ziel des Testprozesses stehen. Leicht nachvollziehen lässt sich der Gehalt dieser Aussage am Beispiel der Fehlerursache *Schreibfehler*, die in anderen Klassifikationsverfahren wie oben erwähnt auch als Fehlerklasse dient. Der Begriff *Schreibfehler* lässt sich hierbei

sicher unterschiedlich weit fassen, jedoch sind *Schreibfehler* im Entwicklungsprozess ab Modultest nicht durch ein spezielles Testverfahren adressierbar. Somit ist die Fehlerklasse *Schreibfehler* zur Benennung von Testverfahren zur Fehleridentifikation nicht geeignet. Schreibfehler können sich in den Quelltextsequenzen zu allen oben genannten ODC-Fehlerklassen befinden und dort Fehlverhalten auslösen. Schreibfehler sind anhand der durchgeführten Fehleranalyse und des ausgelösten Fehlverhaltens gemäß den Angaben in Tabelle 4.1 Spalte 3 z. B. einer der definierten Fehlerklassen von *Checking, Assignment/Initialization, Algorithm/Method, Interface/O-O Messages* zuzuordnen.

Spezifikations- und Entwurfsfehler im Dokument *Quelltext*: Spezifikations- und Entwurfsfehler, die im Dokument *Quelltext* verblieben sind, sollen spätestens im betrachteten Testprozess identifiziert werden. Die Identifikation von Spezifikationsfehlern resultiert vornehmlich aus der Analyse von Spezifikationen im Zuge der Herleitung von Testfällen für den Systemtest aus (nicht-formalen) Anforderungsinspektionen oder dem Aufbau von Testmodellen (Formalisierung von Anforderungen) [176, 229]. Daher werden Spezifikationsfehler, die sich im Dokument *Quelltext* manifestiert haben, in der testfokusorientierten Fehlerklassifikation der verhaltensorientierten ODC-Klasse *Algorithm/Method* und dort der Unterklasse *Spezifikation* mit der Testphase Systemtest zugeordnet.

Entwurfsfehler im Dokument *Quelltext* gehören den spezifizierenden ODC-Klassen *Function/Class/Object* und *Relationship* an, führen jedoch nicht zu separaten testverfahrensorientierten Fehlerklassen, sondern werden anhand ihrer Testbarkeit beurteilt. Testbarkeit meint die Zuordnung zu Fehlerklassen, deren Tests die im Quelltext verbliebenen Entwurfsfehler im Testprozess frühestmöglich aufdecken könnten. Hier münden dann Unterklassen unterschiedlicher ODC-Klassen formal in derselben testverfahrensorientierten Fehlerklasse, da sie durch dieselben Testverfahren in denselben Testphasen adressiert werden.

Abbildung von Fehlerklassen auf Testphasen und Verfahrenskategorien: Die Unterteilung der ODC-Klassen des *Defect Types* in der *Target*-Klasse *Code* führt zu 19 Zuweisungsvorschriften, die die Fehlerursachen auf 13 Fehlerklassen abbilden. In Tabelle 4.2 wird für jede Fehlerklasse (Spalte 2) die zum Testen erforderliche Systemabstraktion in Form der Testphase (Spalte 1) und mögliche Testverfahrenskategorien (Spalte 3 und Beschreibung in Spalte 4) ausgewiesen. Die Zuordnung und Interpretation zur *Auswahl der Testverfahren* wird in Kapitel 4.4.3 beschrieben.

Disjunktheit: Die Disjunktheit von Fehlerklassen ist eine generelle Problematik [53, 98], da beispielsweise aufgrund der Fehlerfortpflanzung eine Fehlerursache unterschiedliches Fehlverhalten erzeugen kann [26]. Auch kann die Subjektivität von klassifizierenden Testern zu unterschiedlichen Ergebnissen führen [53]. Folglich sind Fehlermeldungen nicht immer eindeutig einer Fehlerklasse zuzuordnen [53]. Fehlerklassen werden dennoch häufig erfolgreich eingesetzt [53, 98, 159]. Bei der Zuteilung der Fehlermeldung zu Fehlerklassen ist ausschlaggebend, dass eine Fehlermeldung nach einer Fehleranalyse nur einer der Fehlerklassen zugeordnet wird, auch wenn Fehlermerkmale von verschiedenen Fehlerklassen – in unterschiedlich starker

Ausprägung – erfüllt sind.

Die IBM-Klassifikation anhand des ODC-Schemas definiert orthogonale und nahezu disjunkte (quasi-disjunkte) Fehlerklassen [53]. Die Beschränkung auf nur zwei von acht orthogonalen Klassifikationskriterien des ODC-Schemas von IBM bewahrt die erstrebte Überschneidungsfreiheit. Die in der testfokusorientierten Fehlerklassifikation eingeführten, verfeinernden Unterklassen zu den ODC-Klassen werden als die Überschneidungsfreiheit bewahrend aufgefasst. Somit wird die testfokusorientierte Fehlerklassifikation auch als quasi-disjunkt angesehen.

Ermessensspielräume bei der Klassenzuordnung: Wie bereits bei dem Thema Disjunktheit ausgeführt, können bei der Zuordnung einer Fehlermeldung zu einer Fehlerklasse Ermessensspielräume bestehen. Die Zuordnungsentscheidungen sollten in diesen Fällen auch die Zielsetzung der Klassifikation berücksichtigen. Fehlermeldungen, deren Fehlerbeschreibungen nicht eindeutig einzustufen sind, sind der Klasse mit der spätesten der zur Auswahl stehenden Fehleridentifikationsphasen zuzuweisen, wenn z. B. mithilfe der Fehlerklassifikation das Kosteneinsparpotential abgeschätzt werden soll und eine konservative Abschätzung gewünscht wird.

Verallgemeinerungsfähigkeit: Die ODC von IBM ist für alle Phasen der Softwareentwicklung ausgelegt [53]. Die *testfokusorientierte Fehlerklassifikation* zielt auf die durch den Testprozess identifizierten Fehler für das Dokument *Quelltext*. Die testfokusorientierte Fehlerklassifikation differenziert die ODC-Klassen durch Bildung von Unterklassen. Diese Unterteilung ist orientiert am Testfokus und an der erforderlichen Systemvollständigkeit. Die Klassifikation mit der Verknüpfung zu Testverfahren stellt ein Schema dar, dessen Anpassung an abweichende Rahmenbedingungen generell möglich ist. In der vorliegenden Ausarbeitung jedoch sind die Klassifikationsvorschrift, die Testphasen und die Testverfahrenszuordnung auf ein phasenorientierten Vorgehensmodells für automobile Entwicklungen zugeschnitten. Eine Übertragbarkeit ist möglich, wenn sich quasi-disjunkte Fehlerklassen eindeutig frühestmöglichen Identifikationszeitpunkten zuordnen lassen, die Fehlerbeseitigungskosten mit den Testphasen ansteigen und die Fehlerklassen durch Testverfahren adressierbar sind.

Spalte 1	Spalte 2	Spalte 3	Spalte 4
Target[1] ***(Fehlerhaftes Dokument)***	**Defect Type**[1] ***(Fehlertyp)***	**Zuordnungsvorschrift von ODC-Klassen zu Fehlerklassen – Klassifikationsvorschrift für testfokusorientierte Fehlerfrüherkennung –** Klassenbildung nach Testfokus und frühestmöglichem Testzeitpunkt in Abhängigkeit von Systemvollständigkeit (testfokusorientierte Fehlerfrüherkennung). Zuordnung erfolgt anhand von Fehlerursachen und Fehlereigenschaften (auf Basis der Fehlermeldungen nach erfolgter Fehlerbeseitigung).	**Fehlerklassen**
Specification *(Spezifikation)*	[2]	Unvollständige, widersprüchliche sowie fehlerhafte Anforderungen	Spezifikation
Design *(Entwurf)*	[2]	Entwurfsentscheidungen, die sich nachteilig auf Entwicklung, Test oder Benutzung auswirken.	Entwurf
Code *(Quelltext)*	Assignment/ Initialization *(Zuweisung/ Initialisierung)*	Irrtum bei **(nicht-algorithmischen) Datenzuweisungen**, z. B. Null-Pointer, falsch belegte Systemkonstanten (wie Blinkerfrequenz)	Zuweisung
	Checking *(Programmablaufkontrolle)*	Logikirrtum in der Programmierung von **Bedingungen**, z. B. Abfragen, die unabhängig von den Parametern stets wahr/unwahr sind, fehlerhafte Verwendung von logischen Operatoren (z. B. AND statt OR)	Bedingung
		Unzureichende oder fehlerhafte **Grenzwertüberprüfung**, z. B. fehlerhafte Verwendung von Relationsoperatoren (z. B. „> “anstelle von „≥“)	Grenzwert
	Algorithm/ Method *(Algorithmus)*	Fehlerhafte oder fehlende Algorithmusumsetzung im Quelltext aufgrund einer bislang unerkannten **Fehlerursache in der Systemspezifikation oder deren Verfeinerungen (Spezifikationsfehler)**	Spezifikation (im Testprozess)
		Fehlerhafte oder fehlende Algorithmusumsetzung, die **im Normalbetrieb bei Ausführung eines Moduls** bereits vor Integration in das System diagnostizierbar ist, z. B. Verwendung falscher Berechnungsfunktionen, falscher algorithmischer Lösungsansatz, fehlerhafte Umsetzung des gewünschten Algorithmus	Modulalgorithmus
		Fehlerhafter modulübergreifender Entwurf des Algorithmus, fehlende modulübergreifende Algorithmusumsetzung oder missinterpretierte Systemspezifikation die **im Normalbetrieb** erst **auf Systemebene** durch Beobachtung des Systemverhaltens diagnostizierbar sind, z. B. fehlerhafte Umsetzung von Prozessen auf Systemebene	Systemverhalten
		Fehlerhafte oder fehlende Algorithmusumsetzung, die erst **in forcierten Ausnahmesituationen** im Zusammenspiel von Soft- und Hardware bzw. durch die Umwelt induziert sind, z. B. durch Über-/Unterspannung, Zerstörung von Systemteilen (z. B. Glühbirnen) oder sonstige Störungen der Umwelt (z. B. extremer Frost, hohe Luftfeuchtigkeit)	Robustheit
		Das spezifizierte Systemverhalten wird trotz **fehlerfreiem Algorithmusablauf** bei Testdurchführung **auf Entwicklungshardware** innerhalb definierter Parameter nicht erreicht, wie z. B. Nichteinhaltung von Zeitvorgaben, Speicherverbrauch	Modulperformanz

Tabelle 4.1.: Zuordnungsvorschrift von *Target*- und *Defect-Type*-Klassen der ODC-Ausprägung *Code* zu testfokusorientierten Fehlerklassen (Fortsetzung auf nächster Seite)

		Das spezifizierte Systemverhalten wird trotz **fehlerfreiem Algorithmus** bei Testdurchführung **auf Zielhardware** innerhalb definierter Parameter nicht erreicht, wie z. B. Nichteinhaltung von Zeitvorgaben, Stromspannung, Speicherverbrauch	Systemperformanz
		Das spezifizierte Systemverhalten wird trotz **fehlerfreiem Algorithmus im Zusammenspiel mit anderen, nicht im Projekt entwickelten Systemkomponenten** nicht erreicht, z. B. Überlastung des gemeinsam genutzten Bus-Systems.	Fahrzeugperformanz
	Timing/ Serialization *(Synchronisation)*	Interferenzen paralleler Prozesse bei gemeinsam genutzten Ressourcen, z. B. zeitgleiches Schreiben auf dieselbe Datenvariable **bei der Modulintegration**	Modulintegration
		Interferenzen paralleler Prozesse bei gemeinsam genutzten Ressourcen, z. B. zeitgleiches Schreiben auf dieselbe Datenvariable **bei der Systemintegration**	Systemintegration
		Interferenzen paralleler Prozesse bei gemeinsam genutzten Ressourcen, z. B. zeitgleiches Schreiben auf dieselbe Datenvariable **bei der Fahrzeugintegration**	Fahrzeugintegration
	Interface/ O-O Messages *(Schnittstelle/ Nachrichten)*	Irrtum in der Integration von **Softwaremodulen**, z. B. durch fehlerhafte Bit-Belegungen in Signalen, Protokollfehler (Nachrichtenfolge, Datenbeschreibung, Zeitbeschränkungen)	Modulintegration
		Irrtum in der Integration von **Soft- und Hardwaremodulen**, z. B. durch fehlerhafte Bit-Belegungen in Signalen, Protokollfehler (Nachrichtenfolge, Datenbeschreibung, Zeitbeschränkungen)	Systemintegration
		Irrtum in der Integration von **verschiedenen Systemen im Fahrzeug**, z. B. durch Nutzung gemeinsamer Ressourcen wie doppelt vergebene Nachrichtenkennungen auf einem gemeinsam genutzten Bus-System	Fahrzeugintegration
	Function/Class/ Object *(Funktion/Klasse/ Objekt)*	Unentdeckter Irrtum im Entwurf der Softwaremodule, der die spezifikationsgemäße **Datenhaltung bzw. Nutzung** einschränkt, z. B. fehlende Dateneinheiten oder nicht ausreichende Zeichenkettenlängen	Grenzwert
		Unentdeckter Irrtum im Entwurf der Softwaremodule, der zu unvollständiger Implementierung und infolgedessen zu einem **eingeschränkten Funktionsumfang auf Systemebene** führt, z. B. fehlende Umsetzung einer programmtechnischen Einheit	Systemverhalten
	Relationship *(Beziehung)*	Unentdeckter Irrtum im Entwurf der Softwaremodule – Diskrepanzen im Entwurf von **Modulen** verhindern deren **Integration**.	Modulintegration
(1) Attribute und Ausprägungen nach ODC von IBM [53] (in Klammern deutscher Arbeitsbegriff) (2) Differenzierung zur Klassenzuordnung für vorgesehenes Klassifikationsziel nicht erforderlich.			

Tabelle 4.1.: Zuordnungsvorschrift von *Target-* und *Defect-Type-*Klassen der ODC-Ausprägung *Code* zu testfokusorientierten Fehlerklassen (Fortsetzung der vorherigen Seite)

Phase der frühestmöglichen Fehler-identifikation	Testfokus-orientierte Fehlerklasse	Kategorien von einsetzbaren Qualitäts-sicherungsmaßnahmen (Auswahl)	Charakterisierung der Qualitätssicherungsmaßnahmen
Anforderungs-inspektion	Spezifikation	Formale Spezifikationen	Formale Spezifikationen [55, 194] können auf Widerspruchsfreiheit und Fehlerursachen geprüft werden. Weiterhin können teilweise unvollständige und fehlende Anforderungen identifiziert werden.
		Verhaltensmodell-basierte Spezifikationen	Verhaltensmodelle [41] beschreiben das System üblicherweise weniger präzise als formale Spezifikationen [55, 194]. Folglich kann die Verifikation der Spezifikationen z. B. durch Model Checking [55] nur begrenzt durchgeführt werden. Erfahrungsgemäß werden jedoch einige Spezifikationsfehler durch Aufstellen von Verhaltensmodellen identifiziert (s. Kapitel 6.3.2).
Entwurfsinspektion	Entwurf	Entwurfsinspektion	Durchführung von Inspektionen (Reviews), z. B. Fagan-Inspektion [86].
Modultest	Bedingung	Bedin-gungsüberdeckungstest	Bedingungsüberdeckungen wie z. B. MC/DC [52, 176, 241] prüfen u. a. die Relevanz von Teilbedingungen auf das Gesamtergebnis des Bedingungsausdrucks.
		Grenzwerttest	Grenzwerttests prüfen die korrekte Ergebnisberechnung für Werte an den Rändern von Äquivalenzklassen [149, 176, 241], also Werteintervallen, die identische Ergebnisse liefern sollen, z. B. durch Grenzwertanalysen [149, 176, 222, 241]. Grenzwertanalysen können fehlerhafte Verwendungen von Relationsoperatoren (z. B. „$\geq$“ anstelle von „$>$“) identifizieren. Grenzwertanalysen sind zum Teil kombinierbar mit Bedingungsüberdeckungen.
	Grenzwert	Grenzwerttest	Grenzwerttests prüfen die Grenzen der Wertebereiche von Wertespeichern, können aber auch Definitions- oder Versorgungsirrtümer aufdecken, z. B. fehlerhafte Definition von Datenlängen, Sonderzeichen in Zeichenketten.
	Zuweisung	Statische Analysen(1)	Analysen des Quelltextes können u. U. Datenflussanomalien [176] und Null-Pointer [129] identifizieren.
		Strukturorientierte Test-verfahren	Die im Quelltext enthaltenen Zuweisungen werden unter Variation der Aufrufreihenfolge z. B. durch Quelltextüberdeckungen [176], wie Anweisungs-, Zweig- oder Pfadüberdeckungen geprüft.
	Modulalgorithmus	Statische Analysen(1)	Statische Analysen können u. U. Memory-Leaks, nicht terminierende Schleifen und Rekursionen aufdecken.
		Strukturorientierte Test-verfahren	Die inhaltliche Korrektheit der Rückgabewerte wird durch aus der Struktur des Quelltextes hergeleiteten Testfällen geprüft, z. B. auf Basis von Quelltextüberdeckungen [176], wie Anweisungs-, Zweig- oder Pfadüberdeckungen.
		Funktionsorientierte Test-verfahren	Die inhaltliche Korrektheit der Rückgabewerte wird durch aus der Spezifikation hergeleiteten Testfällen geprüft, z. B. auf Basis von Äquivalenzklassenanalysen [176].
	Modulperformanz	Lasttest (auch Leistungs-test)	Das spezifikationsgemäße Modulverhalten wird unter hoher Auslastung im Rahmen der Möglichkeiten auf der Entwicklungshardware geprüft.
Modul-integrationstest	Modulintegration	Modulintegrationstest	Die Interaktion zwischen jeweils zwei Modulen wird geprüft [176].

Tabelle 4.2.: Zuordnungsvorschrift von Phasen der frühestmöglichen Fehleridentifikation sowie von Kategorien von Qualitätssicherungsmaßnahmen zu Fehlerklassen (Fortsetzung auf nächster Seite)

Systemintegrations-test	Systemintegration	Systemintegrationstest	Die Interaktion zwischen jeweils zwei Systemteilen (z. B. Soft- und Hardware) wird geprüft [176].
Systemtest	Systemverhalten	Funktionsorientierte Test-verfahren	Das Systemverhalten wird durch aus der Spezifikation hergeleiteten Testfällen geprüft, z. B. auf Basis von Äquivalenzklassenanalysen.
		Verhaltensmodellbasier-ter Test	Es wird geprüft, ob das entwickelte System die möglichen Szenarien, die durch die Gesamtsystemanforderungen beschrieben sind, abbildet. Testfälle in Form von auszuführenden Sequenzen von Ein- und Ausgabesignalen wer-den aus Verhaltensmodellen hergeleitet, z. B. auf Basis von Überdeckungskriterien für Verhaltensmodelle [41] wie Transitionsüberdeckungen und Pfadtests.
		Anwendungsfallbasierter Test	Spezifikationen definieren häufig konkrete Anwendungsfälle. Sind diese durch Sequenz- [144] oder Use-Case-Diagramme beschrieben, geben sie Sequenzen von Ein- und Ausgabesignalen vor, die zum Testen verwendet werden können.
	Spezifikation (im Testprozess)	Verhaltensmodellbasier-ter Test	Beim Aufbau von Verhaltensmodellen werden Anforderungen semi-formalisiert und einem Review unterzogen. Semi-formalisierte Verhaltensmodelle können einer automatisierten Prüfung unterzogen werden. Diese Prüfung und Reviews können Spezifikationsfehler identifizieren.
		Anwendungsfallbasierter Test	Beim Aufbau von Szenario-Diagrammen werden Anforderungen semi-formalisiert und einem Review unterzogen. Semi-formalisierte Szenariomodelle können einer automatisierten Prüfung unterzogen werden. Diese Prüfung und Reviews können Spezifikationsfehler identifizieren.
	Robustheit	Robustheitstest	Die Funktionsfähigkeit (im Sinne von korrekter Ergebnisermittlung) des Systems in forcierten Ausnahmesituatio-nen wird z. B. durch Negativtests (Test mit Werten jenseits der intendierten Äquivalenzklassen) oder Umwelttests (umweltbedingte Extremsituationen wie tiefe Temperaturen, hohe Luftfeuchtigkeit, Staub) geprüft.
	Systemperformanz	Lasttest (auch Leistungs-test)	Das spezifikationsgemäße Systemverhalten wird auf der Ziel-Hardware durch Erzeugung einer hohen Auslastung geprüft. Die Tests zielen insbesondere auf die Prüfung, ob zeitliche und ressourcengebundene Vorgaben eingehalten werden.
Fahrzeugintegra-tionstest	Fahrzeugintegrati-on	Fahrzeugintegrationstest	Die gemeinsame Nutzung von Ressourcen (Speicher, Bus-Systeme, Stromanschlüsse, elektromagnetische Strahlung) durch verschiedene, unabhängig voneinander entwickelte Systeme wird geprüft.
Fahrzeugtest	Fahrzeugperfor-manz	Lasttest (auch Leistungs-test)	Lasttests [176] für gemeinsam genutzte Bus-Systeme.
(1) Statische Analysen werden ohne Testfallausführung durchgeführt, sind hier aber als mögliches Verfahren zur Qualitätssteigerung benannt.			

Tabelle 4.2.: Zuordnungsvorschrift von Phasen der frühestmöglichen Fehleridentifikation sowie von Kategorien von Qualitätssicherungsmaßnahmen zu Fehlerklassen (Fortsetzung der vorherigen Seite)

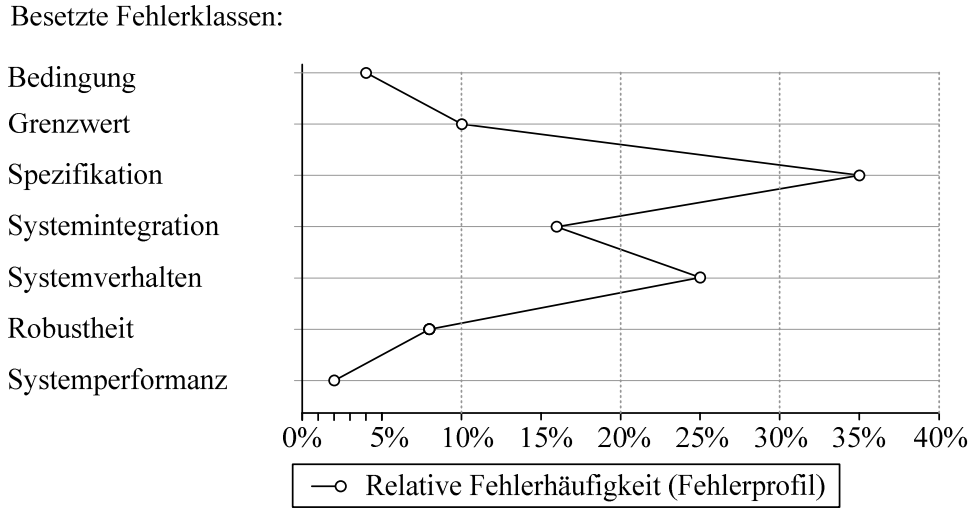


Abbildung 4.2.: Fehlerprofil eines Projekts (BC1) über besetzte Fehlerklassen

4.4.2. Testzielorientierte Fehlerklassengewichtung

Testziel als Gewichtung: Jede Teststrategie sollte das Abbild eines festgelegten Testziels sein. Die Anzahl der Testverfahren, die in den vorgesehenen Testphasen, durchgeführt werden können, sind in der Regel durch Zeit- und Kostenvorgaben begrenzt [183, 190, 247]. Daher sollten die einzusetzenden Testverfahren so ausgewählt werden, dass das Testziel optimal unterstützt wird. Nachfolgend wird eine Methode erarbeitet, die ein Testziel durch Gewichtung des Ergebnisses der testfokusorientierten Fehlerklassifikation, also der Fehlerverteilung in den Fehlerklassen, umsetzt. Aus der veränderten Fehlerverteilung ergeben sich objektiviert Empfehlungen zu den Fehlerklassen, die zur Erreichung des Testziels vorrangig zu testen sind.

Fehlerprofil: Ausgangsbasis für die Ableitung von vorrangig zu adressierenden Fehlerklassen ist die Fehlerverteilung von Fehlerursachen in einem Referenzprojekt, die dort z. B. nicht im Testprozess identifiziert wurden. In einem ersten Verfahrensschritt werden die Fehlerursachen mithilfe der in Kapitel 4.4.1 beschriebenen testfokusorientierten Klassifikation für das Dokument *Quelltext* aufgeteilt. Die Verteilung der relativen Häufigkeiten über die Fehlerklassen lässt sich in einem projektspezifischen Fehlerprofil visualisieren (s. Abbildung 4.2). Hierbei werden nur besetzte Klassen dargestellt. Es werden Fehlerschwerpunkte ersichtlich, die bei der Auswahl der Testverfahren Berücksichtigung finden sollten und über frühzeitige Fehleridentifikation zu einer Kostenreduktion führen sollten.

Definition des Testziels: In der Praxis ist die Verteilung der relativen Fehlerhäufigkeit des gewählten Referenzprojekts jedoch nicht immer eine hinreichende Entscheidungsgrundlage für die Erstellung einer Teststrategie. Das Testziel kann die Berücksichtigung von *Entscheidungsaspekten*, wie die Priorität der Fehlermeldungen,

die zu erwartenden Fehlerbeseitigungskosten oder die Sicherheitsgefährdung, die von einem Fehlverhalten ausgeht, fordern. Ebenso kann die Kombination verschiedener Aspekte für die Entscheidung herangezogen werden, welche Fehlerklasse vorrangig im Test identifiziert werden soll. Hierbei können auch Abhängigkeiten zwischen den gewählten Entscheidungsaspekten auftreten, so z. B. zwischen Fehlerbeseitigungskosten und Testphase der Fehlerbeseitigung oder zwischen Fehlerart und Fehlerkosten.

Gewichtung durch Entscheidungsaspekte: Die Entscheidungsaspekte werden formalisiert und zu Gewichtungen je Fehlerklasse transformiert. Anhand der Gewichtungen werden die Fehleranzahlen der Fehlerklassifikation algorithmisch (s. Anhang B) zu *gewichteten Fehleranzahlen* transformiert und es entsteht ein *gewichtetes Fehlerprofil*, das für die Berücksichtigung des Testziels wie in Kapitel 4.4.3 beschrieben zu interpretieren ist.

4.4.3. Auswahl von Testverfahren

Der letzte Schritt der fehlerklassenorientierten Testverfahrensauswahl besteht in der Interpretation der Häufigkeitsverteilung über die Fehlerklassen mit Bestimmung der optimierenden Testverfahrenskategorien, also der *Auswahl der Testverfahren*.

Interpretation der Fehlerhäufigkeiten: Die Fehlerklassen, die in dem Fehlerprofil bzw. in dem gewichteten Fehlerprofil die höchsten relativen Fehlerhäufigkeiten aufweisen, sind vorrangig zu testen, um die formulierten strategischen Ziele zu berücksichtigen und, sofern möglich, eine Fehlerfrüherkennung zu ermöglichen. Für diese Fehlerklassen sind Testverfahren gemäß Tabelle 4.2 auszuwählen.

Zuordnung von Testphasen und Testverfahren zu Fehlerklassen (Tabelle 4.2): In der hier vorgestellten fehlerklassenorientierten Testverfahrensauswahl wird von den Fehlerklassen, die bei der testfokusorientierten Fehlerklassifikation generiert wurden, und der zum Test der Fehlerklasse erforderlichen Systemabstraktion auf anzuwendende Testverfahren geschlossen. Hierzu ist in Tabelle 4.2 für die Fehlerklassen aus Tabelle 4.1 (Spalte 4) jeweils die bei der Klassenbildung implizierte, notwendige Systemvollständigkeit in Form der Testphase (Spalte 1) ausgewiesen. Die Testphase gibt den Zeitpunkt der frühestmöglichen Fehleridentifikation durch Testverfahren an. Tabelle 4.2 ist nach den Testphasen sortiert und verdeutlicht so die zeitliche Abfolge der durchzuführenden Tests.

Den Fehlerklassen (Spalte 2) sind Kategorien von Testverfahren (Spalte 3) zugeordnet, die in der Testphase zur Fehleridentifikation anwendbar sind. Die Kategorien umfassen die in der automobilen Entwicklungspraxis gängigen Testkategorien, sind jedoch nicht vollständig.

In Spalte 4 der Tabelle 4.2 finden sich Charakterisierungen der Verfahrenskategorien. Eine Verfahrenskategorie ist eine Gruppierung von Testverfahren mit gleichem Fokus auf die Fehlereigenschaften. Zu den Kategorien werden in den Beschreibungen beispielhaft konkrete Testverfahren benannt. Die Auflistungen der Testverfahren

weisen die in der automobilen Entwicklungspraxis gängigen Testverfahren aus, sind jedoch nicht vollständig. Die Präferenz für Testverfahren wird in der Regel je nach Entwicklungsbetrieb, System, Effizienzeinschätzung und Erfahrungen der Tester variieren.

Weist eine Fehlerklasse mehrere Kategorien von Testverfahren auf, kann eine ausgewählt werden. Wird eine bereits im Referenzprojekt eingesetzte Verfahrenskategorie empfohlen, ist zu prüfen, ob Testverfahren der gleichen Kategorie, jedoch mit einer höheren Fehlersensitivität zur Verfügung stehen. Ist dies nicht der Fall, so wäre ein Forschungsbedarf zur Verfahrensoptimierung bzw. -entwicklung erkannt. Zunächst kann ggf. auf eine andere Kategorie ausgewichen werden.

4.4.4. Bewertung

Zusammenfassung: Ziel der fehlerklassenorientierten Testverfahrensauswahl ist es, die Teststrategie für ein künftiges Projekt objektiviert und bedarfsgerecht zu entwickeln. Hierbei werden für ein formuliertes Testziel die vorrangig zu testenden Fehlerklassen auf Basis von Fehlerdaten eines Referenzprojekts identifiziert und geeignete Verfahren zur Identifikation dieser Fehler ausgewählt. Die testfokusorientierte Fehlerklassifikation trennt Fehlerklassen anhand der Testbarkeit durch Verfahrenskategorien und anhand der für den Test notwendigen Systemabstraktionsebene. Durch die *testzielorientierte Fehlerklassengewichtung* werden Testziele zu Gewichten der Fehlerklassen transformiert. Die Fehlerklassen mit den größten relativen Häufigkeiten sind mit den klassenspezifisch zugeordneten Verfahrenskategorien zu testen, um das Testziel, verbunden mit einer Fehlerfrüherkennung, umzusetzen.

Bewertung Vorgehensschema: Aus der Literatur ist kein vergleichbares Verfahren bekannt. Die testfokusorientierte Fehlerklassifikation sowie die Zusammenstellung der Testverfahren sind auf phasenorientierte Vorgehensmodelle der Automobilindustrie ausgerichtet. Das Vorgehensschema kann für andere Bereiche geöffnet werden, wenn die Klassifikation geeignet ausgetauscht wird. Die Gewichtung, also das umzusetzende Testziel, kann anwendungsunabhängig gewählt werden.

Anforderung an Fehlerklassifikation und Testverfahren: Grundvoraussetzung für das Klassifikationsverfahren ist, dass sich quasi-disjunkte Fehlerklassen bilden lassen und diesen eindeutig frühestmögliche Identifikationszeitpunkte zugeordnet werden können. Weiterhin müssen Testverfahren zugewiesen werden können, die Fehlverhalten der entsprechenden Klasse effektiv und damit auch frühzeitig identifizieren.

Anforderung an die Fehlerdaten: Die Fehlerdaten stammen vorzugsweise aus Fehlermeldungen von Kunden, also von im Testprozess des Herstellers nicht gefundenen Fehlern, oder aus dem Feldtest. Weiterhin sollte idealerweise ein Projekt mit vergleichbaren Merkmalen gewählt werden, da z. B. je nach Art des zu entwickelnden Systems unterschiedliche Fehlerklassen vorrangig entstehen können.

In den untersuchten Automobilprojekten besaß nur ein Projekt ausreichend Fehlerinformationen, die zudem manuell aufbereitet werden mussten. Somit ergibt sich auch an dieser Stelle, dass die Verbesserung der Fehlerdokumentation eine wichtige Grundlage für die Steigerung der Testqualität bildet.

Anforderung an Testziele: Testziele müssen durch bewertbare bzw. quantifizierbare Fehlereigenschaften vollständig beschrieben werden können. Beispielsweise dürfte das Testziel, die Fehlerursachen mit den kostenintensivsten Fehlerfolgekosten (Totalausfall, Sachschäden, Rückrufaktionen usw.) zu identifizieren in dieser Allgemeinheit kaum algorithmisch zu formulieren sein. Dahingegen kann das Testziel, Fehler in der Kommunikation zwischen Komponenten zu identifizieren, durch Fehlereigenschaften beschrieben werden.

4.5. Einsparpotentiale durch Fehlerfrüherkennung

Kosteneinsparpotentiale als Grundlage der Strategieoptimierung: Ein anerkannter Ansatz zur Reduktion von testinduzierten Qualitätskosten ist die Fehlerfrüherkennung. Hier werden Fehlerbeseitigungskosten über die Veränderung der Fehlerströme reduziert, da die Fehlerbeseitigungskosten mit den fortschreitenden Testphasen ansteigen [14, 28, 76, 78, 155, 192, 193, 223]. Wird das Fehlverhalten in der Projektpraxis nicht zum theoretisch frühestmöglichen Zeitpunkt identifiziert und die Fehlerursache beseitigt, liegt ein Kosteneinsparpotential in früherer Fehleridentifikation und -beseitigung vor. Das Kosteneinsparpotential ergibt sich dann aus der Differenz der Fehlerbeseitigungskosten zum Zeitpunkt der Ist-Fehlerbeseitigung und der Fehlerbeseitigungskosten zu einem möglichen, früheren Zeitpunkt. Der Zeitpunkt der Fehleridentifikation kann durch die Auswahl der Testverfahren und damit durch die Teststrategie beeinflusst werden. In Zusammenhang mit der angestrebten Optimierung der Teststrategie ist daher zunächst zu prüfen, ob das Referenzprojekt Kosteneinsparpotentiale aufweist.

Testphasenabhängige Bestimmung der Ist-Fehlerbeseitigungskosten: Für die Ist-Kosten der Fehlerbeseitigung eines Projekts gilt:

$$K^{\textit{Ist-Fehlerbeseitigung}} = \sum_{p}^{P} (N_p^{\textit{Beseitigte Fehlerursachen}} \cdot K_p^{\textit{Mittlere Beseitigung einer Fehlerursache}}) \tag{4.2}$$

mit:

N_p	Anzahl der in der Testphase p beseitigten Fehlerursachen
K_p	Mittlere Beseitigungskosten einer Fehlerursache in Testphase p (arithmetischer Mittelwert)
$p = 1, ..., P$	Laufvariable für Testphasen (unabhängig von den Iterationen)

Das Produkt aus Anzahl beseitigter Fehlerursachen und mittleren Kosten je Fehlerbeseitigung liegt in der Praxis wohl meist direkt vor und kann übernommen werden, wird hier aber zur Verdeutlichung der Kostenstruktur benannt. Andernfalls, bzw. wenn die mittleren Beseitigungskosten einer Fehlerursache K_p nicht vorliegen, bietet es sich an, für die arithmetischen Mittel Schätzwerte einzusetzen und die Kostenbetrachtung mit den resultierenden angenäherten, also geschätzten Ist-Kosten vorzunehmen.

Maximales Kosteneinsparpotential: Für den Testablauf relevante Fehlerursachen entstehen nur in den ersten drei Projektphasen der Softwareentwicklung, in Anforderungsdefinition, Entwurf und Implementierung. Fehlerursachen können frühestens in der sich an die Entstehungsphase anschließenden Inspektionsphasen identifiziert und beseitigt werden. Die Kostendifferenz zwischen den Ist-Kosten und den Kosten der Fehlerbeseitigung in den absolut frühestmöglichen Phasen der Qualitätsüberprüfung stellt die maximal mögliche Kostenreduktion dar. Das Kosteneinsparpotential ergibt sich als $\widehat{K}osteneinsparpotential^{Maximal}$ aus der Differenz von $K^{Ist\text{-}Fehlerbeseitigung}$ und $\widehat{K}^{Maximal\ frühe\ Fehlerbeseitigung}$.

Diese maximalen Kosteneinsparpotentiale sind jedoch für viele Fehlerklassen als unrealistisch anzusehen. Für die Testbarkeit vieler Fehlerklassen muss ein gewisser Grad an Abstraktion (z. B. strukturorientiert, funktionsorientiert) und Vollständigkeit (z. B. Funktion, Modul, Gesamtsystem) des zu testenden Gesamtsystems erreicht sein. Demzufolge können bestimmte Fehlerklassen auch erst in späteren Testphasen fokussiert und identifiziert werden [122, 204]. Somit kann die Identifikation sämtlicher Fehlerursachen durch Testen i. Allg. nicht vor Abschluss des Fahrzeugtests als letzte Integrationsstufe erfolgen.

Der Berechnung der *maximalen* Kosteneinsparpotentiale kommt jedoch Bedeutung zu, da sie im Vergleich zu den nachfolgend dargestellten *fehlerklassenoptimierten* Kosteneinsparpotentialen geringere Anforderungen an die Datenbasis stellen. Es sind lediglich die in der Fehlerkorrektur geänderten Dokumente (Spezifikation, Entwurf, Quelltext) zu benennen, diese Informationen ergeben sich bereits nach der Fehleranalyse und liegen in der Regel in den Fehlermeldungen vor. Bereits aus den Häufigkeitsverteilungen zu den Zeitpunkten der Fehlerentstehung und Fehleridentifikation ergibt sich ein Überblick, ob ein Optimierungspotential vorhanden ist, das weitere Analysen sowie ggf. den Einsatz von Optimierungsmaßnahmen als sinnvoll erscheinen lassen.

Kosteneinsparpotential einer Optimierungsmethode: Erfolgen Fehleridentifikation und -beseitigung infolge von Testoptimierungen mit einer zunächst beliebigen Methode x in früheren Testphasen als bei den Ist-Fehlerbeseitigungskosten beschrieben, errechnen sich geringere Fehlerbeseitigungskosten. Das *Kosteneinsparpotential der*

Optimierungsmethode x resultiert somit nach Gleichung (4.3) aus der Differenz der Ist-Fehlerbeseitigungskosten und den analog zu Gleichung (4.2) geschätzten Kosten der mit Methode x optimierten Fehlerbeseitigung [6] $\widehat{K}^{\textit{Mit Methode x optimierte Fehlerbeseitigung}}$:

$$\widehat{K}osteneinsparpotential^{\textit{Optimierungsmethode x}} = K^{\textit{Ist-Fehlerbeseitigung}} - \widehat{K}^{\textit{Mit Methode x optimierte Fehlerbeseitigung}} \tag{4.3}$$

Fehlerklassenoptimiertes Kosteneinsparpotential: Wird die Testphase der frühestmöglichen Fehleridentifikation über die Fehlerklassen der testfokusorientierten Fehlerklassifikation bestimmt, wie in Kapitel 4.4.1 vorgestellt, ergeben sich realisierbare Kosteneinsparpotentiale. Für die Bestimmung des Kosteneinsparpotentials wird angenommen, dass ein Testverfahren alle Fälle von Fehlverhalten der fokussierten Fehlerklasse identifiziert. So wird zum Beispiel unterstellt, dass alle Integrationsfehler in Integrationstests identifiziert werden. Die Anzahl der in jeder Testphase identifizierbaren Ausprägungen von Fehlverhalten ergibt sich aus den Besetzungen in den Fehlerklassen. Die Berechnung des Kosteneinsparpotentials $\widehat{K}osteneinsparpotential^{\textit{Fehlerklassenoptimierte Fehlerbeseitigung}}$ erfolgt nach Gleichung (4.3). Das fehlerklassenoptimierte Kosteneinsparpotential gibt an, welche Kostenreduktionen durch Fehlerfrüherkennung maximal realisierbar sind.

Iterationen: Wird ein Produkt über mehrere Iterationen entwickelt, können Kostenreduktionen durch Fehlerfrüherkennung innerhalb jeder Iteration erwartet werden. Es kann davon ausgegangen werden, dass Fehleridentifikation und -beseitigung in früheren Iterationen auch geringere Kosten bedeuten [122, 241]. Für iterationsabhängige Kostenberechungen ist das Kostenverhalten über die Iterationen jedoch in der beobachteten Projektpraxis nicht hinreichend quantifiziert. Fehleranzahlen und Kosten je Testphase werden daher praxisorientiert als Summe der Werte aus den Iterationen verstanden.

Relatives Kosteneinsparpotential: Durch Optimierung erreichte Kostenreduktionen (Gleichung (4.3)) werden in Relation zu den Ist-Fehlerbeseitigungskosten (Gleichung (4.2)) als *Relatives Kosteneinsparpotential* ermittelt:

$$\textit{Relatives } \widehat{K}osteneinsparpotential^{\textit{Optimierungsmethode x}} = \frac{\widehat{K}osteneinsparpotential^{\textit{Optimierungsmethode x}}}{K^{\textit{Ist-Fehlerbeseitigung}}} \tag{4.4}$$

[6] Zur Notation siehe Anhang A.1.

Effektivität von Optimierungsmaßnahmen: Als Maß für die Effektivität einer Optimierungsmaßnahme kann die *Ausnutzung der maximalen Kostenreduktion* betrachtet werden, ausgedrückt durch die Relation der durch eine Optimierung erreichbaren Kosteneinsparung zu dem maximalen Kosteneinsparpotential (theoretisches Maximum):

$$Ausnutzung\ der\ maximalen\ Kostenreduktion^{Optimierungsmethode\ x} = \frac{\widehat{K}osteneinsparpotential^{Optimierungsmethode\ x}}{\widehat{K}osteneinsparpotential^{Maximal}} \tag{4.5}$$

Berechnung von Kosteneinsparpotentialen in der Literatur: Ein von Schweiggert [233] publiziertes Modell zur Berechnung der Kosteneinsparpotentiale bewertet alternativ die Länge der Zeitspanne, in der eine Fehlerursache im System unentdeckt bleibt. Die automobile Softwareentwicklung ist in ihren Kostenbetrachtungen und in ihrer Datenarchivierung stark testphasenorientiert. Daher sind diskrete Berechnungsvorschriften auf Basis der Testphasen vorzuziehen. Ein anderes diskretes Berechnungsmodell von Schweiggert [233] ließe sich für Testphasen adaptieren. Die in der Projektpraxis vorliegenden Daten weisen jedoch nicht den vorgesehenen Präzisierungsgrad für das Berechnungsmodell auf, so dass zudem algorithmische Veränderungen vorgenommen werden müssten. Daher wurde eine eigene testphasenbasierte Berechnung der Kosteneinsparpotentiale durch Fehlerfrüherkennung vorgezogen.

Bewertung und Verallgemeinerung: Die vorgestellten Algorithmen zur Berechnung von maximalen und fehlerklassenoptimierten Kosteneinsparpotentialen basieren auf dem Kostenanstieg der Fehlerbeseitigung mit fortschreitenden Testphasen [28]. Die Abschätzungen des theoretisch maximalen Kosteneinsparpotentials aus den Ist-Kosten stellen nur vergleichsweise geringe Anforderungen an die Datenbasis, die in vorliegender Praxis meist erfüllbar sind. Liegen Fehlerinformationen vor, die eine Fehlerklassifikation ermöglichen, bietet das fehlerklassenoptimierte Verfahren eine genauere, projektorientierte Abschätzung von möglichen Einsparpotentialen durch Testoptimierung.

Die vorgestellten Verfahren sind nicht durch branchenspezifische Charakteristika beschränkt. Es wird lediglich ein phasenorientiertes Vorgehensmodell mit über die Phasen ansteigenden Kosten vorausgesetzt. Eine Anwendbarkeit für Systementwicklungen auch außerhalb der Automobilbranche ist also gegeben.

4.6. Break-Even-Point bei Testautomatisierung

Neben der Fehlerfrüherkennung gilt die Testautomatisierung als ein weiterer Optimierungsansatz zur Reduktion der testinduzierten Qualitätskosten. In diesem Kapitel wird das Kosten-Nutzen-Verhalten der Testautomatisierung skizziert und ein Modell zur Berechnung des Break-Even-Points aufgestellt, anhand dessen während der Projektplanung der zu erwartende Break-Even-Point ermittelt werden kann.

4.6.1. Kosten-Nutzen-Verhalten

Mehrkosten: Eine Testautomatisierung erzeugt durch die Vorbereitung der Testfälle zur automatisierten Ausführung (Kostenelement: *Vorbereitung zur automatisierten Ausführung*) sowie der Anpassung der Automatisierung nach Systemänderungen (Kostenelement: *Anpassung der Automatisierung*, s. Abbildung 4.1, Kapitel 4.3.2) zunächst Mehrkosten.

Minderkosten: Die Testdurchführungskosten sind i. Allg. geringer, da Testfallausführung und Fehlererkennung einerseits je Testfall weniger Zeit benötigen und andererseits oft ohne Beaufsichtigung durch Personen geschehen können [76]. Diese Minderkosten gleichen jedoch die Mehrkosten der Vorbereitung bei einmaliger Ausführung in der Regel nicht aus.

Break-Even durch wiederholte Ausführung: Iterative Entwicklungsprozesse und die Verifikation der Fehlerbeseitigung mittels Testfallausführung bedingen, dass Testfälle in der Regel mehrmals ausgeführt werden. Testfallautomatisierungskosten fallen hierbei nur einmal an. Kostenreduktionen durch geringere Testfallausführungskosten entstehen jedoch mit jeder wiederholten Ausführung. Mit zunehmender Zahl von Iterationen wird insgesamt ein *Break-Even-Point*, also eine Kostenreduktion erwartet.

Weiterer Nutzen: Neben einer Kostenreduktion nach Erreichen des Break-Even-Points kann auch die Entwicklungszeit beeinflusst werden. Eine Testautomatisierung kann aufgrund der schnelleren Testfallausführungen zu kürzeren Entwicklungszeiten führen, was die Produktentwicklungszeit verkürzen kann (*Time to Market*). Alternativ kann bei unveränderter Projektlaufzeit eine höhere Anzahl Testfälle ausgeführt werden [203], was eventuell zu einer höheren Identifikationsrate von Fehlverhalten im Testprozess und so zu einer gesteigerten Produktqualität bei Markteinführung führen kann.

4.6.2. Kosten der Testautomatisierungen

Für die Quantifizierung der Testautomatisierungskosten wird in Anhang C aus dem Kostenmodell für testinduzierte Qualitätskosten ein Modell zur exakten Berechung und zwei Modelle für Berechnungen in unterschiedlich starken Abstraktionen abgeleitet. Das stärker abstrahierende Modell (Gleichung (C.7) in Anhang C) legt

den Berechnungen ein über Testverfahren, Testphasen und Iterationen aggregiertes Kostenverhalten zugrunde, indem die von Testverfahren, Testphasen und Iterationen abhängigen, präzisen Einzeldaten durch über diese Kriterien (gewichtet) gemittelte Werte substituiert werden. Wenn die mittleren Werte für Kosten, Automatisierungsgrade und Anpassungsgrad als arithmetische Mittelwerte auf den Einzeldaten beruhen, ergibt sich trotzdem ein exakter Wert. Die Verwendung von aggregierten Werten kommt der derzeitigen Praxis entgegen, in der nicht immer iterations-, phasen- und verfahrensabhängige Kosten- und Aufwandsangaben vorliegen. Durch die aggregierten Werte ergeben sich Testautomatisierungskosten, die über Iterationen hinweg konstant sind. In den Berechnungen wird lediglich der Kostenverlauf der Mehr- oder Minderkosten der Testautomatisierung (s. Gleichung (C.7) in Anhang C) über Iterationen hinweg berechnet. Zur Quantifizierung sind letztendlich nur die sechs, in Tabelle 4.3 aufgelisteten Eingangsparameter zu bestimmen.

Bezeichnung im Modell	**Eingangsparameter**
T	Anzahl aller im Projekt erzeugten Testfälle (automatisiert und manuell auszuführende)
$\widehat{G}^4$	Mittlerer Automatisierungsgrad zur Testfallautomatisierung, $(0 < \widehat{G}^4 \leq 1)$, $G^4 = 0$ keine Testfallautomatisierung
$\widehat{G}^{4,Anpassung}$	Mittlerer Anpassungsgrad als Anteil der anzupassenden automatisierten Testfälle $(0 \leq \widehat{G}^{4,Anpassung} \leq 1)$
$\widehat{G}^5$	Mittlerer Automatisierungsgrad der pro Iteration ausgeführten Testfälle, $(0 < \widehat{G}^5 \leq 1)$, $\widehat{G}^5 = 0$ keine automatisierte Testfallausführung
Y	Mittlere Anzahl der pro Iteration ausgeführten Testfälle
I	Anzahl Iterationen

Tabelle 4.3.: Eingangsparameter der vereinfachten Kostenfunktion für die Testautomatisierung

Einschränkungen der vereinfachten Kostenfunktion: In der Praxis nimmt der Fertigstellungsgrad der Systeme mit fortlaufender Iterationsanzahl zu. Für die in jeder Iteration hinzugekommenen Systemfunktionen werden zusätzliche Testfälle erzeugt [17] und zusammen mit den bereits in vorangegangenen Iterationen erzeugten Testfällen ausgeführt. Folglich können Testfälle aus frühen Iterationen öfter ausgeführt werden als Testfälle aus späten Iterationen. Somit erwirken Testfälle aus frühen Iterationen u. U. eine höhere Einsparung als später erzeugte Testfälle. Diese Eigenschaft bleibt in der vereinfachten Kostenfunktion auch im Hinblick auf die vorliegende Datenqualität unberücksichtigt. Stattdessen wird angenommen, dass alle Testfälle gleich häufig ausgeführt werden.

Bewertung: Der vorgestellte Ansatz nimmt Bezug auf die realen Abläufe in der betrachteten Entwicklungspraxis und ist in Bezug auf Kostenaspekte, wie Kosten für Automatisierungswerkzeuge oder Anpassung der Automatisierung von bestehenden Testfällen, umfassender als beispielsweise die Ansätze nach Dustin [76] oder Elfgen [81].

4.6.3. Abschätzung des Break-Even-Points

Grundlage der Abschätzung: Eine Testautomatisierung wird wirtschaftlich lohnend, wenn sich die für die Automatisierung zusätzlich aufgewendeten Kosten (Kosten für Werkzeuge, Testfallautomatisierung und Anpassung) aufgrund der reduzierten Testdurchführungskosten amortisieren, d. h. der Break-Even-Point überschritten wird. Das ist der Fall, wenn die Testkosten mit Testautomatisierung (Automatisierungsgrad G) die ohne Automatisierung unterschreiten, wenn also unter Verwendung von Gleichung (5.11 in Anhang C) gilt:

$$K^{\textit{Test ohne Automatisierung}} > K^{\textit{Test mit Automatisierungsgrad G}} \tag{4.6}$$

Zum Kostenvergleich wird für die beiden zu vergleichenden Szenarien die identische Anzahl Testfälle ausgeführt, in einem Fall alle manuell, im anderen Fall mit dem Anteil G automatisiert. Es wird von identischem Fehleridentifikationsverhalten ausgegangen. Daher müssen die Fehlerbeseitigungskosten nicht berücksichtig werden.

Nach entsprechenden Umformungen der Gleichung (C.7) in Anhang C kann wahlweise bestimmt werden, nach wie vielen Iterationen, Testphasen, Anzahl Testfällen pro Iteration oder bei welchem Automatisierungsgrad ein Break-Even erreichbar ist.

Im Hinblick auf die in der derzeitigen Praxis beobachteten Datenlage bietet sich an, Testautomatisierungskosten und die Automatisierungsgrade über Testverfahren, Testphasen und Iterationen aggregiert (gemittelte Werte) zu betrachten und so die pro Iteration anfallenden Kosten als konstant anzunehmen. Das Kosten-Nutzen-Verhältnis der Testautomatisierung wird dann nach Gleichung (C.7) in Abhängigkeit der ausgeführten Iterationsanzahl abgeschätzt.

Der Break-Even-Point in Abhängigkeit zur Iterationsanzahl $\widehat{I}$ bestimmt sich aus Gleichung (C.9) (in Anhang C). Es ergibt sich folgender funktionaler Zusammenhang der Parameter (Erklärung der Parameter s. Tabelle 4.3):

$$
\begin{aligned}
\widehat{I} = & \\
& \Big(\widehat{K}^{\textit{Testfallautomatisierungswerkzeuge}} \\
& \quad + \widehat{T} \cdot \Big(\widehat{G}^{4} \cdot \widehat{K}^{\textit{Testfallautomatisierung für einen Testfall}} \\
& \qquad + \widehat{G}^{\textit{Anpassung}} \\
& \qquad + \widehat{K}^{\textit{Anpassung der Automatisierung für einen Testfall}} \Big) \Big) \\
& / \widehat{Y} \cdot \widehat{G}^{5} \cdot \Big(\widehat{K}^{\textit{Testfallausführung für einen manuellen Testfall}} \\
& \quad - \widehat{K}^{\textit{Testfallausführung für einen automatisierten Testfall}} \\
& \quad + \widehat{K}^{\textit{Manuelle Fehlerkennung für einen Testfall}} \\
& \quad - \widehat{K}^{\textit{Automatisierte Fehlerkennung für einen Testfall}} \Big)
\end{aligned}
\tag{4.7}
$$

Im Zähler der Gleichung (4.7) stehen die automatisierungsbedingten Kosten (Automatisierungswerkzeuge, Testfallautomatisierung und Anpassung der Automatisierung), im Nenner die Kostenreduktion in der Testfallausführung mit Fehlererkennung als Differenz der manuellen und automatisierten Tätigkeiten. Der Break-Even-Point wird in Iteration $\widehat{I}$ erreicht.

4.6.4. Bewertung

Die Kostenbetrachtung zur Testautomatisierung basiert auf Analysen zu Projektstrukturen, Projektpraxis (z. B. Option, in einer Iteration nicht alle Testfälle auszuführen) sowie Kostenelementen aus Projektbudgetplänen. Die Modellierung der Kosten, ausgehend von iterations-, testphasen- und testverfahrenabhängigen Kostenelementen, die auf den testfallbasierten Test bezogen sind, kann daher als praxiskonform angesehen werden.

Für die Quantifizierung der Testautomatisierungskosten wurde ein Modell zur exakten Berechung und zwei Modelle für Berechnungen in unterschiedlich starken Abstraktionen eingeführt. Auf dem stärker abstrahierten Modell aufbauend, wurde eine Break-Even-Point-Bestimmung abgeleitet, die über Testphasen und Testmethoden projektspezifisch aggregiert. Hier wurden präzise Einzeldaten durch gemittelte Werte substituiert. Für die prospektive Anwendung kommen statt der Mittelwerte abschätzende Werte zur Anwendung. Wie immer bestimmt hier die Güte der Schätzungen die Güte der Aussagen.

Der Break-Even-Point wurde algorithmisch wegen der intendierten Anwendung exemplarisch auf die Iterationsanzahl bezogen. Ebenso kann dieser auch in Bezug zur Testfallanzahl im Projekt, zur Anzahl ausgeführter Testfälle oder zu den Automatisierungsgraden bestimmt werden. Die Betrachtungen können auf testfallbasierte Automatisierungen anderer Branchen übertragen werden, wenn sich die Strukturen der Automatisierung gleichen bzw. einfachere Strukturen vorliegen.

4.7. Zusammenfassung

Kapitel 4 enthält Entwicklungen, die eine Optimierung von Teststrategien unterstützen. Optimiert werden soll durch Fehlerfrüherkennung, durch Einsatz von effizienten Testverfahren, durch Testautomatisierung und durch Identifikation von projektspezifischen Kostenschwerpunkten im Testprozess.

Zunächst wird der Optimierungsprozess für Teststrategien schematisiert, indem eine Abfolge von Tätigkeiten festgelegt wird, durch die sich in Bezug zu einem Referenzprojekt optimierte Teststrategien erreichen lassen. Die Optimierung der Teststrategie erfordert die Quantifizierung und Simulation der Qualitätskosten des betrachteten Testprozesses. Basierend auf Struktur- und Kostenanalysen wurde ein geeignetes Kostenmodell zur Quantifizierung und Simulation testinduzierten Qualitätskosten entwickelt, das je nach Datenlage und Anwendungszeitpunkt exakte wie abschätzende Berechnungsmethoden vorsieht. Die weiteren, erarbeiteten Methoden zur Unterstützung des Optimierungsprozesses beruhen auf den Struktur- und Kostenanalysen in Kapitel 2.5 und beziehen größtenteils algorithmische Entwicklungen aus dem Qualitätskostenmodell in Kapitel 4.3 ein. Konkret wurden entwickelt:

- Eine fehlerklassenorientierte Testverfahrensauswahl inklusive einer
  - testfokusorientierten Fehlerklassifikation zur Früherkennung von Fehlverhalten durch Tests,
  - testzielorientierten Fehlerklassengewichtung zur Transformation von Testzielen zu Gewichtungen der Fehlerklassen,
- Algorithmen für die Berechnung von Kosteneinsparpotentialen durch frühzeitige Fehlerbeseitigung,
- Algorithmen für die Bestimmung von Break-Even-Points von Testautomatisierungen.

Mit Hilfe der fehlerklassenorientierten Testverfahrensauswahl können anhand von Fehlerprofilen Kategorien von Testverfahren identifiziert werden, die hinsichtlich vorab definierter Testziele als optimal angesehen werden.

Die testfokusorientierte Fehlerklassifikation verweist je Fehlerklasse auf Kategorien von effizienten Testverfahren und auf die zum Test erforderliche Systemabstraktion, in der das zu der Klasse gehörige Fehlverhalten frühestmöglich identifiziert werden kann. Klassifiziert werden Fehlerursachen anhand von Fehlereigenschaften. Im Zusammenhang mit der Strategieoptimierung werden Daten von vergleichbaren, abgeschlossenen Projekten klassifiziert. Von besonderem Interesse ist dieses Vorgehen, wenn für Produktfamilien ein typisches Fehlverhalten angenommen werden kann.

Kosteneinsparpotentiale werden als Differenzen zwischen Ist-Kosten und Kosten für als optimal definierte Ziele ermittelt. Es können sowohl theoretisch maximale und fehlerklassenorientierte Kosteneinsparpotentiale quantifiziert werden. Erstere definieren ein theoretisches, durch den Testprozess nicht realisierbares Optimum, letztere ein an Systemabstraktionen orientiertes, realistisch näherungsweise durch

Testen erreichbares Optimum.

Die Break-Even-Point-Berechnung berücksichtigt entwicklungstypische Vorgehensweisen, wie Testfallwiederverwendung oder anteilige Ausführung der vorhandenen Testfälle.

Die entwickelten Methoden werden im Folgenden zur Analyse von Praxisdaten eingesetzt. In Kapitel 5 werden zunächst die untersuchten Projekte beschrieben und charakterisiert.

5. Analyse von Projektdaten

In Kapitel 4 wurden Verfahren für die spätere Simulation von Teststrategien entwickelt. Hier werden als weiterer Schritt der Vorbereitung Daten industrieller Entwicklungsprojekte eines Fahrzeugteilezulieferers analysiert. Es werden

- Kostenschwerpunkte, Fehlerströme sowie Fehlerverteilungen über Fehlerklassen ermittelt (Kapitel 5.2, 5.5 und 5.6),
- Einflussfaktoren auf das Kostenverhalten isoliert (Kapitel 5.4),
- die Realitätstreue von Aufwandsabschätzungen zur Fehlerbeseitigung untersucht (Kapitel 5.3).

Die Ergebnisse bilden Grundlagen für prospektive Quantifizierungen von Aufwandsdaten in Optimierungen und stellen Merkmale des Kostenverhaltens für Vergleiche mit der Literatur zur Verfügung. Die Projekte werden in Kapitel 5.1 zunächst charakterisiert.

5.1. Untersuchte Automobilprojekte

Für die Kostenanalysen in dieser Arbeit konnten Daten aus acht Entwicklungsprojekten von eingebetteten Systemen herangezogen werden. Die Daten stammen aus Systementwicklungen eines großen Konzerns und zwar aus dem Bereich der Fahrzeugteileentwicklung eines Fahrzeugteilezulieferers. Die Entwicklungen wurden für unterschiedliche Fahrzeughersteller durchgeführt und unterscheiden sich in Umfang, Qualität und Produkttyp. Es handelt sich jedoch in allen Fällen um die Entwicklung von Body Controllern oder Infotainment-Systemen. Die Projekte sind hier anonymisiert. Sie werden durch den Produkttyp bezeichnende Kürzel (s. Kapitel 5.1.1) sowie eine Nummerierung innerhalb jeden Kürzels identifiziert.

5.1.1. Projektcharakteristika

Die nachfolgende Charakterisierung der Projekte beschränkt sich auf die für diese Arbeit relevanten Aspekte.

Systemtypen: Es liegen fünf *Body Controller*-Projekte (BC1 bis BC5), zwei *Infotainment-Plattform*-Projekte (Inf1 und Inf2) und ein Mischprojekt (M1) vor. Projekt M1 ist ein Großprojekt zur Qualitätssteigerung, in das sowohl verschiedene Body Controller- als auch Infotainment-Systeme integriert sind. Projekt BC5 ist im Gegensatz zu den übrigen Projekten keine Auftragsentwicklung sondern ein Forschungsprojekt mit dem Ziel, einen Prototyp für einen neuartigen Body Controller zu entwickeln. Rücksprachen und Expertengespräche mit Testmanagern waren für die

Projekte BC1, BC2, BC4 sowie M1 möglich, allerdings aufgrund von Konzernumstrukturierungen nur im ersten Drittel des Bearbeitungszeitraums der Dissertation.

Body Controller-Systeme seien hier als *ereignisgesteuerte Systeme* und Infotainment-Systeme hingegen als *Workflow-Systeme* betrachtet. Ereignisgesteuerte Systeme zeichnen sich in dieser Arbeit dadurch aus, dass sie hauptsächlich auf voneinander unabhängige Signale der Umwelt reagieren [42]. Ein Beispiel sind die Eingaben eines Blinkermoduls (Linksblinken, Warnblinken usw.). Workflow-Systeme hingegen verarbeiten vorrangig vordefinierte Abfolgen von Aktivitäten (Szenarios). Als Beispiel sei das Bedienen eines mehrstufigen Auswahlmenüs zum Abspielen einer Audio-CD genannt.

Die in dieser Arbeit durchgeführten Untersuchungen betrachten *Body Controller*- und *Infotainment*-Projekte je nach Analyseziel sowohl getrennt als auch gemeinsam. Der Begriff *Automobil* fasst Body Controller und Infotainment sowie das Mischprojekt M1 zu einer Projektgruppe zusammen. Die gemeinsame Untersuchung ist als Abgrenzung zu anderen Industriesektoren zu verstehen.

Entwicklungsstadium, Systemgröße (in Anzahl Komponenten), Quelltextumfang, Programmiersprache sowie die im Systemtest verwendeten Testverfahrenskategorien sind für diese Projekte in Tabelle 5.1 aufgelistet und werden nachfolgend diskutiert.

Entwicklungsstadium (Tabelle 5.1, Spalte 2): Alle untersuchten Projekte wurden mit iterativer Anwendung des V-Modells entwickelt. Die Serienvorbereitung der Projekte war zum Zeitpunkt der Datenauswertung gerade erst abgeschlossen. Die Produkte befanden sich daher kurz vor dem *Beginn der Serienproduktion*.

Die Systeme des Projekts M1 befinden sich in unterschiedlichen Entwicklungsstadien, von denen der Großteil SOP erreicht hat.

Die Systeme des Forschungsprojekts BC5 existieren nur als Prototypen. Die Entwicklungszyklen wurden bereits vollständig abgeschlossen. Produktion und Lieferungen an Kunden erfolgten nicht, folglich sind keine Daten zu kundeninduziertem Fehlverhalten, z. B. aus Akzeptanztests, verfügbar.

Systemgröße in Komponenten, Quelltextumfang, Programmiersprache (Tabelle 5.1, Spalten 3 bis 5): Die Body Controller-Projekte unterscheiden sich in den Eigenschaften *Komponentenanzahl, Quelltextumfang, Programmiersprache* deutlich von den Infotainment-Projekten. Infotainment-Systeme weisen mehr Komponenten und entsprechend größeren Quelltextumfang auf. Infotainment-Systeme wurden in höheren Programmiersprachen entwickelt.

Angewendete Testverfahren im Systemtest (Tabelle 5.1, Spalte 6): Für die untersuchten Body Controller-Projekte wurden im Systemtest zwei unterschiedliche Kategorien von Testverfahren angewendet, unsystematische Tests (s. Kapitel 2.4.2) sowie funktionsorientierte und verhaltensmodellbasierte Testverfahren. In

Projekt (Bezeichnung und Anmerkungen)	Entwicklungs-stadium	Systemgröße (in Komponenten)	Quelltextumfang	Pro-gram-mier-spra-che	Angewendete Testkate-gorien im Systemtest
Systemtyp: Body Controller					
BC1 (1) (4)	SOP	12 BCF ($>$ 2.000 Anf.)	ca. 20 kLOC	C	vorrangig unsyste-matisch, z. T. funk-tionsorientiert und verhaltensmodellbasiert
BC2 (4)	SOP	7 BCF	wenige kLOC	C	Funktionsorientiert, verhaltensmodellbasiert
BC3	SOP	16 BCF (ca. 1.650 Anf.)	mehrere kLOC	C	Funktionsorientiert und verhaltensmodellbasiert
BC4 (1)	SOP	ca. 10-15 BCF	ca. 20 kLOC	C	Funktionsorientiert und verhaltensmodellbasiert
BC5 (2) (4)	Forschung (vor SOP)	Einzelnes System	Wenige kLOC	C	Unsystematisch
Systemtyp: Infotainment					
Inf1 (4) (5)	SOP	17 HK	$>$ 1 MLOC	Java, C++, C	verhaltensmodell-basiert
Inf2 (4)	SOP	17 HK (mit 67 SK wie Tourplaner, 3D-Karte usw.)	4,2 MLOC, 500 MB	Java, C++, C	verhaltens-modellbasiert
Systemtyp: Body Controller & Infotainment					
M1 (3) (4)	Subsystem-abhängig: vor SOP und SOP	Subsystem-abhängig	Subsystem-abhängig: von wenigen kLOC bis wenige MLOC	Java, C++, C	unbekannt

(1) System BC1 und BC4 sind funktional nahezu identisch, jedoch für andere Fahrzeugreihen
(2) Forschungsprojekt, Entwicklung eines Prototypen
(3) Vorhaben zur Qualitätssteigerung mit mehreren Body Controller- und Infotainment-Systemen
(4) Auswertbare Fehlerdatenbanken liegen vor
(5) Quelltextumfang: geschätzter Wert, basierend auf Erfahrung und Angaben von General Motors aus dem Jahr 2007 [73]

Abkürzungen:
SOP Start of Production
BCF Body Controller Funktionen
HK Hauptkomponenten
SK Subkomponenten
Anf. Anforderungen
kLOC Thousand Lines of Code
MLOC Million Lines of Code

Tabelle 5.1.: Charakteristika der untersuchten Automobilprojekte

der zuletzt genannten Kategorie handelt es sich um ein funktionsorientiertes und verhaltensmodellbasiertes Testverfahren, das speziell für einen Fahrzeugteilezulieferer für den Test von eingebetteten Systemen definiert wurde und *Äquivalenzklassenanalysen* [149, 176, 241], *Grenzwertanalysen* [149, 176, 222, 241] sowie das *Wp-Pfadüberdeckungskriterium* [101, 102] für zustandsbasierte Verhaltensmodelle umfasst. In Projekt BC1 wurde hauptsächlich mit dem unsystematischen Test und nur in einer Entwicklungsiteration für bestimmte Systemteile mit dem funktionsorientierten und verhaltensmodellbasierten Testverfahren getestet. Die Infotainment-Systeme wurden ausschließlich mit einem verhaltensmodellbasierten Testverfahren, dem *Wp-Pfadüberdeckungskriterium* [101, 102], getestet.

Entwicklerteams: Mit dem zunehmenden Softwareanteil in Automobilen (s. Kapitel 1.1) ist eine Verschiebung des Ausbildungsprofils der Entwicklerteams zu beobachten: Elektronik wurde überwiegend von Elektrotechnikern entwickelt, für Software-Implementierungen werden hauptsächlich Informatiker eingesetzt. Viele Innovationen in Infotainment-Systemen, wie z. B. Navigationssystem und Spracherkennung, sind in den letzten Jahren neu entstanden, ohne unmittelbar aus bereits existierenden Hardware-Komponenten hervorzugehen. Für die Softwareentwicklung dieser Systeme wurde häufig neues Personal benötigt und eingestellt. Vor allem waren dies Softwareentwickler (Informatiker). Die Entwicklerteams der untersuchten Body Controller setzen sich vorwiegend aus den Entwicklerteams früherer Hardwaresysteme zusammen, also aus Elektrotechnikern. Möglicherweise beeinflussen unterschiedliche berufliche Ausbildungen der Entwickler das Herangehen an und die Betrachtungsweise auf die zu entwickelnden Systeme, was zu unterschiedlichen Fehlerprofilen der beiden Systemtypen führen könnte.

5.1.2. Datenverfügbarkeit und -verwendung

Verwendete Daten: Verwendet wurden vor allem Informationen zu Entwicklungskosten, Entwicklungsverläufen und zu Charakteristika von Softwarefehlern sowie deren Identifizierung und Beseitigung. Diese Daten wurden als sog. *Fehlerdaten*, den Projektbudgetplänen, Testprotokollen und Expertengesprächen mit Testmanagern entnommen.

Fehlerdaten: In der Regel werden die Fehlerdaten mit Beginn eines Fehlermanagements als sog. *Fehlermeldungen* in Fehlerdatenbanken gespeichert. Fehlerdatenbanken existieren hier für die Projekte BC1, BC2, BC5, Inf1, Inf2 und M1. Die Fehlerdaten des Projekts M1 referenzieren einzelne Funktionen, die jedoch nicht eindeutig den implementierten Body Controller- bzw. Infotainment-Systemen zugeordnet werden konnten. Somit ist hier eine getrennte Betrachtung des Fehlverhaltens der beiden integrierten Systemtypen nicht möglich. Aufgrund von differierenden Zugriffsrechten und Datenbankkonfigurationen sind für die Projekte unterschiedliche Fehlercharakteristika verfügbar.

Für Projekte BC3 und BC4 wurden zur Beschreibung der Fehler- und Testsituation Expertenbeurteilungen von Projekt- und Testmanagern protokolliert. An diesen Projekten werden ausschließlich Kostenoptimierungen verdeutlicht (s. Tabelle 5.2).

Projektbudgetpläne: Die Projektbudgetpläne beziehen sich auf die gesamte Systementwicklung von Projektanfang bis Projektende. Die Werte wurden mit Projektfortschritt zum Teil ergänzt bzw. aktualisiert. Es konnten lediglich in die Projektbudgetpläne von Projekt BC1 eingesehen werden.

Plan-Werte: Informationen zu den untersuchten Projekten können unterschieden werden in *prospektive* und *retrospektive Projektdaten*. Prospektive Projektdaten sind *Plan-Werte* und werden vor Durchführung der jeweiligen Arbeiten abschätzend festgesetzt. Als prospektive Werte werden z. B. die Aufwandschätzungen für Fehlerbe-

Art der Projektdaten	Verfügbarkeit der Daten nach Projekt							
	BC1	BC2	BC3	BC4	BC5	Inf1	Inf2	M1
Fehlerdatenbank								
Fehlerbeseitigungsaufwand								
Ist	X[2,5]	X[1,2,5]			X	X[2,5]	X[1,2,5]	X[1,2,5]
Plan (Schätzung)	X	X[1]			X	X	X[1]	X[1]
Aufteilung nach Fehleranalyse, Fehlerkorrektur, Korrekturtest						X[7]	X[7]	
Phasen der Fehlerentstehung und -identifikation	X[3]	X[3]			X[3]	X[3]	X[3]	X[3]
Fehlerpriorität	X[5]	X[5]			X	X[5]	X[5]	X[5]
Beschreibung des Fehlverhaltens und der Fehleranalyse	X[4]							
Vergleichende, parallele Anwendung unterschiedlicher Testverfahren	X[9]							
Projektbudgetpläne								
Budgetplanungsdaten: Plan-Daten zu Fixkosten (Werkzeugkosten, Testaufbau usw.) und variable Kosten (Personalaufwand für z. B. Planung, Implementierung und Test)	X[6]							
Testprotokoll								
Testautomatisierungsgrad	X[8]	X[8]						
Aufwand der automatisierten/manuellen Testfallausführung	X[8]	X[8]						
Anzahl der automatisierten/manuellen Testfallausführungen und Anzahl der dabei identifizierten Fehler	X[8]	X[8]						
Expertengespräche mit Testmanagern								
Personalaufwand für Verhaltensmodellerstellung	X[9]	X[9]	X[9]	X[9]				

X[1] Für Analyse der Realitätstreue von Aufwandsabschätzungen verwendet, da ausreichendes Datenaufkommen (s. Kapitel 5.3)
X[2] Bestimmung Kostenwachstum für Fehlerbeseitigung (s. Kapitel 5.4)
X[3] Erstellung von Fehlerstromdiagrammen (s. Kapitel 5.5)
X[4] Fehlerklassenoptimiertes Kosteneinsparpotential (s. Kapitel 6.1.2), fehlerklassenorientierte Testverfahrensauswahl (s. Kapitel 6.2)
X[5] Analyse der Abhängigkeit der Fehlerbeseitigungskosten vom Systemtyp (s. Kapitel 5.4.2)
X[6] Plan-Projektkosten (s. Kapitel 7.1.1)
X[7] Analyse Fehlerbeseitigungskosten (s. Kapitel 5.2.2)
X[8] Bestimmung des Break-Even-Points der Automatisierungskosten (s. Kapitel 6.4)
X[9] Bestimmung der Effektivität von Testverfahren (s. Kapitel 6.3.2)

Tabelle 5.2.: Überblick über verfügbare Daten nach Projekt und deren Verwendungen in den Analysen

seitigungen in den Fehlerdatenbanken und prospektive Daten aus Projektbudgetplänen verwendet.

Ist-Werte: Die realen Werte werden hier als *Ist-Werte* bezeichnet und sind erst retrospektiv mit der Projektdurchführung zu ermitteln. Ist-Werte werden in dieser

Arbeit hauptsächlich aus aktualisierten Projektbudgetplänen, Testprotokollen und Fehlerdatenbanken entnommen.

Verfügbarkeit und Verwendung von Projektdaten: Tabelle 5.2 listet die projektspezifisch verfügbaren, für die vorliegende Arbeit wichtigen Daten und deren Verwendungen in den Analysen auf. Angaben zum Fehlerbeseitigungsaufwand liegen von sechs Projekten vor. Von denen haben nur zwei Projekte detaillierte Angaben zum Aufwand für Fehleranalyse, Fehlerkorrektur und Korrekturtest und ein Projekt eine Beschreibung des Fehlverhaltens und der Fehleranalyse sowie Projektbudgetpläne. Testprotokolle mit Informationen zur Testautomatisierung liegen von zwei Projekten vor. Testmanager von sechs Projekten gaben Expertenberichte zu neu eingeführten Testverfahren ab.

5.1.3. Fehlermeldungen in Fehlerdatenbanken

Fehlerdaten in Fehlermeldungen: *Fehlermeldungen* sind eine fundamentale Informationsquelle dieser Arbeit. Sie wurden mit dem im Systemtest einsetzenden Fehlermanagement in *Fehlerdatenbanken* gespeichert. Unter Fehlermeldungen sind die gespeicherten Fehlerdaten zu verstehen. Sie werden von Testern nach Identifikation eines Fehlverhaltens angelegt und beinhalten Beschreibungen zum identifizierten Fehlverhalten des zu testenden Systems sowie zum Stand und Aufwand der Fehlerbeseitigung. Es finden sich z. B. Informationen zu den verwendeten Testfällen und eine Reihe an Verwaltungsinformationen wie Zeitstempel nach Abschluss von Bearbeitungsschritten, Prioritäten, Versionsnummern der fehlerhaften Systeme und Aufwandsdaten verschiedener Art. Eine Fehlermeldung dient Programmierern zur Lokalisierung von Fehlerursachen während der Fehlerbeseitigung. Art und Umfang der je Fehlermeldung gespeicherten Informationen sind zwischen den Projekten inhomogen. Die projektabhängige Verfügbarkeit ist in Kapitel 5.1.2 beschrieben.

Fehlermeldungen aus dem Modul- und Systemtest, selektive Fehlererfassung: Die vorliegenden Fehlerdatenbanken wurden ausschließlich ab Systemtest verwendet. Für die in dieser Arbeit intendierte Betrachtung des Testprozesses fehlen somit speziell Daten aus den Modul- und Modulintegrationstests. Die zu der Modultestphase in den Fehlerdatenbanken befindlichen Fehlermeldungen entstammen einem erneuten, im Systemtest eingeleiteten Modultest. Dieser wurde durchgeführt, da die Modultests, die im Rahmen einer verteilten Entwicklung an einem anderen Standort durchgeführt wurden, als nicht ausreichend angesehen wurden. Die nachfolgenden Auswertungen der Datenbanken werden im Hinblick auf diese Datenlage interpretiert.

Fehlermeldungen aus Akzeptanz- und Fahrzeugtest: Es liegen Fehlermeldungen zu kundenidentifiziertem Fehlverhalten aus den Akzeptanztests der Musterlieferungen an den Kunden (Fahrzeughersteller) vor. Die Projekte BC1 und BC2 weisen weiterhin kundenidentifiziertes Fehlverhalten aus dem Fahrzeugtest auf, da dieser hauptsächlich vom Kunden (Fahrzeughersteller) durchgeführt wurde. Die hier untersuchten Systeme befinden sich vor Produktionsbeginn (SOP). Es existieren somit noch keine Feldfehler.

Zuordnung von Integrationstests: Fehlermeldungen aus Integrationstests wurden beim Eintragen in die Fehlerdatenbanken nicht durchgängig getrennt von denen der zugehörigen Testphasen ausgewiesen: Das betrifft die Fehlermeldungen aus Modulintegrationstests, Systemintegrationstests und Fahrzeugintegrationstests.

Anzahl der Fehlermeldungen: Die Anzahl der Fehlermeldungen variiert in den Projekten zwischen 78 und 23.726 (s. Tabelle 5.3). Für die beiden Infotainment-Systeme Inf1 und Inf2 liegen mit 23.726 bzw. 18.137 deutlich mehr Fehlermeldungen vor als für die Body Controller-Systeme BC1, BC2 und BC5, die zwischen 78 und 2.659 Fehlermeldungen aufweisen. Aufgrund der etwa tausendfach höheren Anzahl von Quelltextzeilen der Infotainment-Systeme ist dies allerdings durchaus zu erwarten. M1 liegt mit 4.532 Fehlermeldungen erwartungsgemäß dazwischen.

Duplikate: Fehlermeldungen, die ein bereits in einer vorangehenden Fehlermeldung beschriebenes Fehlverhalten erneut melden, werden in den untersuchten Entwicklungen als Duplikate bezeichnet. Für Duplikate werden in der Praxis keine gesonderten Aktivitäten zur Fehlerbeseitigung ausgelöst [136, 140, 271]. Sie werden hauptsächlich zur Verifikation der Fehlerbeseitigung verwendet (Ausführung des fehleridentifizierenden Testfalls), wenn die Fehlerkorrektur abgeschlossen ist. Der relative Anteil an Duplikaten ist für die beiden Infotainment-Systeme mit 25% bzw. 44% wesentlich höher als für Body Controller-Systeme mit Werten zwischen 0% und 12% (s. Tabelle 5.3).

Abgelehnte Fehlermeldungen: Fehlerdatenbanken enthalten auch Fehlermeldungen, die als abgelehnt gekennzeichnet sind und daher nicht zu einer Fehlerbearbeitung führten. Bei abgelehnten Fehlermeldungen basiert das Fehlverhalten auf Ursachen, die außerhalb des getesteten Systems (z. B. Konfigurationsfehler) liegen, oder es wird ein Fehlverhalten bemängelt, das jedoch spezifikationsgemäß ist. Der Anteil abgelehnter Fehlermeldungen liegt in Body Controller-Projekten bei 28%, bei Infotainment-Projekten bei 17% (s. Tabelle 5.3). Abgelehnte Fehlermeldungen werden in den Auswertungen nicht berücksichtigt.

Anzahl akzeptierter Fehlermeldungen: Für die akzeptierten Fehlermeldungen, die also keine Duplikate oder abgelehnte Fehlermeldungen sind, wurden in den Projekten die Fehlerursachen lokalisiert und beseitigt. Je nach Projekt lösen zwischen 35% und 95% der Fehlermeldungen eine Fehlerbeseitigung aus. Die zur Fehleridentifikation und -beseitigung gespeicherten Informationen werden in dieser Arbeit ausgewertet.

Änderungswünsche: Die Fehlerdatenbanken enthalten neben Fehlermeldungen auch *Änderungswünsche.* Änderungswünsche werden im Hinblick auf die Zielsetzung dieser Arbeit nicht als Fehlermeldungen beurteilt und nicht betrachtet.

Projekt	Anzahl Fehlermeldungen						
	gesamt	Duplikate		Abgelehnte Fehlermeldungen		Akzeptierte Fehlermeldungen (1)	
		Anzahl	%	Anzahl	%	Anzahl	%
Body Controller							
BC1	725	86	12%	147	20%	492	68%
BC2	2.659	287	11%	818	31%	1.554	58%
BC5	78	0	0%	4	5%	74	95%
Body Controller (gesamt)	3.462	373	11%	969	28%	2.120	61%
Infotainment							
Inf1	23.726	5.874	25%	3.585	15%	14.267	60%
Inf2	18.137	8.055	45%	3.683	20%	6.399	35%
Infotainment (gesamt)	41.863	13.929	33%	7.268	17%	20.666	50%
Body Controller und Infotainment							
M1	4.532	329	7%	808	18%	3.395	75%
Automobil							
Alle Projekte	49.857	14.631	29%	9.045	18%	26.181	53%

(1) Entspricht: Anzahl der Fehlermeldungen mit Fehlerbeseitigung

Tabelle 5.3.: Anzahl der Fehlermeldungen in den Projekten

5.2. Identifizierung von Kostenschwerpunkten

In diesem Kapitel werden anhand der Projektdaten die Kostenverteilungen auf die Projektphasen der Softwareentwicklung, die einzelnen Testphasen innerhalb des Testprozesses und auf die Aktivitäten in der Fehlerbeseitigung analysiert. Die ersten beiden Kostenverteilungen sollen u. a. die in den bisherigen Kapiteln unterstellten Kostenschwerpunkte verifizieren. Ansonsten werden die Kostenverteilungen in späteren Quantifizierungen und im Vergleich von Kostenentwicklungen aus der Literatur einbezogen.

5.2.1. Projekt- und Testphasen

Die Kostenverteilungen auf die einzelnen Projektphasen der Softwareentwicklung sowie auf die Testphasen werden herausgestellt und die Kostenschwerpunkte identifiziert.

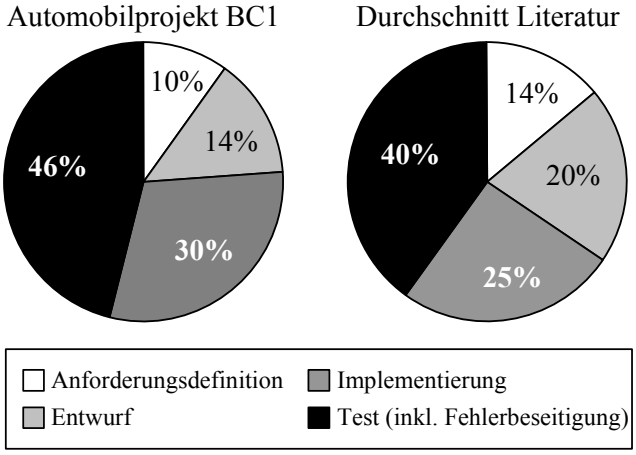


Abbildung 5.1.: Verteilung der Projektkosten auf die Projektphasen im Projekt BC1 und in publizierten Projekten (s. Tabelle I.1, Anhang I)

Datenbasis: Kostenbetrachtungen über alle betrachteten Projektphasen hinweg können in der vorliegenden Datenbasis nur aus Budgetplanungsdaten entnommen werden. Budgetpläne liegen jedoch nur für das Projekt BC1 vor. Somit bezieht sich die Auswertung auf dieses Projekt.

Projektphasen: Die Budgetplanungsdaten separieren allgemeine Projektkosten mit 30%. Die verbleibenden 70% der Projektkosten verteilen sich, wie in Abbildung 5.1 (s. auch Tabelle I.1 in Anhang I) dargestellt, auf die Projektphasen der Softwareentwicklung – auf Anforderungsdefinition, Entwurf, Implementierung – und Test. Der Test beinhaltet die Fehlerbeseitigung und ist mit 46% die kostenintensivste Projektphase.

Nach Angaben in der Literatur werden etwa 50% ($\pm$ 20%) des Entwicklungsaufwands für den Test aufgewendet [8, 15, 199, 214, 215, 262]. Konkreter aufgeschlüsselte Projektangaben aus der Literatur schätzen den Aufwand mit durchschnittlich 40% geringer ein (s. Abbildung 5.1 und Tabelle I.1 in Anhang I). Abweichungen können u. a. durch Unterschiede in der Datenerfassung, aber auch durch unterschiedliche Auffassungen, welche Tätigkeiten (z. B. Fehlerbeseitigung) dem Test zuzuordnen sind, verursacht sein. Zu beachten ist zudem, dass die Testzeiten und ihre Anteile am Projektaufwand zweifelsfrei von der Programmgröße beeinflusst sind, vgl. u. a. McConnell [186]. Die Kostenverteilung aus Projekt BC1 kann als vergleichbar zu durchschnittlichen Werten aus der Literatur angesehen werden.

Testphasen: Laut Budgetplänen des Projekts BC1 verteilen sich die Kosten im Testprozess auf die Testphasen *Modultest* bis *Akzeptanztest,* wie in Abbildung 5.2 dargestellt. Die Kosten sind nicht nach Iteration getrennt erfasst und beinhalten den beim Hersteller durchzuführenden Testvorgang (also ohne Fahrzeug- und Akzeptanztest) und die Beseitigung identifizierter Fehlerursachen. Der Systemintegrationstest ist im

Budgetplan separat ausgewiesen und kann daher getrennt dargestellt werden.

Der Kostenschwerpunkt wird entsprechend den Budgetplänen in Projekt BC1 eindeutig beim Systemtest erwartet, gefolgt von dem Modultest. Dieser verursacht etwa halb so hohe Kosten wie der Systemtest. Die anderen Testphasen sind kostenmäßig von untergeordneter Bedeutung. Fahrzeug- und Akzeptanztest inkludieren lediglich Fehlerbeseitigungskosten, da die Testdurchführung vom Kunden geleistet wird. Die Kostenaufteilung im Budget spiegelt mit Sicherheit die Erfahrung aus abgeschlossenen Projekten wider. Das bestätigt auch eine Umfrage [80] in verschiedenen Geschäftsbereichen der Siemens AG: Am häufigsten werden große Einsparpotentiale im Test eingebetteter Systeme in den Testphasen Systemtest und Modultest gesehen (s. Tabelle I.2 in Anhang I).

Schweiggert [233] ermittelte für IT-Systeme, dass auf den Systemtest etwa zwei Drittel und auf den Modul-/Modulintegrationstest etwa ein Drittel der Testkosten entfallen. In der Arbeit von McConnell [186] wird der Aufwand des Tests in Abhängigkeit zur Projektgröße beschrieben. Die mit *Developer Testing, Integration* und *System Testing* bezeichneten Phasen umfassen bei einer mit Body Controllern vergleichbaren Projektgröße ($\approx$0 kLOC, s. Tabelle 5.1, Kapitel 5.1.1) jeweils etwa ein Drittel. Die Ergebnisse von Schweiggert und McConnell widersprechen sich somit zunächst hinsichtlich des Anteils des Systemtests. Daher ist zu untersuchen, auf welcher Grundlage die Einteilung der Phasen definiert wurde. Da McConnell an dieser Stelle von *Integration* und nicht von *Integrationstest* spricht, kann vermutet werden, dass an dieser Stelle weitere Tätigkeiten in den ausgewiesenen Aufwand eingerechnet sind. *Developer Testing* umfasst bei McConnell zahlreiche Testvorgänge durch die Entwickler, typischerweise Unit Tests, Component Tests und Integrationstests, aber mitunter auch Regressionstests und Systemtests. Testteams übernehmen Beta Tests, Kundenakzeptanztests, Performanztests, Konfigurationstests und vieles mehr [186]. Die Fehlerbeseitigung wird den Entwicklertätigkeiten im Großbereich *Construction* zugerechnet. Daraus ergibt sich eine Verschiebung des Aufwands weg vom Definitionsbereich des Systemtestaufwands in dieser Arbeit. Dennoch deuten die Ergebnisse von Schweiggert und McConnell darauf hin, dass große Varianzen in der Aufwandsverteilung existieren können. Das hier untersuchte Projekt BC1 steht im Einklang mit Schweiggert. Eine Verallgemeinerung auf eingebettete Systeme erfordert jedoch eine breitere Datenbasis.

5.2.2. Fehlerbeseitigung

Im Hinblick auf spätere Quantifizierungen mit dem Kostenmodell für testinduzierte Qualitätskosten und die Möglichkeit, Optimierungsmaßnahmen daraus abzuleiten, ist die Verteilung der Fehlerbeseitigungskosten auf die einzelnen Arbeitsschritte von Interesse. Die Fehlerdatenbanken der Infotainment-Projekte Inf1 und Inf2 dokumentieren detailliert den Aufwand für die Arbeitsschritte *Fehleranalyse, Fehlerkorrektur, Korrekturtest* (Verifikation der Fehlerbeseitigung und Regressionstests) und können daher analysiert werden.

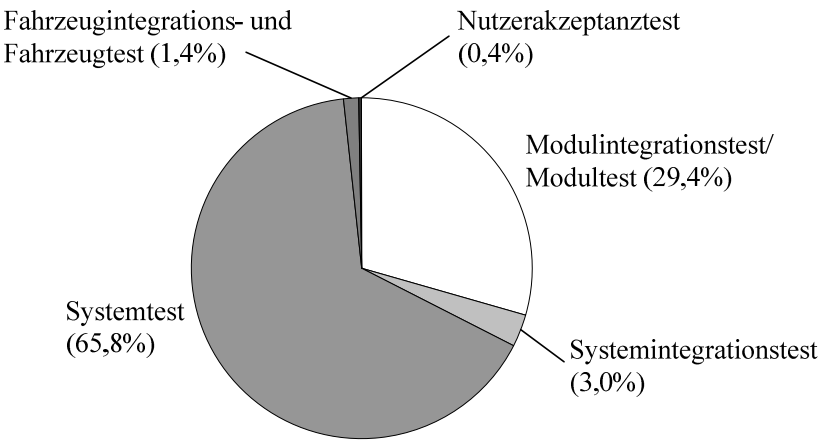


Abbildung 5.2.: Verteilung der Testkosten im Projekt BC1 auf die Testphasen nach Budgetplänen

Vorgehensweise: Die Aufwandsanteile für die jeweiligen Arbeitsschritte wurden zunächst je Projekt, dann zusammenfassend über beide Projekte ermittelt. Eine genaue Beschreibung der Vorgehensweise wird in Anhang D gegeben.

Ergebnis: Die Verteilungen der relativen Aufwandsanteile auf die Arbeitsschritte unterscheiden sich zwischen den Projekten Inf1 und Inf2. Dennoch ist in beiden Projekten der Aufwand für den *Korrekturtest* (Verifikation der Fehlerbeseitigung und Regressionstests) mit unter 10% im Vergleich zur *Fehleranalyse* (Mittelwert 27%) und zur *Fehlerkorrektur* (Mittelwert 66%) deutlich geringer (Tabelle D.1 in Anhang D). Allerdings ist zu beachten, dass die aufwändigeren Regressionstests nur in wenigen Fällen ausgeführt wurden.

Werden nur die Fehlermeldungen betrachtet, die eine Aufwandsangabe für alle drei Arbeitsschritte (Fehleranalyse, Fehlerkorrektur und Korrekturtest) aufweisen (s. Abbildung 5.3 und Tabelle D.2 in Anhang D), ergeben sich für die Projekte Inf1 und Inf2 bzgl. der Fehlerkorrektur ähnliche Kostenanteile. Allerdings ist die zugrunde liegende Datenbasis gering. Der Anteil der Fehleranalyse in Projekt Inf1 gleicht dem Wert auf der Basis von allen Fehlermeldungen, in Projekt Inf2 hat sich der Aufwand von der Fehleranalyse hin zu den Korrekturtests verschoben und dessen Anteil mehr als verdoppelt. Eine Analyse der absoluten Aufwandswerte (in Tabellen nicht angegeben) zeigte, dass in Projekt Inf2 hauptsächlich die Fehlermeldungen mit hohem Beseitigungsaufwand eher vollständig dokumentiert wurden. Diese Tendenz deutet sich auch in Projekt Inf1 an. Hierbei handelt es sich vornehmlich um Fehlerbeseitigungen mit Regressionstests, was die Verschiebung der Aufwandsverteilung zu den Korrekturtests erklärt. Aber auch bei Durchführung von Regressionstests bleibt die Fehlerkorrektur der aufwändigste Arbeitsschritt.

Verallgemeinerung: Untersuchungen von Fagan [86] ergaben, dass 50% des Aufwands für *Testaktivitäten* (Testfallgenerierung, Testdurchführung, Fehleranalyse,

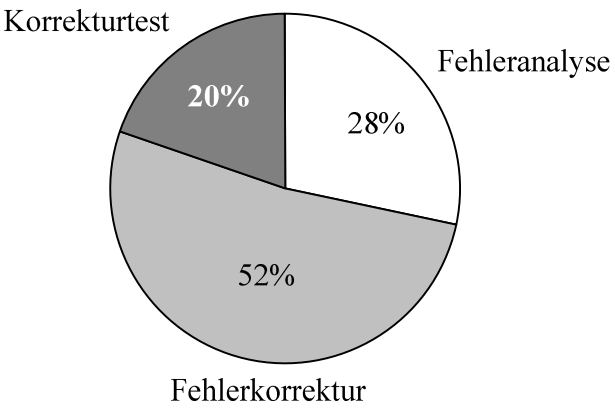


Abbildung 5.3.: Anteile von Fehleranalyse, Fehlerkorrektur und Korrekturtest (jeweils mittlerer Aufwand) an der Fehlerbeseitigung von Infotainment-Projekten, auf Basis von Fehlermeldungen mit vollständigen Angaben

Fehlerkorrektur, Verifikation der Fehlerbeseitigung und Regressionstests) den Arbeitsschritten Fehleranalyse und -korrektur zuzurechnen sind. Somit kann, sowohl aus den untersuchten Projekten als auch aus der Literaturquelle, geschlossen werden, dass eine Optimierung des Arbeitsschrittes Fehlerkorrektur (Systemänderung), ergänzt um Optimierungen des Arbeitsschrittes Fehleranalyse, ggf. erhebliche Kostenreduktionen mit sich bringen könnte. Ein Ansatz zur Optimierung der Fehleranalyse wird durch Einführung halbformaler Methoden mit Verfolgung von Mustern (s. Kapitel 9.3) vorgeschlagen.

5.2.3. Bewertung

Die durchgeführte Kostenanalyse für das untersuchte Projekt weist die Testphase als die kostenintensivste Projektphase der Softwareentwicklung aus. Der Kostenanteil von 46% kann als valide und übertragbar angesehen werden, da der Kostenanteil vergleichbar mit dem durchschnittlichen Wert von publizierten Projekten ist. Der Anteil des Systemtests an den Testphasen wird in der Literatur mit Werten zwischen ein und zwei Dritteln angegeben. Das untersuchte Projekt BC1 liegt mit zwei Dritteln somit am oberen Grenzwert.

Fehlerbeseitigungskosten stellen einen großen Anteil der Kosten des Systemtests dar. Anhand von zwei Infotainment-Projekten wurden die Kostenanteile für die zugehörigen Aktivitätsbereiche Fehleranalyse (28 %), Fehlerkorrektur (52 %) und Korrekturtest (20 %) isoliert.

5.3. Realitätstreue von Abschätzungen des Fehlerbeseitigungsaufwands

Bedeutsamkeit von Aufwandsabschätzungen: Fehlerbeseitigungskosten haben im Kostenmodell für testinduzierte Qualitätskosten eine hohe Bedeutung, da Schätzungen direkt als Kostenelemente in die Kostenberechnung einfließen können.

Ziel der Analyse der vorliegenden Aufwandsabschätzungen: Es soll überprüft werden, ob individuelle Schätzwerte einen verlässlichen Beitrag zur Abschätzung von Qualitätskosten liefern können. Untersuchungen zur Vorhersage von Fehlerbeseitigungskosten liegen bisher kaum vor [260]. Die Qualität der von Entwicklern vorgenommenen Zeitaufwandsschätzungen für die Fehlerbeseitigung soll anhand der Projekte BC1, BC2, Inf1, Inf2 sowie M1 vorgenommen werden, da für diese Projekte sowohl Aufwandsabschätzungen als auch der tatsächlich erbrachte Aufwand vorliegen. Die Beschreibung der Datenbasis und der Vorgehensweise der Datenanalyse sowie die Darstellung der Ergebnisse befinden sich in Anhang E.

Ergebnis: Die Ergebnisse aus Anhang E zeigen, dass die Entwickler in den untersuchten Projekten den Fehlerbeseitigungsaufwand realistisch abschätzen, sich damit positiv von den durchschnittlichen Werten in der Literatur [76, 247] abheben. Somit können derartige Abschätzungen durchaus vertrauenswürdige Grundlagen für summarische Betrachtungen sein. Bei summarischen Betrachtungen heben sich die bei wenigen Fehlern auftretenden großen Über- und Unterschätzungen des Fehlerbeseitigungsaufwands weitgehend auf. Im Mittel wird der Aufwand für die Fehlerbeseitigung um 2% unterschätzt. Selbstverständlich kann dieses Ergebnis nicht als gültig für andere Unternehmen angesehen werden. Jedoch motiviert das Ergebnis zu analogen Untersuchungen, um die Vertrauenswürdigkeit von hausinternen Schätzungen kennen zu lernen.

5.4. Einflussfaktoren auf die Fehlerbeseitigungskosten

Die hersteller- und kundeninduzierten Fehlerbeseitigungskosten sind wichtige Hauptkostengruppen der testinduzierten Qualitätskosten. Fehlerbeseitigungskosten sind durch zahlreiche Einflussfaktoren bestimmt [4, 28, 163, 206]. Die Kenntnis der Einflussfaktoren und deren Abhängigkeiten unterstützt u. a. maßgeblich die Qualität von Äquivalenzschlüssen, wenn Referenzprojekte gewählt werden, um von den Fehlerbeseitigungskosten eines abgeschlossenen Projekts auf die eines geplanten zu schließen. Im Folgenden werden die Fehlerbeseitigungskosten der vorliegenden, automobilen Projektdaten hinsichtlich verschiedener Einflussfaktoren untersucht.

5.4.1. Testphase der Fehleridentifikation

Der Einfluss der Testphase auf die Fehlerbeseitigungskosten kann als unbestritten angesehen werden (s. Kapitel 3.3). Im Hinblick auf die Quantifizierung der Qua-

litätskosten sollen Erkenntnisse zum Wachstumsfaktor der Kosten zwischen den Testphasen gewonnen werden.

Datenbasis: Zur Untersuchung des Einflusses der Testphase auf die Fehlerbeseitigungskosten weisen die Fehlerdatenbanken der Projekte BC1, BC2, Inf1, Inf2 sowie M1 die notwendigen Daten auf.

Vorgehensweise: Die Fehlerbeseitigungskosten wurden nach manueller Datenaufbereitung je Testphase (ab dem Modultest) innerhalb der Projekte gemittelt. Um die Kosten in den Testphasen vergleichen zu können, wurden alle derart ermittelten Aufwandsdaten auf die Modultestphase normiert (s. Tabelle 5.4).

Ergebnis: Zwischen Modultest und Systemtest ergibt sich, gemittelt über alle untersuchten Projekte, kein Anstieg des Aufwands für die Fehlerbeseitigung. Das könnte auf die spezifische Testsituation in den Projekten zurückzuführen sein (s. Kapitel 5.1.2): Hier wurden in der Systemtestphase zusätzlich Methoden des Modultests eingesetzt. Das dadurch identifizierte Fehlverhalten wurde, dem Fokus der Testverfahren und dem üblichen Einsatzzeitpunkt der Testverfahren entsprechend, als Modultestfehler gekennzeichnet, obwohl die Fehleridentifikationen zeitlich dem Systemtest zuzuordnen sind. Damit verbunden ist, dass die Systementwicklung entsprechend fortgeschritten ist und alle Fehlerbeseitigungen annähernd einen, dem Systemtest entsprechenden Aufwand erfordern.

Die Fehlerbeseitigungskosten steigen ab Fahrzeugtest im Vergleich zu den vorangegangenen Testphasen an. Der relative Kostenzuwachs ist bei den Body Controllern mit 2,5 stärker ausgeprägt als bei Infotainment-Projekten mit 1,2 (s. Tabelle 5.4), doch sind die Fehlerbeseitigungskosten für Akzeptanztests aufgrund weniger eingetragener Aufwandsdaten nur durch 10 (Body Controller) bzw. 15 (Infotainment-Systeme) Fehlermeldungen gesichert. Das deutliche generelle Kostenwachstum, bezogen auf den Systemtest, kann jedoch durch 132 Fehlermeldungen für den Fahrzeugtest von Body Controllern bestätigt (s. Tabelle 5.4) und damit als gesichert angesehen werden. Fahrzeugtest und Akzeptanztest für vom Hersteller vor SOP gelieferte Muster weisen hier, über alle Projekte betrachtet, im gewichteten Mittel rechnerisch gleiche Aufwandsdaten auf. Dies ergibt sich, da für die Infotainment-Projekte und das Mischprojekt M1 keine Daten aus dem Fahrzeugtest vorliegen und die Body Controller einen stärkeren Anstieg über die Testphasen aufweisen.

Schlussfolgerung: Anzeichen für einen schwach ausgeprägten Anstieg der Kosten über die Phasen sind gegeben. Die Datenbasis der Automobilprojekte ist jedoch zu gering, um eine belastbare Aussage zur Höhe des Kostenwachstum der Fehlerbeseitigung (s. Kapitel 6.1) in Verbindung zur Testphase der Fehleridentifikation ableiten zu können. Es wurde aufgrund der Datenlage das Kostenwachstum nur vom Modultest bis zum Akzeptanztest von Mustern (Musterlieferungen) berechnet.

Vergleich mit der Literatur: In den untersuchten Automobilprojekten wurden die Fehlermeldungen nur im wiederholten Modultest, im Systemtest, Fahrzeugtest und Akzeptanztest erfasst (s. Kapitel 5.1.2). Die betrachtete Literatur umfasst zusätzlich

die Fehleridentifikationen aus den davorliegenden Entwicklungsphasen und Feldeinsatz, jedoch nicht aus Fahrzeugtest- und Akzeptanztest. In den Datenquellen kann ein stetiges, wenn auch quantitativ unterschiedliches Kostenwachstum für die Fehlerbeseitigungskosten festgestellt werden (s. Kapitel 3.3). Das hier für die Automobilprojekte beobachtete Kostenwachstum für die Fehlerbeseitigung zeigt ein weniger ausgeprägtes phasenspezifisches Kostenwachstum als es in der Literatur berichtet wird. Zusammenfassend kann geschlossen werden, dass die Untersuchungsergebnisse nicht quantitativ jedoch qualitativ mit den Ergebnissen aus der Literatur übereinstimmen.

Projekt	Relatives Kostenwachstum je Projektphase								Summe
	Modultest(2)		Systemtest		Fahrzeugtest		Akzeptanztest		
	W	F	W	F	W	F	W	F	F
Body Controller									
BC1	1,0	5	0,9	5	1,9	2	2,6	1	13
BC2	1,0	38	1,2	129	1,6	130	2,5	9	306
Body Controller (1)	1,0	43	1,2	134	1,6	132	2,5	10	319
Infotainment									
Inf1	1,0	63	1,0	1.105	-	-	1,2	14	1182
Inf2	1,0	2	1,3	27	-	-	1,8	1	30
Infotainment (1)	1,0	65	1,0	1.132	-	-	1,2	15	1.212
Body Controller und Infotainment									
M1	1,0	8	0,3	5	-	-	1,5	68	81
Automobil (1)	1,0	116	1,0	1.271	1,6	132	1,6	93	1.612

W Relatives Kostenwachstum je Projekt, auf Modultest bezogen
F Fehleranzahl
\- Aufwandsdaten liegen nicht vor
(1) W ist entsprechend der Fehleranzahl der Testphase gewichtet.
(2) Nur Daten aus einem erneuten Modultest in der Systemtestphase

Tabelle 5.4.: Relatives Kostenswachstum des Fehlerbeseitigungsaufwands nach Testphase der Fehleridentifikation für die untersuchten Projekte, normiert auf den Modultest

5.4.2. Systemtyp

Datenbasis: Die Abhängigkeit des Fehlerbeseitigungsaufwands von den beiden betrachteten Systemtypen Body Controller und Infotainment wird anhand der Fehlerdatenbanken der Projekte BC1, BC2, Inf1, Inf2 sowie M1 untersucht.

Ergebnis: Laut den untersuchten Fehlerdatenbanken beträgt der mittlere Fehlerbeseitigungsaufwand für Automobilprojekte 5,6 Stunden (s. Tabelle 5.5). Der Median ist mit lediglich 1,0 Stunden mehr als 5-fach kleiner. Dies spricht für eine rechtsschiefe Verteilung des Fehlerbeseitigungsaufwands. Wie Tabelle 5.5 zeigt, variieren die Mittelwerte und Medianwerte des Fehlerbeseitigungsaufwands nicht nur deutlich zwischen den untersuchten Projekten, sondern Projekte mit nur wenigen auswertbaren Fehlermeldungen weisen gleichzeitig die höchsten Maximalwerte auf. Es ist ein deutlicher Unterschied zwischen Body Controller- und Infotainment-Systemen zu beobachten. Body Controller besitzen im Median einen achtmal so hohen Fehlerbeseitigungsaufwand wie Infotainment-Systeme. Der entsprechende Mittelwert ist sogar etwa 11-mal (=14,5/1,3) höher.

Projekt	Anzahl bewerteter Fehlermeldungen	Fehlerbeseitigungsaufwand pro Fehlermeldung in Stunden						
		Mittelwert	Standard-abweichung	Min (2)	25%-Quartil	Median	75%-Quartil	Max
Body Controller								
BC1	31	38,0	74,2	0,0	2,0	10,0	29,5	327,0
BC2	686	13,9	23,9	0,0	3,0	8,0	16,0	240,0
Kennwerte BC (1)	717	14,5	27,2	0,0	3,0	8,0	16,0	327,0
Infotainment								
Inf1	1.325	1,0	0,8	0,1	1,0	1,0	1,0	20,0
Inf2	79	10,2	37,8	0,3	0,3	1,0	1,0	200,0
Kennwerte Infotainment (1)	1.404	1,3	6,2	0,1	1,0	1,0	1,0	200,0
Body Controller und Infotainment								
M1	474	4,9	13,0	0,0	0,1	0,3	4,0	120,0
Automobil(1)	2.594	5,6	17,0	0,0	1,0	1,0	4,0	327,0

(1) Entsprechend der Fehleranzahl gewichtet
(2) Aufwandswerte kleiner als 0,05 Stunden (3 Minuten) sind als 0,0 Stunden dargestellt.

Tabelle 5.5.: Statistische Kennwerte des Fehlerbeseitigungsaufwands der Projekte BC1, BC2, Inf1, Inf2 sowie M1

Dieser Unterschied zwischen Body Controller- und Infotainment-Projekten zeigt sich auch in den abweichenden Verteilungen des Fehlerbeseitigungsaufwands: Mehr als 75% der Fehlerursachen in Infotainment-Systemen (s. 75%-Quartil, Tabelle 5.5) wurden innerhalb einer Aufwandsstunde beseitigt. In Body Controller-Systemen weisen 75% der Fehlerursachen (s. 25%-Quartil, Tabelle 5.5) drei oder mehr Auf-

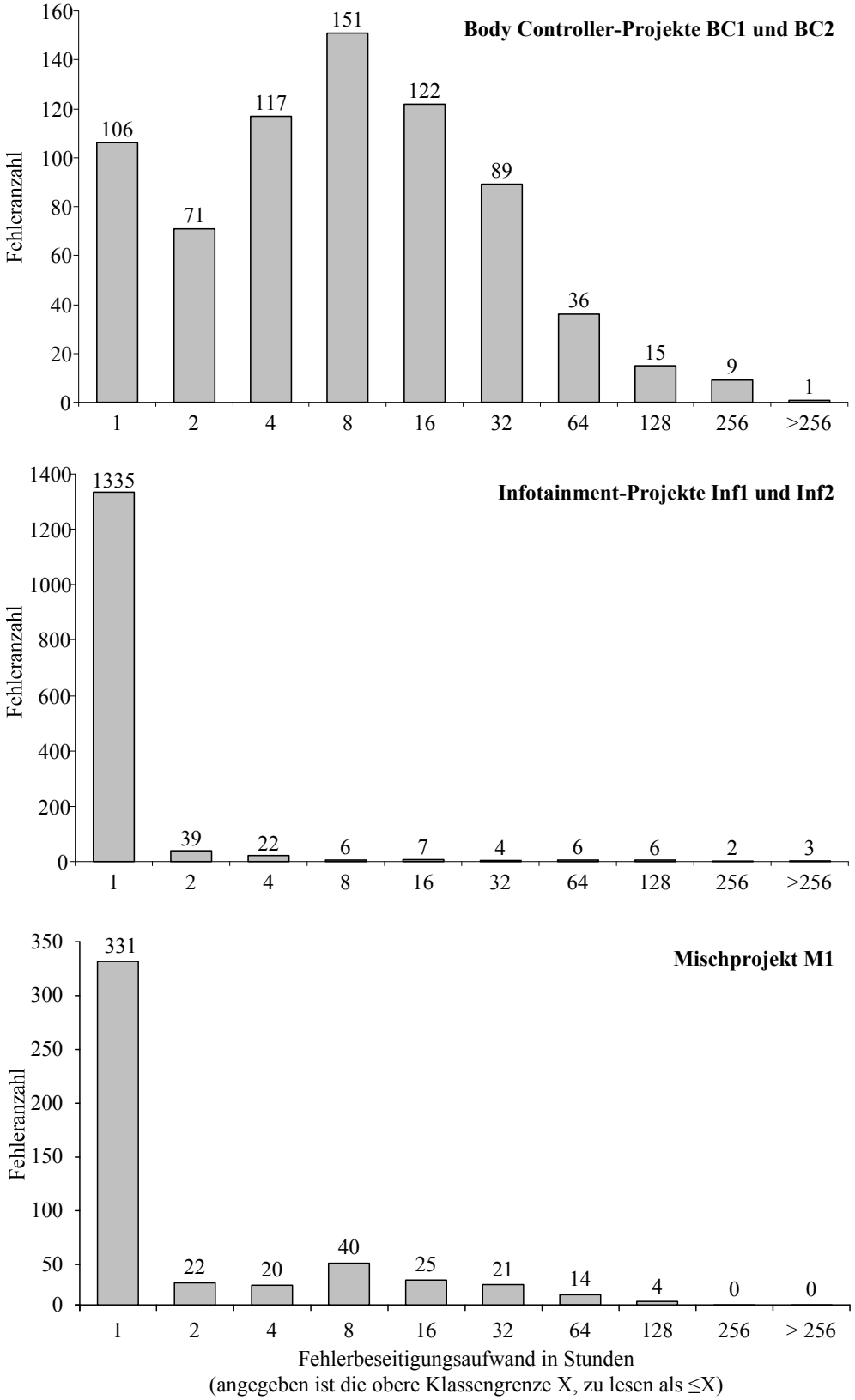


Abbildung 5.4.: Vergleich der Systemtypen. Verteilung des Fehlerbeseitigungsaufwands über logarithmischer Zeitachse

wandsstunden auf.

In der Abbildung 5.4 ist je Systemtyp die Verteilung des Fehlerbeseitigungsaufwands in logarithmisch äquidistanten Klassen dargestellt. Es zeigt sich eine sehr unterschiedliche Verteilung für Body Controller und Infotainment-Systeme. Die Fehlermeldungen aus Projekt M1 beziehen sich sowohl auf Body Controller- als auch auf Infotainment-Systeme. Die Verteilung des Fehlerbeseitigungsaufwands lässt eine Vermischung der systemtypischen Verteilungen von Body Controller- und Infotainment-Projekten erkennen.

Die Unterschiede in den Verteilungen der Fehlerbeseitigungskosten könnten durch die Fehlerklassen der untersuchten Fehlermeldungen begründet sein. Demnach müssten Body Controller vornehmlich andere Fehlerklassen aufweisen als Infotainment-Systeme. Diese Annahme kann jedoch aufgrund der Datenlage nicht überprüft werden, da Beschreibungen des Fehlverhaltens nur für Projekt BC1 vorliegen. Eine Klassifizierung der Fehlermeldungen aus Infotainment-Systemen ist daher nicht möglich.

Schlussfolgerung: Der Einfluss des Systemtyps auf die Fehlerbeseitigungskosten kann für die vorliegenden Daten als erwiesen angesehen werden. Obwohl dieser Befund nicht ohne Weiteres verallgemeinert werden kann, empfiehlt sich aber dennoch für eine prospektive Qualitätskostenschätzung ausschließlich retrospektive Daten aus Projekten mit identischen Systemtypen zu verwenden.

5.4.3. Fehlerpriorität

Die Fehlerpriorität ist ein Ergebnis der Fehleranalyse und wird unter Berücksichtigung verschiedener Fehlereigenschaften wie z. B. Kritikalität, Zentralität und Größe des Wirkungsbereichs bestimmt. Tester klassifizierten Fehlverhalten in drei (Projekt BC1) bzw. vier (Projekte BC2, Inf1, Inf2 und M1) Prioritätsstufen: *gering, mittel, hoch* und ggf. *top*. Ein Fehlverhalten der Prioritätskategorie *top* – auch *Showstopper* genannt – führt i. Allg. zu einem Programmabsturz und bedingt die Beseitigung der Fehlerursache, bevor die Software für den Kunden freigegeben wird. Die anderen drei Kategorien sind nicht strikt abgegrenzt und haben daher eine größere subjektive Komponente.

Datenbasis: Die Tester hinterlegten die Prioritäten in den Fehlermeldungen der untersuchten Fehlerdatenbanken der Projekte BC1, BC2, Inf1 und Inf2. Das Mischprojekt M1 wird in dieser Untersuchung nicht betrachtet, da die deutlich unterschiedlichen Kostenniveaus der Fehlerbeseitigung aufgrund der Vermischung von Systemtypen eine mögliche Abhängigkeit von der Priorität überlagern könnten.

Vorgehensweise: Die Fehlermeldungen wurden je Projekt nach Prioritätsklasse gruppiert und zu jeder Klasse der Mittelwert und der Median des Fehlerbeseitigungsaufwands bestimmt (s. Tabelle 5.6). Die Kennwerte der Systemtypen wurden mit der Anzahl der Fehlermeldungen mit Prioritätsangaben je Projekt gewichtet. Das

relative Kostenwachstum der Klassen wurde auf die Prioritätsstufe *gering* normiert. Weiterhin wird überprüft, ob systematisch für bestimmte Fehlerprioritäten vermehrt Aufwandseintragungen vorgenommen wurden.

Prüfung auf systematisch selektive Datenerfassung: Die Eintragung des Fehlerbeseitigungsaufwands ist lückenhaft und es könnte vermutet werden, dass die Eintragung in Abhängigkeit von der Fehlerpriorität vorgenommen wurde, also beispielsweise hoch priorisierte Fehlermeldungen häufiger vollständig erfasst wurden. Dadurch können Ergebnisse aus diesen Daten möglicherweise verfälscht werden. In statistischen Tests wurde daher ein möglicher Zusammenhang zwischen der Vollständigkeit des in Datenbanken eingetragenen Fehlerbeseitigungsaufwands und der Fehlerpriorität untersucht. Die Ergebnisse sind in Anhang I Tabelle I.3 dargestellt. Sowohl der Chi-Quadrat-Test [237] als auch der Trendtest [237] ergaben nur für das Projekt Inf1, dass die Angabe der Fehlerbeseitigungskosten signifikant ($p \leq 0{,}01$) von der Priorität abhängt. In diesem Projekt weist die mittlere Prioritätsklasse einen besonders hohen Anteil von Fehlermeldungen mit vollständig eingetragenem Aufwand auf. Aus der Untersuchung aller betrachteten Projekte kann aber geschlossen werden, dass die Vollständigkeit der Eintragungen nicht generell über alle Projekte hinweg gleich gerichtet von der Prioritätsklasse abhängt und dass damit keine Anzeichen vorliegen, die eine systematische Verzerrung der Ergebnisse erwarten lassen.

Ergebnis: Nach den durchgeführten Untersuchungen (s. auch Tabelle 5.6) wächst der mittlere Fehlerbeseitigungsaufwand für die Body Controller- und Infotainment-Projekte mit steigender Priorität an. Der relative Anstieg der Mittelwerte des Fehlerbeseitigungsaufwands zur nächst höheren Priorität ist für beide Systemtypen ähnlich ausgeprägt, obwohl der absolute Fehlerbeseitigungsaufwand deutlich zwischen den Systemtypen abweicht (s. Kapitel 5.4.2). Im Body Controller-Projekt BC2 mit einer definierten Showstopper-Kategorie ist der Anstieg der Mittelwerte des Fehlerbeseitigungsaufwands von der Prioritätsstufe *hoch* zur höchsten Priorität, den Showstoppern, sehr ausgeprägt. Das relative Kostenwachstum von der zweithöchsten zur höchsten Priorität beträgt bzgl. des Mittelwerts für Body Controller das 1,4-fache (=1,9/1,3), für Infotainment-Systeme das 1,2-fache (=1,7/1,4). Bei den Infotainment-Systemen zeigt sich der mediane Fehlerbeseitigungsaufwand als unabhängig von der Priorität. Dies weist darauf hin, dass Extremwerte den mittleren Fehlerbeseitigungsaufwand geprägt haben. Der nahezu konstante Fehlerbeseitigungsaufwand für Fehlermeldungen der mittleren und hohen Priorität ist durch die Mittelwerte angedeutet und evident in den Medianwerten.

Schlussfolgerung: Aus den Untersuchungen folgt, dass bei steigender Priorität systemtypübergreifend mit wachsenden Fehlerbeseitigungskosten zu rechnen ist. Dennoch existieren Unterschiede: Auch wenn das auf Basis der Mittelwerte berechnete relative Kostenwachstum unabhängig vom Systemtyp ist, wachsen die Medianwerte der Body Controller mit steigender Priorität an, die für Infotainment-Systeme jedoch nicht. Hier sind die Fehlerbeseitigungszeiten so kurz (s. Kapitel 5.4.2), dass möglicherweise kleine Unterschiede von der Zeiterfassung nicht mehr dokumentiert wurden.

	Fehlerbeseitigungsaufwand in Stunden bzw. *relatives Kostenwachstum* Mittelwert Priorität				Median Priorität				Anzahl Fehlermeldungen Priorität				Summe
	Gering	Mittel	Hoch	Top	Gering	Mittel	Hoch	Top	Gering	Mittel	Hoch	Top	
Body Controller													
BC1	3,4	3,5	8,0	n.r.	2,0	16,0	16,0	n.r.	8	21	2	n.r.	31
BC2	12,0	13,8	15,7	21,7	4,0	8,0	8,0	10,0	177	317	181	11	686
Gewichteter[1] Mittelwert/Median	11,6	13,4	15,4	21,7	3,9	8,3	8,3	10,0					
Relatives Kostenwachstum[2]	*1,0*	*1,1*	*1,3*	*1,9*	*1,0*	*2,1*	*2,1*	*2,6*					
Summe									185	338	183	11	717
Infotainment													
Inf1	0,7	0,9	1,0	1,2	1,0	1,0	1,0	1,0	67	348	509	210	1.134
Inf2	k.A.	1,1	1,3	k.A.	k.A.	2,0	2,0	k.A.	0	2	3	0	5
Gewichteter[1] Mittelwert/Median	0,7	0,9	1,0	1,2	1,0	1,0	1,0	1,0					
Relatives Kostenwachstum[2]	*1,0*	*1,3*	*1,4*	*1,7*	*1,0*	*1,0*	*1,0*	*1,0*					
Summe									67	350	512	210	1.139

[1] Als Gewichte für die Berechnung der gewichteten Mittelwerte/Mediane wurde die Fehleranzahl der Projekte, und nicht die Fehleranzahl der Prioritätsstufen eingesetzt
[2] Normiert auf Prioritätsstufe „gering“
n.r. Nicht relevant, Prioritätsklasse für Projekt nicht definiert
k.A. keine Angabe

Tabelle 5.6.: Fehlerbeseitigungsaufwand in Stunden und Kostenwachstum nach Priorität der Fehlermeldung für die untersuchten Automobilprojekte

Infotainment-Projekt Inf1 hebt sich von den übrigen Projekten ab, indem knapp ein Fünftel der Fehlermeldungen (19%=210/1.134, s. Tabelle 5.6) als Showstopper eingestuft wurden. Im Body Controller-Projekt BC2 liegt dieser Wert mit knapp 2% (=11/686) deutlich darunter. Hierfür könnten unterschiedliche Entwicklungsstrategien ursächlich sein, da Infotainment-Systeme andere Eigenschaften aufweisen. In Infotainment-Systemen ist der Fehlerbeseitigungsaufwand je Fehler i. Allg. deutlich geringer als in Body Controller-Systemen (s. Kapitel 5.4.2). Workflow-Systeme sind zudem oft weniger sicherheitskritisch, dafür in den Abhängigkeiten der Funktionen komplexer.

Untersuchungen von Hampp und Knauß [122] bestätigen, dass der Aufwand der Fehlerbeseitigung mit wachsender Fehlerpriorität ansteigt. Das Kostenwachstum mit steigender Priorität könnte möglicherweise auf Korrelationen zwischen Fehlerpriorität und Fehlerklasse beruhen. Dies konnte jedoch aufgrund mangelnder Daten nicht untersucht werden. Das nachgewiesene Kostenwachstum mit steigender Fehlerpriorität indiziert den Einsatz von Teststrategien, die hochpriorisiertes Fehlverhalten speziell bei Body Controllern kosteneffizient finden.

5.4.4. Bewertung

Die Fehlerbeseitigungskosten je Fehler steigen in den durchgeführten Untersuchungen mit den Testphasen an. Bemerkenswert ist hierbei, dass kein Kostenwachstum zwischen Modul- und Systemtest identifiziert werden konnte. Dies kann vielfältige Gründe haben, z. B.:

- Modul- und Systemtestphase wurden quasi gleichzeitig und von den gleichen Testern durchgeführt. Somit wurden auch die Fehlerbeseitigungen im gleichen Entwicklungsstand, dem des Systemtests, durchgeführt. Die beobachteten, identischen Fehlerbeseitigungskosten stehen somit im Einklang mit den Beobachtungen anderer Untersuchungen, die aufzeigen, dass die Fehlerbeseitigungskosten nahezu ausschließlich von der Testphase [14, 28, 76, 78, 155, 192, 193, 223] oder der Verweildauer im System [1] abhängen.
- Es wird angenommen, dass die Testverfahren im Modul- und Systemtest vorrangig Fehlerursachen unterschiedlicher Fehlerklassen identifizieren [26, 53, 53, 112, 176] (s. Kapitel 4.4). Der Einfluss der Fehlerklasse auf die Fehlerbeseitigungskosten könnte gegenläufig sein und den Phaseneffekt aufheben. Dies kann hier jedoch aufgrund der zu geringen Datenbasis nicht getrennt werden.

Als sicher ist weiterhin der Einfluss des Systemtyps auf den absoluten Fehlerbeseitigungsaufwand anzusehen. Ebenso nachgewiesen ist ein Einfluss der Fehlerpriorität auf den Fehlerbeseitigungsaufwand. Es ist sehr wahrscheinlich, dass die drei Einflussfaktoren Testphase, Systemtyp und Fehlerpriorität miteinander korrelieren. Eine Analyse der Abhängigkeiten zwischen den Einflussfaktoren und die Bestimmung des partiellen Effekts dieser Einflussfaktoren auf die Fehlerbeseitigungskosten, z. B. in multiplen Regressionsanalysen, konnte aufgrund der geringen Datengrundlage nicht durchgeführt werden.

Die angenommene [122, 204, 252] Abhängigkeit der Fehlerbeseitigungskosten von den Fehlerklassen wird hier nicht diskutiert. Grund hierfür ist, dass die testfokusorientierte Fehlerklassifikation (s. Kapitel 4.4.1) nach Fehlerursache *und* Testphase der möglichen Fehleridentifikation klassifiziert. Da bereits eine Abhängigkeit von der Testphase nachgewiesen wurde, kann der Einfluss der Testphase den Einfluss der Fehlerursache überdecken. Aufgrund der geringen Datenmenge (19 Fehlermeldungen) kann kein belastbares Ergebnis abgeleitet werden (s. Tabelle I.4 in Anhang I), das um den Einfluss der Testphase bereinigt ist. Die Abhängigkeit der Fehlerbeseitigungskosten von den Fehlerursachen bleibt somit unbetrachtet.

Fazit: Obwohl die Fehlerbeseitigungskosten durch komplexe Kostenabhängigkeiten bestimmt werden, wird für Kostenkalkulationen in der Literatur häufig eine eindimensionale Betrachtung bevorzugt, und zwar der deutliche und praxistaugliche Kostenanstieg über die Testphasen [14, 28, 76, 78, 155, 192, 193, 223]. Diesem Vorgehen schließen sich die Quantifizierungen in dieser Arbeit an.

5.5. Erhebung von Fehlerströmen

Fehlerströme für die Automobilprojekte werden hergeleitet und interpretiert. Die Fehlerströme gehen später in dieser Arbeit in Quantifizierungen zur Strategieoptimierung ein.

Datenbasis und Vorgehen: Für die Projekte BC1, BC2, Inf1, Inf2 sowie M1 liegen Fehlerdatenbanken vor, in denen die Projektphasen der Fehlerentstehung und die Testphasen der Fehleridentifikation dokumentiert sind. Zur Erhebung der Fehlerströme werden die relativen Anteile der in jeder Projektphase entstandenen und der in jeder Testphase identifizierten Fehlerursachen getrennt für Body Controller- und Infotainment-Systeme gemittelt.

Zuordnungskriterien für die Projektphasen der Fehlerentstehung: Die Projektphase der Fehlerentstehung wurde von Testern bzw. Entwicklern im Rahmen einer Fehleranalyse identifiziert und in die Fehlerdatenbanken eingetragen. Unterschieden wurden die Projektphasen *Anforderungsdefinition*, *Entwurf* und *Implementierung*. Waren Spezifikationen fehlerhaft oder unvollständig, wurde die Fehlermeldung der *Anforderungsdefinition* zugeordnet. Analog wurden Fehlerursachen im Quelltext der *Implementierung* zugeordnet. Die Erkennung von Entwurfsfehlern ist weniger eindeutig [53], da klassische Entwurfsfehler, wie z. B. Zeitverhalten, wahlweise durch Entwurfsänderungen als auch durch Implementierungen behoben werden können. In den betrachteten Projekten wurden Fehlermeldungen der Entwurfsphase zugeordnet, wenn die Fehlerbeseitigung durch eine Entwurfsänderung erfolgte.

Ergebnis Fehlerentstehung: In den Fehlerströmen zur Fehlerentstehung weichen die gemittelten Anteile der Body Controller- und Infotainment-Systeme voneinander ab (s. Tabelle 5.7). Infotainment-Systeme zeichnen sich im Vergleich zu Body Controller-Systemen durch deutlich geringere Fehleranteile während der Anforderungsdefinition und des Entwurfs und durch entsprechend mehr Fehleranteile während der Implementierung aus. Dies könnte durch systemtypische Eigenschaften ereignisgesteuerter Systeme (Body Controller) und Workflow-Systeme (Infotainment) bedingt sein. Mit zunehmender Größe eines Workflow-Systems wächst i. Allg. die Anzahl der Systemzustände oder Anwendungskontexte, in denen eine Funktion aufgerufen werden kann. Beispielsweise könnte der Beginn eines Telefongesprächs über Sprachkommandos, Eingabetasten oder eingehenden Anruf geschehen. Letzterer kann weiterhin in unterschiedlichen Systemzuständen oder Kontexten (Navigation, Radiobedienung, CD-Wiedergabe usw.) eintreten. Die Funktionen ereignisgesteuerter Systeme dagegen erlauben einen Funktionsaufruf i. Allg. nur in wenigen Systemzuständen. Beispielsweise kann das Richtungsblinken eines Blinkers nur aus wenigen unterschiedlichen Situationen aktiviert werden. Die beschriebene Komplexität der Workflow-Systeme vergrößert den erforderlichen Quelltextumfang (s. Kapitel 5.1.1) und verstärkt damit möglicherweise die Fehlerentstehung in der Implementierung im Vergleich zu der in Entwurfs- und Anforderungsdefinitionsphase. Andererseits kann der höhere Anteil von Implementierungsfehlers auch durch eine effektivere konstruktive Qualitätssicherung in Anforderungsdefinitions- und Entwurfsphase bedingt sein. Dies lässt sich jedoch an den vorliegenden Daten nicht überprüfen.

Projekt	Projektphase der Fehlerentstehung			Gewichtung (Fehleranzahl)
	Anforderungsdefinition	Entwurf	Implementierung	
Body Controller				
BC1	22%	10%	68%	366
BC2	11%	27%	62%	554
Gewichteter Mittelwert Body Controller	15%	20%	65%	920
Infotainment				
Inf1	5%	12%	83%	2.887
Inf2	2%	3%	95%	546
Gewichteter Mittelwert Infotainment	5%	11%	84%	3.433
Body Controller und Infotainment				
M1	16%	14%	70%	791
Gewichteter Mittelwert Automobil	8%	13%	79%	5.144

Tabelle 5.7.: Fehlerströme zur Fehlerentstehung in den untersuchten Automobilprojekten für Fehleridentifikation ab Systemtest (inkl. erneutem Modultest)

Ergebnis Fehleridentifikation: Abbildung 5.5 visualisiert die gemittelten und entsprechend der Fehleranzahl im Projekt gewichteten Fehlerströme der Automobilprojekte. Die Fehleridentifikationsphasen sind anteilig aufgeteilt auf die Fehlerentstehungsphasen, in denen das jeweils identifizierte Fehlverhalten verursacht wurde. Da die Fehlerdaten der Projekte in gleicher Weise für die aufgeführten Testphasen erfasst wurden, können die Projekte und auch die Systemtypen untereinander verglichen werden. Die Fehlerstromdiagramme der Body Controller-Projekte sind sehr heterogen (s. Tabelle 5.8). Die Verteilung in Projekt M1 weicht mit einem deutlich kleineren Fehleranteil in der Phase Systemintegrations-/Systemtest von den übrigen Ergebnissen ab.

Es ist somit – im Gegensatz zur Fehlerentstehung – in den Verteilungen der Identifikation der Fehlerursachen auf die Identifikationsphasen kein wesentlicher Unterschied zwischen Body Controller- und Infotainment-Systemen nachweisbar. Insgesamt betrachtet, werden nahezu 40% der Fehlerursachen erst durch ein Fehlverhalten beim Kunden, also in späten Testphasen identifiziert. Die Verteilung der Fehlermeldungen durch die testfokusorientierte Fehlerklassifikation (s. Kapitel 5.6) zeigt für Projekt BC1 auf, dass der Großteil der Fehler vor dem beim Kunden durchgeführten Fahrzeugtest hätte identifiziert werden können.

	Fehleridentifikationsphase (Testphasen mit Softwareausführung)			Gewichtung (Fehler-anzahl)
	Modultest (1)	Systemtest	Fahrzeugtest(2)/ Akzeptanztest	
Body Controller				
BC1	31%	47%	22%	366
BC2	12%	46%	42%	554
Gewichteter Mittelwert Body Controller	20%	46%	34%	920
Infotainment				
Inf1	10%	57%	33%	2.887
Inf2	7%	58%	35%	546
Gewichteter Mittelwert Infotainment	10%	57%	33%	3.433
Body Controller und Infotainment				
M1	24%	12%	64%	791
Gewichteter Mittelwert Automobil	14%	48%	38%	5.180

(1) Nur Daten aus einem erneuten Modultest in der Systemtestphase
(2) In den Projekten BC1 und BC2 wurde der Fahrzeugtest beim Fahrzeughersteller durchgeführt, bei den Projekten Inf1 und Inf2 beim Fahrzeugteilezulieferer.

Tabelle 5.8.: Fehlerströme zur Fehleridentifikation in den untersuchten Automobilprojekten

Die Fehlererfassung erst ab Systemtest, ohne Feldeinsatz und nur eingeschränkt im Modultest, führt bei relativen Betrachtungen dazu, dass die Prozentzahlen ab Integrations-/Systemtest höher sind, als wenn alle Phasen erfasst worden wären. Die vergleichende Interpretation des Fehlerflusses, z. B. mit Fehlerströmen der Literatur, ist somit aufgrund der unvollständigen Daten erschwert.

5.6. Analyse von Fehlerklassen

In diesem Kapitel werden die Fehlerursachen des kundenidentifizierten Fehlverhaltens klassifiziert und analysiert. Das vom Kunden (Fahrzeughersteller) festgestellte Fehlverhalten zeigt auf, welche Fehlerursachen nicht in dem beim Hersteller durchgeführten Testprozess erkannt wurden, und ist von Interesse für Optimierungen von Teststrategien. Das Klassifikationsergebnis wird dem aus Untersuchungen in der Literatur (s. Kapitel 3.4) gegenübergestellt.

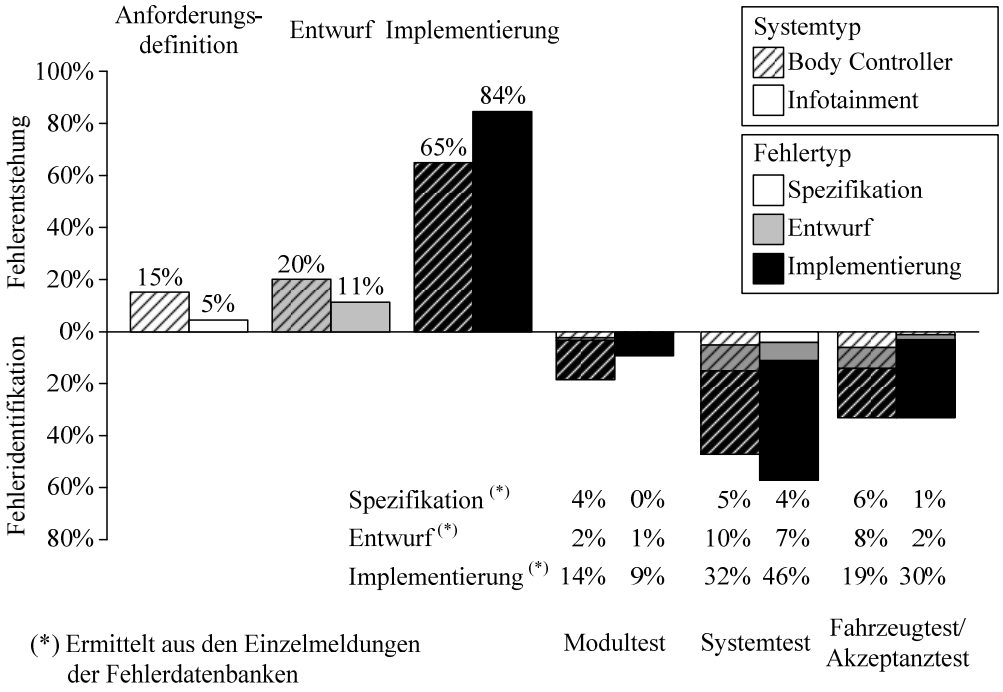


Abbildung 5.5.: Fehlerströme (gewichtete Mittelwerte) zur Fehlerentstehung und -identifikation, getrennt für Body Controller- und Infotainment-Systeme

Datenbasis: Die für eine Klassifikation erforderlichen Fehlerinformationen stehen nur für das Projekt BC1 zur Verfügung und beschreiben Fehlverhalten aus Fahrzeug- und Akzeptanztest. Die Fehlerursachen wurden mit Hilfe der testfokusorientierten Fehlerklassifikation (s. Kapitel 4.4.1) klassifiziert.

Literaturdaten: In der Literatur werden unterschiedliche Fehlerklassifikationen verwendet, insbesondere weichen diese auch von der testfokusorientierten Fehlerklassifikation ab, die ein anderes Klassifikationsziel verfolgt (s. Kapitel 4.4.1). Für den Vergleich der Fehlerverteilungen zwischen Literaturquellen und Projekt BC1 mussten daher Fehlerklassen zusammengefasst bzw. untergliedert. Außerdem waren in den Literaturangaben nicht immer alle Fehlerklassen genannt, so dass die relativen Angaben zu den Fehlerklassen in Tabelle 5.9 nicht zwangsläufig hundert Prozent ergeben.

Ergebnis: Die Verteilungen der Fehlerursachen auf die Fehlerklassen (s. Tabelle 5.9) weisen in Untersuchungen der Literatur und des Projekts BC1 sehr divergierende Anteile auf, was zum Teil auch auf die unterschiedlichen Charakteristika der Datenbasen (z. B. Fahrzeugtest und Akzeptanztest versus Feldeinsatz, unterschiedliche Fachgebiete) und Fehlerklassifikationen (Klassifikationskriterien) zurückgeführt werden kann. Dennoch machen Spezifikations- und Systemverhaltensfehler einen Großteil Fehlverhaltens aus.

Fazit: Aus den Ergebnissen kann abgeleitet werden, dass Testverfahren, die vornehmlich auf Spezifikations- und Systemverhaltensfehler fokussieren, verstärkt eingesetzt werden sollten, um die Fehleridentifikationsrate zu steigern. Zu Bedenken ist jedoch, dass die Erhöhung der Anzahl der identifizierten Fehler nicht das alleinige Ziel optimierter Teststrategien ist. In der Regel können Auswirkungen von Fehlverhalten unterschiedlich kritisch (Sicherheitsaspekt) sein und unterschiedlich hohe Kosten verursachen. Folglich kann das Beseitigen weniger, aber kritischer Fehlerursachen (z. B. Systemausfälle) der Beseitigung vieler, aber unkritischer Fehlerursachen (z. B. orthographische Fehler der Bedienoberfläche) vorgezogen werden. Folglich ist die Wahl der einzusetzenden Testverfahren auch abhängig von einer spezifisch zu formulierenden Zielsetzung und kann nicht nur allgemein aus der Verteilung der Fehlerursachen aus Fehlerklassen abgeleitet werden (s. Kapitel 6.2).

5.7. Zusammenfassung

Im Hinblick auf die Simulation testinduzierter Qualitätskosten wurden in diesem Kapitel die vorliegenden Projektdaten auf kostenstrukturelle Zusammenhänge untersucht und diese – soweit möglich – quantifiziert. Hierfür wurden die Entwicklungsdaten eines Fahrzeugteilezulieferers analysiert. Es liegen verwendbare Fehlerdaten aus zwei Body Controller- und zwei Infotainment-Projekten vor. Ein weiteres Projekt umfasst Systeme beider Typen.

Die untersuchten Fehlerdaten entstammen System-, Fahrzeug- und Akzeptanztests sowie den zugehörigen Integrationstests. Der Systemtest umfasste einen erneuten, intensivierten Modultest. Die Fehlermeldungen weisen Dokumentationslücken auf. Vollständigere und präzisere Daten wären für die quantitativen Auswertungen hier und somit auch für die im nachfolgenden Kapitel 6 intendierten Optimierungen sicherlich von Vorteil gewesen und würden die Aussagekraft der Ergebnisse unterstreichen.

Die zentralen Ergebnisse dieses Kapitels sind:

- Knapp die Hälfte der Projektkosten entfallen auf den Testprozess inklusive Fehlerbeseitigung. Dies entspricht Beobachtungen aus der Literatur.
- Bei den von unabhängigen Testteams verursachten Testkosten ist der Systemtest mit fast zwei Drittel der Testkosten die kostenintensivste Testphase, gefolgt vom Modultest mit fast einem Drittel. Die durch die Tests ausgelösten Fehlerbeseitigungskosten sind jeweils einbezogen. Auf Integrationstests, Fahrzeugtests und Akzeptanztests entfallen insgesamt weniger als 5% der testinduzierten Qualitätskosten. Hierbei sind die Fehlerbeseitigungskosten in die Testphasen inkludiert, nicht jedoch die Testkosten für die beim Kunden durchgeführten Fahrzeug- und Nutzerakzeptanztests.
- In der Fehlerbeseitigung ist etwa die Hälfte der Kosten der Fehlerkorrektur und jeweils etwa ein Viertel der Fehleranalyse und dem Korrekturtest zuzuordnen. Der Korrekturtest wurde hierbei jedoch überwiegend ohne Regressionstest durchgeführt.

Fehlerklassen *(Untergliederungen kursiv)*	Relative Fehleranzahl (%)															
			Gesamte Entwicklung													
	Projekt BC1 (1)	Fahrzeugtest, Akzeptanztest	Basili und Perricone [11, 11]	Standish Group, 1995 [248]	IEEE [79]	Endres, 1975 [82]	Lutz [180], Voyager	Lutz [180], Galileo	Lutz [180], Voyager	Lutz [180], Galileo	Nakajo und Kume, 1991 [200]	Basili, 1984 [11]	Möller, 1993 [193]	Ebert, 1996 [78]	Burghardt, 1997 [47]	Freimut, 2005 [98]	MethodPark, 2007 [187]
Spezifikation	34	48	44	40	46						34	10	20	18	15	20
Neu hinzugekommene Spezifikationen	*16*															
Fehlerhaft/ unvollständig	*18*			*40*	*46*											
Bedingungen	14					17	8	20	14	16						
Grenzwertanalyse	*11*															
MC/DC	*3*															
Systemverhalten	26					34	37	33	32	26						
Integration	16					34	18	35	19							
Performanz	2			4												
Robustheit	8															
Schreibfehler		12								2						
Entwurf		4								20						
Zuweisungen				9												
Methode				12						8						
Absturz																
Nicht zuordenbar										28						

(1) Die Unterteilung nach Testphasen in den Klassen Integration, Performanz wird in diesem Zusammenhang außer Acht gelassen.

Tabelle 5.9.: Prozentuale Anteile der Fehlerklassen, aus Untersuchungen der Literatur sowie Projekt BC1

- Die prospektive Abschätzung des Fehlerbeseitigungsaufwands durch die Tester kann für vorliegende Daten als verlässlich gelten. Im Mittel liegt der real angefallene Aufwand in den untersuchten Projekten 2% über der Schätzung. Für den Großteil der Fehlermeldungen ist die Aufwandsabschätzung präzise, für wenige Fehlermeldungen existieren jedoch deutliche Abweichungen. Die Abweichungen nach oben und nach unten gleichen sich jedoch in etwa aus, so dass die Abschätzung besonders bei summarischen Betrachtungen geeignet ist.
- Die Abhängigkeit der Fehlerbeseitigungskosten von Testphase, Systemtyp und Fehlerpriorität wurde untersucht. Eine Abhängigkeit lässt sich für die Testpha-

se nachweisen. Die Kosten steigen über den vorliegenden Bereich der Testphasen schwach monoton an, wobei kein signifikantes Kostenwachstum zwischen dem im Systemtest durchgeführten Modultest und dem Systemtest beobachtbar war. Die Literatur weist gelegentlich für große Projekte ein stärkeres Kostenwachstum aus. Dies könnte auch für die vorliegenden Daten, in denen die Infotainment-Projekte die größeren Projekte darstellen, gegeben sein, da sich Unterschiede in den Fehlerbeseitigungskosten in Abhängigkeit vom Systemtyp feststellen lassen. Die Ursache des Unterschieds lässt sich jedoch aus den vorliegenden Daten nicht analysieren. Mit steigender Fehlerpriorität steigen ebenfalls die Fehlerbeseitigungskosten. Wegen des geringen Datenumfanges konnte der Zusammenhang von Fehlerbeseitigungskosten und möglichen Einflussfaktoren nur einzeln analysiert werden, obwohl wegen anzunehmender Korrelationen zwischen Testphase, Systemtyp und Fehlerpriorität mehrdimensionale Analysen angebracht wären. Von den untersuchten Einflussfaktoren kann ausschließlich die Testphase der Fehleridentifikation beeinflusst werden und kann somit als Ansatzpunkt für die Optimierung von Teststrategien gelten.

- Im Hinblick auf die intendierten Strategieoptimierungen (s. Kapitel 7) wurden für die fünf Projekte mit Fehlerdaten Fehlerströme ermittelt. Diese erfassen die Fehleridentifikation ab dem erneuten Modultest. Die Fehlerströme sind in den Phasen der Fehlerentstehung stärker von den untersuchten Systemtypen abhängig als in den Phasen der Fehleridentifikation. Im Mittel wurden fast 40% des festgestellten Fehlverhaltens durch den Kunden identifiziert.

Die aufbereiteten Projektdaten aus Kapitel 5 werden in Kapitel 6 und 7 in Zusammenhang mit den Entwicklungen zur Testoptimierung in Kapitel 4 verwendet.

6. Evaluierung von Optimierungsansätzen anhand von Projektdaten

Den später in Kapitel 7 exemplarisch durchgeführten Strategieoptimierungen werden hier einige flankierende Untersuchungen vorangestellt. Der in Kapitel 4 erarbeitete Optimierungsprozess strebt im ersten Schritt eine kostenreduzierende Optimierung durch Fehlerfrüherkennung und den Einsatz von effizienteren Testverfahren an. In Kapitel 6.1 wird zunächst überprüft, ob die vorliegenden Projekte ein Kosteneinsparpotential durch Früherkennung aufweisen:

- Es werden die maximalen Kosteneinsparpotentiale für alle Projekte mit Fehlerdaten überprüft,
- dann wird für ein Projekt ein fehlerklassenoptimiertes Einsparpotential ermittelt.

Eine optimierende Auswahl von Testverfahren muss nicht nur über die im Test unzureichend adressierten Fehlerklassen bestimmt werden, sondern es können z. B. auch Testziele, die festlegen, welche Fehlereigenschaften prioriisert zu identifizieren sind, einbezogen werden:

- In Kapitel 6.2 wird demonstriert, wie Testziele mithilfe des Verfahrens zur testzielorientierten Fehlerklassengewichtung (s. Kapitel 4.4.3) zu Klassengewichtungen transformiert werden, um auf dieser Basis eine zielgerichtete Verfahrensauswahl vornehmen zu können.

In den vorliegenden Projekten werden im Systemtest sowohl Testverfahren mit unsystematischer Testfallerstellung als auch ein funktionsorientiertes und verhaltensmodellbasiertes Verfahren eingesetzt (s. Kapitel 5.1.1). Mit der unsystematischen Testfallerstellung wird ein hoher Anteil der im System befindlichen Systemverhaltensfehler nicht im Testprozess erkannt (s. Tabelle 5.9 in Kapitel 5.6). Daher wird die Effizienz des in den Projekten eingesetzten, verhaltensorientierten Testverfahrens - es handelt sich um das GATE-Verfahren - anhand der vorliegenden Daten analysiert:

- Die Effizienz des funktionsorientierten und verhaltensmodellbasierten GATE-Verfahrens sowie dessen Eigenschaften bezüglich einer Fehlerfrüherkennung werden eingehend mit der entwicklungstypischen, unsystematischen Testmethode verglichen (beides in Kapitel 6.3).

In einem weiteren Schritt des Optimierungsprozesses werden niedrige Testkosten durch Testautomatisierungen verschiedener Art angestrebt. Das Kostenverhalten der bereits in zwei Projekten durchgeführten Testautomatisierungen wird aufgezeigt:

- Die Testausführungskosten werden quantifiziert, Break-Even-Point-Bestimmungen durchgeführt und kostenbestimmende Faktoren diskutiert (Kapitel 6.4).

6.1. Einsparpotentiale durch Fehlerfrüherkennung

6.1.1. Maximales Kosteneinsparpotential

Die Testphase der Fehlerbeseitigung ist bestimmend für die Fehlerbeseitigungskosten [14, 28, 76, 78, 155, 192, 193, 223] und ist durch die Auswahl von Testverfahren beeinflussbar. Somit kann die Testphase der Fehleridentifikation Ziel von Optimierungen sein [17, 177]. Fehlerbeseitigungskosten werden auch durch Faktoren wie Priorität und Systemtyp beeinflusst, wie in den vorliegenden Projektdaten zumindest in Ansätzen nachgewiesen (s. Kapitel 5.4). Diese Faktoren sind aber unveränderliche Fehler- und Projekteigenschaften und können somit nicht direkt das Ziel von Optimierungen sein.

Gemäß den Betrachtungen in Kapitel 4.5 wird in den untersuchten Automobilprojekten anhand des maximalen Kosteneinsparpotentials zunächst grob geprüft, ob erhebliche Kosteneinsparpotentiale in den Fehlerbeseitigungskosten vorliegen. Diese können dann anhand des fehlerklassenoptimierten Kosteneinsparpotentials für Optimierungsmaßnahmen verfeinert bestimmt werden.

Vorgehen, verwendete Daten: Für alle Projekte mit Fehlerdatenbanken werden die maximalen Kosteneinsparpotentiale ermittelt, deren Berechnung in Anhang F detailliert an einem Beispiel erläutert ist. Die Ergebnisse werden für die Projekte einzeln sowie für die Systemtypen und über alle Projekte gemittelt in Tabellen F.1 und F.2 in Anhang F sowie als Auszug in Tabelle 6.1 dargestellt. Das auf die Ist-Kosten bezogene Kosteneinsparpotential wird aus den ermittelten Fehlerströmen (s. Kapitel 5.5) einmal auf Basis des relativen Kostenwachstums in den Projekten (aus Tabelle 5.4 in Kapitel 5.4.1) und einmal auf Basis des relativen Kostenwachstums aus der Literatur (Median aus Tabelle 3.3 in Kapitel 3.3) berechnet, um den Einfluss von Kostenverläufen und Fehlerströmen zu diskutieren. Zudem werden die Kosteneinsparpotentiale den entsprechenden Einsparpotentialen von Literaturprojekten gegenübergestellt. Wegen des intendierten Vergleichs zwischen den Projekten werden relative Kosteneinsparpotentiale nach Gleichung (4.4) in Kapitel 4.5 angegeben.

Substitution von fehlenden Daten zum relativen Kostenwachstum: Für das Projekt BC5 sind die Fehlerbeseitigungskosten der Testphasen ab Modultest unbekannt. Sie wurden durch gemittelte, automobilspezifische Werte abgeschätzt (s. Tabelle 5.4 in Kapitel 5.4.1). In den Literaturprojekten sind die Testphasen Fahrzeug- und Akzeptanztest nicht enthalten und das relative Kostenwachstum muss substituiert werden. Da es sich um kundeninduzierte Fehlerbeseitigungen handelt, wird mit dem Kostenfaktor des Feldtests (einzige Kostenbasis für kundeninduzierte Fehlerbeseitigung in der Literatur) gerechnet.

Ergebnis: Auf Basis der projektspezifischen Kostenverläufe ergeben sich für die untersuchten Automobilprojekte relative maximale Kosteneinsparpotentiale von etwa 63% (Tabelle 6.1). Nach dieser Berechnung liegt das automobile Einsparpotential ab Anforderungsinspektion unter dem der Literaturprojekte (73%). Für den Modultest liegen nur die im Systemtest erfassten Daten vor. Folglich beziffern die berechneten relativen Einsparpotentiale (Tabelle 6.1) – genau betrachtet – ausschließlich das relative maximale Einsparpotential des im Systemtest und in Musterlieferungen (Fahrzeugtest, Akzeptanztest) identifizierten Fehlverhaltens. Wird auch für die Projektgruppe *Literatur* nur das ab Modultest identifizierte Fehlverhalten betrachtet, erhöht sich das Kosteneinsparpotential der Literatur sogar auf 83%. Wie bereits ausgeführt, sind diese maximalen Kosteneinsparpotentiale, die für den konstruierten Fall der Fehlerbeseitigung direkt bei der Fehlerentstehung gelten, keine realisierbaren Größen sondern als einfach zu berechnende Maßzahlen für ein Kosteneinsparpotential zu werten.

In den Automobilprojekten steigt das relative Kostenwachstum in den Phasen mit erfassten Daten relativ schwach an. Wie bereits in Kapitel 5.5 festgestellt, sind die Fehlerbeseitigungskosten des Modultests für Projekt BC1 vermutlich aufgrund der Fehleridentifikation im Systemtest nahezu identisch mit dem der Systemtestphase. Weiterhin sind die kostenintensivsten Fehlermeldungen, die Feldfehler, noch nicht erfasst. Wird für alle Projekte und Projektgruppen die deutlich progressivere Kostenentwicklung der Literatur (s. Tabellen 3.3 und 5.4 sowie Abbildung 6.1) zugrunde gelegt, wird das Einsparpotential der automobilen Projekte stark angehoben. Mit 86% (s. Tabelle 6.1) übertrifft das Kosteneinsparpotential der Automobile den Wert der Literatur mit 83% bei Betrachtung ab Modultest. Zurückzuführen ist dies auf den im Vergleich zum Feldtest der Literatur höheren Fehleranteil im Fahrzeugtest/Akzeptanztest – dies sind hier vor allem die Infotainment-Projekte – bei Verwendung des hohen Kostenfaktors des Feldeinsatzes für den Fahrzeugtest/Akzeptanztest (in Literatur keine adäquateren Phasen vorhanden). Die Verwendung des gleichen testphasenabhängigen Kostenwachstums für alle Projekte macht nun den Effekt der Fehlerströme mit einer prozentualen Überbewertung der Fehleranteile von Systemtest und Musterlieferungen bei den automobilen Projekten sichtbar. Dies zeigt auch auf, dass sich realistische Projektvergleiche nur bei gleichen Phasenunterteilungen ergeben können.

Schlussfolgerung: Die deutlichen Unterschiede der relativen maximalen Einsparpotentiale aus Tabelle 6.1 sprechen dafür, dass bei Abschätzungen eine möglichst projektadäquate Funktion für das Kostenwachstum zugrunde zu legen ist. Dies sollte das Projektmanagement veranlassen, einerseits eine hochwertige Protokollierung der Fehlermeldungen und Aufwandsdaten anzustreben, andererseits als Grundlage für die Ermittlung von Kosteneinsparpotentialen möglichst gleichartige Referenzprojekte heranzuziehen.

Unabhängig vom gewählten Kostenwachstum kann aus den Ergebnissen jedoch geschlossen werden, dass hohe Kosteneinsparpotentiale in jedem Fall vorhanden sind. Die geschätzten Kosteneinsparpotentiale motivieren daher die Durchführung weiterer, präzisierender Kostenuntersuchungen, um die quantitativen Aussagen realitätsnäher

Projekt/-gruppe	Relatives maximales Kosteneinsparpotential bezogen auf die Ist-Kosten(2), berechnet mit dem testphasenspezifischen Kostenwachstum	
	der Projekte	der Literatur (Median)
Body Controller		
BC1	68%	83%
BC2	76%	88%
BC5	61%	75%
Gesamt	74%	86%
Infotainment		
Inf1	57%	85%
Inf2	66%	85%
Gesamt	57%	85%
Body Controller und Infotainment		
M1	66%	89%
Automobil	63%	86%
Literaturprojekte		
ab Modultest (1)	83%	
ab Anforderungsinspektion (1)	73%	
(1) Daten aus Tabellen 3.1 - 3.3 in Kapitel 3 (2) Erläuterungen zur Berechnung in Anhang F		

Tabelle 6.1.: Relative maximale Kosteneinsparpotentiale der Fehlerbeseitigung, berechnet mit dem testphasenspezifischen Kostenwachstum aus den Projekten bzw. aus der Literatur (Auszug aus Tabellen F.1 und F.2 in Anhang F)

abzuschätzen, z. B. über fehlerklassenoptimierte Kosteneinsparpotentiale.

6.1.2. Fehlerklassenoptimiertes Kosteneinsparpotential

Vorgehen: Für die Planungspraxis interessieren im Hinblick auf zukünftige Entwicklungen vor allem realistisch erreichbare Kosteneinsparpotentiale. Exemplarisch wird das Kosteneinsparpotential nach der in Kapitel 4.5 entwickelten Methode der fehlerklassenoptimierten Fehlerbeseitigung (*Kosteneinsparpotential* $_{\textit{Fehlerklassenoptimierte Fehlerbeseitigung}}$) aufgezeigt. Aus dieser ergibt sich das Kosteneinsparpotential, wenn im Testprozess die durch die erforderliche Systemabstraktion vorgegebene, frühestmögliche realisierbare Fehleridentifikation stattgefunden hätte.

Datenbasis: Die Berechnung erfolgt für das Body Controller-Projekt BC1, dessen Datenbasis als einzige eine Klassifikation der Fehlermeldungen erlaubt (s. Tabelle 5.2, Kapitel 5.1.2). Es wurden alle Fehleridentifikationen ab wiederholtem Modultest einbezogen

Berechnungsvorschriften: Die Klassifikation der Fehlermeldungen erfolgt nach manueller Bewertung der Eigenschaften der Fehlursachen mit der in Kapitel 4.4.1 entwickelten, testfokusorientierten Fehlerklassifikation. Die Klassenbesetzungen stellen die Fehlerströme bei optimaler, realistisch möglicher Fehleridentifikation dar und dienen der Berechnung der optimierten Fehlerbeseitigungskosten. Die Berechnung erfolgt auf Basis des relativen Kostenwachstums für die Fehlerbeseitigung der Literatur (s. Tabelle 3.3). Dieses wird verwendet, da es auch Aufwandswerte für die Phasen der Fehlerentstehung enthält. Die Fehlerbeseitigungskosten berechnen sich wie in Kapitel 6.1.1 als Summe der je Testphase gebildeten Produkte aus den Fehleranteilen und dem relativen Kostenwachstum. Das relative Kosteneinsparpotential ergibt sich wiederum nach Gleichung (4.4), bezogen auf die Ist-Kosten der Fehlerbeseitigung.

Ergebnis: Tabelle 6.2 gibt für das maximale und für das fehlerklassenoptimierte Kosteneinsparpotential den geschätzten Anteil der in jeder Testphase identifizierbaren Fälle von Fehlverhalten an. Bei der fehlerklassenoptimierten Fehlerbeseitigung ist zu erkennen, dass im Vergleich zum Ist eine deutlich frühzeitigere Fehlererkennung möglich ist. Werden die Fehleranteile mit den testphasenspezifischen Fehlerbeseitigungskosten (in relativem Kostenwachstum nach Literatur) veranschlagt, ergibt sich gegenüber den Ist-Kosten ein fehlerklassenoptimiertes Kosteneinsparpotential von 50% (s. auch Abbildung 6.1). Dies ist zwar deutlich weniger als die mit 83% (Tabelle 6.2) berechnete maximale Kostenreduktion bei Fehlerbeseitigung in der Fehlerentstehungsphase, dennoch offenbart der realitätsnähere Wert ein durchaus nennenswertes Einsparpotential.

Wird das realisierbare, fehlerklassenorientierte Kosteneinsparpotential voll ausgenutzt, wird analog nach Gleichung (4.5) das theoretisch maximale Kosteneinsparpotential zu 60% (=50%/83%) ausgeschöpft.

Für Projekte aus der Literatur liegen keine Fehlerströme unter Angabe der Verteilung auf Fehlerklassen vor. Somit kann kein fehlerklassenoptimiertes Kosteneinsparpotential für Literaturprojekte zu Vergleichszwecken berechnet werden.

Schlussfolgerung: Es ist evident, dass eine frühestmögliche Fehlerbeseitigung in Projekt BC1 auch unter realistisch erreichbaren Bedingungen zu hohen Kostenreduktionen führen kann.

Die Quantität der Kostenreduktion kann nur auf andere Body Controller oder auf andere automobile Projekte übertragen werden, wenn eine Vergleichbarkeit der Fehlerströme und des Kostenwachstums vorliegt. Aus den Ergebnissen des theoretisch maximalen Kosteneinsparpotentials in kann einerseits gefolgert werden, dass eine quantitative Übertragbarkeit auf andere Projekte begrenzt sein wird, da das Kostenwachstum über die Testphasen maßgeblich das Ergebnis beeinflusst und je

Kosteneinsparpotentiale der Fehlerbeseitigungskosten für Projekt BC1									
Methode zur Berechnung	Testphase der Fehlerbeseitigung	Anforderungs-definition	Entwurf	Implementierung	Modultest	Systemtest	Fahrzeugtest/ Akzeptanztest	Normierte Fehler-beseitigungskosten[4]	Relatives Kosten-einsparpotential
Optimierte Fehleridentifikation									
		Geschätzter Anteil Fehleridentifikationen							
Maximales Kosteneinsparpotential	Fehler-entstehung[1]	22%	10%	68%				0,42	83%
Fehlerklassen-optimiertes Kosteneinsparpotential	frühestmögliche Identifikation nach Fehlerklassen[2]	-	-	-	76%	24%	0%	1,24	50%
Ist-Fehleridentifikation									
		Ist-Anteil Fehleridentifikationen							
Ist-Fehlerbeseitigung	Ist-Identifikation[1]	-	-	-	31%	47%	22%	2,46	
Daten zur Berechnung									
		Testphasenabhängiges Kostenwachstum für Fehlerbeseitigung							
Relatives Kostenwachstum[3]		0,2	0,3	0,5	1,0	2,0	5,5		

(1) Fehleranteile aus Tabelle 5.7, Kapitel 5.5
(2) Klassifikation aller ausreichend beschriebenen Fehlermeldungen in BC1 durch manuelle Bewertung der Fehlereigenschaften und Fehlerursachen
(3) Relatives Kostenwachstum der Literatur (Median) nach Testphase aus Kapitel 3.3, Tabelle 3.3
(4) Summe aus den je Testphase gebildeten Produkten aus Fehleranteil und relativem Kostenwachstum (Erläuterung in Anhang F)
- Keine Phase des Testprozesses

Tabelle 6.2.: Maximales und fehlerklassenoptimiertes Kosteneinsparpotential der Fehlerbeseitigungskosten für Body Controller-Projekt BC1

nach Projekt und Unternehmen stark divergieren kann, jedoch scheint andererseits ein Kosteneinsparpotential vorhanden zu sein, das wert ist, genutzt zu werden.

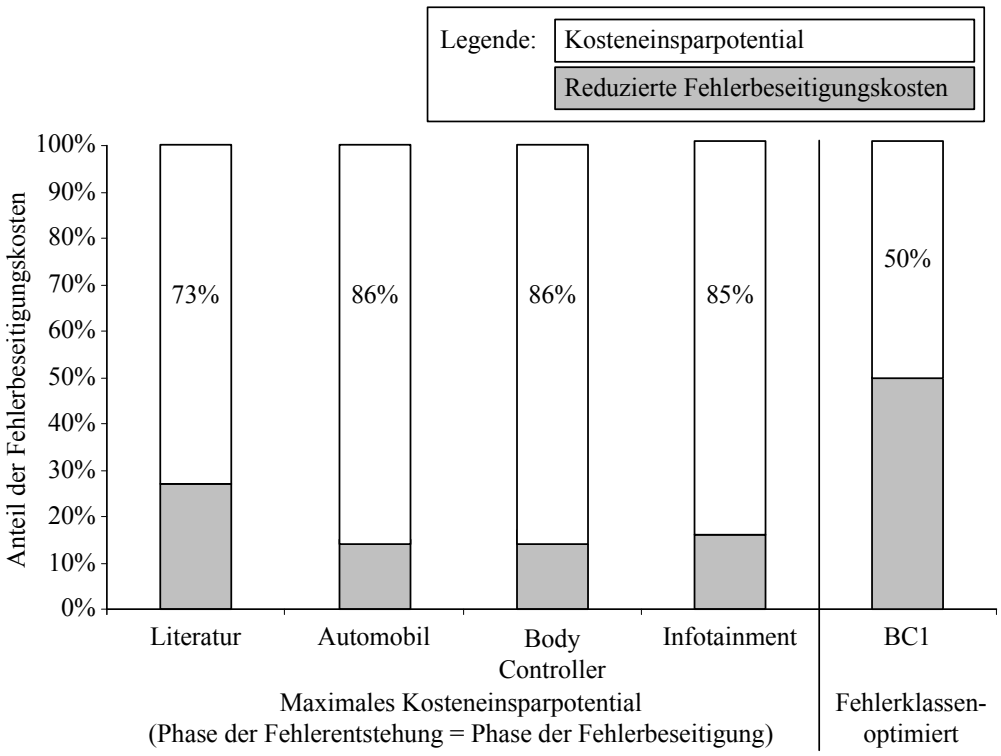


Abbildung 6.1.: Relative Kosteneinsparpotentiale der Fehlerbeseitigung, berechnet anhand der testphasenspezifischen Kostenentwicklung (relatives Kostenwachstum) der Literatur (Median) (Ist-Kosten = 100%)

6.2. Priorisierung von Fehlerklassen durch Testziele

Wie in Kapitel 5.6 aufgezeigt, sollte die Wahl der einzusetzenden Testverfahren nicht nur von den Häufigkeitsverteilungen der Fehlerursachen, die in vorangegangenen Projekten nicht im Testprozess identifiziert wurden, abhängig gemacht werden, sondern zudem auch von einem Testziel, das projektspezifisch Präferenzen festlegt. Nachfolgend wird an Projektdaten exemplarisch demonstriert, wie ein Testziel mit Hilfe des in Kapitel 4.4.2 entwickelten Verfahrens zur testzielorientierten Fehlerklassengewichtung als Klassengewichtung abgebildet werden kann. Die gewichteten Fehlerklassen können dann als Grundlage für eine Verfahrensauswahl herangezogen werden, s. Kapitel 4.4.3.

Definition Testziel bzw. Präferenz zur Testverfahrensauswahl: Die Ergebnisse in Kapitel 6.1 zeigen ein erhebliches Kosteneinsparpotential durch Fehlerfrüherkennung

auf. Somit wird hier die Fehlerfrüherkennung als primäres Ziel definiert. Um die Einsatzmöglichkeiten der testzielorientierten Fehlerklassengewichtung umfassender zu demonstrieren, wird für die exemplarische Anwendung ein zweiteiliges Testziel angenommen:

- Es soll Fehlverhalten mit hoher Fehlerpriorität mit gewissem Vorrang identifiziert werden (Motivation ergibt sich aus den Beobachtungen in Kapitel 5.4.3).
- Es sollen die Fehlerklassen frühzeitiger identifiziert werden, die ein hohes Kosteneinsparpotential erwarten lassen.

Dabei soll exemplarisch die Fehlerpriorität doppelt so stark in die Berechnung einfließen wie das Kosteneinsparpotential.

Verwendete Fehlerdaten zur Fehlerklassifikation: Die Methode zur testzielorientierten Fehlerklassengewichtung erfordert die Klassifizierung von Fehlermeldungen aus einem Referenzprojekt. Als Referenzprojekt ist nur BC1 möglich, da nur in diesem Projekt ausreichend Fehlerinformationen für eine Fehlerklassifikation verfügbar sind (s. Kapitel 5.1.2). Für die Fehlerklassifikation wird das vom Kunden im Fahrzeugtest und Akzeptanztest identifizierte Fehlverhalten betrachtet. Dieses Fehlverhalten wurde im bisherigen Testprozess nicht erkannt und verursachte einen höheren Beseitigungsaufwand als Fehlverhalten, das frühzeitig vom Hersteller im Test identifiziert wurde. Es werden Fehlermeldungen aller Iterationen betrachtet.

Fehlerklassengewichtung: Die Klassifikation erfolgte durch manuelle Sichtung der Fehlerbeschreibungen und ergibt ein ungewichtetes Fehlerprofil. Die Berechnung der anhand des Testziels gewichteten Häufigkeitsverteilungen ist detailliert in Anhang G aufgeführt.

Ergebnis: Das Fehlerprofil hat sich durch die einzelnen Gewichtungen mit den Aspekten *Priorität* und *Frühe Identifikation* sowie deren Kombination jeweils leicht geändert (s. Tabelle 6.3 und Abbildung 6.2). Die Gewichtungen beeinflussen Systemintegrations-, Systemrobustheits- und Systemperformanzfehler nicht messbar, wohl aber Bedingungs-, Grenzwert-, Spezifikations- und Systemverhaltensfehler (s. Spaltendifferenz, Tabelle 6.3). Die Fehlerklassen *Bedingung* und *Spezifikation* werden in diesem Beispiel überwiegend aufgrund der geringen Fehlerpriorität abgewertet (s. Spalte 2), die Fehlerklasse *Systemverhalten* aus gegenteiligem Grund stärker gewichtet. Vor allem *Grenzwertfehler* werden aufgrund der frühen Fehleridentifikation stärker gewichtet (s. Spalte 3). Die Fehlerklasse *Grenzwert* gewinnt durch beide Gewichtungsaspekte an Gewicht. Die stärksten Abweichungen bei gleichzeitig weiterhin den deutlich höchsten Gewichtungen weisen Spezifikations- und Systemverhaltensfehler auf.

Schlussfolgerung: Diese Ergebnisse zeigen, dass die Zielsetzung, nämlich Fehlverhalten mit hoher Fehlerpriorität und mit hohen Fehlerbeseitigungskosten im Testprozess effektiv zu identifizieren, gewährleistet ist, wenn im Testprozess effiziente Methoden zur Identifikation von Spezifikations- und Systemverhaltensfehler verstärkt eingesetzt werden. Die Gewichtung verändert die Aussage der relativen

Besetzte Fehlerklasse(1)	Fehlerverteilung				Einfluss der Gewichtungs-methode (Spaltendiffe-renz (4)-(1))
	Ungewichtet (bzw. mit *Fehleranzahl* gewichtet) Spalte 1	Mit *Fehlerpriorität* gewichtet (normiert) Spalte 2	Mit *Frühe Identifikation* gewichtet (normiert) Spalte 3	Gesamt-gewichtung G_k (normiert) Spalte 4	
Modultest					
Bedingung	4%	3%	4%	3%	-1%
Grenzwert	10%	11%	13%	12%	2%
Systemtest					
Spezifikation(2)	35%	29%	33%	30%	-5%
Systemintegration	16%	16%	16%	16%	0%
Systemverhalten	25%	31%	24%	29%	4%
Systemrobustheit	8%	8%	8%	8%	0%
Systemperformanz	2%	2%	2%	2%	0%
Summe	100%	100%	100%	100%	0%

(1) Geordnet nach frühestmöglicher Testbarkeit im Testprozess
(2) Im Rahmen des Testprozesses frühestens im Systemtest durch Testverfahren adressierbar.

Tabelle 6.3.: Vom Kunden (Fahrzeughersteller) induzierte Fehlermeldungen des Projekts BC1. Exemplarische Gewichtung aller besetzten Fehlerklassen

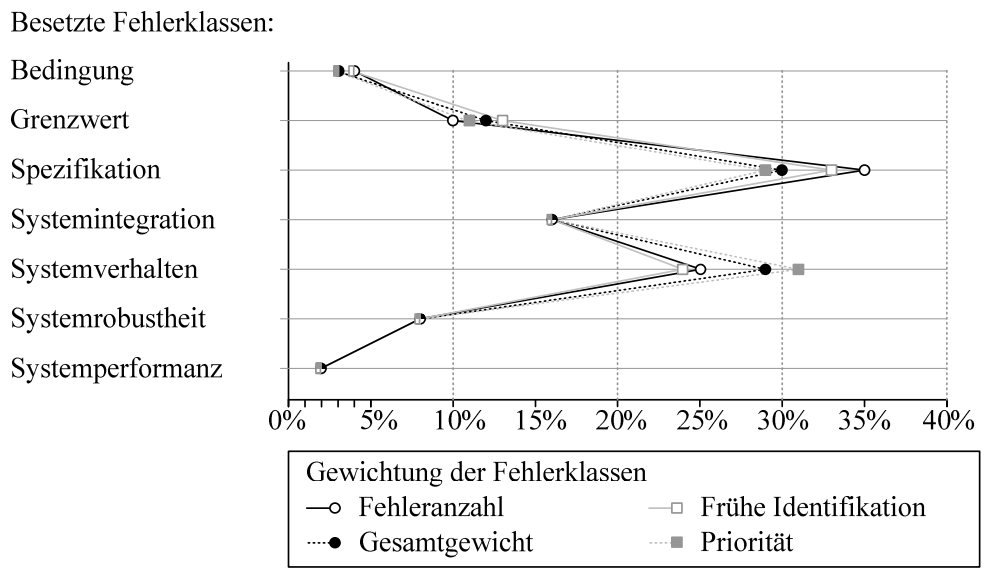


Abbildung 6.2.: Fehlerprofil des Projekts BC1 vor (=Fehleranzahl) und nach (=Gesamtgewicht) Anwendung der Gewichtungsmethode

Fehlerhäufigkeit (ungewichtetes Fehlerprofil) in diesem Fall nicht grundlegend, bestätigt aber das Ergebnis und verstärkt daher die Aussagekraft.

Die Literatur [7,41,59,60,91,100,174,214,215,231] tendiert zur Fokussierung beider Fehlerklassen durch Verhaltensmodellbetrachtungen. Im hier betrachteten Testprozess (beginnend mit dem Modultest) kann somit der Einsatz von verhaltensmodellbasierten Testverfahren zum Testziel beitragen.

In der Literatur wurde keine Bearbeitung zu dem hier exemplarisch gewählten Ziel der Strategieoptimierung gefunden. Daher kann das Resultat nicht vergleichend diskutiert werden. Allerdings ergibt sich eine Bestätigung bezüglich der zu fokussierenden Fehlerklassen: Die Literatur verweist auf die besondere Bedeutung von Spezifikations- und Systemverhaltensfehler im Umgang mit eingebetteten Systemen [11, 82, 91]. Die hier dargelegten Ergebnisse stützen die Erkenntnisse aus der Literatur in besonderer Weise, da die hier analysierten Fälle von Fehlverhalten nicht im Testprozess sondern erst vom Kunden entdeckt wurden.

6.3. Fehlerfrüherkennung durch ein funktionsorientiertes und verhaltensmodellbasiertes Testverfahren

Identifikation von Systemverhaltensfehlern: Untersuchungen in Kapitel 5.6 belegen sowohl für Projekt BC1 als auch für die Literatur, dass neben Spezifikationsfehlern Systemverhaltensfehler dominieren. Da diese Arbeit auf den dynamischen Test fokussiert, sind hier vornehmlich Möglichkeiten zur Kostenreduktion durch Früherkennung von Systemverhaltensfehlern im Testprozess, in diesem Falle im Systemtest, aufzuzeigen. Das besagt nicht, dass nicht auch in eine Fehlervermeidung zu investieren wäre. Für die Identifikation von Systemverhaltensfehlern gelten spezifikationsbasierte und verhaltensmodellbasierte Testverfahren als effektiv [13, 241]. Verhaltensmodellbasierte Testverfahren erzeugen häufig Testfälle aus zustandsbasierten Verhaltensmodellen wie Sequenzdiagrammen oder Zustandsautomaten [241]. Zustandsautomaten beschreiben im Gegensatz zu Sequenzdiagrammen das Systemverhalten umfassend und nicht nur partiell [131, 232]. In dem Großkonzern, aus dem die vorliegenden Entwicklungsdaten stammen, werden derzeit Zustandsautomaten vorgezogen. Ein funktionsorientiertes und verhaltensmodellbasiertes Testverfahren wurde bereits ansatzweise integriert.

Das GATE-Verfahren: In dem Konzernbereich, in dem die Entwicklung von eingebetteten Systemen derzeit einen Schwerpunkt bildet, wurde speziell für den Systemtest von Steuergeräten ein funktionsorientiertes und verhaltensmodellbasiertes Testverfahren (*GATE-Verfahren – Generic and Automated Test Environment*) entwickelt. Das GATE-Verfahren fokussiert auf das Systemverhalten, welches durch hierarchische Zustandsautomaten [41, 202] zu beschreiben ist. Darüber hinausgehend, wird der verhaltensmodellbasierte Ansatz mit einer funktionsorientierten Komponente in Form einer *Äquivalenzklassenanalyse* [149, 176, 241] gekoppelt. Hierbei werden die Wertebereiche der Eingabevariablen unter Zuhilfenahme der *Klassifikationsbaummethode* [113] in Äquivalenzklassen unterteilt. Die Grenzen der Äquivalenzklassen werden durch eine *Grenzwertanalyse* [149, 176, 222, 241] validiert. GATE generiert,

u. a. auf den Verhaltensmodellen basierend, automatisiert Testfälle entsprechend dem *Wp-Pfadüberdeckungskriterium* [101, 102].

Untersuchungsziel: Die Effektivität der systematisch auf dieser Basis erzeugten Testfälle wird nachfolgend exemplarisch in einer iterativen Systementwicklung untersucht. Hierbei werden die Fehleridentifikationsraten des systematischen Testverfahrens mit denen eines angewendeten, entwicklungstypischen, unsystematischen Tests verglichen.

6.3.1. Vorgehen Effektivitätsprüfung

Voraussetzungen für den validen Vergleich von Testverfahren: Ein valider Vergleich von Testverfahren verlangt die Ausführung der zu vergleichenden Testverfahren unter identischen Rahmenbedingungen: gleiches System unter Test, zeitgleiche Ausführung zwecks Verhinderung von Lessons Learned-Testfällen [156, 241], mit identischen Personen und identischem Wissenstand usw. Dies ist jedoch aufgrund wirtschaftlicher Gesichtspunkte meist nur in künstlichen Umgebungen, z. B. anhand von kleinen Beispielprojekten im akademischen Umfeld möglich. Die Übertragbarkeit der Ergebnisse aus kleinen Beispielprojekten auf die Praxis ist allerdings nicht uneingeschränkt gegeben [28].

Für den intendierten Methodenvergleich wird ein Versuch im industriellen Umfeld (Testteam, Zeit- und Budgetbeschränkungen usw.) vorgezogen, der allerdings nicht kompromissfrei durchzuführen ist.

Projektauswahl: Das GATE-Verfahren wurde in den Body Controller-Projekten BC1, BC2, BC3 und BC4 eingesetzt, allerdings in unterschiedlichem Umfang und verschiedener Weise. Die Projekte BC2, BC3 und BC4 liefern dazu keine analysierbaren Fehlerdaten. In Projekt BC1 wurde das GATE-Verfahren in der dritten Iteration für etwa ein Drittel des Systems eingesetzt. Dadurch ist ein Vergleich der Testverfahren im selben Projekt, mit identischem Testteam und vergleichbaren Systemteilen möglich. Daher wird Projekt BC1 für die Analyse herangezogen.

Beschreibung Systementwicklung: Der Body Controller BC1 umfasst zwölf *Body Controller-Funktionen (BCF)*. Die BCFs wurden in einer iterativen Anwendung des *V-Modells* [37] entwickelt. In jeder Iteration wurde der Funktionsumfang der BCFs erweitert und getestet. Betrachtet werden hier die Systemtests von sechs V-Modell-Iterationen. Es wurde durchgängig – mit der im nachfolgenden Absatz beschriebenen Ausnahme – ein derzeit im Entwicklungsumfeld entwicklungstypischer, unsystematischer Test angewendet. Dieser beinhaltet eine unsystematische Testfallgenerierung, die auf der Erfahrung der Tester sowie auf der Vorgabe basiert, dass für jede Anforderung mindestens ein Testfall auszuführen ist. Die Wiederholbarkeit der Testdatenherleitung ist bei diesem Vorgehen nicht gegeben.

Untersuchungsgruppen, Bewertungskriterium: Die BCFs 1-4 wurden für die intendierte Effektivitätsuntersuchung in der dritten Iteration im Systemtest mit

dem GATE-Verfahren validiert. Verglichen mit dem unsystematischen Test soll die Qualität der BCFs durch Verringerung der Restfehleranzahl erhöht werden. Die zur automatisierten Testfallgenerierung notwendigen Verhaltensmodelle, z. B. Zustandsautomaten [41, 202], wurden von den Testern aus textuellen Spezifikationen manuell aufgestellt. Die Tester wurden hinsichtlich der Modellerstellung und Testfallgenerierung vor Durchführung der empirischen Untersuchung durch eine zweitägige Schulung befähigt, unter Coaching die notwendigen Verhaltensmodelle aufzustellen und die Testfälle automatisiert zu generieren. Im Falle des Projekts BC1 erwies sich die resultierende Testfallanzahl jedoch als zu hoch. Daher wurde die Testfallanzahl auf die *paarweise Überdeckung von Parameterwerten* [266] reduziert und anschließend manuell ausgeführt.

Die BCFs 5-12 wurden in der dritten Iteration unter Anwendung des herkömmlichen, unsystematischen Tests getestet.

Vergleichbarkeit der Gruppen: Die BCFs der beiden Gruppen (BCFs 1-4 und BCFs 5-12) sind identisch hinsichtlich der Systemeigenschaften wie Programmiersprache oder Erfahrung der Entwickler. Unterschiede finden sich hauptsächlich bzgl. der Anzahl BCFs, Systemgröße, Systemkomplexität (Zyklomatische Zahl [185]: CYC_{BCF1-4}/CYC_{BC5-12}=0,55) und Anzahl Systemanforderungen (Verhältnis: 0,59). Die Systemkennzahlen beschreiben einen proportionalen Zusammenhang zur Anzahl der BCFs pro Gruppe, was auf eine generelle Vergleichbarkeit der beiden Systemgruppen schließen lässt.

Eigenschaften der Verhaltensmodelle: Das GATE-Verfahren verwendet u. a. Verhaltensmodelle für die Testfallherleitung. Die dafür verwendeten deterministischen Zustandsmaschinen umfassten bis zu vier Hierarchieebenen, wofür je nach BCF zwischen 10 und 40 Zustände, die durch etwa 25 bis 70 Transitionen verbunden waren, verwendet wurden. Die Größe der Zustandsmaschinen ist entsprechend zum Funktionsumfang der BCFs als eher gering einzustufen. Zum Vergleich: der Zustandsautomat, der das Infotainment-System Inf1 beschrieb, umfasste mehr als 30.000 Zustände und mehr als 100.000 Transitionen.

6.3.2. Ergebnisse

Effektivere Fehleridentifikation: Abbildung 6.3 zeigt je BCF-Gruppe das im Systemtest identifizierte Fehlerverhalten nach Iterationen. In den ersten beiden Iterationen wurden alle zwölf BCFs entsprechend des unsystematischen Tests validiert. Die Zahl der Fehleridentifikationen ist in den BCF 5-12 jeweils höher. Die Werte der prozentualen Verteilungen gleichen sich.

In der dritten Iteration wurde der Funktionsumfang erheblich erweitert, was sich in allen zwölf BCFs durch eine deutlich erhöhte Anzahl von Fehleridentifikationen widerspiegelt. Mit dem Ziel der Qualitätsverbesserung wurden die BCFs 1-4 mit einem systematischen Testverfahren (hier: GATE-Verfahren) getestet. Die Referenz-BCFs 5-12 wurden weiterhin mit unsystematisch erstellten Testfällen getestet. Die restlichen

Rahmenbedingungen wie Programmier- und Testteams sowie Automatisierungen blieben konstant. Mit dem systematischen GATE-Verfahren (BCFs 1-4) wurden im Gegensatz zu den vorhergehenden Iterationen häufiger Fehlverhalten in BCF 1-4 identifiziert als in BCF 5-12 (s. Abbildung 6.3). Der prozentuale Wert der BCF 1-4 ist 1,5-mal höher als der für BCF 5-12. Die Effektivitäten der Testverfahren, als Anteil der herstellerinduzierten Fehleridentifikationen an den gesamten Fehleridentifikationen in der Iteration, von 75% bzw. 62% sind in Tabelle 6.4 dargestellt.

BCF-Gruppe	Anzahl Fehleridentifikationen in Iteration 3			Anteil der herstellerinduzierten Fehleridentifikationen
	Hersteller-induziert im Systemtest	Kundeninduziert im Fahrzeug- oder Akzeptanztest	Gesamt	
BCF 1-4	50	17	67	75%
BCF 5-12	42	26	68	62%

Tabelle 6.4.: Effektivität der Testverfahren im Systemtest

In der vierten Iteration wurde der Funktionsumfang nur wenig erweitert. Sämtliche zwölf BCFs wurden hier wieder entsprechend dem unsystematischen Test getestet. Die Anzahl Fehleridentifikationen sank in den zuvor mit dem GATE-Verfahren getesteten BCFs 1-4 deutlicher als in den restlichen BCFs 5-12. Diese Beobachtung ist darauf zurückzuführen, dass das GATE-Verfahren Fehlverhalten effektiver und somit früher identifiziert hat und damit die Restfehlerzahl gesunken ist bzw. möglicherweise eine Verbreitung von Folgefehlern auf die nächste Iteration unterblieben ist.

In der fünften Iteration wurde der Funktionsumfang beider BCF-Gruppen erweitert. Beide BCF-Gruppen wurden entsprechend dem unsystematischen Test getestet und weisen in etwa wieder gleich hohe Fehleranzahlen auf.

In der sechsten Iteration wurden nur wenige neue Funktionen hinzugefügt. Weiterhin lagen zur Zeit der Datenerfassung noch nicht alle Fehlerdaten dieser Iteration vor. Daher sind die Anzahlen der Fehleridentifikationen dieser Iteration derart gering.

Reduktion kundeninduzierter Fehleridentifikationen: Zur Bewertung eines Testverfahrens ist neben der Anzahl der durch Tests identifizierten Fehlerursachen (s. o.) auch die Anzahl der Fehleridentifikationen durch den Kunden (hier im Fahrzeugtest und Akzeptanztest) zu betrachten, da sich die Gesamtzahl der im System befindlichen Fehler verändert haben kann – wie hier durch die Weiterentwicklungen von Iteration zu Iteration. Folglich kann aus einer gestiegenen Anzahl Fehleridentifikationen im Hersteller-Test nicht geschlossen werden, dass die Anzahl der kundeninduzierten Fehleridentifikationen sinken wird.

Abbildung 6.4 zeigt die absoluten Anzahlen der nach Musterlieferung an den Fahrzeughersteller gemeldeten Fehlverhalten, aufgeschlüsselt nach Iteration. Es

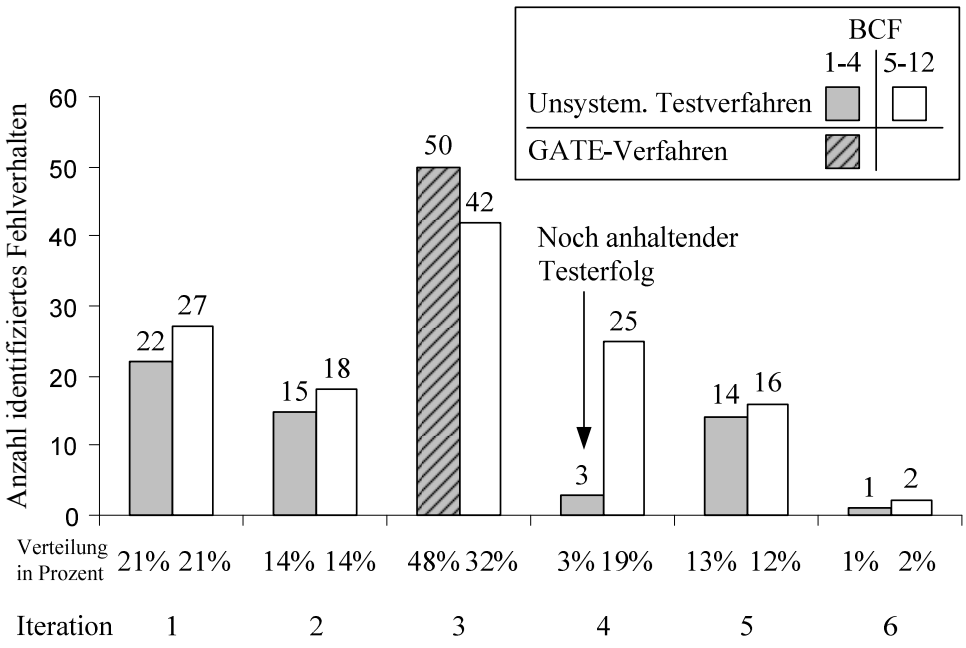


Abbildung 6.3.: Fehleridentifikationen im Systemtest nach Iterationen und deren prozentuale Verteilung (Projekt BC1)

ist zu erkennen, dass in den ersten beiden Iterationen in den BCFs 1-4 häufiger Fehlverhalten vom Fahrzeughersteller identifiziert wurde als in den BCFs 5-12. In Iteration drei wurden nach Anwendung des GATE-Verfahren in den BCFs 1-4 weniger Fehlverhalten durch den Kunden gemeldet als bei Anwendung des unsystematischen Tests für BCFs 5-12. Der gemeinsame Anstieg der beiden Gruppen im Vergleich zur zweiten Iteration ist auf den deutlich erhöhten Funktionsumfang zurückzuführen. Der Testerfolg des GATE-Verfahrens bei den BCFs 1-4 scheint auch in Iteration vier noch durch deutlich geringere Beanstandungen vom Fahrzeughersteller nachzuwirken. Zum Zeitpunkt der Datenerfassung lagen für die fünfte und sechste Iteration noch keine Fehlermeldungen des Kunden (Fahrzeughersteller) vor.

Fehleridentifikationen durch den Kunden nach Fehlerklassen: Weitere Aufschlüsse zur Effektivität des eingesetzten, verhaltensmodellbasierten Testverfahrens ergeben sich aus der Fehlerklassifikation des durch den Kunden (Fahrzeughersteller) gemeldeten Fehlverhaltens. Hierzu werden die nach Abschluss der dritten Iteration identifizierten Fehlerursachen mittels der testfokusorientierten Fehlerklassifikation den Fehlerklassen zugeordnet, getrennt nach BCF-Gruppen.

Unter Abwägung der geringen Datengrundlage lassen sich aus Abbildung 6.5 folgende Auswirkungen des in Projekt BC1 angewendeten Testverfahrens ziehen:

- Nach Anwendung des GATE-Verfahrens bleibt im Vergleich zum unsystematischen Test ein geringerer Anteil an Systemverhaltensfehlern im Systemtest

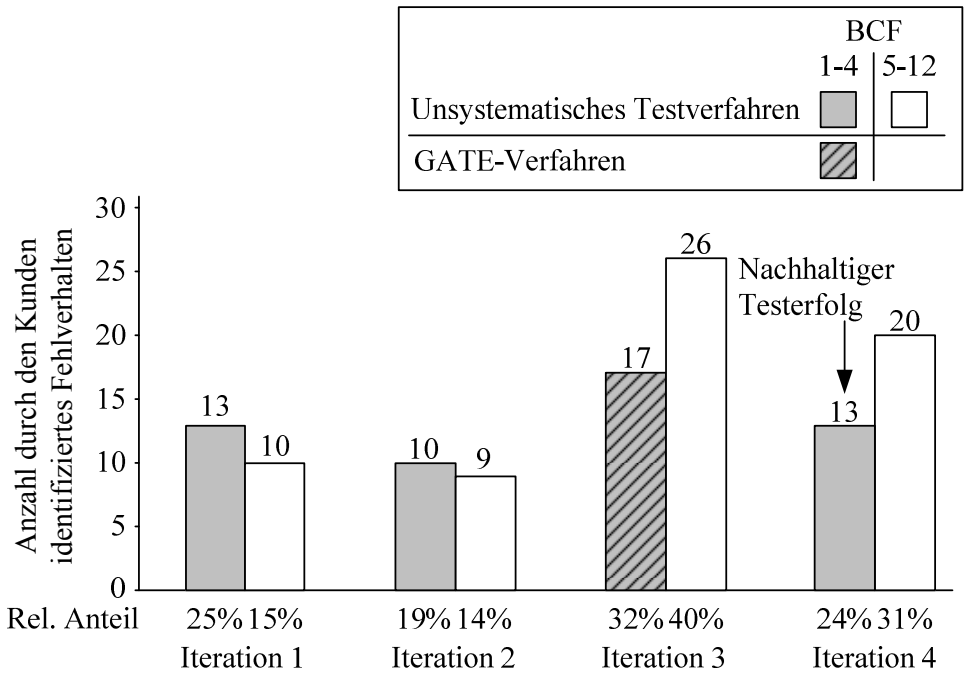


Abbildung 6.4.: Fehleridentifikationen durch den Kunden im Projekt BC1 (im Fahrzeugtest und Akzeptanztest beim Kunden identifiziertes Fehlverhalten nach Iterationen)

unentdeckt. Folglich scheint das GATE-Verfahren Systemverhaltensfehler zwar nicht vollständig, jedoch verstärkt zu identifizieren, was zu einer Reduktion der Fehleranzahl im Fahrzeugtest und Akzeptanztest führte. Das Untersuchungsergebnis muss wegen der geringen Datenbasis in zukünftigen Projekten weiter abgesichert werden.

- In Projekt BC1 wurden Spezifikationsfehler durch das Aufstellen von Verhaltensmodellen evident. Die deutlich geringere Anzahl an nicht identifizierten Spezifikationsfehlern spiegelt den Effekt des Aufstellens von Verhaltensmodellen wider.
- Performanz und Robustheitsfehler werden vom GATE-Verfahren nicht fokussiert. Unsystematische Tests, die auf der Erfahrung der Tester basieren, identifizieren daher mehr Fehlerursachen dieser Fehlerklassen und verringern somit die Anzahl nicht identifizierter Fehlerursachen.

Vergleichbare Analysen aus der Literatur liegen nicht vor.

Kostenreduzierte Fehlerbeseitigung: Wie in diesem Kapitel verdeutlicht, meldet der Kunde (Fahrzeughersteller) nach Anwendung des GATE-Verfahren weniger Fehlverhalten im Vergleich zum herkömmlichen Testverfahren (s. Abbildung 6.4), da die Fehlerursachen bereits im Systemtest identifiziert und beseitigt wurden. Folglich ist von einer Kostenersparnis durch die frühe Fehlerbeseitigung auszugehen.

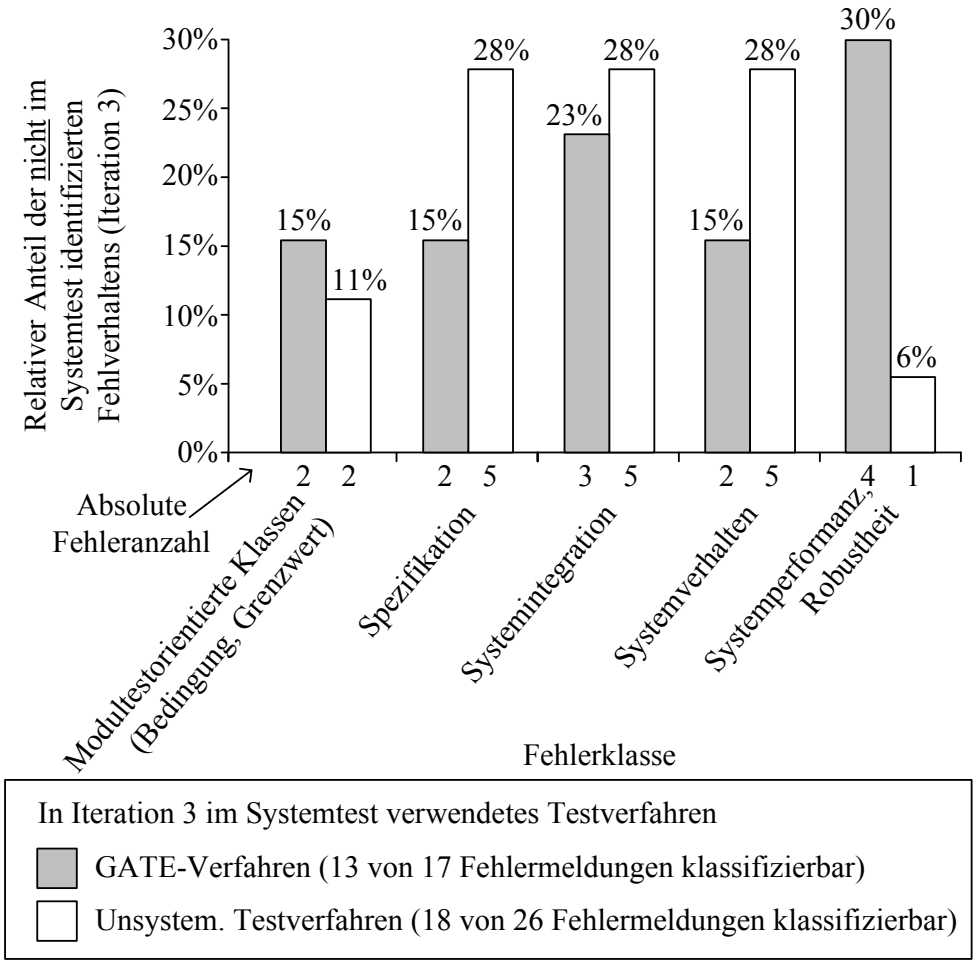


Abbildung 6.5.: Verteilung der vom Kunden (Fahrzeughersteller) identifizierten Fehlverhalten in Iteration 3 auf die Fehlerklassen mit Identifikationsmöglichkeit im Systemtest

Die Kostenersparnis beträgt unter Verwendung des relativen Kostenwachstums der Literatur für Systemtest und Feldeinsatz (Tabelle 3.3 in Kapitel 3.3) 13,5%, s. Tabelle 6.5.

Kostenreduktion durch verkürzte Entwicklungszeiten: Der Testmanager des Projekts BC2, in dem das GATE-Verfahren ebenfalls angewendet wurde, bestätigte die gesteigerte Effektivität: durch die Anwendung des Testverfahrens konnten zwei Iterationen, die zur Steigerung der Qualität eingeplant waren, aufgrund schneller erreichter Qualitätsziele eingespart werden. Auch in den Untersuchungen im Projekt BC1 wurde gesteigerte Effektivität durch eine Fehleridentifikation in früheren Iterationen sichtbar (s. Abbildung 6.3). Hieraus kann sich dann u. U. eine größere Kostenreduktion ergeben, als durch Fehleridentifikation in der frühestmöglichen

Testphase innerhalb einer Iteration [139].

BCF-Gruppe	Identifizierte und beseitigte Fehlerursachen in Testphasen der Iteration 3				Relative Beseitigungskosten einer Fehlerursache	Relative Kostenreduktion
	Systemtest		Fahrzeugtest und Akzeptanztest			
	abs.	rel.	abs.	rel.		
BCF 1-4 (GATE-Verfahren)	50	74,6%	17	25,4%	2,89(2)	(3,34-2,89)/ 3,34 = 13,5%
BCF 5-12 (unsystematischer Test)	42	61,8%	26	38,2%	3,34(2)	
Relatives Kostenwachstum(1)	2,0		5,5			

(1) Relatives Kostenwachstum (Median) der Literatur für Systemtest und Feldeinsatz, Tabelle 3.3 in Kapitel 3.3
(2) Berechnet als Summe der je Testphase gebildeten Produkte aus relativem Fehleranteil und relativen Kostenwachstum

Tabelle 6.5.: Relative Kostenreduktion durch frühere Fehlerbeseitigung nach Anwendung des GATE-Verfahrens (Projekt BC1).

Bewertung der Ausnutzung des Kosteneinsparpotentials: In Kapitel 6.1.2 wurde für Projekt BC1, basierend auf dem Kostenwachstum der Literatur, ein fehlerklassenoptimiertes Kosteneinsparpotential von 50% (s. Tabelle 6.2, Kapitel 6.1.2) berechnet. Nach Tabelle 6.5 konnten jedoch nur 13,5% der Fehlerbeseitigungskosten ab Systemtest durch Anwendung des GATE-Verfahrens eingespart werden. Dies könnte auf den ersten Blick auf eine geringe Kostenreduktion und gar eine ineffektive Fehleridentifikation des GATE-Verfahrens hindeuten. Das GATE-Verfahren adressiert jedoch nur Systemverhaltens- und Grenzwertfehler und das nur während des Systemtests. So kann selbst bei optimaler Fehleridentifikation beider Fehlerklassen im Systemtest nicht das gesamte fehlerklassenoptimierte Kosteneinsparpotential von 50% erreicht werden, da dies sich testphasenübergreifend und auf alle Fehlerklassen bezieht. Das realistisch maximal erreichbare Kosteneinsparpotential im Systemtest ergibt sich für das GATE-Verfahren bei Identifikation sämtlicher Systemverhaltens- und Grenzwertfehler. Dies ist damit das Einsparziel des GATE-Verfahrens. Dieses Kosteneinsparpotential kann wie folgt abgeschätzt werden: Entsprechend der bereits in BC1 durchgeführten Analysen sind 26% der nach dem Systemtest eingetragenen Fehlermeldungen auf Systemverhaltensfehler zurückzuführen, weitere 11% auf Grenzwertfehler (s. Kapitel 5.6, Tabelle 5.9). Folglich sollte der Anteil des im Systemtest nicht identifizierten Fehlverhaltens bei optimaler Fehleridentifikation um 37% (=26%+11%) reduziert werden. In Tabelle 5.9 wird die Kostenreduktion bei Identifikation sämtlicher Systemverhaltens- und Grenzwertfehler abgeschätzt. Auszugehen

ist von den Daten der BCF 5-12, da diese BCFs in Iteration 3 mit dem herkömmlichen Testverfahren getestet wurden. Nach den vorangegangenen Überlegungen sollten 37% der 26 im Systemtest mit herkömmlichen Testverfahren nicht identifizierten Fehler durch das GATE-Verfahren im Systemtest identifiziert werden. Dies führt zu 52 im und 16 nach dem Systemtest identifizierten Fehlverhalten und folglich zu einer Kostenreduktion von etwa 15% (15,4% fehlerklassenoptimiertes Einsparpotential, s. Tabelle 6.6, Abbildung 6.4). Tatsächlich erreichte die praktisch durchgeführte Anwendung des GATE-Verfahrens in den BCF 1-4 eine nahe heranreichende Kostenreduktion von 13,5% (s. Tabelle 6.5). Die Effektivität des GATE-Verfahrens bzgl. der Identifikation von Systemverhaltens- und Grenzwertfehlern wird analog zu Gleichung (4.5) mit 88% (=13,5/15,4) berechnet und kann damit als gut angesehen werden.

Das in Tabelle 6.2 ermittelte, fehlerklassenoptimierte Kosteneinsparpotential in Höhe von 50% ist nur durch ein Zusammenspiel mit weiteren effizienten Testverfahren erreichbar (s. Abbildung 6.6): Nach Tabelle 6.6 ließen sich 15,4% der Fehlerbeseitigungskosten durch Identifikation aller Systemverhaltens- und Grenzwertfehler im Systemtest einsparen. Würden sämtliche vom Kunden identifizierten Fehler im Systemtest gefunden, wäre eine Kosteneinsparung von weiteren 25% möglich. Die Identifikation von Spezifikationsfehlern während der Anforderungsdefinition sowie der Bedingungs- und Grenzwertfehler im Modultest kann weitere 10% der Fehlerbeseitigungskosten einsparen. Dadurch wäre das gesamte fehlerklassenoptimierte Kosteneinsparpotential aus Kapitel 6.1.2 mit 50% ausgenutzt.

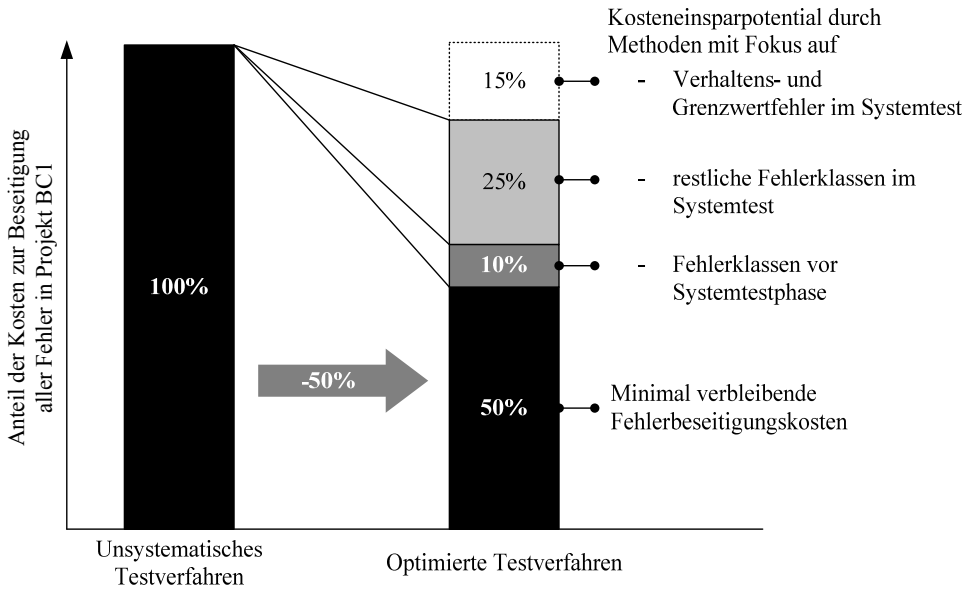


Abbildung 6.6.: Verteilung des fehlerklassenoptimierten Kosteneinsparpotentials in Projekt BC1

Testverfahren im Systemtest in Iteration 3 in den BCF 5-12	Identifizierte Fehlerursachen in den Testphasen				Rel. Beseitigungskosten einer Fehlerursache (2)	Relative Kostenreduktion
	Systemtest		Fahrzeugtest und Akzeptanztest			
	abs.	%	abs.	%		
Ist-Zustand: Durch unsystematischen Test identifizierte Fehlerursachen	42	61,8%	26	38,2%	3,34	(3,34 - 2,82) / 3,34 = 15,4%
GATE (optimal): Alle Systemverhaltens- und Grenzwertfehler identifiziert	52 (=42 + MAX * 26)	76,5%	16 (=(1 - MAX) * 26)	23,5%	2,82	
Relatives Kostenwachstum (1)	2,0		5,5			

(1) Relatives Kostenwachstum (Median) der Literatur für Systemtest und Feldeinsatz, Tabelle 3.3 aus Kapitel 3.3

(2) Berechnet als Summe der je Testphase gebildeten Produkte aus relativem Fehleranteil und relativem Kostenwachstum

MAX Maximaler Anteil der durch GATE fokussierten Fehler. Hier 0,37 (37%) – ergibt sich aus 26% Systemverhaltensfehlern und 11% Grenzwertfehlern.

Tabelle 6.6.: Kostenreduktion durch frühere Fehlerbeseitigung bei Identifikation aller Systemverhaltens- und Grenzwertfehler im Systemtest der BCF 5-12 (Projekt BC1)

Effizientere Testfallerstellung – Erfahrungsberichte: Das GATE-Verfahren wurde außer in Projekt BC1 auch in den Projekten BC2, BC3, BC4 und BC5 eingesetzt. Im Vorfeld des Methodeneinsatzes wurde von Testmanagern häufig die Erstellung der Verhaltensmodelle für den Systemtest als zu aufwändig kritisiert. Die Erstellung der verhaltensorientierten Testfälle erwies sich jedoch in allen Projekten innerhalb des ohnehin knappen, für die herkömmliche Testfallerstellung kalkulierten Zeitrahmens als möglich. Die Testmanager beurteilten den Zeitaufwand der Verhaltensmodellierung mit dem leicht zu bedienenden Standardwerkzeug Microsoft Visio [189] als wirtschaftlich positiv. Der Testmanager des Projekts BC2 urteilte nach der Einführung des GATE-Verfahrens, dass die Testfallerstellung zeiteffizienter gewesen sei. In Projekt BC3 benötigten Tester während der Verfahrenseinführung vier anstelle der nach herkömmlichen Verfahren veranschlagten acht Tage (Zeitersparnis 50%). In Projekt BC3 benötigte weiterhin ein hinzugezogener, erfahrener Tester für die Testfallgenerierung inkl. Modellerstellung zwei von zehn geplanten Tagen (Zeitersparnis 80%).

Berichte über deutliche Kostenreduktionen finden sich auch in der Literatur. Bei Leirios Technologies wurde die Beobachtung gemacht, dass das Erstellen eines Testfalls auf herkömmliche Weise etwa eine halbe Stunde dauerte [173]. Mit modellbasierter Testfallgenerierung dagegen seien hundert Testfälle an einem Nachmittag möglich. Folglich sei eine *andere Größenordnung* [173] erreicht worden. Nach einer anderen Literaturquelle können durch modellbasierte Testfallgenerierung in der Zeit, in der mit herkömmlichen Verfahren etwa 20 Testfälle erzeugt wurden, 100 bis 300 Testfälle erzeugt werden [174]. Dies ist eine Effizienzsteigerung um den Faktor fünf bis 15.

Zusammenfassend kann geschlossen werden, dass das Aufstellen von Verhaltensmodellen und die Herleitung von modellbasierten Testfällen weniger aufwändig ist als unsystematisches Herleiten von Testfällen. Gleichzeitig identifizieren die aus Verhaltensmodellen erstellten Testfälle nach den in diesem Kapitel durchgeführten Analysen Fehlerursachen der fokussierten Fehlerklasse effektiver.

6.3.3. Bewertung

In der empirischen Gegenüberstellung eines in der Praxis angewendeten unsystematischen Tests und eines eingeführten systematischen Testverfahrens wird ersichtlich, dass ein systematisches Testverfahren die fokussierten Fehlerursachen deutlich effektiver und früher identifizieren kann. Das maximal zu erreichende Kosteneinsparpotential für Systemverhaltensfehler im Systemtest konnte zu einem Großteil ausgeschöpft werden. Frühzeitige Beseitigung der Fehlerursachen verringert zudem die Anzahl der Folgefehler (*Fault Propagation* [26, 51, 188, 257]) und erreicht einen iterationsübergreifenden, kostenreduzierenden Effekt.

Ein erhöhter Aufwand zur Modell- und Testfallerstellung wurde nicht beobachtet. Vielmehr wurden sogar bereits während des ersten Einsatzes von Zeit- und damit Kostenreduktionen berichtet. Das Aufstellen – ohne Ausführung – von verhaltensorientierten Testfällen erwies sich durch die explizite Betrachtung von Abhängigkeiten zwischen Anforderungen bereits als Mittel zur Identifikation von Spezifikationsfehlern. In Anbetracht des hohen Anteils von kundenidentifizierten Spezifikationsfehlern sollten Modell- und Testfallerstellung in frühen Projektphasen parallel zur Anforderungsinspektion erfolgen. Heute ist zwar weitläufig bekannt [182], dass frühzeitige Investitionen in den Entwicklungs- und Testprozess Fehlerbeseitigungskosten in späten Phasen deutlich senken können, jedoch ist die Umsetzung noch nicht ausreichend in der Entwicklungspraxis angekommen.

Es existieren zahlreiche Untersuchungen, die einen Effektivitätszuwachs bzgl. der Fehleridentifikation durch Anwendung modellbasierter Testtechniken beschreiben [16, 20, 27, 54, 62, 64, 75, 87, 95, 210, 218, 235]. Diese Befunde können mit dem auf Verhaltensmodellen basierenden GATE-Verfahren bestätigt werden. Die Wirksamkeit anderer Testverfahren kann in analoger Weise mit den gezeigten Methoden ermittelt werden.

6.4. Optimierung durch Testautomatisierung

Die Industrie strebt niedrige Testkosten durch Automatisierungen an [76]. Kosten und Nutzen von Testautomatisierungen sollen in diesem Kapitel auf Basis der in Kapitel 4.6 eingeführten Berechnungsvorschriften zur Break-Even-Point-Bestimmung an den vorliegenden Entwicklungsdaten analysiert werden. Testautomatisierungen in zwei Projekten werden auf Wirtschaftlichkeit untersucht und verglichen.

6.4.1. Break-Even-Point für Projekt BC1

Kosten und Nutzen der in Projekt BC1 im Systemtest vorgenommen Testautomatisierungen sollen bewertet werden durch Berechnung der Anzahl an Iterationen, die zur Erreichung des Break-Even-Points notwendig ist. Die Testautomatisierung wird als wirtschaftlich angesehen, wenn die Entwicklung mindestens die berechnete Anzahl Iterationen erreicht.

Automatisierungseigenschaften: Automatisierungsdaten des Projekts BC1 liegen für den Systemtest in Form von Testprotokollen und Projektbudgetplänen vor. Im Systemtest standen insgesamt 2.537 definierte Testfälle zur Verfügung (s. Tabelle 6.7). Es wurden jedoch nicht in jeder Iteration alle Testfälle, sondern im arithmetischen Mittel nur 2.039 Testfälle (80%) ausgeführt (670 manuelle und 1.369 automatisierte Testfallausführungen). Dies entspricht einem „allgemeinen“Automatisierungsgrad von G^A=67%. Die Anzahl der insgesamt automatisiert ausführbaren Testfälle (von den 2.537 Testfällen) liegt nicht vor. Wird angenommen, dass die Automatisierungsgrade für Testfallerstellung G^{4A} und TestdurchführungG^{5A} gleich sind (G^{4A}=G^{5A}), so kann die Anzahl der automatisierten Testfälle für die Berechnung der Automatisierungskosten aus dem Automatisierungsgrad hochgerechnet werden.

Anzahl erstellter Testfälle	2.537
davon manuelle Testfälle (hochgerechnet)	834
davon automatisierte Testfälle (hochgerechnet)	1.703
Mittlere Anzahl ausgeführter Testfälle je Iteration	2.039
davon manuelle Testfälle (arithm. Mittel)	670
davon automatisierte Testfälle (arithm. Mittel)	1.369
Automatisierungsgrad, G^{4A}=G^{5A}	67%
Prozent ausgeführter Testfälle pro Iteration	80%

Tabelle 6.7.: Testfälle und deren Automatisierungen für Projekt BC1

Für Projekt BC1 liegen keine differenziert erfassten Aufwandsinformationen zur *Anpassung der Automatisierung* vor. Sie sind vielmehr in den Testfallautomatisierungskosten enthalten. Daher werden diese zu amortisierenden Kosten in der exemplarischen Break-Even-Point-Berechnung nicht explizit ersichtlich.

Kosten der Testautomatisierung: Die Kosten der Testautomatisierung mit Automatisierungsgrad G^A=*67%* berechnen sich nach Gleichung (C.7) in Anhang C und werden in Tabelle 6.8 aufgeführt. Bei der Variante ohne Testautomatisierung (Automatisierungsgrad = *0%*) fallen keine Testfallautomatisierungskosten an.

Testfallautomatisierung für die automatisierte Durchführung Automatisierungsgrad 67%	
Automatisierungskosten eines Testfalls	74 €[(1)]
Zu automatisierende Testfallanzahl	1.703
$K^{Testfallautomatisierung}$	126.022 €
$K^{LizenzenTestfallautomatisierungswerkzeuge}$ (entnommen aus Projektbudgetplänen)	211.200 €
Summe Testfallautomatisierungskosten und Lizenzkosten	337.222 €

[(1)] Wert ist abgeleitet aus den Projektbudgetplänen. Dies entspricht fast neun Testfällen pro Tag (=7,5h pro Tag * 85€ pro h/74€ pro Testfall).

Tabelle 6.8.: Abschätzung Testfallautomatisierungskosten für Projekt BC1

Durchführungskosten manuell bzw. automatisiert ausgeführter Testfälle: Die Durchführung eines Testfalls verursacht Personalaufwand. Es wird angenommen, dass sowohl für die manuelle als auch die automatisierte Testdurchführung genau ein Tester anwesend ist, obwohl es auch denkbar wäre, die automatisierte Testdurchführung unbeaufsichtigt auszuführen [76]. Die Berechnung schätzt also den Personalaufwand zur automatisierten Testdurchführung nach oben hin ab.

Die Testdurchführungskosten berechnen sich aus dem Stundensatz eines Testers (entnommen aus Projektbudgetplänen) und dem angefallenen Zeitaufwand zur Testdurchführung in Minuten (gemittelter Wert aus den Testprotokollen). Entsprechend den Testprotokollen benötigte ein manuell ausgeführter Testfall nahezu 15-mal so viel Zeit wie ein automatisiert ausgeführter Testfall (s. Tabelle 6.9) oder anders ausgedrückt: die Testautomatisierung kann die Testdurchführungskosten um etwa 93% reduzieren.

Testdurchführungskosten gesamt: Die Testdurchführungskosten der gesamten Testfallmenge berechnen sich aus der Anzahl manuell und automatisiert ausgeführter Testfälle, jeweils multipliziert mit den Durchführungskosten eines einzelnen Testfalls (s. Tabelle 6.10). Die Testautomatisierung in Projekt BC1 spart etwa 63% der im Systemtest anfallenden Testdurchführungskosten gegenüber dem ausschließlich manuellen Test ein.

	Testdurchführung	
	Manuell	Automatisiert
$A^{Testdurchführung}$ (in Minuten)	10,4	0,7
$K^{StundensatzeinesTesters}$	85 €	85 €
Personalkosten pro Testfall	14,73 €	0,99 €
Absolute Kostenreduktion	13,74 €	
Relative Kostenreduktion	93%	

Tabelle 6.9.: Reduktion der Personalkosten in der Durchführung eines Testfalls durch Testautomatisierung für Projekt BC1

	Automatisierungsgrad		
	G^4 = 0%	G^{4A} = 67%	
	Manuell	Manueller Anteil	Automatisierter Anteil
Personalkosten pro Testdurchführung	14,73 €	14,73 €	0,99 €
Testfallanzahl (zur Durchführung)	2.039	670	1.369
Personalkosten aller Testdurchführungen	30.034 €	9.869 €	1.355 €
$K^{Testdurchführung}{}_G$	30.034 €	11.224 €	
Absolute Kostenreduktion	18.810 €		
Relative Kostenreduktion	63 %		

Tabelle 6.10.: Kostenreduktion in der Testdurchführung einer Iteration durch Testautomatisierung bei einem Automatisierungsgrad von 67% für Projekt BC1

Break-Even-Point: Die Anzahl der Iterationen, die notwendig sind, um die zusätzlichen Kosten der Automatisierung zu amortisieren (Break-Even-Point), wird durch Einsetzen der berechneten Werte in die Gleichung (4.7) in Kapitel 4.6.3 ermittelt:

$$\begin{aligned} I &= \frac{\textit{Testfallautomatisierungskosten aus Tabelle 6.8}}{\textit{Absolute Kostenreduktion aus Tabelle 6.10}} \\ &= 17,9\ \textit{Iterationen} \end{aligned} \tag{6.1}$$

Aus Gleichung (6.1) ergibt sich 18 als die notwendige Zahl an Iterationen (s. auch Abbildung 6.7). In der Praxis durchlief der Systemtest des Projekts BC1 jedoch nur sechs Iterationen, somit war die Testautomatisierung unter diesen Bedingungen nicht wirtschaftlich.

Abhängigkeit des Break-Even-Points von Lizenzkosten: Die Automatisierung in Projekt BC1 ist laut Projektbudgetplänen mit sehr hohen Lizenzkosten für Automatisierungswerkzeuge belastet. Dies motiviert, die Abhängigkeit des Break-Even-Points von den Fixkosten zu untersuchen. Das Diagramm in Abbildung 6.8 visualisiert das Kostenverhältnis automatisierter Test zu manuellem Test (Break-Even-Points) in Abhängigkeit von Lizenzkosten und Iterationsanzahl. Die helle Fläche signalisiert einen erreichten Break-Even-Point. Der Faktor, um den die Automatisierung teurer ist als der manuelle Test, liegt hier zwischen 0,0 und 1,0. Es ist evident, dass geringe Lizenzkosten einen Break-Even-Point bereits mit wenigen Iterationen (so z. B. ca. sieben Iterationen beim Wegfallen der Lizenzkosten) ermöglichen. Projekt BC1 durchlief bis zum Zeitpunkt der Analyse sechs Iterationen, was – ohne Lizenzkosten – bis dahin fast zur Erreichung des Break-Even-Points geführt hätte.

Abhängigkeit des Break-Even-Points von Iterationen, Testfallanzahl und Lizenzkosten: Der Break-Even-Point hängt neben der Iterationsanzahl auch vom Automatisierungsgrad ab. Wird auch die Anzahl der automatisiert ausgeführten Testfälle als variabel angesehen, so ergibt sich eine Kostenabhängigkeit nach Abbildung 6.9. Es ist deutlich zu erkennen, dass ein Break-Even-Point (graue Fläche) bei einer hohen Anzahl von Testfällen nach wenigen Iterationen erreicht wird. Die hier aufgezeigten Abhängigkeiten machen deutlich, dass Kosten und Nutzen von Testautomatisierungen nach den spezifischen Gegebenheiten abgewogen werden müssen.

Ergebnisbewertung und Verallgemeinerung: Die Kostenentwicklungen der Testdurchführungen mit und ohne Automatisierungen verdeutlichen in Abbildung 6.7 für das Projekt BC1, dass bei geringen Iterationsanzahlen die manuelle Testdurchführung wesentlich geringere Kosten verursacht. Der Grund hierfür ist, dass die Automatisierung in Projekt BC1 mit hohen Fixkosten belastet ist, welche hauptsächlich durch die sehr hohen Lizenzkosten für Automatisierungswerkzeuge (Quelle: Projektbudgetpläne) entstehen. Diese fallen aufgrund firmeninterner Kostenkalkulationen und

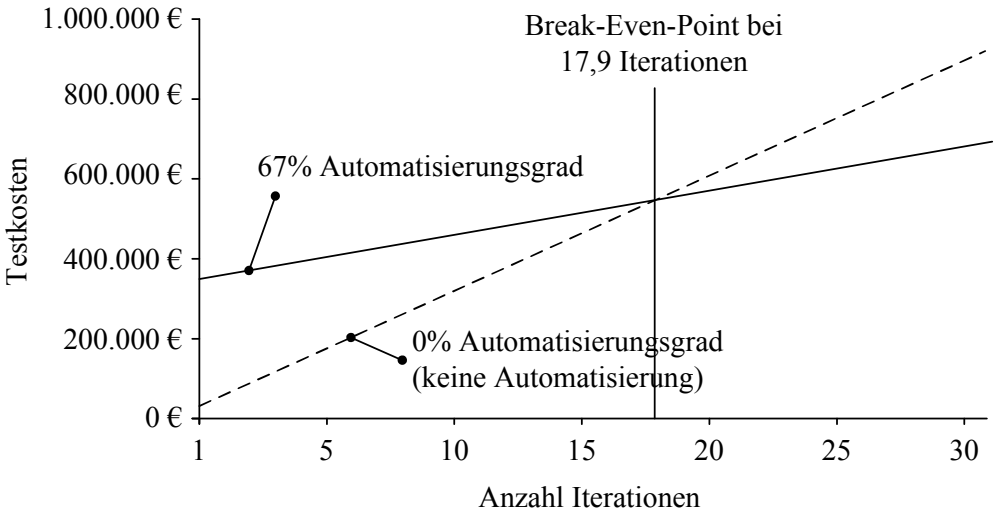


Abbildung 6.7.: Testkosten mit und ohne Testautomatisierung für Projekt BC1

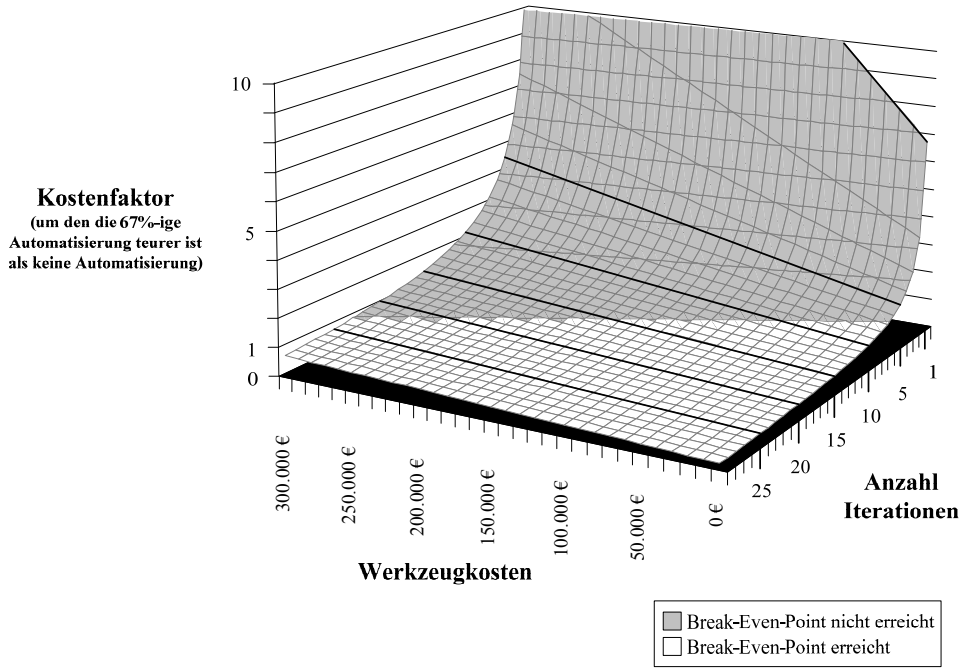


Abbildung 6.8.: Kostenverhältnisse und Break-Even-Points der Testautomatisierung in Abhängigkeit von Iterationsanzahl und Lizenzkosten bei einem Automatisierungsgrad von 67% für Projekt BC1

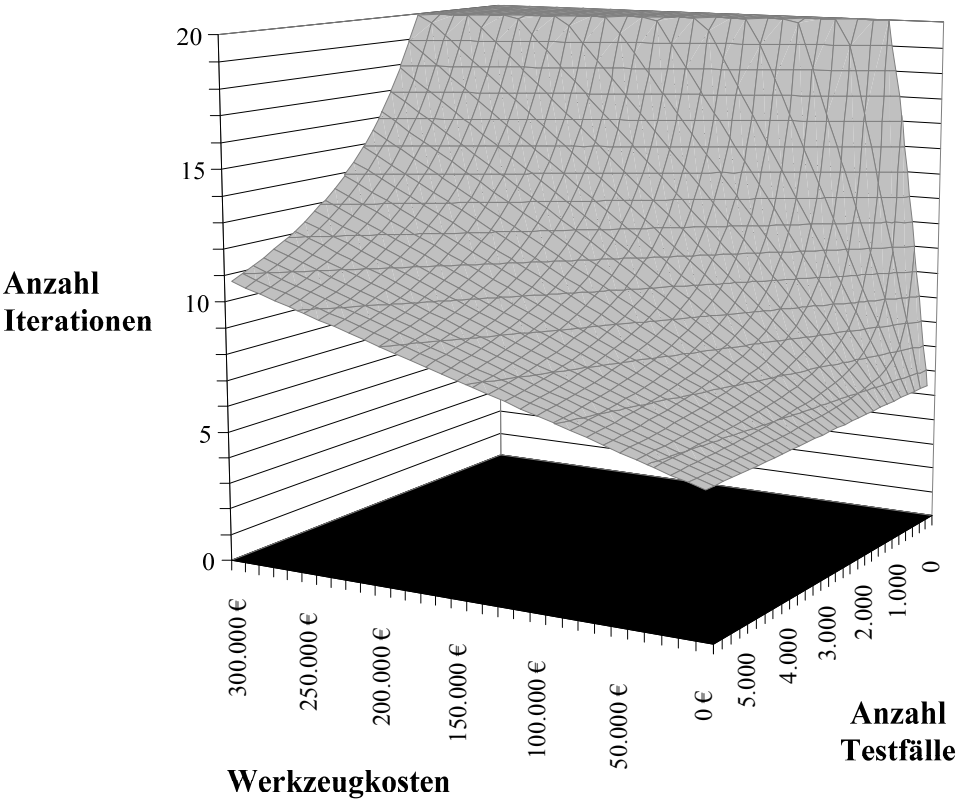


Abbildung 6.9.: Break-Even-Points in Abhängigkeit von Iterationsanzahl, Anzahl Testfälle und Fixkosten bei 67%-iger Testautomatisierung

Gegebenheiten sehr hoch aus. Zwar umfassen die Lizenzkosten bereits Kosten für Wartung und Hardware, dennoch sind i. Allg. geringere Kosten anzunehmen (s. Projekt BC2 in Kapitel 6.4.2). Die Break-Even-Point-Bestimmung wurde für den im Projekt BC1 vorliegenden Automatisierungsgrad von 67% ermittelt. Selbst bei einem Automatisierungsgrad von 100%, also 2.039 automatisierten Testfällen und sonst unveränderten Bedingungen, sind 16 Iterationen zum Erreichen des Break-Even-Points notwendig (s. Abbildung 6.9). Die Testautomatisierung rentiert sich also bei den ungewöhnlich hohen Kosten für Automatisierungswerkzeuge und geringen Iterationszahlen nicht. Im Falle geringerer Lizenzkosten ist jedoch mit realistisch erreichbaren Kostenreduktionen zu rechnen. Die Ergebnisse lassen sich aufgrund der spezifischen Gegebenheiten nur eingeschränkt auf andere Projekte übertragen.

6.4.2. Break-Even-Point für Projekt BC2

Abschätzung der Durchführungskosten: Bei der Abschätzung des Break-Even-Points für Projekt BC2 wird analog zu Kapitel 6.4.1 vorgegangen, allerdings kann nur auf eine Datenzusammenstellung des zuständigen Testmanagers zur Kosten-Nutzen-

Betrachtung der Automatisierung zurückgegriffen werden, die um Annahmen ergänzt werden muss: In Projekt BC2 wurden 2.021 Testfälle erstellt, von denen etwa 80% automatisiert wurden. Ein Tester automatisierte nach Angaben des Testmanagers in Projekt BC2 pro Tag durchschnittlich fünf Testfälle. Damit ist die Testfallautomatisierung in Projekt BC2 fast doppelt so kostenintensiv wie in Projekt BC1 (fünf bzw. mehr als neun Testfallautomatisierungen pro Tag). Der Testmanager gab an, dass in jeder Iteration alle Testfälle ausgeführt wurden. Die Testdurchführung eines manuellen Testfalls benötigte fünf Minuten, die eines automatisierten Testfalls 18 Sekunden. Dies entspricht einer Ersparnis der Durchführungszeit von 94%. Die Lizenzkosten für Werkzeuge wurden mit 17.400€ angegeben, und sind somit äußerst gering. Die Automatisierungskosten eines Testfalls wurden auf den beschriebenen Daten basierend mit 136€ (=8h * 85€/h / 5 Testfälle pro Tag) angenommen. Der Aufwand zur Anpassung der Automatisierung wird analog zur Berechnung in Kapitel 6.4.1 nicht berücksichtigt. Die absolute Kostenreduktion in der Testdurchführung pro Iteration beträgt 10.770 €.

Break-Even-Point: Auf diesen Daten berechnet sich entsprechend dem in Kapitel 6.4.1 angewendeten Verfahren ein Break-Even-Point nach etwa 23 Iterationen (exakt: 22,03 Iterationen, s. Abbildung 6.10). Auch in Projekt BC2 wurde unter den gegebenen Bedingungen mit knapp zehn Iterationen keine Amortisation der Testautomatisierungskosten erreicht.

Wie Abbildung 6.10 zeigt, haben geringe Steigerungen des Automatisierungsgrades bei niedrigen Automatisierungsgraden einen größeren Effekt auf die benötigte Iterationsanzahl. Diese Betrachtung ist hier jedoch weniger von praktischem Interesse, da die berechneten Iterationsanzahlen unrealistisch hoch sind.

6.4.3. Vergleich mit verwandten Arbeiten

Schnelle Amortisation in Literaturprojekten: Für einen Vergleich der hier beschrieben Ergebnisse mit anderen Projekten aus der Automobilindustrie stehen kaum konkrete Zahlen zur Verfügung [203]. DSpace gibt für die Testautomatisierung bei Automobilsystemen durch HIL-Systeme (Hardware-in-the-Loop) einen üblichen Break-Even-Point nach drei bis sechs Monaten an [168]. Auch wenn die Dimension Zeit nicht direkt mit der ermittelten Anzahl Iterationen vergleichbar ist, scheint dennoch ein deutlicher Unterschied zwischen beiden Ergebnissen zu liegen, da erfahrungsgemäß in einem halben Jahr deutlich weniger als die kalkulierten 18 bzw. 23 Entwicklungsiterationen durchgeführt werden.

Werden eingebettete Systeme generell betrachtet, so gibt die *is Industrie Software GmbH* einen durchschnittlichen Break-Even-Point mit der vierten Iteration an [141]. Auch dies steht nicht im Einklang mit den hier beobachteten Ergebnissen, die eine deutlich spätere Amortisation darstellen.

Größere Einsparungen in der Testdurchführung in IT-Systemen: Beim Vergleich von eingebetteten Systemen mit IT-Systemen ist zu erkennen, dass IT-Systeme frühere

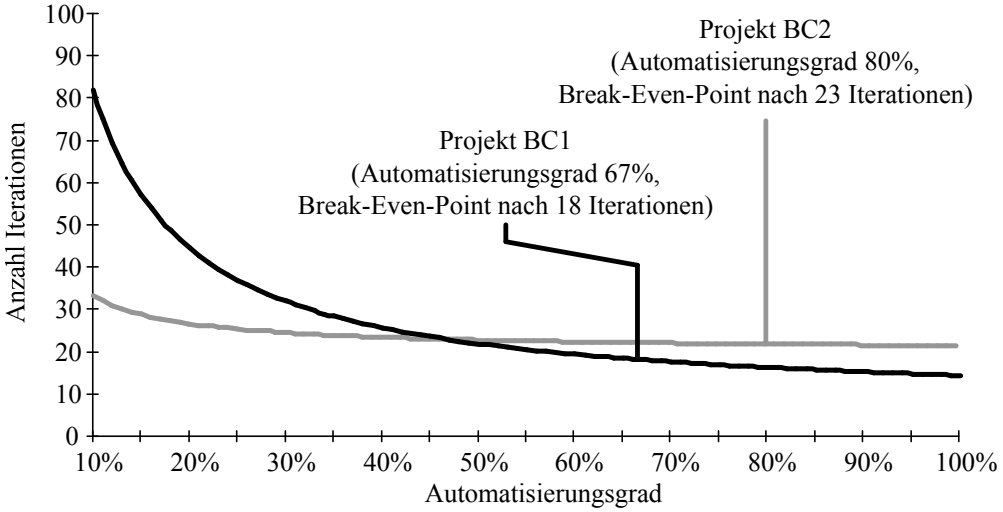


Abbildung 6.10.: Kurvenverläufe zum Break-Even-Point der Testautomatisierung für Projekte BC1 und BC2 nach Automatisierungsgrad und Iterationsanzahl

Amortisationen der Investitionskosten aufweisen:

- Elfgen [81] beobachtete im automatisierten GUI-Test von IT-Applikationen einen Break-Even-Point nach durchschnittlich 2,03 Iterationen.
- HP [130] beziffert den möglichen ROI (Return of Investment) der Testautomatisierung bei 5-jährigen Workflow-Projekten mit 526%, wobei der Break-Even-Point nach weniger als einem Jahr erreicht wird.
- Dustin [76] legte detaillierte Werte zur Kostenreduktion bei vollständiger Automatisierung offen (s. Tabelle 6.11). Die Automatisierung eines Testprozesses mit einer unbekannten Anzahl von Iterationen einer IT-Entwicklung ergab eine Kostenersparnis von 75% gegenüber dem manuellen Test. Lizenzkosten und ggf. spezielle Testumgebungen für die Automatisierungen werden nicht erwähnt. Es bleibt offen, ob hierfür Kosten angefallen sind.

Mögliche Ursachen für spätere Amortisation: Gründe für die spätere Amortisation in den untersuchten Automobilprojekten könnten sein:

- Nicht optimal umgesetzte Entwicklungs- und Testprozesse können die Wirksamkeit von Testautomatisierungen reduzieren, ja sogar kontraproduktiv werden lassen [112, 212].
- Eine ungenügende Integration von Testautomatisierungswerkzeugen kann zu einer Verteuerung des Tests anstelle der erhofften Einsparung führen [152, 153].
- Automatisierungswerkzeuge können sehr unterschiedliche Kosten verursachen, von kostenfrei bis sehr kostenintensiv, z. B. aufgrund hoher Lizenzkosten.
- Automatisierungsspezifische Testaufbauten (u. a. Testbench) können hohe Kosten verursachen.

Testschritte	Manuell (h)	Automatisiert (h)	Ersparnis
Testplanung	32	40	-25%
Entwicklung von Testverfahren	262	117	55%
Testfallausführung	466	23	95%
Fehleridentifikation	117	58	50%
Überwachung von Maßnahmen	117	23	80%
Berichterstellung	96	16	83%
Gesamtsdauer	1.090	277	75%

Tabelle 6.11.: Aufwand des manuellen und automatisierten Tests sowie Zeitersparnis für eine iterative IT-Entwicklung (nach Dustin) [76]

- Der Automatisierungsgrad bestimmt maßgeblich die Anzahl der Iterationen, die benötigt werden, um die Investitionskosten zu amortisieren (s. Abbildung 6.10). In den zitierten Literaturquellen wurde der Automatisierungsgrad jedoch nicht angegeben. Daher sind die Ergebnisse der Literatur nicht vollständig interpretierbar.
- Das Verhältnis zwischen Umwelt- und Systemaktivität bei der Ausführung der Testfälle könnte Einfluss auf das Optimierungspotential haben, da Testautomatisierungen den Umweltaufwand (z. B. Nutzereingaben, Rüstzeiten), nicht jedoch den Rechenaufwand des Systems reduzieren [215]. Systeme mit überwiegender Umweltaktivität besitzen daher ein größeres Einsparpotential.
- Die Kosten für die Wartung der Testfälle von Iteration zu Iteration können unter Umständen die Testkosten mehr als verdoppeln [76]. Dies kann die Testautomatisierung unwirtschaftlich machen.

Fazit: Die aufgestellte Liste mit maßgeblichen Einflussfaktoren auf die Kostengestaltung bei Testautomatisierungen verdeutlicht nochmals, dass eine Verallgemeinerung der Ergebnisse nur sehr begrenzt möglich ist.

Die Aufwandsreduktion für die automatisierte Testdurchführung wurde in den untersuchten Projekten mit 93% (BC1) und 94% (BC2) berechnet. Dieser Wert stimmt mit den Untersuchungen von Hörcher [132] mit einer 90%-igen und Dustin [76] mit einer 95%-igen Reduktion im Test eingebetteter Systeme überein. Dagegen beobachtete Gühmann [116] nur eine Reduktion um 50%. Die Systeme wurden nicht ausreichend detailliert beschrieben, um Gründe für den Unterschied bei Gühmann zu erkennen.

6.4.4. Fehleridentifikationsrate

Die Testautomatisierung eröffnet die Möglichkeit, eine höhere Anzahl Testfälle in gleicher Entwicklungszeit durchführen zu können. Dies führt jedoch nicht zwingend zu einem entsprechend höheren Umfang an identifiziertem Fehlverhalten [215].

Fehleridentifikationsrate in dem Projekt BC1: Im Projekt BC1 identifizierten etwa halb so viele manuell ausgeführte Testfälle etwa dreimal so oft Fehlverhalten wie die automatisiert ausgeführten Testfälle (s. Tabelle 6.12). 22% der manuell und nur 4% der automatisiert ausgeführten Testfälle identifizierten Fehlverhalten. Somit ist die Effektivität der automatisierten Testfälle für Projekt BC1 deutlich geringer. Zurückzuführen ist dies auf das Vorgehen bei der Testfallgenerierung. Die automatisiert ausgeführten Testfälle wurden nicht testverfahrenoptimiert erstellt (z. B. verhaltensmodellbasiert). Es wurden z. B. Testfallreihen generiert, die auf den vollständigen Test der Parameterkombinationen zielen.

Dustin [76] hingegen berichtet für automatisiert durchgeführte Testfälle eine Fehleridentifikationsrate von 40% (700 Fehlermeldungen bei 1.750 Testfällen). Damit liegt die Fehleridentifikationsrate sogar höher als die der manuellen Durchführung in Projekt BC1.

Geringe Effektivität: Wie bereits erwähnt, wurde im Projekt BC1 die Testautomatisierung verwendet, um Testfälle auszuführen, die bestimmte Parameter vollständig testen. Auf Testfallreduktionsverfahren wurde also verzichtet. Das führte dazu, dass vermehrt Testfälle mit geringerer Fehleridentifikationswahrscheinlichkeit automatisiert ausgeführt wurden [56, 57]. Die manuell erstellten und nicht vollständig automatisierten Testfälle entstanden z. T. unsystematisch, aber auch anhand von Lessons Learned-Testfällen [156, 241] und Negativtestfällen [241] sowie mit verhaltensmodellbasierten Testfallgenerierungsmethoden. Insbesondere durch Lessons Learned-Testfälle werden naheliegende Fehlerursachen fokussiert, was bei einer begrenzten Testfallanzahl häufig zu einer hohen Fehleridentifikationsrate führt.

Projekt BC1	Durchführung Testfall	
	Manuell(1)	Automatisiert(2)
Ausgeführte Testfallanzahl	670	1.369
Anzahl Fehlermeldungen	150	54
Fehleridentifikationsrate	22%	4%

(1) Unsystematisch hergeleitete Testfälle, Lessons Learned-Testfälle, Negativtestfälle, verhaltensmodellbasierte Testfälle

(2) Hauptsächlich vollständige Parameterüberdeckungen, kleiner Anteil an unsystematisch hergeleiteten Testfällen

Tabelle 6.12.: Fehleridentifikationsrate der in Projekt BC1 pro Iteration ausgeführten Testfälle

Fazit: Die Fehlersensitivität von Testfällen ist per se unabhängig von der Automatisierung, da sie nur ein der Testfallgenerierung nachgelagerter Schritt ist. Dennoch können Effektivitätsanalysen die Sinnhaftigkeit der automatisierten Testfälle bewerten, da Automatisierungen dazu verleiten können, redundante oder wenig fehlersensitive Testfälle auszuführen, was zu vermeidbaren Mehrkosten führen kann.

Im Vergleich des Projektes BC1 mit der Literatur zeigte sich, dass automatisierte Testfälle offensichtlich situations- und projektbedingt sehr unterschiedlich fehlersensitiv sind. Am Beispiel des Projekts BC1 kann vermutet werden, dass die Testautomatisierung dazu verleitet, auch Testfälle mit geringer Fehleridentifikationswahrscheinlichkeit auszuführen. Testfallqualität wird also durch Testfallquantität ausgeglichen. Dieses Beispiel legt offen, dass der Schlüssel zu guten Fehleridentifikationsraten in einer effizienten Testfallgenerierung zu suchen ist. Eine Optimierung der Fehleridentifikationsrate kann – wie am Beispiel des GATE-Verfahrens in Kapitel 6.3 gezeigt – erwartet werden, wenn die zu automatisierenden Testfälle anhand von optimierten (fehlersensitiven) Testverfahren hergeleitet werden. Im Ausblick (s. Kapitel 9.2) wird ein Ansatz zur Bestimmung der Qualität von Testfällen skizziert.

6.4.5. Bewertung

Vergleich der untersuchten Projekte: Die Berechnung der Break-Even-Points für die Projekte BC1 und BC2 war unter Einbeziehung einiger Annahmen möglich. Divergierende Projekteigenschaften (s. Tabelle 6.13) führen in den Projekten BC1 und BC2 zu unterschiedlichen Abhängigkeiten zwischen der Anzahl Iterationen und dem Automatisierungsgrad und zu deutlich unterschiedlichen Kurvenverläufen für den Break-Even-Point (s. Abbildung 6.10). Hoher Automatisierungsgrad, ausschließlich automatisierte Testausführung und niedrige Lizenzkosten bevorzugen das Projekt BC2 bezüglich einer schnellen Amortisation. Jedoch weist BC2 höhere Testfallautomatisierungskosten auf. Diese höheren Investitionskosten werden in jeder Iteration durch Einsparungen in der Testdurchführung stückweise amortisiert. Die Testdurchführungskosten eines manuellen Testfalls sind in Projekt BC2 nur etwa halb so hoch wie in Projekt BC1. Ein Grund für die verkürzte Durchführungszeit im Projekt BC2 konnte nicht ermittelt werden. Die relative Kostenersparnis ist in beiden Projekten mit 93% bzw. 94% als gleich anzusehen. Die absolute Kostenersparnis pro Iteration fällt durch die geringeren Durchführungskosten in Projekt BC2 trotz ausschließlich automatisierter Testausführung geringer aus (10.770€ vs. 18.810€). Insgesamt ergibt sich aus den unterschiedlichen Einflussfaktoren eine steilere Amortisationskurve für Projekt BC1 (s. Abbildung 6.10). Der Break-Even-Point wird dadurch in Projekt BC2 trotz des hohen Automatisierungsgrades erst später erreicht.

Validität des Algorithmus durch Ergebnisvergleich: Die Validität des Berechnungsalgorithmus kann anhand der Berechnungen des Testmanagers des Projekts BC2 exemplarisch überprüft werden. Diese weisen aus, dass 23 Iterationen zum Erreichen des Break-Even -Points notwendig seien. Diese vom Testmanager durchgeführte Berechnung bestätigt das Ergebnis des angewendeten Algorithmus (22,03 Iterationen bis zum Break-Even-Point). In Projekt BC1 wurde von der Projektleitung keine Bestimmung des Break-Even-Points durchgeführt.

Fazit: Das Verfahren zur Berechnung des Break-Even-Point lieferte mit der Rechnung eines Testmanagers vergleichbare Ergebnisse und kann daher als an einem Projekt validiert gelten. Die hier berechneten Ergebnisse hängen sehr von den projektspezifischen Einflussfaktoren wie Testfallanzahl, Automatisierungsgrad und

Kostenwirksame Eigenschaften und Break-Even	Projekt BC1	Projekt BC2
Testfallanzahl	2.537	2.021
Automatisierungsgrad	67%	80%
Anteil automatisiert ausgeführter Testfälle	80%	100%
Automatisierte Testfälle pro Tag und Tester	9	5
Dauer Testdurchführung manuell	10,4 min	5,0 min
Dauer Testdurchführung automatisiert	42 s	18 s
Lizenzkosten	211.200 €	17.400 €
Absolute Kostenreduktion in der Testdurchführung pro Iteration	18.810 €	10.770 €
Break-Even-Point (Iterationen)	18	23

Tabelle 6.13.: Kostenwirksame Eigenschaften der Projekte BC1 und BC2 bzgl. der Testautomatisierung

Testfallautomatisierungskosten ab. Eine Übertragung auf andere Projekte wird in seltenen Fällen möglich sein. Die beiden Break-Even-Point-Bestimmungen verdeutlichen eingehend, dass die Kosten-Nutzen-Bewertungen von Testautomatisierungen am jeweiligen Projekt erfolgen sollten.

Durch Testautomatisierung kann branchenübergreifend eine Kostenreduktion in der Testausführung von mehr als 90% erreicht werden. Dies führt entsprechend der Literatur üblicherweise zu einem Break-Even-Point nach wenigen Iterationen. In den betrachteten Automobilprojekten konnte jedoch entgegen der Erwartungen kein Break-Even erreicht werden. Dies konnte vorrangig auf projektbedingt sehr hohe Lizenzkosten (Projekt BC1) bzw. Testfallautomatisierungskosten (Projekt BC2) zurückgeführt werden. Auch dies belegt, dass Kosten-Nutzen-Rechnungen projektbedingt abgeschätzt werden sollten und Testautomatisierungen nicht unbedacht angewendet werden sollten.

In den untersuchten Projekten konnte beobachtet werden, dass die Testautomatisierung dazu verleitete, auf Verfahren zur Reduktion der Testfallanzahl zu verzichten und einen vollständigen Test anzustreben. Im Sinne der Kostenoptimierung sollten für die Herleitung der Testfälle trotz Testautomatisierung optimierte Verfahren verwendet werden.

6.5. Zusammenfassung

In diesem Kapitel wurden die zur Optimierung der Teststrategie ausgewählten Ansätze (Fehlerfrüherkennung, Testautomatisierung) sowie zwei vorbereitende Arbeitsvorgänge (Ermittlung von Einsparpotentialen, Gewichtung von Fehlerklassen durch Testziele, Break-Even-Point-Berechnung) an Projektdaten evaluiert:

Einsparpoteniale durch Fehlerfrüherkennung:

- Im Falle der frühestmöglichen Identifikation und Beseitigung aller im Test gefundenen Fehlerursachen, also in der Phase der Fehlerentstehung, ergäbe sich im Vergleich zur gängigen Praxis eine Kostenreduktion in den untersuchten Automobilprojekten von etwa 86% unter Verwendung des relativen Kostenwachstums der Literatur (s. Tabelle 6.1). Für die Literaturprojekte ergibt sich, wenn ebenfalls nur die Testphasen ab Modultest betrachtet werden, eine mittlere Kostenreduktion von 83%. Diese Werte stellen ein theoretisches Optimum dar, dessen Umsetzung jedoch nicht realistisch ist.
- Eine realistischere Abschätzung wird durch das fehlerklassenoptimierte Kosteneinsparpotential angegeben. Hier wird angenommen, dass die eingesetzten Testverfahren das fokussierte Fehlverhalten (z. B. Integrationsfehler) vollständig identifizieren und diese Testverfahren zum frühestmöglichen Zeitpunkt eingesetzt werden, also bei Erreichen der erforderlichen Systemvollständigkeit. Für das Body Controller-Projekt BC1, das einzige vorliegende Projekt, in dem Daten für eine Fehlerklassifizierung existieren, resultiert eine fehlerklassenoptimiertes Kosteneinsparpotential von 50% (s. Tabelle 6.2, Kapitel 6.1.2).

Gewichtung von Fehlerklassen durch Testziele:

- Exemplarisch wurde für das Projekt BC1 als Testziel definiert, dass Fehlerklassen mit hohem Kosteneinsparpotential durch Fehlerfrüherkennung und Fehlverhalten mit hoher Fehlerpriorität durch die Testverfahrensauswahl vorrangig adressiert werden sollen. Die Fehlerpriorität sollte doppelt so stark in die Gewichtung eingehen wie die Fehlerfrüherkennung. Die beiden Entscheidungsaspekte wurden mittels der testzielorientierten Fehlerklassengewichtung auf Fehlerklassen abgebildet und ergeben, dass Spezifikations- und Systemverhaltensfehler vorrangig zu reduzieren sind. Es ist zu beobachten, dass das Ziel, Fehlermeldungen mit hoher Priorität vorrangig zu entdecken, Systemverhaltensfehler stärker gewichtet und Spezifikationsfehler geringer. Das Ziel der Kosteneinsparung hat einen gegenteiligen Effekt auf die Gewichtung. Systemverhaltensfehler werden besonders effektiv durch verhaltensorientierte Testverfahren im Systemtest gefunden.

Fehlerfrüherkennung durch ein funktionsorientiertes und verhaltensmodellbasiertes Testverfahren:

- Die Wirksamkeit eines verhaltensorientierten Testverfahrens im Systemtest wurde anhand des funktionsorientierten und verhaltensmodellbasierten Testverfahrens GATE (*Generic and Automated Test Environment*) untersucht. Es wurden zwei parallel entwickelte Teilprojekte eines Body Controller-Projekts verglichen, von denen eines in einer Iteration mit dem GATE-Verfahren getestet wurde. Die Verfahrenseinführung erforderte keine längere Einarbeitungszeit und die Effektivität der Fehleridentifikation konnte sofort angehoben werden. Ohne GATE wurden 62% der zu diesem Zeitpunkt sich im System befindlichen Fehlerursachen identifiziert, durch Anwendung des GATE-Verfahrens stieg die Effektivität auf 75% (s. Tabelle 6.4). Die Anzahl der vom Kunden identifizierten Fehlerursachen wurde um etwa ein Drittel gesenkt.

- Das GATE-Verfahren reduzierte die Fehlerbeseitigungskosten allein durch Fehlerfrüherkennung um 13,5% (reduzierte Testfallerstellungskosten bleiben unberücksichtigt, s. Tabelle 6.5). Ein optimales verhaltensorientiertes Testverfahren, das alle Systemverhaltensfehler im Systemtest identifiziert, führt zu einer Kostenreduktion von 15,4% (fehlerklassenoptimiertes Kosteneinsparpotential, s. Tabelle 6.6). Folglich wurden 88% des im Systemtest vorhandenen Kosteneinsparpotentials durch Anwendung eines Testverfahrens, das Systemverhaltensfehler identifiziert, genutzt.
- Insgesamt ergaben sich eine kostenreduzierte Fehlerbeseitigung und eine effizientere Testfallerstellung.

Optimierung durch Testautomatisierung:

- Von Testautomatisierungen werden i. Allg. Kostenreduktionen erhofft. Das vorgestellte Verfahren zur Abschätzung des Break-Even-Points wurde an einem Projekt validiert und zeigte, dass in beiden betrachteten Body Controller-Projekten BC1 und BC2 der Break-Even-Point nicht erreicht wurde. Hauptgrund hierfür sind hohe Lizenz- bzw. Automatisierungskosten. Ein Break-Even-Point hätte sich erst bei hohen Iterationsanzahlen ergeben. Vor allem in IT-Systemen werden laut Literatur weitaus frühere Break-Even-Points beobachtet.
- Das Ergebnis ist als projektspezifisch einzustufen und belegt die Notwendigkeiten projektspezifische Kosten-Nutzen-Betrachtungen vorzunehmen.

7. Simulationsgestützte Teststrategieoptimierung

Aufbauend auf den bisher in Kapitel 5 und 6 gewonnenen Erkenntnissen wird in Kapitel 7.1 der in Kapitel 4.2 dargestellte Optimierungsprozess exemplarisch an einem Entwicklungsprojekt validiert, indem schrittweise eine konkrete, optimierte Teststrategie entworfen wird. Anschließend werden die Kosteneinsparungen derselben Optimierungen auf Basis von publizierten Fehlerströmen und Kostenverläufen abgeschätzt und die Ergebnisse erörtert (Kapitel 7.2). Die Verallgemeinerbarkeit der Strategieoptimierungen wird diskutiert (Kapitel 7.3) und die Ergebnisse zusammengefasst (Kapitel 7.4).

7.1. Erprobung des Optimierungsprozesses an Projektdaten

Vorgehensweise: Die ausgewählten Ansatzpunkte für Optimierungen von Teststrategien (Fehlerfrüherkennung mit Einsatz von effizienten Testverfahren, Identifikation von isolierten Kostenschwerpunkten, Testautomatisierung) werden in den nachfolgenden Kapiteln sukzessive an einem Referenzprojekt simuliert und die resultierenden Qualitätskosten mit den Ist-Kosten des Projekts verglichen und diskutiert. Die Quantifizierungen der Qualitätskosten erfolgen – nach Erfordernis retrospektiv und prospektiv – mit dem Qualitätskostenmodell aus Kapitel 4.3. Hiermit wird sogleich die Praxistauglichkeit des Modells überprüft. Zur Demonstration der Funktionsweise des Optimierungsprozesses werden alle in Kapitel 4.2 genannten Schritte des Optimierungsprozesses und somit alle in Kapitel 4 durchgeführten Entwicklungen zur Testoptimierung referiert.

Referenzprojekt: Das Projekt BC1 bietet, im Vergleich zu den übrigen Projekten, die besten Voraussetzungen für die Umsetzung, da hier Projektbudgetpläne sowie Fehlermeldungen mit Fehlerbeschreibungen und Aufwandsbestimmungen vorliegen, die für die Qualitätskostenberechnung benötigt werden. Zudem kann für die intendierten Quantifizierungen auf zahlreiche, bereits in dieser Arbeit ermittelten Untersuchungsergebnisse aufgebaut werden.

Es sei nochmals erwähnt, dass nicht alle Attribute für jede Fehlermeldung eingetragen wurden und diese nur von Systemtestphase (inkl. erneutem Modultest) bis Akzeptanztest vorliegen (s. Kapitel 5.1.2). Aus diesem Grund wird die Fehlerbeseitigung nur für die drei Phasen Systemtest, Fahrzeugtest und Akzeptanztest betrachtet, wobei der erneut im Systemtest durchgeführte Modultest der Modultestphase zugeordnet wird und sich somit eine vierte Phase ergibt.

7.1.1. Quantifizierung der testinduzierten Ist-Qualitätskosten

Der erste Schritt des Optimierungsprozesses besteht in der Bestimmung der testinduzierten Ist-Qualitätskosten für das gewählte Referenzprojekt.

Berechnungsgrundlage: Die retrospektive Berechnung der testinduzierten Ist-Qualitätskosten basiert auf den während der Entwicklung erfassten Ist-Projektdaten. Fehlende Ist-Daten werden in der Berechnung durch prospektiv erfasste Planungsdaten oder durch Erfahrungswerte abgeschätzt.

Die Projektbudgetpläne gehen von einem durchgehenden Einsatz des herkömmlichen, unsystematischen Tests aus, da vor Projektbeginn nicht geplant war, das systematische GATE-Verfahren einzusetzen. Die Kostenreduzierung durch Einsatz des GATE-Verfahrens in einer von sechs Iterationen für vier von zwölf Body Controller-Funktionen wird mangels Ist-Daten in der Ist-Kostenrechnung nicht berücksichtigt. Es ergeben sich also die Ist-Kosten ohne der Anwendung des GATE-Verfahrens.

Kosten für die Elemente der Hauptkostengruppen *Testaufbau, Testvorbereitung* sowie *Testdurchführung* liegen im Budgetplan aggregiert über die Testphasen vor. Die Kosten für den ursprünglichen Modultest sind in den Kostenelementen der Projektbudgetpläne und somit auch in der Aufstellung enthalten. Die *Hersteller-* und *Kundeninduzierten Fehlerbeseitigungskosten* stehen für System-, Fahrzeug- und Akzeptanztest sowie den wiederholten Modultest zur Verfügung. Für die Hauptkostengruppe *Testmanagement* liegen keine Daten vor. Aufgrund des geringen Zusammenhangs zur Optimierung von Teststrategien werden die Kosten nicht berücksichtigt. Die Hauptkostengruppen *Fehlervermeidung* und *Fehlerfolgekosten* fließen konzeptgemäß nicht in die Rechnung ein, da sie nicht im Fokus dieser Arbeit liegen.

Die wesentlichen Abschätzungen zu den Kostenelementen sind im Anhang H.1 beschrieben. Wegen der Abschätzungen werden die testinduzierten Qualitätskosten nicht mittels des iterationsorientierten, exakten Quantifizierungsalgorithmus sondern mittels der vereinfachten Berechnungsalgorithmen für retrospektiv geschätzte Qualitätskosten ermittelt (s. Kapitel 4.3.3).

Interpretation der Ergebnisse: Die abgeschätzten testinduzierten Ist-Qualitätskosten (s. Tabelle 7.1) wurden auf 1.000 Währungseinheiten normiert. Daraus ergibt sich eine vereinfachte Lesart des Promilleanteils der Kostenelemente an den testinduzierten Ist-Qualitätskosten.

	Ist-Kosten	GATE-Verfahren	GATE-Verfahren, Lizenzkosten-optimierung	GATE-Verfahren, Lizenzkosten-optimierung, Automatisierungen
Testinduzierte Qualitätskosten	**1000**	**958**	**841**	**686**
Herstellerinduzierte Qualitätskosten	**909**	**899**	**782**	**627**
Testmanagement	**n. b.**	**n. b.**	**n. b.**	**n. b.**
Testaufbau	**285**	**321**	**204**	**202**
Projektspezifische Ausführungsumgebung	138	138	138	124
Prüfstand	40	40	40	40
Prüfling	82	82	82	82
Treiber und Platzhalter	16	16	16	2
Werkzeuge	147	183	66	78
Testmanagement	n. b.	n. b.	n. b.	n. b.
Testvorbereitung	65	101	36	48
Testdurchführung	60	60	8	8
Fehlerbeseitigung	22	22	22	22
Testvorbereitung	**263**	**203**	**203**	**127**
Modellbasierter Test	0	61	61	81
Modellerstellung	0	61	61	81
Testfallgenerierung	137	16	16	11
Manuelle Testfallerstellung	137	16	16	11
Automatisierte Testfallerstellung	–	–	–	–
Testfallanpassung	–	–	–	–
Testautomatisierung	126	126	126	35
Vorbereitung zur automatisierten Ausführung	126	126	126	35
Anpassung der Automatisierung	0	0	0	0
Testdurchführung	**180**	**180**	**180**	**103**
Testfallausführung und Fehleridentifikation	180	180	180	103
Manuell	113	113	113	22
Automatisiert	67	67	67	81
Herstellerinduzierte Fehlerbeseitigung	**181**	**195**	**195**	**195**
Fehlermanagement	30	30	30	30
Datenpflege	30	30	30	30
Fehlerbearbeitung	121	132	132	132
Fehleranalyse	42	46	46	46
Fehlerkorrektur	79	86	86	86
Korrekturtest	30	33	33	33
Verifikation der Fehlerbeseitigung und Regressionstest	30	33	33	33
Kundeninduzierte Qualitätskosten	**91**	**59**	**59**	**59**
Kundeninduzierte Fehlerbeseitigung	**91**	**59**	**59**	**59**
Fehlermanagement	1	0	0	0
Datenpflege	1	0	0	0
Fehlerbearbeitung	72	47	47	47
Fehleranalyse	25	16	16	16
Fehlerkorrektur	47	31	31	31
Korrekturtest	18	12	12	12
Verifikation der Fehlerbeseitigung und Regressionstest	18	12	12	12

n. b. Nicht berücksichtigt. Kosten hierfür sind nicht in anderen Kostenelementen enthalten.
– Keine explizite Kostenaufschlüsselung. Kosten sind in anderen Kostenelementen enthalten.

Tabelle 7.1.: Testinduzierte Qualitätskosten des Body Controller-Projekts BC1 in verschiedenen, optimierenden Teststrategien

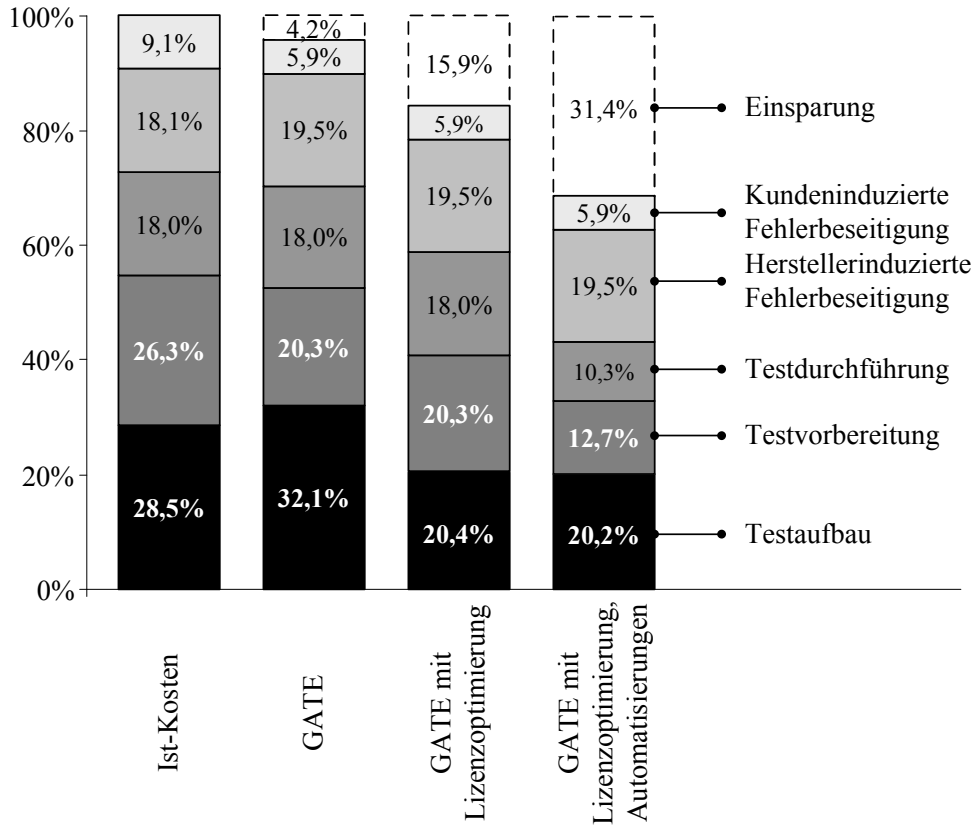


Abbildung 7.1.: Verteilung der Testkosten auf die Kostengruppen (Projekt BC1) in verschiedenen, optimierenden Teststrategien

Die Ist-Qualitätskosten in Projekt BC1 (s. Tabelle 7.1) verteilen sich zu 90,9% auf die *Herstellerinduzierten Qualitätskosten* und zu 9,1% auf die *Kundeninduzierten Qualitätskosten*. Das Projekt BC1 befindet sich noch vor dem Start der Serienproduktion (SOP), daher sind Feldfehler nach SOP nicht vorhanden. Insofern liegt eine untypische Projektsituation vor, da die *Kundeninduzierten Qualitätskosten* noch unterrepräsentiert sind.

Die Qualitätskosten entfallen etwa wie folgt auf die Hauptkostengruppen: 28,5% auf *Testaufbau*, 26,3% auf *Testvorbereitung,* 18,0% auf *Testdurchführung* sowie 18,1% auf *Herstellerinduzierte* und 9,1% auf *Kundeninduzierte Fehlerbeseitigung* (s. Abbildung 7.1). Es wird bereits deutlich, dass den *Hersteller-* und *Kundeninduzierten Fehlerbeseitigungskosten* (inkl. Fehlermanagement) in Summe eine hohe Bedeutung zuzuordnen ist. Sie weisen zusammen bereits ohne die kostenintensive Beseitigung der Feldfehler mehr als ein Viertel der testinduzierten Qualitätskosten auf (18,1% + 9,1% = 27,2%).

Die Kostengruppen mit nennenswerten Qualitätskostenanteilen sind breit gestreut: Testfallausführung (manuell und automatisiert zusammen) 18,0%, *Werkzeugkosten* 14,7%, *Projektspezifische Ausführungsumgebung* 13,8%, *Testfallgenerierungskosten* 13,7%, *Testautomatisierungen* 12,6% und *Fehlerbearbeitung* 12,1%. Es ist zu bedenken, dass jeder Einsatz eines Testverfahrens mehrere dieser Gruppen beeinflusst.

Die feingranulare Kostenstruktur des Qualitätskostenmodell macht diese Kostenbildung nachvollziehbar und lässt z. B. erkennen, dass die Lizenzkosten für Testautomatisierungswerkzeuge (*Testvorbereitung* und *Testdurchführung* in der Kostengruppe *Werkzeuge*) aufgrund interner Kostenkalkulationen mit 12,5% (=6,5%+6,0%) hohe Beträge aufweisen.

Quantitativer Vergleich mit der Literatur, Verallgemeinerbarkeit: Eine detaillierte und umfassende Kostenrechnung für den Testprozess, wie sie hier im Qualitätskostenmodell durchgeführt wurde, ist in der Literatur nicht beschrieben. Die Analyse der detaillierten Kostenverteilung lässt Kostenschwerpunkte erkennen, an denen Modifikationen der Teststrategien ansetzen können.

Da die Ergebnisse nach Kostenkategorien differenziert und auf verschiedenen Ebenen aggregiert vorliegen, können Teilaspekte mit Literaturwerten verglichen werden:

- Wallmüller [258] gibt das Kostenverhältnis von Testvorbereitung zu Testdurchführung mit 67% zu 33%, Biffl [24] mit 50% zu 50% an. Für Projekt BC1 berechnet sich auf Basis des Qualitätskostenmodells ein Verhältnis von 59,4% (=26,3%/(26,3%+18,0%)) zu 40,6% (=100%-59,4%). Somit liegen die Werte aus Projekt BC1 mittig zwischen beiden Literaturwerten und erscheinen daher plausibel. Abweichungen zwischen den einzelnen Werten können durch unterschiedliche Entwicklungsdetails, wie z. B. Testautomatisierungsgrade oder durch Wiederverwendung von Testfällen aufgrund iterativer Entwicklungen, entstehen.
- Schweiggert [233] beziffert das Kostenverhältnis von Testfallvorbereitung mit Testdurchführung zu Fehlerbeseitigung (hersteller- und kundeninduziert) mit 23% zu 77%. Für Projekt BC1 errechnen sich die Anteile für Testfallvorbereitung und -durchführung zu 62,0% (=(26,3%+18,0%)/(26,3%+18,0%+18,1%+9,1%)) für Fehlerbeseitigung demnach zu 38,0% (=100%-62,0%). Die in Projekt BC1 beobachteten Werte weichen somit von denen von Schweiggert ab. Gründe hierfür sind sicherlich, dass ausschließlich die Fehlerbeseitigung im erneuten Modultest, System-, Fahrzeug- und Akzeptanztest betrachtet wurde, und die sehr ungünstige Lizenzkostensituation.

Zusammenfassendes Fazit ist, dass die Kostenverteilungen im Vergleich von Projekten offensichtlich deutlich projektspezifische Komponenten haben, somit die Ansatzpunkte für Optimierungen projektspezifisch überprüft und beurteilt werden sollten.

7.1.2. Einsparpotential durch Fehlerfrüherkennung

Aus den testinduzierten Ist-Qualitätskosten kann das Testziel hergeleitet werden, die kostenintensiven, kundeninduzierten Fehlerbeseitigungen, die fast 10% der Projektkosten ausmachen, zu reduzieren. Somit sieht der zweite Schritt des Optimierungsprozesses vor, zunächst die Sinnhaftigkeit einer Strategieoptimierung durch Fehlerfrüherkennung zu überprüfen, indem für das Referenzprojekt das Kosteneinsparpotential auf Basis einer fehlerklassenorientierten Fehlerfrüherkennung (s. Kapitel 4.5) bestimmt wird. Die Ergebnisse für das Projekt BC1 können aus Kapitel 6.1 übernommen werden. Datenbedingt wurde das Kosteneinsparpotential auf der Basis des relativen Kostenwachstums der Literatur durchgeführt. Somit ist das Ergebnis zwar quantitativ nicht belastbar, jedoch ist aufgrund des hohen Wertes (fehlerklassenoptimiertes Kosteneinsparpotential 50%, s. Tabelle 6.2 in Kapitel 6.1.2) davon auszugehen, dass ein Kosteneinsparpotential vorhanden ist, das genutzt werden sollte.

7.1.3. Kostenreduktion durch Früherkennung

Der dritte Schritt des Optimierungsprozesses besteht in der Auswahl von effizienten Methoden, die die Fehlerklassen, die ein Früherkennungspotential aufweisen, adressieren. Für diesen Arbeitsschritt wurde die fehlerklassenorientierte Testverfahrensauswahl (s. Kapitel 4.4) entwickelt. Für die testfokusorientierte Fehlerklassifikation (s. Kapitel 4.4.1) kann für Projekt BC1auf die Häufigkeitsverteilung in Kapitel 6.2 zurückgegriffen werden. Die Dominanz der Systemverhaltensfehler indiziert gemäß Tabelle 4.2 in Kapitel 4.4 eine Optimierung des Testprozesses über Testverfahren, die auf diese Fehlerklasse zielen, z. B. die Anwendung von funktionsorientierten oder verhaltensmodellbasierten Tests. Aufgrund des Effizienznachweises in Kapitel 6.3 bietet sich der Einsatz des GATE-Verfahrens an, das zudem mehrere Kostengruppen reduziert, nämlich nicht alleinig die Kosten für die Fehlerbeseitigung durch Fehlerfrüherkennung sondern auch die Testfallerstellungskosten.

Optimierungsvariante: Anhand des Kostenmodells für testinduzierte Qualitätskosten soll die mögliche Kostenreduktion bei einer durchgängigen Anwendung des GATE-Verfahrens im Systemtest in allen Iterationen des Projekts BC1 und für alle BCFs abgeschätzt werden.

Berechnungsgrundlage: Das Einsparpotential durch den Einsatz des GATE-Verfahrens wird aufgrund der Beobachtungen in Kapitel 6.3.2 wie folgt abgeschätzt:

- Der Einsatz des GATE-Verfahrens bewirkte eine effektivere Fehleridentifikation. Dies führte zu einer verringerten *Anzahl an kundenidentifiziertem Fehlverhalten* (s. Abbildung 6.4, Kapitel 6.3.2). Die Anzahl der vom Kunden gemeldeten Fälle von Fehlverhalten konnte auf 65% (=17/26) des Vergleichwertes reduziert werden. Für die Berechnung wird daher eine Reduktion der vom Kunden identifizierten Fälle von Fehlverhalten um 35% angesetzt.
- Die *Testfallgenerierung* mittels verhaltensbasierter Modelle reduzierte den Testfallgenerierungsaufwand (inkl. *Modellerstellung*) in den untersuchten Projekten

beträchtlich (s. Kapitel 6.3.2). Hier wird eine Reduzierung des Testfallgenerierungsaufwands von 50% angesetzt. Die Kosten der *Modellerstellung* reduzieren sich in der Entwicklungspraxis mitunter durch das Wiederverwenden von Verhaltensmodellen. In den untersuchten Projekten wurden z. B. die Verhaltensmodelle des Projekts BC1 für Projekt BC4 wieder verwendet, dies wird hier jedoch nicht berücksichtigt. Aufgrund der Verwendung von Modellierungswerkzeugen sind in der *Testvorbereitung* zudem *Werkzeugkosten* für die Verwendung von Modellierungswerkzeugen zu berücksichtigen.

Das GATE-Verfahren wirkt sich somit auf die Kostenelemente *Modellerstellung*, *Testfallgenerierung, Werkzeugkosten* sowie die *Hersteller-* und *Kundeninduzierten Fehlerbeseitigungskosten* aus. Diese Kostenelemente gehen gegenüber denen in Kapitel 7.1.1 modifiziert in die Berechnung ein. Die Kostenberechnungen und deren Verteilung auf die Kostenelemente sind im Anhang H.2 beschrieben.

Interpretation der Ergebnisse: Werden die bisherigen, unsystematischen Tests im Systemtest durchgängig durch ein systematisches Testverfahren (hier durch das funktionsorientierte und verhaltensmodellbasierte GATE-Verfahren) ersetzt, können die *Testinduzierten Qualitätskosten* um 4,2% [7] (=100,0%-95,8%) gesenkt werden (s. Tabelle 7.1 und Abbildung 7.1). Eine relativ hohe Kostenreduktion durch frühe Fehleridentifikation verzeichnet die *Kundeninduzierte Fehlerbeseitigung* mit 3,2%-Punkten (=9,1%-5,9%). Charakteristisch für die Fehleridentifikation in frühen Testphasen sind erhöhte *Herstellerinduzierte Fehlerbeseitigungskosten* (hier: 1,4%=19,5%-18,1%). Es fand also eine Verlagerung der Kosten statt, wobei die Summe beider Kostengruppen eine Einsparung anzeigt (1,8%=3,2%-1,4%).

Auf der Ebene der Hauptkostengruppen betrachtet, erhöhten sich die Kosten für den *Testaufbau* durch Lizenzkosten um 3,6%-Punkte. Gleichzeitig werden die Kosten für die *Modellerstellung* in der *Testvorbereitung* aufgrund effektiverer Testfallgenerierung durch Modellierung um insgesamt 6,0%-Punkte reduziert.

Die mit der Einführung des GATE-Verfahrens erreichten Einsparungen hätten zu einer Kostenreduktion von 1,9% (=4,2%*0,46) aller in Projekt BC1 angefallenen Kosten geführt, da in Projekt BC1 die Testkosten der rechten V-Modell-Seite inkl. Fehlerbeseitigung 46% der Projektkosten ausmachen (s. Kapitel 5.2.1). Diese Kostenreduktion ist im Wesentlichen der Fehlerfrüherkennung, die geringere Fehlerbeseitigungskosten bewirkt, zuzurechnen. Zudem verringern sich durch den GATE-Einsatz gleichzeitig die Testvorbereitungskosten.

Die durchgeführte Strategieoptimierung wird als effizient angesehen.

[7] In Kapitel 6.3.2 wurde durch die Anwendung der GATE-Methodik eine Einsparung von 13,5% der Fehlerbeseitigungskosten berechnet. Die hier berechnete Kosteneinsparung von 4,2% bezieht sich auf die gesamten testinduzierten Qualitätskosten, von denen die Fehlerbeseitigung nur eine Kostengruppe ist.

7.1.4. Isolierung von Kostenschwerpunkten

Im vierten Schritt des Optimierungsprozesses wurden die Kostendetails der testinduzierten Ist-Qualitätskosten kritisch unter dem Aspekt inspiziert, ob die Kostenkategorien Schwerpunkte aufweisen, die auf die spezifische Situation des Projekts bzw. des Unternehmens zurückzuführen sein könnten.

In Projekt BC1 sind, wie bereit in Kapitel 6.4.1 festgestellt, die Lizenzkosten für Testautomatisierungswerkzeuge (Kostenelemente: *Werkzeuge Testvorbereitung* und *Werkzeuge Testdurchführung*) aufgrund firmeninterner Kalkulationen sehr hoch. Für Veränderungen sind hier individuelle Lösungen zu suchen. Ebenso spezifisch werden die Kostenauswirkungen sein, die dann durch das Qualitätskostenmodell überprüft werden können. Um den beachtlichen Einfluss, der von einer Kostenkomponente ausgehen kann, zu demonstrieren, wird im Folgenden angenommen, die Lizenzkosten der Testautomatisierungswerkzeuge seien analog zu denen aus Projekt BC2 (s. Kapitel 6.4.2).

Berechnungsgrundlage: Die Lizenzkosten aus Projekt BC1 werden in den Kostenelementen *Werkzeugkosten Testvorbereitung* und *Werkzeugkosten Testdurchführung* durch die von Projekt BC2 ersetzt. Die restlichen Kostenelemente bleiben unverändert.

Interpretation der Ergebnisse: Die Werkzeugkosten für die Testautomatisierung betragen 12,5% der testinduzierten Qualitätskosten. Mit analogen Lizenzgebühren zu BC2 hätten zusammen mit den Einsparungen durch das GATE-Verfahren 15,9% der Testkosten eingespart werden können (Tabelle 7.1 und Abbildung 7.1). Dieses Ergebnis verdeutlicht, dass Kosten für Werkzeuglandschaften einen hohen Kostenanteil an den Qualitätskosten darstellen können.

7.1.5. Kostenreduktion durch Automatisierungen

Im letzten Optimierungsschritt wird auf Basis einer Inspektion der Teststrategie des Referenzprojektes entschieden, an welchen Stellen weitere Testautomatisierungen vorgenommen werden können. Um die Qualitätskosten der Automatisierungen mit den übrigen testinduzierten Qualitätskosten differenziert ausweisen zu können, wird die Wirtschaftlichkeit der Automatisierungen nicht durch Break-Even-Point-Bestimmungen überprüft sondern durch Simulation der Kosten mittels des Kostenmodells für testinduzierte Qualitätskosten.

Optimierungsvariante und Berechnungsgrundlage: Ausgangsbasis für die Untersuchungen in diesem Kapitel ist Projekt BC1 nach den Optimierungen durch das GATE-Verfahren im Systemtest und durch die Werkzeugauswahl, wie in den Kapiteln 7.1.3 bzw. 7.1.4 beschrieben. BC1 ist bereits mit einem Automatisierungsgrad von 67% durchgeführt worden. Potential für zusätzliche Automatisierungen manueller Arbeitsschritte liegt in Projekt BC1 vorwiegend in den Teststrategien des Modultests. Folgende Automatisierungen werden integriert:

Testfallgenerierung mit automatisierter Ermittlung von Testeingabedaten, Treibern und Platzhaltern sowie automatisierter Testfallautomatisierung im Modultest: Für einen effektiveren Modultest des Projekts BC1 sollen in dieser Berechnung Testreihen erzeugt werden, die dem Quelltextüberdeckungskriterium der Zweigüberdeckung [176] genügen. Die Testfallgenerierung umfasst i. Allg. die Ermittlung der Testeingabewerte, bestehend aus Funktionsparametern, Rückgabewerten während der Ausführung aufgerufener Funktionen (für Platzhaltergenerierung notwendig) sowie Systemvariablen, die den Systemzustand beschreiben. Die manuelle Ermittlung dieser Daten für Testreihen ist ein zeitaufwändiger Arbeitsschritt [269]. Werden Herleitung der Testfälle, Testfallautomatisierung und automatisierte Testausführung gemeinsam betrachtet, entfallen 5% der Kosten auf die Testausführung und 95% auf die Testfallgenerierung mit Testfallautomatisierung [269].

Automatisierte Generierung von Platzhaltern im Modultest: Im Modultest sind häufig Platzhalter einzusetzen, um fehlende Funktionen und Module zu simulieren. Platzhalter werden häufig mit erheblichem Aufwand manuell erzeugt. Ein bei einem Fahrzeugteilezulieferer entwickelter Generator soll zusammen mit der automatisierten Ermittlung von Testeingabedaten verwendet werden.

Erhöhung des Testautomatisierungsgrades im Systemtest: In Kapitel 6.4 wurde nachgewiesen, dass sich Automatisierungskosten bei einem hohen Automatisierungsgrad schneller amortisieren. Dieser Effekt soll im Systemtest dargestellt werden, indem der Automatisierungsgrad erhöht wird.

Berechnungsgrundlage und Einzelergebnisse: Die Umsetzung der angestrebten Automatisierungsansätze wird im Anhang H.3 beschrieben. Zusammenfassend ergeben sich folgende Effekte:

Testfallgenerierung mit automatisierter Ermittlung von Testeingabedaten und automatisierte Testautomatisierung im Modultest: Mit einem bei einem Fahrzeugteilezulieferer entwickelten Testfallgenerator werden 87,5% der Testfälle im Modultest generiert (s. Anhang H.3). Die Kosten der Testfallgenerierung sinken durch die Automatisierung um 70% (s. Anhang H.3). Durch Ausgabe der Testfälle in einem Format, das eine automatisierte Ausführung erlaubt, können weiterhin 72% der Testautomatisierungskosten im Modultest eingespart werden (s. Anhang H.3). Diese Automatisierungen reduzieren den Aufwand für die Testfallautomatisierung derart, dass eine Einsparung bereits nach der ersten Iteration möglich ist. Bei einer zu 100% automatisierten Testfallausführung reduzieren sich die Kosten der Testausführung nach Untersuchungen in Kapitel 6.4.3 um 93%. Die zusätzlichen Werkzeugkosten durch den Einsatz der erforderlichen Testfallgeneratoren können in diesem Fall durch Verwendung von Werkzeugen, die in Beratungsleistungen inbegriffen waren, niedrig gehalten werden. Die resultierenden Kosten in Tabelle 7.1 ergeben sich nach Gleichungen (A.29) und (A.31) bzw. Gleichung (A.32) in Anhang A.3.

Entwicklungskosten für Treiber und Platzhalter im Modultest: Die Testeingabeermittlung kann nach gemittelten Erfahrungswerten zu 87,5% automatisiert werden (s. Anhang H.3). Damit verwoben ist auch die automatisierte Generierung von Treibern

und Platzhaltern, so dass diese ebenfalls für 87,5% der Testfälle automatisiert generierbar sind und ohne weiteren Aufwand zur Verfügung stehen. Die resultierenden Kosten in Tabelle 7.1 ergeben sich nach Gleichung (A.7), s. Anhang A.2.2.

Testautomatisierung im Systemtest: Der Automatisierungsgrad wird im Systemtest von den derzeit gegebenen 67% auf 90% angehoben. Dadurch steigen die Modellerstellungskosten im Systemtest um 34% an (s. Anhang H.3), die automatisierte Testausführung senkt aber die Kosten der Testausführung nach Kapitel 6.4.3 um 93%, was insgesamt zu einer Kostenreduktion führt. Die resultierenden Kosten in Tabelle 7.1 ergeben sich nach Gleichung (A.31), Anhang A.3.2.

Interpretation der Ergebnisse: Die genannten Abschätzungen zu den Automatisierungskosten verringern die bereits reduzierten Qualitätskosten nochmals um 15,5% (=84,1%-68,6%) (s. Tabelle 7.1). Die Berechnung (s. Anhang H.3) basiert zum Teil auf Erfahrungswerten aus einem Nicht-Automobilprojekt des gleichen Konzerns, weshalb ein von Kapitel 6.4 abweichendes Ergebnis ermittelt wurde. Daher ist die hier durchgeführte Berechnung zwar als eine mögliche Abschätzung von Kosten der Testautomatisierung anzusehen, aber nur eingeschränkt als quantitativ belastbare Kostenberechnung zu verstehen. Testautomatisierungen sind immer in Abhängigkeit von Projekteigenschaften zu sehen, wie bereits in Kapitel 6.4 aufgezeigt.

7.1.6. Bewertung

Der Optimierungsprozess für Teststrategien hat sich als praxisrelevanter Leitfaden erwiesen, um schrittweise und objektiviert zu einer projektspezifisch effektiv optimierten Teststrategie zu gelangen.

Die exemplarische Anwendung des systematischen GATE-Verfahrens in einer Entwicklungsiteration reduzierte die testinduzierten Qualitätskosten durch Reduzierung von kundeninduziertem Fehlverhalten. Durch das Kostenmodell war eine Simulation der bei einer durchgängigen Anwendung des GATE-Verfahrens entstehenden Kosten unter Berücksichtigung der in Projekt BC1 gegebenen Umstände strukturiert und nachvollziehbar möglich. Weitere Kostenreduktionen sind durch eine Reduktion der Feldfehler und durch Erreichen des Testziels in weniger Iterationen realisierbar. Diese Aspekte lassen sich prinzipiell durch das Qualitätskostenmodell, nicht jedoch bei vorliegender Datenlage quantifizieren.

In Projekt BC1 wies das Qualitätskostenmodell Werkzeuge für Testautomatisierungen als projektspezifische Kostenschwerpunkte aus. Werkzeugkosten sollten daher potentielle Ansatzpunkte für kritische Überprüfungen sein. Um den Einfluss zu demonstrieren wurde als theoretische Alternative der Werkzeugeinsatz zu den Lizenzkosten aus Projekt BC2 simuliert. Zusammen mit den Reduktionen durch das GATE-Verfahren und den simulierten Testautomatisierungen hätten rechnerisch 31,4% der Testkosten eingespart werden können (s. Abbildung 7.1). In Bezug zu den gesamten Projektkosten bedeutet dies eine Kostenreduktion um 14,4% (=31,4%*46%

[8]). Bei Vernachlässigung der Lizenzkostenoptimierung verbliebe immer noch eine Kostenreduktion von 19,7% der testinduzierten Qualitätskosten bzw. ca. 9% (=19,7%*46%) der Projektkosten. Somit wird die durchgeführte Strategieoptimierung als zufriedenstellend angesehen.

Das Qualitätskostenmodell hat sich in der Bewertung von Kosteneinsparpotentialen im Rahmen der Validierung am Projekt BC1 als ein hilfsreiches und gut anwendbares Werkzeug erwiesen. Die testinduzierten Qualitätskosten können transparent und strukturiert retrospektiv berechnet und prospektiv abgeschätzt werden. Die Kostenverteilungen sind differenziert einsehbar. Das additive Konzept der Berechnungsvorschriften ermöglicht flexible Anpassungen an die Detaillierung projektspezifischer Datenbasen. Jedoch müssen Plandaten sowie erfasste Projektdaten für sehr realitätsnahe Kostenabschätzungen eine ausreichende Detaillierung aufweisen, was sich in den untersuchten Projekten als nicht ausreichend gegeben herausstellte. Präzisere Kostenberechnungen hätten durchgeführt werden können, wenn die Projektbudgetpläne feingranularer und die Fehlerdaten vollständiger wären. Für eine verbesserte Projektplanung sollte eine Optimierung von Projektbudgetplänen angestrebt werden. Vorschläge zur Gewinnung präziser Fehlerdaten in Fehlerdatenbanken werden im Ausblick (s. Kapitel 9.1) skizziert.

7.2. Kostensimulationen nach Kennwerten aus der Literatur

Das Projekt BC1 stellt sich als Projekt mit vielen Besonderheiten dar. So flossen Fehlerbeseitigungskosten ausschließlich für identifiziertes Fehlverhalten im System-, Fahrzeug- und Akzeptanztest sowie für den im Systemtest nachgeholten Modultest ein. Es wurden hierbei das projektspezifische Fehleridentifikationsverhalten und das projektspezifische Kostenwachstum verwendet. In den folgenden Berechnungen soll von der vorliegenden Projektstruktur (Fehlerstromdiagramme, Kostenwachstum der Fehlerbeseitigung) abstrahiert werden, indem diese Werte durch Fehlerströme und Kostenverläufe der Literatur (nach der Zusammenstellung in Kapiteln 3.2 und 3.3) ersetzt werden unter Beibehaltung der übrigen projekteigenen Aufwandsdaten. Es werden vier Simulationsrechnungen analog zu den Betrachtungen in den Kapiteln 7.1.3 bis 7.1.5 durchgeführt:

- Es wird eine Ist-Kostenberechnung auf Basis des in Projekt BC1 durchgeführten unsystematischen Tests, wie in Kapitel 7.1.1 beschrieben, durchgeführt.
- Anschließend wird der unsystematische Test im Systemtest durch den durchgehenden Einsatz des GATE-Verfahrens, wie in Kapitel 7.1.3 beschrieben, ersetzt.
- In der dritten Simulation werden zusätzlich die Lizenzkosten, wie in Kapitel 7.1.4 beschrieben, reduziert.
- In der letzten Simulation werden zusätzlich die in Kapitel 7.1.5 beschriebenen Optimierungen durch Automatisierungen hinzugefügt.

[8] Anteil der Testkosten an den Projektkosten beträgt 46% (s. Kapitel 5.2.1).

Annahmen zu Projekteigenschaften sowie Berechnungsgrundlage: Da gemäß dem Fokus dieser Arbeit die testinduzierten Qualitätskosten optimiert werden sollen, werden für die Fehlerbeseitigung die Testphasen ab Modultest berücksichtigt, nämlich Modultest, Systemtest und Feldeinsatz. In den Fehlerströmen der Literatur stehen als *Kundeninduzierte Fehlerbeseitigungskosten* nur die kostenintensiven Fehlermeldungen des Feldeinsatzes zur Verfügung. Allerdings ist deren Anteil deutlich geringer als die kundeninduzierten Fehlermeldungen im Fahrzeug- und Akzeptanztest in den vorliegenden Projekten. Für die Fehlerströme werden die normierten Mediane aus Tabelle 3.2 (s. Kapitel 3.2) und für das relative Kostenwachstum die Faktoren (Mediane) aus Tabelle 3.3 (s. Kapitel 3.3) verwendet. Die übrigen Projektkosten werden den ermittelten Werten aus Projekt BC1 entnommen. Die beschriebene Änderung der Kennzahlen im Vergleich zur Ist-Kostenberechnung des Projekts BC1 in Kapitel 7.1.1 wirkt sich auf die Hauptkostengruppen der *Hersteller-* und *Kundeninduzierten Fehlerbeseitigung* aus. Die unter diesen Bedingungen ermittelten Ist-Qualitätskosten werden wiederum als Bezugsbasis angenommen und die Testoptimierungen auf die neu simulierten Ist-Kosten normiert.

Ergebnisse: Die in Abbildung 7.2 dargestellten Kosten zu den vorgenommenen Optimierungen lassen sich wie folgt interpretieren:

- Der geänderte Fehlerstrom sowie das geänderte Kostenwachstum bewirken, dass die Kosten der *Hersteller-* und *Kundeninduzierten Fehlerbeseitigung* beim unsystematischen Test zusammen 35,8% (=24,8%+11,0%) der Testkosten verursachen. Der Anteil der Fehlerbeseitigungskosten ist also im Vergleich zu Projekt BC1 um 8,6%-Punkte höher. Zu begründen ist dies vorrangig durch das progressivere Kostenwachstum in der Literatur.
- Die Testoptimierung mit dem GATE-Verfahren bewirkt basierend auf den Literaturdaten eine vergleichbare Kostenreduktion von 4,5% (Projekt BC1: 4,2%). Dieses Ergebnis wird im vorliegenden Fall durch gegenläufige Kostenentwicklungen (z. B. weniger, aber kostenintensivere Kundenfehler) beeinflusst. Absolut betrachtet, bedeutet der gleiche Prozentsatz dennoch eine höhere Einsparung, da die Projektkosten durch die kostenintensivere Fehlerbeseitigung angestiegen sind.

Die nachfolgenden Optimierungen werden nicht direkt von den geänderten Eingangsdaten beeinflusst.

- Die Lizenzoptimierungen senken die Kosten des Testaufbaus weiterhin deutlich.
- Die Automatisierung zusätzlicher Arbeitsschritte senkt wiederum vor allem die Kosten für Testvorbereitung und -durchführung.

Zusammenfassung: Verglichen mit den Simulationen in den vorangegangenen Kapiteln fallen die Kostenreduktionen geringer aus, da den Kostenelementen mit höherem Optimierungspotential geringere Kostenanteile zukommen und der Anteil der vergleichsweise wenig optimierten Fehlerbeseitigungskosten anstieg. Diese Berechnung verdeutlicht, dass relative Kosteneinsparungen bereits durch Änderung weniger Eingangsdaten stark abweichen können und daher nur schwer auf andere

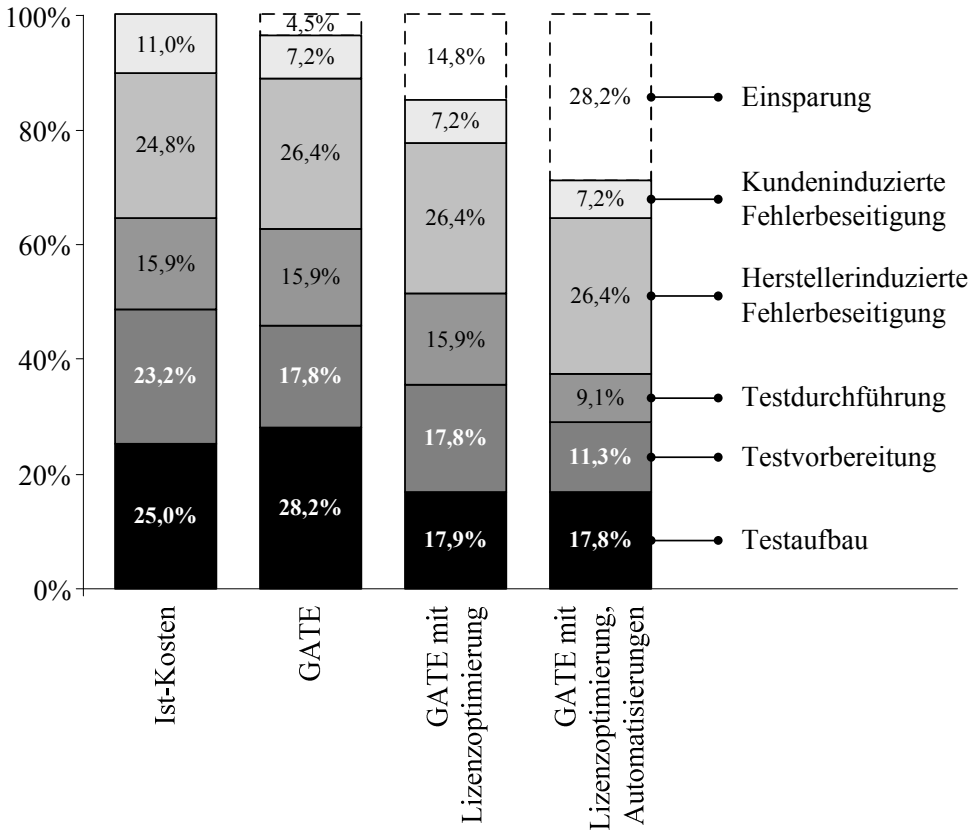


Abbildung 7.2.: Verteilung der Testkosten auf die Kostengruppen (Simulation auf Basis von Fehler- und Kostenverteilungen der Literatur)

Projektgegebenheiten übertragbar sind und Untersuchungen an den vorliegenden Projektstrukturen unabdingbar sind. In diesem Fall wird wiederum aufgezeigt, dass vollständige Projektdaten mit Fehlerdaten, die den Feldtest einschließen, für die Beurteilung der Kostenabhängigkeiten von großem Vorteil sind.

7.3. Verallgemeinerbarkeit der Strategieoptimierungen

Der Optimierungsprozess für Teststrategien zielt auf eine projektspezifische Optimierung ab. Daher ist die Verallgemeinerung der projektspezifisch ermittelten Optimierungen nicht Intention der Anwendung, vielmehr soll das Verfahren als solches projektübergreifend einsetzbar sein. Vor diesem Hintergrund werden die Optimierungsergebnisse hinsichtlich der Übertragung im Folgenden diskutiert.

Anhand von Simulationen lassen sich für das Projekt BC1 wirksame Kosteneinspareffekte durch das GATE-Verfahren und durch Testautomatisierung ermitteln.

Dieser Effekt ergibt sich ebenfalls, wenn für Kostenwachstum und Fehlerverteilungen durchschnittliche Werte aus Projektdaten aus der Literatur eingesetzt werden. So kann von qualitativ vergleichbaren Ergebnissen gesprochen werden, zumal die Literaturzusammenstellung auf recht unterschiedlichen Projekten beruht und sich dennoch gleichgerichtete Wirkungen zeigen. Aufgrund der durchgeführten Untersuchungen wird davon ausgegangen, dass das GATE-Verfahren projektunabhängig Verhaltensfehler im Systemtest effizient identifiziert. Testautomatisierungen und Lizenzkostenoptimierungen weisen jedoch eine engere Bindung an Projekt- oder Unternehmensgegebenheiten auf und sollten jeweils überprüft werden. Quantitative Übertragungen können für alle betrachteten Optimierungsansätze nur sehr eingeschränkt vorgenommen werden, da hier die Projektstrukturen maßgeblich das Ergebnis beeinflussen. Hinzu kommt, dass die erforderlichen Kostendetails zur Ermittlung der Qualitätskosten projektspezifisch unterschiedlich verfügbar sind und somit die Abschätzungen der Qualitätskosten nicht für alle Kostenbereiche gleichermaßen realitätsnah sind. Die detailgetreue Archivierung von Kostendaten ist zumindest in den untersuchten Projekten noch sehr verbesserungsbedürftig.

7.4. Zusammenfassung

Wirksame Ansätze zu Effizienzsteigerungen im Testprozess eingebetteter Systeme wurden simuliert. Hierbei wurden die in den vorangegangenen Kapiteln ermittelten Kosten und entwickelten Methoden eingesetzt.

Anwendung des Optimierungsprozesses auf Projektdaten: Zunächst wurde der in Arbeitsschritte untergliederte Optimierungsprozess gemäß den Ausführungen in Kapitel 4.2 für das Body Controller-Projekt BC1 ausgeführt, beginnend mit der Quantifizierung der testinduzierten Ist-Qualitätskosten. Große Einsparpotentiale für Fehlerfrüherkennung bestehen vor allem für Spezifikations- und Systemverhaltensfehler. Mit einer durchgängigen Anwendung des GATE-Verfahrens können laut Simulation 4,2% der testinduzierten Qualitätskosten eingespart werden (s. Tabelle 7.1).

Die Lizenzkosten für Automatisierungswerkzeuge belaufen sich auf 12,5% der testinduzierten Qualitätskosten und stellen einen Kostenschwerpunkt des Projekts dar, für den individuelle Lösungen zur Reduktion gesucht werden müssen. Diese quantitative Beobachtung ist stark projekt- bzw. unternehmensbedingt und nicht verallgemeinerbar.

Automatisierungen von manuellen und kostenintensiven Testaktivitäten des Modul- und Systemtests – der Systemtest hatte bereits einen Automatisierungsgrad von 67% – hätten weitere 15,5% der testinduzierten Qualitätskosten einsparen können. Kosten-Nutzen-Beziehungen von Testautomatisierungen weisen immer eine enge Bindung an Projekt- oder Unternehmensgegebenheiten auf und sollten ebenfalls jeweils überprüft werden. Die hier berechneten Einsparungen stellen u. a. aufgrund der hohen Fixkosten nur eingeschränkt belastbare Kostenaussagen dar. Diese Kostenaussagen geben lediglich eine qualitative Richtung vor.

Kostensimulationen nach Kennwerten aus der Literatur: Die Simulation der Optimierungsschritte auf Basis der Fehlerströme und relativen Kostenzuwächse der Literatur zeigen den maßgeblichen Einfluss dieser Kennwerte und somit die Grenzen der quantitativen Übertragbarkeit auf. Dies demonstriert, dass ein Referenzprojekt dem intendierten Entwicklungsprojekt möglichst ähnlich sein sollte.

Einsatz der entwickelten Methoden: Mit den in Kapitel 4 entwickelten Methoden wurde eine Optimierung der Teststrategie von Projekt BC1 durchgeführt. Die Methoden führten zu überprüfbaren, praxisrelevanten Ergebnissen. Aufgrund der feingranularen Struktur des Qualitätskostenmodells mit den zahlreichen Aggregationsebenen sowie den abschätzenden Algorithmen ließen sich die Kostenquantifizierungen problemlos durchführen. Die im Laufe der Quantifizierung gewonnenen Detailerkenntnisse zu den Kosten legen Kostenschwerpunkte offen und lassen erkennen, wo Modifikationen von Teststrategien anzusetzen sind. Die schrittweise Durchführung des Optimierungsprozesses hat mithilfe der unterstützenden Verfahren, speziell des Qualitätskostenmodells, exemplarisch eine effiziente Optimierung der Teststrategie des Referenzprojekts erbracht. Kostensimulationen wie diese sollen in der Projektplanung motivieren, kostenreduzierende Maßnahmen abzuwägen und die Wirksamkeit von Maßnahmen zu bewerten.

Datenarchivierung: In den Kostensimulationen musste auf aggregierte Kostenelemente sowie auf Referenzdaten aus anderen Projekten zurückgegriffen werden, da eine detailgetreue Archivierung von Kostendaten zumindest in den vorliegenden Projektdaten in weiten Bereichen fehlte.

Fazit: Die Strategieoptimierungen, die in dieser Arbeit durch die vorgegebene Abfolge von Optimierungsschritten und unterstützenden Verfahren identifiziert und demonstriert wurden, beruhen im Besonderen auf der Früherkennung von Systemverhaltensfehlern und Testautomatisierungen. Insgesamt können diese Maßnahmen als qualitätssteigernd angesehen werden und es kann eine Kostenreduktion, bezogen auf derzeitig praktizierte Entwicklungsprozesse, erwartet werden. Somit ist die Optimierung des Testprozesses mit den unterstützenden Verfahren an einem Entwicklungsprojekt validiert. Hier vorgelegte Kostenquantifizierungen mit relativen Kosteneinsparungen können dazu anregen, die vorgeschlagenen Strategieoptimierungen für geplante Projekte mit den entwickelten Hilfsmitteln für die eigene Entwicklungspraxis zu simulieren.

8. Aspekte zur Verallgemeinerbarkeit

Diese Arbeit analysiert die dynamische, testfallbasierte Qualitätssicherung in der Entwicklung von eingebetteten Systemen für automobile Body Controller und Infotainment-Systeme im Hinblick auf eine Kosten- und Effektivitätsoptimierung. Charakteristisches Merkmal der betrachteten Entwicklung ist die iterative V-Modell-Anwendung, wie in Kapitel 2.2.1 beschrieben. Im Fokus dieser Arbeit steht ein Ausschnitt aus der Entwicklung, und zwar der Testprozess als phasenorientiertes Vorgehensmodell, in dem Tester die Software in den Iterationen testen.

Zum Testprozess von eingebetteten Systemen im automobilen Umfeld wurden bisher wenig quantifizierende Untersuchungen publiziert [203]. Entwicklungsdaten werden von den Herstellern unter großer Zurückhaltung für Analysen freigegeben. Somit stellen die hier vorliegenden Daten, die erst nach Schaffung einer Vertrauensbasis von den Entwicklungsabteilungen zur Analyse überlassen wurden, trotz Datenmängel eine wertvolle Informationsquelle dar. Die vorliegenden Projektdaten wurden bezüglich Kostenverteilung und Kostenabhängigkeiten untersucht.

Ein Vorgehensschema für die projektspezifische Optimierung von Teststrategien wurde dargestellt und unterstützende Verfahren für die Umsetzung entwickelt. Ansatzpunke für die Optimierung sind Fehlerfrüherkennung, Testautomatisierung und projektspezifische Kostenschwerpunkte. Der beschriebene Optimierungsprozess wurde anhand der Daten eines Entwicklungsprojekts umgesetzt. Die Teststrategie wurde sukzessive optimiert und die testinduzierten Qualitätskosten simulativ quantifiziert.

Eine Diskussion des Gültigkeitsbereichs bzw. der Verallgemeinerungsfähigkeit der Untersuchungsergebnisse aus den Kapiteln 5 bis 6 und der entwickelten Methoden aus dem Kapitel 4 wurde jeweils am Ende dieser Kapitel unter Einbeziehen von Ergebnissen aus der Literatur angeschlossen. Kapitel 8.1 diskutiert kapitelübergreifend die Verallgemeinerbarkeit der Datenanalysen und der entwickelten Verfahren. In Kapitel 8.2 werden aus den gewonnenen Erkenntnissen allgemeingültige Empfehlungen für eine kostenoptimierte Qualitätssicherung im Testprozess eines phasenorientierten Vorgehensmodells abgeleitet.

8.1. Gültigkeit der Untersuchungen und Verfahren

Kostenverteilung in Projekt und Testphasen: Die Kostenverteilung in der Produktentwicklung für das hier untersuchte Projekt BC1 gleicht publizierten Ergebnissen zu Projekten unterschiedlicher Branchen in dem Punkt, dass die Kosten der Testphase deutlich vor den übrigen Phasen rangieren (Kapitel

5.2.1, [3, 28, 34, 145, 146, 181, 213, 227, 239, 253, 254, 262]). Dieser Gleichklang mit den zahlreichen Ergebnissen der Literatur liefert zwar keinen Beweis für eine über das Projekt hinausgehende Gültigkeit, motiviert aber die Annahme, dass diese Kostenverteilung nicht nur für das untersuchte Projekt sondern auch in anderen automobilen Entwicklungen zu finden sein wird.

Innerhalb des Testprozesses sind nach Angaben der Literatur zwischen ein [186] und zwei [80, 233] Drittel der Kosten der Systemtestphase zuzuordnen (Kapitel 5.2). Der Wert für Projekt BC1 ist mit zwei Dritteln somit durchaus plausibel und weist keinerlei Anzeichen für Besonderheiten auf. Für eine verallgemeinerungsfähige Aussage wäre jedoch die Untersuchung weiterer Projekte erforderlich.

Einflussfaktoren auf Fehlerbeseitigungskosten: Fehlerbeseitigungskosten steigen in der Abfolge der Testphasen an [28, 192, 193]. Eine deutliche Abhängigkeit der Fehlerbeseitigungskosten von der Testphase bestätigen auch die durchgeführten Analysen an Body Controllern und Infotainment-Systemen (s. Kapitel 5.4.1). Der Kostenanstieg unterscheidet sich jedoch sowohl stark zwischen den hier untersuchten Projekten als auch zu den ebenfalls sehr inhomogenen Werten in der Literatur. In den vorliegenden Projekten weisen die Body Controller-Projekte sowohl einen stärkeren Kostenanstieg als auch höhere absolute Fehlerbeseitigungskosten als Infotanment-Systeme auf. Es besteht also eine Abhängigkeit der Fehlerbeseitigungskosten vom Systemtyp. Ein Einfluss der Fehlerpriorität auf Fehlerbeseitigungskosten lässt sich in den Projektdaten nachweisen und wird auch in der Literatur [122] bestätigt.

Fehlerklassenorientierte Testverfahrensauswahl: Die in Kapitel 4.4.1 eingeführte testfokusorientierte Fehlerklassifikation verbindet die Fehlerklassen mit Kategorien von Testverfahren, die das Fehlverhalten mit Erreichen der erforderlichen Systemabstraktion, also frühestmöglich, adressieren. Die Basis bildet ein hier ein aus dem ODC-Verfahren von IBM problemadäquat abgeleitetes Klassifikationsverfahren (s. Kapitel 4.4.1). Das entwickelte Verfahren ist nicht an V-Modellentwicklungen gebunden, jedoch ist die Formulierung der Klassifikationsvorschrift auf phasenbezogene, automobile Entwicklungen, bei denen das System sukzessive vervollständigt und damit testbar wird, bezogen. Eine von dieser Anwendung abstrahierende, sinnstiftende Neuformulierung ist möglich, wenn die Restriktionen berücksichtigt werden. Das heißt, es müssen sich quasi-disjunkte Fehlerklassen eindeutig frühestmöglichen Identifikationszeitpunkten zuordnen lassen, die Fehlerbeseitigungskosten müssen mit den Testphasen ansteigen und die Fehlerklassen müssen durch Testverfahren adressierbar sein. Die Liste der für jede Fehlerklasse beispielhaft empfohlenen Testverfahren innerhalb der Testverfahrenkategorien kann problemlos den Präferenzen in Projektentwicklungen angepasst werden.

Kosteneinsparpotential in der Fehlerbeseitigung durch Fehlerfrüherkennung: Das theoretisch maximale Kosteneinsparpotential durch Fehlerfrüherkennung ist für die untersuchten Automobil- und Literaturprojekte nahezu identisch, sofern bei der Literatur ebenfalls die Fehlerverteilung ab Modultest herangezogen wird. Da die Fehlerdaten der Automobilprojekte erst ab dem wiederholten Modultest (s. Kapitel 5.1.2) vorliegen, ist eine quantitative Aussage sehr speziell und es wird kaum Anlass

zur Übertragung geben. Das Verfahren zur Abschätzung des Kosteneinsparpotentials kann jedoch branchenübergreifend eingesetzt werden, da die notwendigen Daten schon heute für unterschiedliche Softwareprojekte [11, 47, 78, 98, 99, 187, 193] zur Verfügung stehen.

Für ein Body Controller-Projekt, für das detaillierte Fehlerdaten vorlagen, wurden auf eine Fehlerfrüherkennung abzielende Fehlerklassen gebildet. Auf Basis dieser Klassen wurden erhebliche Kosteneinsparpotentiale durch Fehlerfrüherkennung (fehlerklassenoptimiertes Kosteneinsparpotential) nachgewiesen. Die quantitativen Ergebnisse wurden in Kapitel 6.3.2 berichtet. Eine Übertragbarkeit kann mangels Vergleichsdaten nicht untersucht werden. Die Methode zur Berechnung des fehlerklassenoptimierten Kosteneinsparpotentials ist vom Grundgedanken auf phasenorientierte Vorgehensmodelle, bei denen die Fehlerbeseitigungskosten mit den Testphasen ansteigen, beschränkt.

Effektivität von verhaltensmodellbasierten Testverfahren: Gemäß den Untersuchungen ist dringlich ein verstärkter Einsatz von verhaltensmodellbasierten Testverfahren zu empfehlen. Exemplarisch wurde die Effektivität des funktionsorientierten und verhaltensmodellbasierten Testverfahrens GATE (s. Kapitel 6.3) mit dem entwicklungstypischen, unsystematischen Testverfahren verglichen. In parallelen Entwicklungsabschnitten eines Body Controller-Projekts konnte eine deutlich überlegene Fehleridentifikation in den adressierten Fehlerklassen nachgewiesen werden. Dieses Ergebnis entspricht Erkenntnissen zur Anwendung des GATE-Verfahrens in anderen Anwendungsbereichen. Die generellen Vorteile von verhaltensmodellbasierten Testverfahren sind in der Literatur [16, 20, 27, 54, 62, 64, 75, 87, 95, 210, 218, 235] belegt. Jedoch verläuft die Übernahme dieser Verfahren in die Entwicklungspraxis schleppend, so dass der überzeugende Nachweis im Bereich der automobilen Systementwicklung als Argumentationshilfe von großem Wert sein kann.

Testautomatisierung: Der entwickelte Algorithmus zur Berechnung von Break-Even-Points von Testautomatisierungen (s. Kapitel 4.6) berücksichtigt iterative Entwicklungsprozesse differenziert. Die Methode ist für alle testfallbasierten Testprozesse mit entsprechenden Kostenelementen einsetzbar.

Die Untersuchungen zur Wirtschaftlichkeit deuten darauf hin, dass die untersuchten Body Controller unter derzeitigen Projektbedingungen den Break-Even-Point wesentlich später erreichen als in der Literatur untersuchte IT-Systeme. Die beiden Projektuntersuchungen belegen, dass die Wirtschaftlichkeit von Testautomatisierungen sehr von den Projektgegebenheiten abhängt, wie zum Beispiel von den Kosten für Automatisierungswerkzeuge.

Bestimmung testinduzierter Qualitätskosten: Das Kostenmodell für testinduzierte Qualitätskosten aus Kapitel 4.3 wurde an den Daten eines Body Controller-Projekts erprobt. Es wurden eine retrospektive Ist-Kostenbestimmung und sieben prospektive Kostensimulationen für Strategieoptimierungen vorgenommen (s. Kapitel 7). Das Kostenmodell zielt bewusst auf das automobile Umfeld und enthält daher spezifische Kostenelemente und -strukturen. Eine Verallgemeinerung auf andere Branchen war

nicht intendiert. Jedoch bietet die additive Verknüpfung der Kostenelemente in den Kostengruppen sowie den darüberliegenden Hierarchieebenen die Möglichkeit, die Zusammenstellung der Kostengruppen sowie der Kostenelemente in den Gruppen problemadäquat zu verändern. Der Großteil des Qualitätskostenmodells dürfte allerdings unmittelbar für andere Abwendungsgebiete einsetzbar sein, da die Auswahl der betrachteten Kostenelemente im Einklang mit den Qualitätskostenmodellen der untersuchten Literatur sowie verschiedenen Praxisprojekten stehen. Dies gilt insbesondere für die Entwicklung eingebetteter Systeme anderer Branchen, z. B. in der Eisenbahn-, Anlagen- oder Medizintechnik, die ebenfalls häufig der iterativen Systementwicklung nach dem V-Modell folgen. Eine Validierung des Qualitätskostenmodells muss mangels ausreichender Praxisdaten ausbleiben. Aspekte der Validierung wurden eingehend in Kapitel 4.3.5 diskutiert.

Strategieoptimierungen: Die Kosteneinsparungen durch Strategieoptimierungen, die an einem Projekt demonstriert wurden, lassen sich quantitativ nicht auf andere Projekte oder gar andere Anwendungsgebiete übertragen. Eine Übertragbarkeit ist bereits per se aufgrund der projektsensitiven Testautomatisierungen nicht zu erwarten. Es kann jedoch von dem in dieser Arbeit demonstrierten Vorgehen bei der Strategieoptimierung mit den gleichen Optimierungsansätzen auch in anderen Entwicklungsprojekten eine Kostenreduktion erwartet werden.

Im untersuchten Projekt zeigt sich anhand des Fehlverhaltens, das von Kunden – also nicht von den Entwicklern bzw. Testteams – entdeckt wurde, dass effektivere Spezifikations- und Verhaltenstests einzusetzen wären. Dieses Ergebnis, das sich mit Erfahrungen aus der Literatur [11, 19, 82] deckt, hat offensichtlich eine deutlich über das Projekt hinausgehende Bedeutung. Ebenso lassen die Ergebnisse den Bedarf nach einer verstärkten konstruktiven Qualitätssicherung erkennen, wie sie in Kapitel 8.2 diskutiert wird.

Fazit: Zusammenfassend lässt sich feststellen, dass die Validierung der entwickelten Verfahren beschränkt auf den Rahmen der vorliegenden Daten durchgeführt wurde und dabei keine grundsätzlichen Einschränkungen auf dem Gebiet der eingebetteten Systeme identifiziert wurden. Die qualitativen Ergebnisse, die beim Einsatz dieser Verfahren in der Entwicklung von eingebetteten, automobilen Systemen erzielt wurden, sind in großen Teilen nach einem Vergleich mit Literaturwerten nicht auf die untersuchten Projekte und auch nicht auf die Automobilbranche beschränkt. Sie können als Basis für Empfehlungen für eine kostenoptimierte Qualitätssicherung im Testprozess herangezogen werden. Die quantitativen Ergebnisse dagegen bleiben weitgehend auf die vorliegenden Projekte beschränkt.

8.2. Folgerungen für einen kostenoptimierten Testprozess

Die in dieser Arbeit nachgewiesenen Kosteneinsparpotentiale sowie die aufgezeigten optimierenden Teststrategievariationen lassen erwarten, dass von den nachfolgend

aufgeführten Maßnahmen für künftige Softwareentwicklungen von eingebetteten Systemen generell eine kostenoptimierende Wirkung im Testprozess ausgeht.

Nutzung der Kosteneinsparpotentiale durch Fehlerfrüherkennung: Die Fehlerfrüherkennung wird als ein effektiv kostenreduzierender Optimierungsansatz zur Nutzung der derzeit hohen Kosteneinsparpotentiale nachgewiesen (s. Kapitel 6.1) und sollte daher ein zentrales Ziel der Qualitätssicherung sein. Zur Unterstützung der Testverfahrensauswahl kann die in dieser Arbeit vorgestellte fehlerklassenorientierte Testverfahrensauswahl eingesetzt werden, die durch die Analyse von Fehlermeldungen eines Referenzprojekts für ein formuliertes Testziel Kategorien optimierender Testverfahren für den gesamten Testzyklus aufdeckt (s. Kapitel 4.4.2).

Einsatz von Verfahren zur Reduktion von Spezifikationsfehlern: Derzeit identifizieren Testprozesse Spezifikations- und Systemverhaltensfehler in unzureichendem Umfang. Eine diesbezügliche Qualitätsoptimierung muss zweifelsfrei in der Anforderungsdefinition mit dem Ziel der Fehlervermeidung beginnen. Dadurch könnten einerseits der Umfang und die Kosten der späten Fehlerbeseitigung reduziert werden sowie andererseits aufgrund frühzeitig erhöhter Qualität ein weniger intensiver und somit kostenreduzierter Systemtest durchgeführt werden. Zudem kann aus der Verbesserung der Spezifikationsqualität eine optimierte Testfallqualität folgen. Die Fokussierung auf Fehlervermeidung setzt jedoch einen Paradigmenwechsel in den entwickelnden Unternehmen voraus, der sich nur längerfristig durchsetzen lassen wird. Parallel dazu und kurzfristig praktikabel ist an einer Optimierung des Testprozesses zu arbeiten mit dem Einsatz effizienterer Methoden und mit Fokussierung auf Spezifikationsfehler im Testprozess. Die Zielsetzung dieser Arbeit konzentriert sich zwar auf den Testprozess, dennoch wird ausblickend für die Anforderungsdefinition ein Optimierungsansatz durch eine halbformale Spezifikationsmethode skizziert (s. Kapitel 9.3).

Einsatz von Verfahren zur Reduktion von Systemverhaltensfehlern: Ein großer Anteil der im Testprozess nicht entdeckten Fehler sind Systemverhaltensfehler. Systemverhaltensfehler sind vorrangig im Systemtest durch funktionsorientierte und verhaltensmodellbasierte Tests diagnostizierbar. Hier besteht offensichtlich ein Bedarf für eine Änderung der Teststrategie. Von den hier analysierten Testphasen mit unabhängigen Testteams ist der Systemtest nun bereits die kostenintensivste Testphase, somit erscheint das Aufzeigen von weiterem Testbedarf zunächst kontraproduktiv zur Zielsetzung der Kostenreduktion. Heute bereits vorliegende, jedoch nur zögernd in die Praxis Eingang findende Testverfahren [178, 197, 214] für den Systemtest bergen Potential in vielerlei Hinsicht: (a) Reduktion der Testfallerstellungskosten, (b) Reduktion der Testdurchführungskosten, (c) Steigerung der Effektivität der Fehleridentifikation und somit (d) Reduktion der Fehlerbeseitigungs- und Fehlerfolgekosten für nach dem Systemtest identifiziertes Fehlverhalten. Für ein funktionsorientiertes und verhaltensmodellbasiertes Testverfahren wurde nachgewiesen, dass die Testkosten bei deutlich stärkerer Effektivität des Testverfahrens geringer sind als bei dem verwendeten, für die derzeitige Praxis typischen [214, 259], unsystematischen, auf Erfahrung basierenden Test (s. Kapitel 6.3). Für die Softwareentwicklung ergibt sich hieraus die Notwendigkeit, Kosten und Effektivität der derzeit eingesetzten Testverfahren zu evaluieren

und ggf. kostenoptimierendere, zu den unternehmensspezifischen Bedürfnissen passende Testverfahren zu wählen. Aufgrund der verstärkten Fehleridentifikation mit Fehlerbeseitigung entstehen allerdings im Systemtest zusätzlich Kosten, die u. U. erst in Bezug auf die Gesamtqualitätskosten eine Kostenreduktion darstellen.

Verfahrensrevision für den gesamten Testbereich: In der Regel kann nur ein Teil des vorhandenen Kosteneinsparpotentials ausgeschöpft werden, wenn die eingesetzten Testverfahren in nur einer Testphase durch effizientere Testverfahren ersetzt werden, wie hier für die Systemverhaltensfehler im Systemtest (s. Kapitel 6.3.2) nachgewiesen. Daher sollte eine umfassende, alle Testphasen einschließende Verfahrensrevision durchgeführt werden.

Projektadäquate Ausweitung von Testautomatisierungen: Die Kosten der Testdurchführung können wirksam durch projektadäquate Ausweitung des derzeitig beobachteten Einsatzes von Automatisierungen reduziert werden. Da die Kosten der Testautomatisierung sehr projektsensitiv sind (s. Kapitel 6.4.3), sollte vor Festlegung des Automatisierungsumfanges die Wirtschaftlichkeit mit einer Break-Even-Point-Berechnung oder durch Simulation der zu erwartenden testinduzierten Entwicklungskosten evaluiert werden.

Revision der Versorgung mit Testwerkzeugen: Bedeutende Kostenwirkung kann auch von Fixkosten ausgehen, wie in einem der untersuchten Projekte an den Lizenzkosten für Automatisierungswerkzeuge gezeigt wurde (s. Kapitel 6.4.1). Das Konzept der firmenüblichen (in diesem Fall zentralen) Versorgung mit Entwicklungswerkzeugen sollte überprüft und mit Alternativen verglichen werden.

Qualitätssteigerung in der Kosten- und Fehlerdokumentation: Es sollte an einer problemadäquat differenzierten Kosten- und Fehlerdokumentation gearbeitet werden. Speziell wären in den untersuchten Datenbanken die Beschreibungen zu Fehlverhalten bzw. Fehlerursachen zu optimieren. Die Beschreibungen sollten aussagekräftig für eine Fehlerklassifikation sein oder direkt mit Fehlerklasse hinterlegt sein und Angaben zum Fehlerbeseitigungsaufwand und die Dokumentation fehleridentifizierender Testfälle enthalten. Die Anforderungen sind nachfolgend in Kapitel 9.1 erläutert. Diese Forderungen nach optimierten Fehlerbeschreibungen und Testfalldokumentationen finden sich häufig in der Praxis und werden von Zeller [271] diskutiert. Einige Forderungen, wie verbesserte Beschreibung des Fehlverhaltens, dienen einer effizienten Fehlerbeseitigung. Aufwandsinformationen sowie Fehlerklassifikationen sind nicht nur für die Fehlerbeseitigung sondern auch für die Erforschung wirtschaftlicher Zusammenhänge und die Optimierung von Teststrategien erstrebenswert. Letztere stellen eher einen indirekten Nutzen dar und finden sich vielleicht aus diesem Grunde seltener als Forderung.

Auswahl von Referenzprojekten: Aus der in den vorliegenden Projekten eindeutig festgestellten Systemabhängigkeit der Fehlerbeseitigungskosten wird die Empfehlung abgeleitet, Kostenprognosen und Empfehlungen zur Teststrategie vorrangig auf Basis von Entwicklungen mit identischem Systemtyp durchzuführen.

Zusammenfassung: Zusammenfassend kann festgestellt werden, dass die betrachtete Softwareentwicklung erhebliche Kosteneinsparpotentiale aufweist. Verfahren zur Identifizierung von Systemverhaltensfehlern und Automatisierungen können die Softwarequalität im Testprozess effizient steigern. Jedoch sollten Auswahl und Ausmaß der Maßnahmen jeweils an den projektspezifischen Gegebenheiten überprüft werden und ggf. weitere Maßnahmen, wie z. B. das Eliminieren von projektspezifischen Kostenschwerpunkten, inkludiert werden. Im Rahmen dieser Arbeit wurden geeignete Werkzeuge zur Unterstützung einer Kostenoptimierung im Testprozess entwickelt.

9. Ausblick

In diesem Kapitel werden Ansätze für weiterführende Untersuchungen benannt sowie Lösungsansätze zu nahe verwandten Forschungsgebieten skizziert. Letztere traten während der Analyse der Fehlerdaten und der Anwendung des Qualitätskostenmodells ins Blickfeld, gehören jedoch nicht zur engeren Thematik dieser Arbeit.

9.1. Datenqualität

Optimierung von Fehlerdatenbanken: Die analysierten Fehlerdatenbanken und die daraus gewonnenen Daten zeigen einen Optimierungsbedarf für den Datenbankentwurf. Ein Ziel von Datenbankoptimierungen sollte sein, die Vollständigkeit der Fehlerdaten zu erhöhen, um eine automatisierte Auswertung der Daten zur projektbegleitenden Qualitäts- und Kostenkontrolle – auch unter Einsatz des Qualitätskostenmodells – zu ermöglichen. Hierzu sollten die Fehlerdatenbanken im Besonderen in folgenden Punkten kritisch geprüft werden:

- Die Daten sind eindeutig hinsichtlich Syntax, Semantik und Dimension zu hinterlegen.
- Die Dateneingabe ist zu unterstützen, um die geforderte Eindeutigkeit von Syntax, Semantik und Dimension zu erreichen.
- Fehlverhalten ist kurz, präzise, mit allen relevanten Informationen zu beschreiben, um eine Fehleranalyse und die Identifikation von Fehlerduplikaten optimal zu unterstützen.
- Umfassende Information zur Fehlerursache sind zu hinterlegen, um eine zutreffende Einordnung von Fehlermeldungen in definierte quasi-disjunkte Fehlerklassen zu gewährleisten.
- Die Motivation zur Eingabe von Aufwandsdaten (aufgeschlüsselt nach Analyse, Beseitigung und Test) ist zu verstärken.
- Die Dokumentation fehleridentifizierender Testfälle (Testschritte, erwartetes und beobachtetes Ergebnis) und die Verfolgbarkeit von Fehlermeldungen zu fehleridentifizierenden Testfällen und Regressionstestfällen sind zu verbessern, um Fehleranalysen und Korrekturtests zu unterstützen.

Zahlreiche weitere Forderungen und Lösungsansätze zur Beschreibung von Fehlverhalten und Testfällen befinden sich bei Zeller [271].

9.2. Qualität von Testfallmengen

Testfallgenerierung und -ausführung können erhebliche Kosten verursachen, wie auch in den Untersuchungen dieser Arbeit ersichtlich ist. Daher kann die Reduktion der Anzahl Testfälle wirtschaftlich interessant erscheinen. Durch etwaige Testfallreduktionen darf die Fehleridentifikation jedoch nicht derart abnehmen, dass die Kostenreduktionen in Testfallgenerierung und -ausführung durch erhöhte Kosten für späte Fehlerbeseitigungen mehr als ausgeglichen werden. Ein Ziel ist es also, für den Test ein Subset der möglichen Testfälle zu ermitteln, das das konkret formulierte Testziel optimal umsetzt [178]. Die Bewertung eines Testfalls bzw. einer Testfallmenge hängt also von der Zielsetzung ab. So kann beispielsweise eine optimale Testfallmenge unterschiedlich definiert sein, wenn sie vorrangig sicherheitskritische oder möglichst viel Fehlverhalten identifizieren soll [178]. Häufig wird die Anzahl der Fälle von identifiziertem Fehlverhalten als Bewertungskriterium für Testfallmengen angeführt [209]. Werden, wie in dieser Arbeit, Zeit und Kosten der Testfallausführung prospektiv optimiert, müssen die Testfallmengen bereits während der Projektplanung bewertet werden können. Hierfür ist das genannte Kriterium jedoch kein geeignetes Gütekriterium, denn die Anzahl der Fälle von identifiziertem Fehlverhalten lässt sich erst retrospektiv nach der Testfallausführung definitiv bestimmen. Zudem ist bei diesem Bewertungskriterium die Qualität der identifizierten Fehlerursachen beliebig. Es können trotz hoher Identifikationsrate Fehlerursachen bestimmter Qualität, auch bedeutsamer Kategorie, nicht identifiziert worden sein. Weiterhin beinhaltet das Kriterium keine Aussagen zur Wirtschaftlichkeit, also dem Verhältnis von Aufwand und Nutzen. Auch das häufig verwendete Bewertungskriterium der Überdeckung (z. B. von Quelltext) ist als Maß für die Testfallqualität ungeeignet [209], da eine hohe Überdeckung nicht unbedingt die Identifikation vieler Fälle von Fehlverhalten bedeutet [13, 218].

Es wird evident, dass Bewertungskriterien für Testfälle und Testfallmengen bisher nur marginal untersucht wurden [96, 134, 151]. Bislang konnten keine Kriterien, die für eine verlässliche Bewertung von Testfallmengen ausreichend sind [209], gegeben werden. Daher ist die Identifizierung von Eigenschaften *guter* Testfälle ein lohnendes Forschungsziel.

Ein allgemeines Bewertungsverfahren, das auch eine Bewertung bei divergierenden Testzielen einschließt, kann vom Bewertungsverfahren für anzuwendende Testverfahren aus Kapitel 4.4 abgeleitet werden. Hierbei wird die Klassifikation von Fehlermeldungen durch eine Klassifikation von Testfällen bzw. Testfallmengen ersetzt. Folglich sind Eigenschaften von Testfällen und Testfallmengen zu identifizieren, die in Bezug zur Testfallqualität gesetzt werden müssen. Um als Entscheidungsaspekt zu fungieren, muss der Bezug bestimmte Testziele umsetzen. Beispiele für derartige Qualitätsmetriken können sein:

- *Testfalllänge:* Bei verhaltensorientierten Testfällen könnte sich die Länge der auszuführenden Szenarien auf die Testfallqualität auswirken, z. B. könnten längere Testfälle mehr Fälle von Fehlverhalten pro Ausführungszeit identifizieren. So ergaben Analysen von Testverfahren, dass das Zusammenfassen von kur-

zen zu längeren Testfällen 5-10% der Testzeit reduzieren kann, ohne dabei die Testqualität zu verringern, da die Überdeckung erhalten bleibt [133].

- *Hamming-Distanz der Eingabewerte:* Knuth [160] beobachtete, dass in seiner Untersuchung 95% der Testfälle nur 4% des Quelltextes überdeckten. Der Test sollte jedoch tendenziell den gesamten Testraum überdecken [115]. Als Maß für die Verteilung der Eingabewerte und somit für die Überdeckung des Testraums könnte die Bestimmung der Hamming-Distanz oder der kartesischen Entfernung hilfreich sein [115]. Diesen Testfallqualitätsmaßen liegt auch die Testfallerzeugung des *Anti-Random-Tests* zugrunde [115].
- *Robustheitstestfälle:* Bestimmte Datenwerte werden erfahrungsgemäß häufig für Robustheitstest verwendet. Beispiele für Eingabedaten für Robustheitstestfälle sind negative Zahlen, die Zahl Null, große Werte (Überläufe), leere Zeichenketten, Null-Zeiger. Das Vorhandensein derartiger Werte könnte betrachtet werden.
- *Kombination von Parameterwerten:* Die Kombination von Parameterwerten kann von Interesse sein. Studien [114, 267] zufolge ist Fehlverhalten häufig schon durch die Existenz zweier bestimmter Parameterwerte induziert. Die Wahrscheinlichkeit, dass drei oder mehr Parameter bestimmte Werte annehmen müssen, um ein Fehlverhalten zu erzeugen, nimmt ab.

Weitere Qualitätskriterien für Testfälle könnten aus der Analyse anerkannter Testverfahren gewonnen werden. Es wäre zu ermitteln, welche Merkmale besonders zielführenden Testverfahren zugrunde liegen und wie sich diese von weniger erfolgreichen unterscheiden. Anschließend kann ermittelt werden, wie sich die relevanten Eigenschaften von Testverfahren in Testfallmengen identifizieren lassen. Beispielsweise könnte eine Qualitätsmetrik *Kombination von Parameterwerten* erarbeitet werden. Die paarweise Überdeckung von Parameterwerten [267] ist ein in der Praxis erfolgreich angewendetes Testfallreduktionsverfahren [56, 57].

9.3. Spezifikationsqualität durch ein halbformales Spezifikationsmodell

9.3.1. Motivation

Spezifikationsfehler stellen zusammen mit den Systemverhaltensfehlern im Systemtest die Fehlerklassen mit den größten Häufigkeiten dar (s. Tabelle 5.9 in Kapitel 5.6). Spezifikationsfehler wurden unterteilt in unvollständige und fehlerhafte Spezifikationen, wobei erstere häufiger auftreten. Auch die NASA beobachtete in einer Auswertung [2] von 45 Anforderungsinspektionen, dass 66% der Spezifikationsfehler auf unvollständige und nur 28% auf fehlerhafte Spezifikationen zurückzuführen sind. Die verbleibenden 6% der Spezifikationsfehler ergeben sich durch nachträglich hinzukommende Funktionen. Als Optimierungsziel von Spezifikationsverfahren lässt sich demnach primär die Vermeidung von unvollständigen Spezifikationen ableiten.

Derzeitige Entwicklungen fokussieren die Reduktion von unvollständigen Spezifikationen durch Formalisierung [229]. Spezifikationen liegen nach wie vor meist nur in natürlicher Sprache vor [21, 177, 179, 255, 259]. Laut einer Untersuchung [229] des

Bundesministeriums für Bildung und Forschung werden in 30% der Unternehmen nur die wesentlichen Eigenschaften eines Systems beschrieben. Formale Beschreibungen werden nur in 5% der Unternehmen angefertigt, in weiteren 5% der Unternehmen wird vollkommen informal, sogar ohne schriftliche Dokumente, spezifiziert. Es existiert offensichtlich eine Divergenz zwischen theoretisch verfügbaren und in der Praxis angewendeten Spezifikationsverfahren [178]. Das Erstellen formaler Spezifikationen wird in der Praxis häufig als zu komplex und zu aufwändig angesehen [9, 32, 119]. Auch wenn Studien diese Annahme widerlegen [5, 32, 32, 88, 94, 120, 123], kommt formalen Spezifikationen derzeit eine dem dynamischen Test untergeordnete Bedeutung zu [178].

Einen Mittelweg könnten hier halbformale Modelle darstellen, die einen in der Praxis akzeptablen Aufwand mit einem respektablen, zu erwartenden Nutzen vereinen. *Halbformale Modelle* werden hier analog zu Wieringa [264] als Diagramme und Tabellen zur strukturierten Darstellung von Daten verstanden. Beispiele für halbformale Modelle sind *Zustandsautomaten* [41, 164, 202], *Nachrichten-Reihenfolge-Diagramme* [144, 164], *Anwenderfalldiagramme* [202] oder *strukturierte Entscheidungstabellen* [198, 270]. Neben Regeln für die strukturierte Darstellung von Informationen können Manipulationsregeln zur Überführung in andere Darstellungsformen formuliert werden.

Aufbauend auf halbformalen Verhaltensmodellen soll effizient die Identifikation unvollständiger und fehlerhafter Spezifikationen erreicht werden. Hierfür bieten sich u. a. an:

Verifikation: Für die Verifikation der Korrektheit kann auf verschiedene Korrektheitsbeweise (Verifikation) durch *Modellprüfung* [33, 55] zurückgegriffen werden. Die Möglichkeiten für Korrektheitsbeweise können eingeschränkt sein [264], da die hier betrachteten Verhaltensmodelle im Gegensatz zu formalen Modellen nur halbformal sind. Es wird davon ausgegangen, dass nachfolgende Analyseverfahren praxisrelevant auch für halbformale Verhaltensmodelle eingesetzt werden können, um die Spezifikation durch graphentheoretische und statische Verfahren zu verifizieren. Dies kann mit und ohne Betrachtung von semantischen Annotationen, wie Bedingungen, geschehen: Erreichbarkeitsanalysen [55], Deadlock-Analysen [71], Kompatibilitätsprüfung an Schnittstellen [172], Robustheitsprüfung bzgl. Eingaben und Signalen [128] sowie Identifikation von Nicht-Determinismus [127].

Verfolgbarkeit: Die manuelle Verfolgbarkeit einzelner Anforderungen bis zum Quelltext ist bereits Stand der Praxis und sogar in Normen gefordert, so z. B. in der Norm IEC 61508 für sicherheitskritische Systeme [72]. Neu hingegen wäre eine automatisierte Verfolgbarkeit von gruppierten Elementen, dargestellt in halbformalen Verhaltensmodellen oder auch anderen Formen, da Elemente untereinander in Beziehung zueinander stehen können. So kann beispielsweise eine Anforderung andere einschränken bzw. ergänzen oder ein Subzustand eines Automaten einen anderen Zustand verfeinern. Die Automatisierbarkeit der Verfolgbarkeit von Elementen zwischen verschiedenen Dokumenten erfordert eine maschinenlesbare Beschreibung. Eine solche könnte in zukünftigen Arbeiten untersucht und definiert werden.

Ein halbformales Spezifikationsverfahren, wie es im Folgenden skizzenhaft entwickelt wird, könnte einen Weg zur Verbesserung der derzeitigen Situation weisen.

9.3.2. Modellfunktion

Das Spezifikationsverfahren soll die Überführung informaler Anforderungen in halbformale Verhaltensmodelle unterstützen. Dabei sind hinsichtlich späterer Automatisierungen in jedem Arbeitsschritt maschinenlesbare Dokumente zu verwenden. Zu Beginn einer Systementwicklung existiert eine *informale Systembeschreibung* des zu entwickelnden Systems z. B. in Form einer Vision (s. Abbildung 9.1). Diese informalen Systembeschreibungen werden durch Definieren in Anforderungen überführt (s. Abbildung 9.1, Nummer 1), häufig als *informale, textuelle Anforderungen* [229]. Diese informalen Anforderungen sind im nächsten Schritt in *halbformale Anforderungen* zu überführen (s. Abbildung 9.1, Nummer 3).

Besondere Bedeutung kommen Beziehungen zwischen informalen Anforderungen [77] sowie deren Übertragung in halbformale Beziehungen zu. In Abbildung 9.1, Nummer 2 stehen exemplarisch *Anforderung 1* und *Anforderung 2* in Beziehung zueinander (z. B. Verfeinerung, Nebenbedingung usw.). Solche Beziehungen sind nicht zwingend evident, aber dennoch essentiell für das Verstehen und Umsetzen der gesamten Spezifikation [77]. Die Beziehungen zwischen den Anforderungen sind umfassend in einem Modell darzustellen (s. Abbildung 9.1, Nummer 4). Als besonders intuitiv und praxisorientiert haben sich zustandsbasierte *halbformale Verhaltensmodelle* herausgestellt [22, 35]. Zu diesen gehören u. a. Zustandsautomaten [41, 202], Beschreibungen von *Diensten* [124] und Nachrichten-Reihenfolge-Diagramme [144].

Die erstellten, halbformalen Verhaltensmodelle sollen für eine Reihe von entwicklungsunterstützenden Maßnahmen eingesetzt werden:

- Die halbformalen Verhaltensmodelle sollen ausführbar sein, so dass *Simulationen* durchgeführt werden können (s. Abbildung 9.1, Nummer 5). Die sich ergebenen Simulationsläufe sollen die Aussagekraft der halbformalen Verhaltensmodelle ergänzen und somit das Systemverständnis steigern [60,180]. Beispielsweise sollen halbformale Verhaltensmodelle und Simulationsläufe als Grundlage für Diskussionen und Inspektionen eingesetzt werden [201].
- In Abhängigkeit vom Detaillierungsgrad der halbformalen Verhaltensmodelle sollen diese, wie bereits heute Stand der Praxis, zur (ggf. eingeschränkten) Quelltextgenerierung verwendet werden [18, 60, 211, 217] (s. Abbildung 9.1, Nummer 6). Eine eingeschränkte Quelltextgenerierung sollte wenigstens Klassen- und Methodenrümpfe umfassen. Darauf aufbauend soll die Ableitung ganzer Algorithmen angestrebt werden.
- Weiterhin sollen halbformale Verhaltensmodelle verwendet werden, um Testfälle für den Systemtest automatisiert zu generieren [102] (s. Abbildung 9.1, Nummer 7).

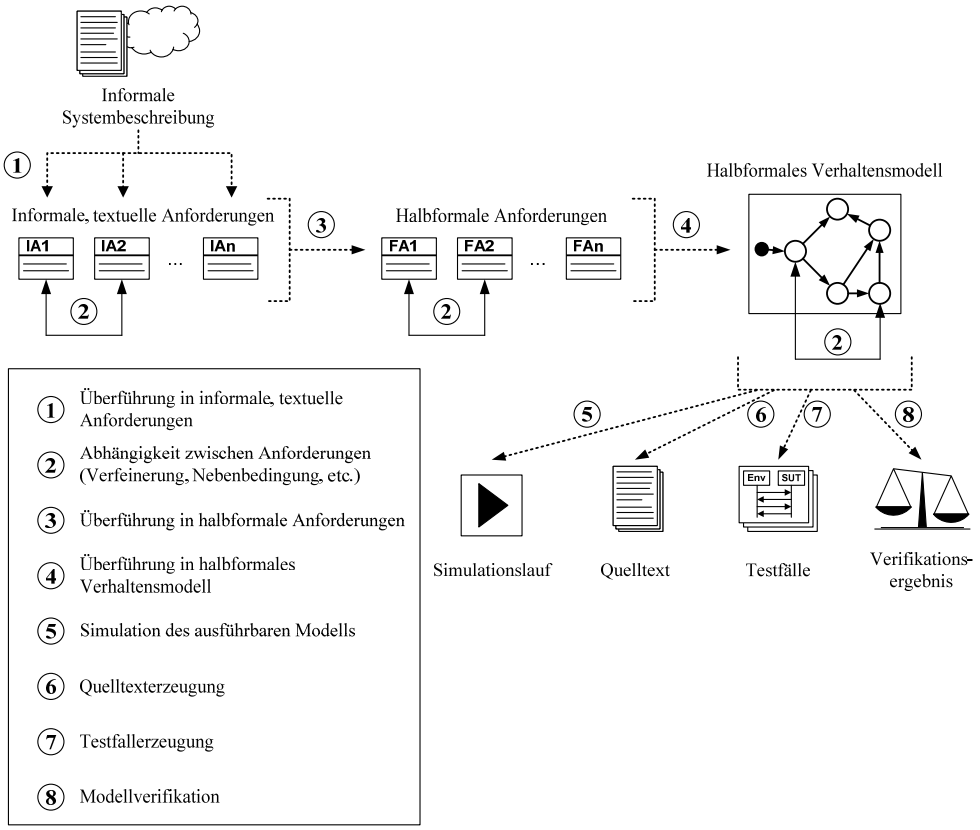


Abbildung 9.1.: Überführung von informalen Anforderungen in maschinenlesbare Dokumente

- Halbformale Verhaltensmodelle sollen Grundlage für Verifikationen sein, die zu Verifikationsergebnissen führen (s. Abbildung 9.1, Nummer 8). So sollen beispielsweise Widersprüche oder Uneindeutigkeiten im Modell identifiziert werden, um mögliche Fehlerursachen in den Spezifikation und der Implementierung zu erkennen.

9.3.3. Strukturbausteine und Muster

Die beschriebenen Dokumente (Anforderungen, Verhaltensmodelle, Simulationsläufe, Quelltext, Testfälle, Verifikationsergebnisse) stellen das zu entwickelnde System unter verschiedenen Zielaspekten wie Simulation, Test oder Verifikation dar. Diese Dokumente bestehen aus Elementen wie Transitionen, Signalen, Satzbausteinen, die hier als *Strukturbausteine* bezeichnet werden. Strukturbausteine können nach zeitlicher Abfolge, inhaltlicher Beziehung usw. gruppiert werden und ergeben so *Muster*.

Ein Ziel des zu entwickelnden Spezifikationsverfahrens ist die Verfolgbarkeit von Strukturbausteinen und Mustern in den verschiedenen Dokumenten (s. Abbildung 9.2). Hierbei soll die Semantik der betrachteten Strukturbausteine und Muster

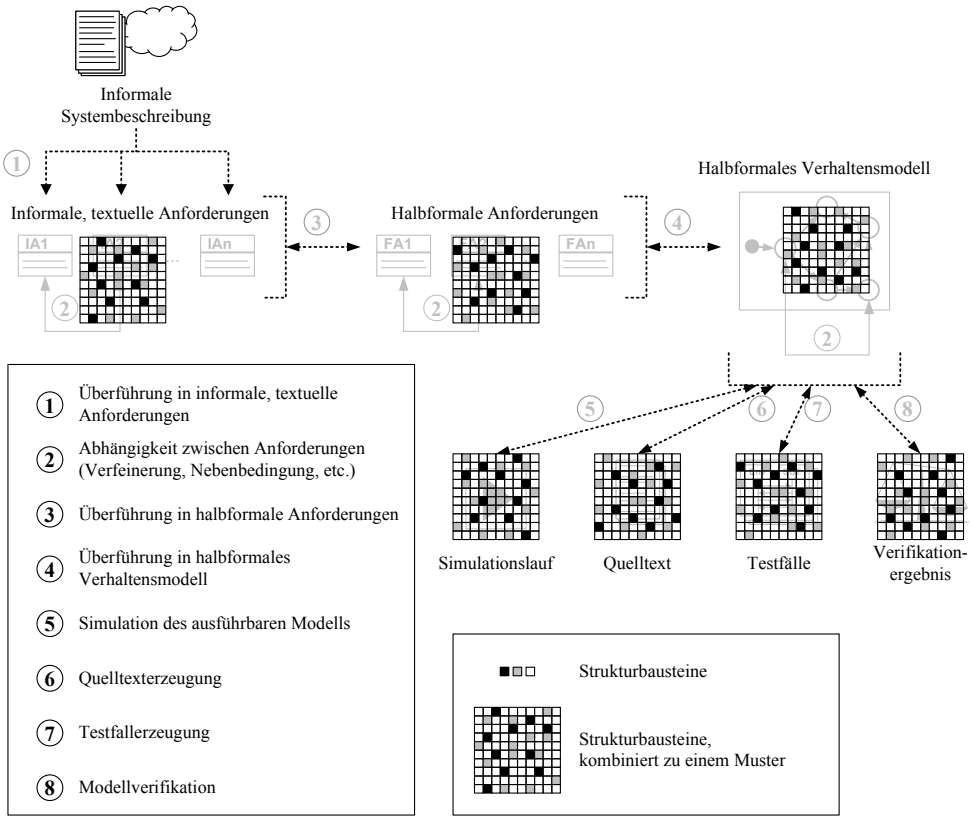


Abbildung 9.2.: Durchgängigkeit von Strukturbausteinen und Mustern über verschiedene Darstellungsformen hinweg

erhalten bleiben. Es werden also Strukturbausteine und Muster gleicher Semantik in unterschiedlichen Dokumenten markiert. Die Darstellung wird jedoch aufgrund der unterschiedlichen Elementtypen (Transitionen, Satzbausteine usw.) variieren.

Die Verfolgbarkeit zwischen Dokumenten könnte auf Transformationsregeln basieren. Es wird eine bidirektionale Verfolgbarkeit zwischen allen Dokumenten angestrebt, daher ist auf die Umkehrbarkeit der Transformationsregeln zu achten.

Werden durch Verifikationen oder Testfälle fehlerhafte Strukturen festgestellt, könnten involvierte Strukturbausteine und Muster mit geringem Aufwand in anderen Dokumenten identifiziert werden. Damit ließen sich z. B. Simulationsläufe mit nicht intendierten Ergebnissen mit weniger Aufwand auf einzelne Anforderungen zurückführen als nach derzeitiger Praxis. Dieses Szenario des halbformalen Spezifikationsverfahrens kombiniert verschiedene Ansätze, die zum Teil Gegenstand aktueller Forschungen sind. Für den praktischen Einsatz des Spezifikationsverfahrens sind weiterführende Analysen und Weiterentwicklungen notwendig.

10. Zusammenfassung

Die vorliegende Arbeit leistet einen praxisorientierten Beitrag zu Optimierungen in der Qualitätssicherung der Softwareentwicklung von eingebetteten Systemen im automobilen Umfeld. Fokussiert wird der phasenorientierte, mit dem Modultest beginnende Testprozess. Softwarequalität, Entwicklungskosten und -zeit werden im Testprozess wesentlich durch die Auswahl der Testverfahren bestimmt. Eine projektspezifische Optimierung der Teststrategie soll eine effektivere Fehleridentifikation mit einer Reduktion der Qualitätskosten verbinden. Es wird ein Optimierungsprozess definiert, der das Vorgehen für die Optimierung durch Fehlerfrüherkennung, Testautomatisierung und ggf. Eliminierung von projektspezifischen Kostenschwerpunkten schrittweise vorgibt.

Die heutige Entwicklungspraxis für eingebettete Systeme wurde anhand von Projekten eines Automobilzulieferers analysiert. Für Struktur- und Kostenanalysen standen Daten aus acht industriellen Entwicklungsprojekten zur Verfügung. Fehlerdatenbanken, Budgetpläne sowie Expertengespräche mit Projektbeteiligten konnten projektabhängig herangezogen werden. Diese Projektdaten sowie Entwicklungsdaten aus der Literatur bilden die Basis zur Quantifizierung von testphasenabhängigen Kosten und für Fehlerströme. Testphasen, Systemtypen und Fehlerpriorität konnten als Einflussfaktoren auf Fehlerbeseitigungskosten nachgewiesen werden.

Strategieoptimierungen setzen die prospektive Berechnung von testinduzierten Qualitätskosten, i. Allg. über Qualitätskostenmodelle, voraus. Die in der Literatur diskutierten Qualitätskostenmodelle tragen weder dem automobilen Kontext noch dem vorliegenden Fokus und den Spezifika von iterativen, phasenorientierten Entwicklungen hinreichend Rechnung. Es wird daher ein Algorithmus zur Berechnung der testinduzierten Qualitätskosten entworfen, der über branchenspezifische Kostenelemente und -strukturen detaillierte Kostenberechnungen für Testaufbau, Testvorbereitung, Testdurchführung samt Fehlerbeseitigung erlaubt. Der Algorithmus lässt sowohl eine exakte retrospektive Berechnung als auch die prospektive Abschätzung mit Plan- und Erfahrungsdaten zu. Trotz einer primären Branchenabhängigkeit ist das Berechnungsmodell zumindest bedingt auf andere Bereiche übertragbar. Dies gilt insbesondere für eingebettete Systeme in der Eisenbahn-, Anlagen- oder Medizintechnik, die in der Regel ebenfalls mit einem phasenorientierten, iterativen Prozess entwickelt werden. Auf den Algorithmus des Kostenmodells aufbauend wurden mehrere, nachfolgend genannte Verfahren entwickelt, die projektspezifische Strategieoptimierungen unterstützen. Das Vorgehensschema zur Ermittlung von projektspezifischen Optimierungen von Teststrategien bietet mit den methodischen Entwicklungen ein Rahmenwerk für die optimierende Planung von Entwicklungsprojekten.

Mit Hilfe der entwickelten Verfahren werden die vorliegenden Projektdaten zunächst bezüglich einzelner Aspekte untersucht. Ein Verfahren zur Quantifizierung von Kosteneinsparpotentialen durch Fehlerfrüherkennung weist in den Fehlerbeseitigungskosten des untersuchten Projekts erhebliche, realistisch erreichbare Kosteneinsparpotentiale nach. An parallel durchgeführten Entwicklungsabschnitten eines Projekts wird gezeigt, dass das funktionsorientierte und verhaltensmodellbasierte GATE-Verfahren im Systemtest Systemverhaltensfehler mit guter Effizienz identifiziert und das vorhandene Kosteneinsparpotential nahezu ausschöpfen kann im Vergleich zu einer auf Erfahrung basierenden, jedoch unsystematischen Testfallerstellung. Mit einem Verfahren zur Break-Even-Point-Bestimmung für Testautomatisierungen wird an zwei Projekten die Wirtschaftlichkeit der durchgeführten Testautomatisierungen geprüft. Ein Break-Even-Point kann für die beiden Projekte – aufgrund extrem hoher Lizenzkosten bzw. hoher Testfallautomatisierungskosten – nicht während der Projektlaufzeit erreicht werden, obwohl die niedrigeren Kosten des eigentlichen Testvorgangs hohe Einsparungen aufweisen. Eine weitere Verfahrensgruppe weist fehlerklassenbasiert Kategorien von Testverfahren aus, die ein definiertes Testziel effizient unter Einbeziehung einer Fehlerfrüherkennung umsetzen.

Der definierte Optimierungsprozess wird unter Einbeziehung der genannten Entwicklungen und Analysen an einem Projekt erprobt. Als Bezugsbasis werden die Ist-Qualitätskosten des Projekts retrospektiv bestimmt. Anschließend werden die testinduzierten Qualitätskosten sukzessive durch eine durchgängige Anwendung des funktionsorientierten und verhaltensmodellbasierten GATE-Verfahrens im Systemtest und durch einen weitgehend automatisierten Testprozess reduziert. Zudem wird der Kosteneinfluss der projektspezifisch hohen Lizenzkosten demonstriert. Abschließend wird das Kostenverhalten der gleichen Strategievarianten auf Basis von Fehler- und Kostenverteilungen aus der Literatur analysiert. Die testinduzierten Qualitätskosten werden jeweils mit dem entwickelten Kostenmodell abgeschätzt. Es erweist sich für den quantitativen Vergleich von Teststrategien als sehr gut geeignet. Die zugrunde liegende Struktur ermöglicht algorithmische Anpassungen in Hinblick auf die vorliegende Detaillierung der Datenbasis. Die Kosten werden detailliert aufgeschlüsselt dargestellt.

Die Erkenntnisse zur Optimierung von Teststrategien sind qualitativ weitgehend auf andere Projekte und großteils auch auf andere Fachgebiete übertragbar. Quantitative Ergebnisse sind eher an Projektgegebenheiten gebunden. Auf Basis der Erkenntnisse werden verallgemeinert Maßnahmen abgeleitet, die auf einen kostenoptimierten Testprozess für künftige Softwareentwicklungen eingebetteter Systeme hinzielen. Zur weiterführenden Forschung werden Optimierungsansätze in verwandten Forschungsgebieten skizziert: Ansätze für eine Verbesserung der Datenqualität in Fehlerdatenbanken, Ansätze zur Bewertung der Qualität von Testfallmengen und Ansätze für ein Verfahren zur Optimierung der Spezifikationsqualität. Letzteres soll auf Basis von halbformalen Modellen zu einer Qualitätssteigerung in frühen Phasen der Entwicklung beitragen.

A. Algorithmen zur Berechnung von testinduzierten Qualitätskosten (zu Kapitel 4.3.3)

Dieser Anhang beinhaltet die Algorithmen des Kostenmodells zur Ermittlung von testinduzierten Qualitätskosten aus Kapitel 4.3.3 sowie deren Herleitung. Die Algorithmen berücksichtigen und formalisieren die Beziehungen der Kostenstruktur in Kapitel 2.5.

A.1. Notation in der Kostenermittlung

Abkürzungen in Quantifizierungsalgorithmen: Bei der Herleitung der Quantifizierungsalgorithmen werden die Abkürzungen aus Tabelle A.1 verwendet.

Abkürzung	Interpretation
K	Kosten in Währungseinheiten
A	Aufwand in Stunden (falls nicht anders angegeben)
W	Wahrscheinlichkeiten aus [0, 1]
DV	Duale Variablen mit dem Wert null oder eins
N	Anzahl der Elemente in einer Menge
G	Relative Anteile wie Testfallautomatisierungsgrad und Anpassungsgrad aus [0, 1]
$\widehat{\ }$	Geschätzter Wert; geschätzt wurde die mit „$\widehat{\ }$“gekennzeichnete Variable

Tabelle A.1.: Abkürzungen zur Herleitung der Quantifizierungsalgorithmen der testinduzierten Qualitätskosten

Indizes: Die Zugehörigkeit der Kostenelemente zu den Beziehungshierarchien wird durch Indizes (Laufvariablen) wiedergegeben, z. B. in der Notation K_p für Kosten K in Testphase p. Der Endwert der Laufvariablen wird durch den Großbuchstaben der Laufvariablen symbolisiert, im Falle der Testphasen also durch P. Existiert eine übergeordnete Hierarchieebene, wird diese als Laufvariable dem jeweiligen Endwert (höchste Ausprägung) der Laufvariable der untergeordneten Ebene angefügt, im Falle der Testphasen p innerhalb der Iterationen i als P_i (s. auch Abbildung A.1). Innerhalb jeder Ausprägung der übergeordneten Hierarchie beginnt die Laufvariable wieder mit

1, also in einer neuen Iteration i beginnt p immer mit 1. Diese Formalisierungen dienen der Darstellung der Berechnung der Qualitätskosten, speziell den Summenbildungen. Wenn Ausprägungen eines Kostenelements K in den verschiedenen Testphasen der Iterationen auftreten, wird die Summenbildung über die Iterationen i und Phasen p wie folgt dargestellt:

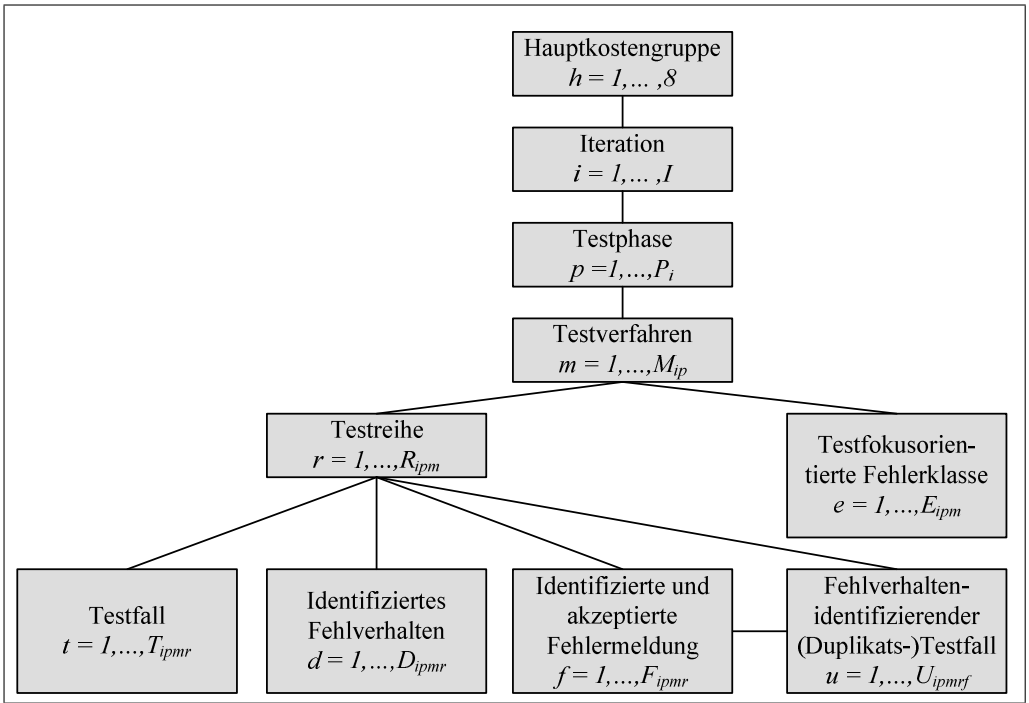
$$\sum_{i}^{I} \sum_{p}^{P_i} K_{ip}$$

Abhängigkeiten der Indizes: Die Abbildung in Tabelle A.2 visualisiert die Abhängigkeiten der Indizes. Die Tabelle listet anschließend die verwendeten Laufvariablen auf. Bei der Ermittlung der Qualitätskosten werden, je nach Zielsetzung, bei dem Testverfahren m entweder die Testreihe r oder die effektiv identifizierte Fehlerklasse e und in der Testreihe r entweder die Testfälle t, die identifizierten Ausprägungen des Fehlverhaltens d oder die akzeptierten Fehlermeldungen f betrachtet.

Spezifizierungen zu Kosten, Aufwand und weiteren Variablen: Aus Gründen der Übersichtlichkeit wird folgende Notation verwendet: Spezifizierungen zu Kosten, Aufwand usw. sind als hochgestellte Zeichenketten angegeben, die Laufvariablen werden tiefgestellt dargestellt, z. B. sind $K_{ipm}^{Testvorbereitung}$ die Kosten der Testvorbereitung der Methode m in der p-ten Testphase der i-ten Iteration.

Geschätzte Werte: Geschätzte Werte (s. Anhang A.3) werden mit dem Zeichen „$\widehat{\ }$“ gekennzeichnet. Einige Berechnungen basieren auf Daten, die abgeschätzt wurden, z. B. aus Erfahrungswerten wie prospektiv anzugebende Testfallanzahlen $\widehat{N}^{Testfallanzahl}$. Ergebnisse aus Berechnungen mit geschätzten Daten sind aufgrund der Datenbasis stets geschätzte Werte, also ebenfalls mit „$\widehat{\ }$“ gekennzeichnet.

Index zur Hauptkostengruppe: Die Endindizes von Kostenelementen in den Hauptkostengruppen *Testaufbau, Testvorbereitung, Testdurchführung, Herstellerinduzierte Fehlerbeseitigung* sowie *Kundeninduzierte Fehlerbeseitigung* sind in den einzelnen Iterationen, Testphasen, Testverfahren usw. abhängig vom spezifischen Projektablauf. Sei beispielsweise der Endindex T_{ipmr} betrachtet. T_{ipmr} stellt die Anzahl der Testfälle der Testreihe r aus Testverfahren m in der Testphase p und Iteration i dar. Im Falle einer vollständigen Automatisierung definiert dieser Index die Anzahl der zu automatisierenden und die der automatisiert auszuführenden Testfälle. Die Anzahl der in einer Iteration zu automatisierenden Testfälle weicht jedoch in der Regel aufgrund der Wiederverwendung von Testfällen aus vorangegangenen Iterationen von der Anzahl automatisiert ausgeführter Testfälle ab. Dies bedingt, dass die Endwerte der Laufvariablen in den verschiedenen Hauptkostengruppen unterschiedlich definierbar sein müssen. Diese Abhängigkeit von der Hauptkostengruppe wird nach Tabelle A.2 durch den hochgestellten Index h an dem Endwert der Laufvariablen symbolisiert, z. B. T_{ipmr}^{h}. Auf diese Darstellung wird nachfolgend jedoch verzichtet, da die Kosten für jede Hauptkostengruppe getrennt berechnet werden.



Laufvariable	**Interpretation**
h = 1, ..., 8	Laufvariable der **Hauptkostengruppen**: 1 = Testmanagement 2 = Fehlervermeidung (im Qualitätskostenmodell nicht ausgeführt) 3 = Testaufbau 4 = Testvorbereitung 5 = Testdurchführung 6 = Herstellerinduzierte Fehlerbeseitigung 7 = Kundeninduzierte Fehlerbeseitigung 8 = Fehlerfolgekosten (im Qualitätskostenmodell nicht ausgeführt) Die Endwerte der Laufvariablen können z. B. im Falle von Iterationen, Testphasen, Testverfahren in Abhängigkeit von der Hauptkostengruppe variieren. Daher werden die Endwerte der Laufvariablen, im Sinne einer Verallgemeinerung des Modells, für alle Indizes von der Hauptkostengruppe h abhängig gemacht. Diese Abhängigkeit, gekennzeichnet z. B. als I^h für Iterationen und P_i^h für Testphasen, wird der Übersichtlichkeit halber nur bei Bedarf dargestellt.
i = 1, ..., I	Laufvariable für die **Iteration**

Tabelle A.2.: Laufvariablen zur Berechnung der Qualitätskosten (Fortsetzung auf nächster Seite)

p = *1, ..., P_i*	Laufvariable für die durchzuführenden **Testphasen** in der Iteration *i*. Die Laufvariable *p* steht sowohl für die entwicklungsbegleitenden Testphasen TP_1 als auch für die Testphasen der Kundeninduzierten Fehlerbeseitigung TP_2 (s. Kapitel 4.2).
m = *1, ..., M_{ip}*	Laufvariable für **Testverfahren** in der Testphase *p* der Iteration *i*
r = *1, ..., R_{ipm}*	Laufvariable für erzeugte **Testreihen** für ein Testverfahren *m* in Testphase *p* der Iteration *i.*
e = *1, ..., E_{ipm}*	Laufvariable für die **Fehlerklasse** der testfokusorientierten Fehlerklassifikation (s. Kapitel 4.4.1), die durch ein Testverfahren m in der Testphase *p* der Iteration *i* erkannt wird.
t = *1, ..., T_{ipmr}*	Laufvariable für **Testfälle** der Testreihe *r* aus Testverfahren *m* in der Testphase *p* und Iteration *i*.
d = *1, ..., D_{ipmr}*	Laufvariable für Ausprägungen von **identifiziertem Fehlverhalten** (entspricht den beim Hersteller vorliegenden Fehlermeldungen), die durch eine Testreihe *r* der Methode m in Testphase *p* und Iteration *i* identifiziert werden.
f = *1, ..., F_{ipmr}*	Laufvariable für **akzeptierte Fehlermeldungen**, die durch eine Testreihe *r* der Methode *m* in Testphase *p* und Iteration *i* identifiziert werden.
$u = 1, ..., U_{ipmrf}$ *bzw.* $u = 1, ..., U_{ipmr}$	Laufvariable für Fehlverhalten **identifizierende Testfälle** (ggf. in Duplikatsfehlermeldungen beschrieben) einer akzeptierten Fehlermeldung *f*. Die Laufvariable wird generiert, nachdem ein Fehlverhalten durch eine Testreihe *r* des Testverfahrens *m* in der Testphase *p* und Iteration *i* identifiziert wird. Die Fehlverhalten identifizierenden Testfälle bestehenaus einem Testfall der Testreihe *ipmr* und in Duplikaten beschriebenen Testfällen, die u. U. anderen Testreihen *ipmr* zuzuordnen sind. Der Fehlverhalten identifizierende Testfall wird bei der Kostenberechnung der Fehlerbeseitigung entweder einer Fehlermeldung *f* innerhalb einer Testreihe *r* zugeordnet oder direkt der Testreihe *r*.
Anmerkung zum Endwert von Laufvariablen: Laufvariablen, deren Endwert die Laufvariable einer übergeordneten Hierarchie tragen, z. B. P_i, beginnen innerhalb jeder Ausprägung der übergeordneten Hierarchie wieder bei 1.	
Anmerkung zu den Laufvariablen ab m: Für kundenidentifiziertes Fehlverhalten werden in der Regel primär keine das Fehlverhalten identifizierenden Testfälle oder gar Testreihen vom Kunden übergeben. Zumindest *ein* fehleridentifizierender Testfall ergibt sich jedoch aus der Fehlerreproduktion. Ebenso liegt nicht zwingend die Angabe eines Testverfahrens vor bzw. es wurde keines eingesetzt. Da hier jeweils der Endindex 1 für erforderliche Laufzahlen gewählt werden kann, ergibt sich daraus (mathematisch) formal keine Sonderstellung.	

Tabelle A.2.: Laufvariablen zur Berechnung der Qualitätskosten (Fortsetzung der vorherigen Seite)

Null als neutrales Element: Für die Formulierung der Gleichungen wird folgende Voraussetzung gemacht: Fallen bestimmte Kosten nicht an, z. B. werden bestimmte Werkzeuge nicht angewendet, ist das entsprechende Kostenelement mit dem Wert

null definiert. Die testinduzierten Qualitätskosten ergeben sich weitgehend durch Summenbildungen über Kostenelemente. Die Formulierung der Teilalgorithmen nutzt intensiv die mathematische Eigenschaft, dass Null in der Addition ein neutrales Element ist. Diese Vorgehensweise ist bei der Interpretation der Summanden in den Gleichungen zu beachten. Die gewählte Darstellung vereinfacht jedoch den Algorithmus und ist prädestiniert für die angestrebte, mittels Software automatisierte Qualitätskostenermittlung.

A.2. Ermittlung testinduzierter Qualitätskosten

Die Modellbildung folgt der Struktur des Testablaufs und somit den in Kapitel 2.5 beschriebenen Eigenschaften und Abhängigkeiten der Kostenelemente.

Exakte retrospektive Berechnung: Es wird zunächst ein exakter Algorithmus für die retrospektive Berechnung von Qualitätskosten vorgestellt. Dieser Algorithmus stellt eine feingliedrige Summe über alle Kosten dar. Feingliedrig bedeutet, dass die angefallenen Kosten für jeden Testfall und jeden identifizierten Fehler explizit eingehen. Das Modell wird top-down von den Hauptkostengruppen zu den Kostenelementen verfeinert. Die Algorithmen sind dadurch gut überschaubar und lassen die logische Struktur der Berechnung und die zu betrachtenden Komponenten deutlich erkennen.

Testinduzierte Qualitätskosten: Die testinduzierten Qualitätskosten berechnen sich nach Kapitel 2.5 aus *Hersteller-* und *Kundeninduzierten Qualitätskosten* (s. Gleichung (A.1)).

$$K^{\textit{Testinduzierte Qualitätskosten}} = K^{\textit{Herstellerinduzierte Qualitätskosten}} + K^{\textit{Kundeninduzierte Qualitätskosten}} \tag{A.1}$$

Die *Herstellerinduzierten Qualitätskosten* umfassen alle Kosten aus dem Testmanagement und allen Testphasen TP_1 (Modul- und Systemtest inkl. Integrationstests) über alle Iterationen. Zu den *Kundeninduzierten Qualitätskosten* gehören alle beim Hersteller anfallenden Kosten der Testphasen TP_2 (Fahrzeugtest mit Integrationstest bis Feldeinsatz) über alle Iterationen (s. Kapitel 4.3.2).

Herstellerinduzierte Qualitätskosten: Die *Herstellerinduzierten Qualitätskosten* beinhalten die Kosten für die fünf Hauptkostengruppen *Testmanagement, Testaufbau, Testvorbereitung, Testdurchführung* und *Herstellerinduzierte Fehlerbeseitigung* (s. Abbildung 4.1). Da in der Automobilindustrie der Testprozess iterativ durchgeführt wird, sind die Hauptkostengruppen *Testvorbereitung, Testdurchführung* und *Herstellerinduzierte Fehlerbeseitigung* abhängig von Iterationen, Testphasen (der Menge TP_1) und den Testverfahren. Bei den Kosten des Testaufbaus und des Testmanagements sind diese Abhängigkeiten für die Kostenelemente nur bedingt gegeben und

erscheinen erst bei Modellverfeinerung. Die unterschiedlichen Ausprägungen der Laufvariablen h für die vier Hauptkostengruppen sind in der Gleichung (A.2) jeweils explizit angegeben (zur Definition der Indizes siehe Tabelle A.2).

$$
\begin{aligned}
K^{\textit{Herstellerinduzierte Qualitätskosten}} = \\
& K^{\textit{Testmanagement},\ h=1} \\
& + K^{\textit{Testaufbau},\ h=3} \\
& + \sum_{i}^{I^{h=4}} \sum_{p}^{P_i^{h=4}} \sum_{m}^{M_{ip}^{h=4}} K_{ipm}^{\textit{Testvorbereitung}} \\
& + \sum_{i}^{I^{h=5}} \sum_{p}^{P_i^{h=5}} \sum_{m}^{M_{ip}^{h=5}} K_{ipm}^{\textit{Testdurchführung}} \\
& + \sum_{i}^{I^{h=6}} \sum_{p}^{P_i^{h=6}} \sum_{m}^{M_{ip}^{h=6}} K_{ipm}^{\textit{Herstellerinduzierte Fehlerbeseitigung}}
\end{aligned}
\tag{A.2}
$$

Die Hauptkostengruppen aus Gleichung (A.2) werden in den nachfolgenden Kapiteln sukzessive verfeinert bis letztendlich monetär oder äquivalent messbare Kostenelemente erreicht werden. Zunächst wird hierbei für jeden Berechnungsalgorithmus die exakte Quantifizierung angegeben.

In den nachfolgenden Algorithmen werden die Indizes h der Hauptkostengruppen (z. B. I^h, P_i^h) in den Gleichungen der Übersichtlichkeit halber weggelassen.

A.2.1. Hauptkostengruppe *Testmanagement*

Die Kosten des *Testmanagement* sind laut Kapitel 2.5.1 zwar durch projektspezifisches Vorgehen mitbestimmt, aber vom Testprozess weitgehend entkoppelt. Das *Testmanagement* gliedert sich daher nicht in die durch die Laufvariablen i, p, m bestimmte Kostenhierarchie ein. Die Testmanagementkosten umfassen Kosten für *Testplanung* und *Verwaltung*:

$$
K^{\textit{Testmanagement}} = K^{\textit{Testplanung}} + K^{\textit{Verwaltung}} \tag{A.3}
$$

Die Kostengruppen *Testplanung* und *Verwaltung* sind stark projekt- und firmenabhängig und werden in dieser Arbeit daher nicht verfeinert. Aufgrund der geringen Abhängigkeit zu anderen Kostenelementen können jedoch leicht im Einzelfall fehlende Kostenelemente ergänzt werden.

A.2.2. Hauptkostengruppe *Testaufbau*

Die Kosten für die Hauptkostengruppe *Testaufbau* ($h = 3$) setzten sich aus zwei Kostengruppen zusammen:

$$K^{Testaufbau} = K^{Projektspezifische\ Ausführungsumgebung} + K^{Werkzeuge} \tag{A.4}$$

Projektspezifische Ausführungsumgebung: Die *Projektspezifische Ausführungsumgebung* umfasst nach Kapitel 2.5.1 die Kostenelemente *Prüfstand*, *Prüfling* sowie *Treiber und Platzhalter*.

$$\begin{aligned} K^{Projektspezifische\ Ausführungsumgebung} = \\ K^{Prüfstand} \\ + \sum_{i}^{I} \sum_{p}^{P_i} \left(K_{ip}^{Prüflinge} + K_{ip}^{Treiber\ und\ Platzhalter} \right) \end{aligned} \tag{A.5}$$

Prüfstand: Die Kosten für Prüfstände liegen in der Regel als Paketpreise direkt monetär in den Projektunterlagen vor und gehen in diesem Fall als Fixkosten in die Berechnung ein.

Prüflinge: Prüflinge sind im Falle einer kontinuierlichen Weiterentwicklung der Hardware je Iteration zu erstellen und können in Abhängigkeit zur Testphase unterschiedliche Komponenten (inkl. Hardware) umfassen. Beispielsweise könnten für einen Modultest eines Blinkers keine Leuchteinheiten benötigt werden, im System(integrations-)test dagegen schon. Somit gilt:

$$\begin{aligned} K_{ip}^{Prüflinge} = \\ K_{ip}^{Materialkosten\ für\ Prüflinge} \\ + A_{ip}^{Entwicklungsaufwand\ für\ Prüflinge} \cdot K^{Stundensatz\ eines\ Testers} \end{aligned} \tag{A.6}$$

Treiber und Platzhalter: Treiber und Platzhalter bezeichnen hier die notwendige Software, die zur Ausführung von Testfällen benötigt wird. In Abhängigkeit von Iteration i, Testphase p und Testverfahren m werden unterschiedliche Treiber und Platzhalter benötigt. So sind beispielsweise je nach Integrationsmethode, z. B. Bottom-Up- [176] oder Top-Down-Integrationsstrategien [176], unterschiedliche Treiber und Platzhalter zu erzeugen. Die Entwicklungskosten ergeben sich in den

Testphasen und Iterationen aus der Summe der bei den Testverfahren angefallenen Kosten:

$$K_{ip}^{\textit{Treiber und Platzhalter}} = \sum_{m}^{M_{ip}} \Big(A_{ipm}^{\textit{Entwicklungsaufwand für Treiber und Platzhalter}} \cdot K^{\textit{Stundensatz eines Testers}} \Big) \tag{A.7}$$

Werkzeuge: Die anfallenden Werkzeugkosten sind stark projekt- und firmenspezifisch. Die Kosten für die vier Hauptkostengruppen *Testmanagement*, *Testvorbereitung*, *Testdurchführung* und *Fehlerbeseitigung* werden getrennt aufgeführt:

$$K^{\textit{Werkzeuge}} = K^{\textit{Werkzeuge Testmanagement}} + K^{\textit{Werkzeuge Testvorbereitung}} + K^{\textit{Werkzeuge Testdurchführung}} + K^{\textit{Werkzeuge Fehlerbeseitigung}} \tag{A.8}$$

Die stark projekt- und firmenspezifisch bestimmten Werkzeugkosten werden am Beispiel der in dieser Arbeit untersuchten Projekte verfeinert. Dabei sind Lizenzkosten üblicherweise an die Zeitdauer der Verwendung gekoppelt. Daher wird jeweils die Zeitdauer als Bemessungsgrundlage für die Werkzeugkosten verwendet. In diesen Fällen geht, wie in nachfolgenden Beispielen, die Nutzungsdauer als Faktor in die Berechnung ein.

$$K^{\textit{Werkzeuge Testmanagement}} = K^{\textit{Jahreslizenzen für Projektplanungssoftware}} \cdot A^{\textit{Nutzungsdauer in Jahren, Projektplanungssoftware}} \tag{A.9}$$

$$
\begin{aligned}
K^{\textit{Werkzeuge Testvorbereitung}} = \\
& K^{\textit{Jahreslizenzen für Modellierungswerkzeuge}} \\
& \quad \cdot A^{\textit{Nutzungsdauer in Jahren, Modellierungswerkzeuge}} \\
& + K^{\textit{Jahreslizenzen für Testfallgeneratoren}} \\
& \quad \cdot A^{\textit{Nutzungsdauer in Jahren, Testfallgeneratoren}}
\end{aligned}
\tag{A.10}
$$

$$
\begin{aligned}
K^{\textit{Werkzeuge Testdurchführung}} = \\
& K^{\textit{Jahreslizenzen für Testfallautomatisierungswerkzeuge}} \\
& \cdot A^{\textit{Nutzungsdauer in Jahren, Testfallautomatisierungswerkzeuge}}
\end{aligned}
\tag{A.11}
$$

$$
\begin{aligned}
K^{\textit{Werkzeuge Fehlerbeseitigung}} = \\
& K^{\textit{Jahreslizenzen für Datenbanksystem}} \\
& \cdot A^{\textit{Nutzungsdauer in Jahren, Datenbanksystem}}
\end{aligned}
\tag{A.12}
$$

Bei den Werkzeugkosten wird die Möglichkeit des Anpassens an besondere Situationen als praxisrelevant angesehen.

A.2.3. Hauptkostengruppe *Testvorbereitung*

Die Hauptkostengruppe *Testvorbereitung* ($h = 4$) setzt sich aus der Summe von drei Kostengruppen zusammen. Für ein Testverfahren m in der Testphase p und der Iteration i gilt:

$$
K_{ipm}^{\textit{Testvorbereitung}} = K_{ipm}^{\textit{Modellbasierter Test}} + K_{ipm}^{\textit{Testfallgenerierung}} + K_{ipm}^{\textit{Testfallautomatisierung}} \tag{A.13}
$$

Die gesamten Kosten der Testvorbereitung des Projekts ergeben sich aus der Summe über die eingesetzten Testverfahren m, die Testphasen p und Iterationen i in Gleichung (A.2). Dies gilt entsprechend für alle nachfolgenden Kostengruppen mit diesen Indizes.

Modellbasierter Test: Beim modellbasierten Test müssen Modelle zum Zwecke der Testfallgenerierung erstellt bzw. angepasst werden. Die Kosten für die Modellerstellung sind abhängig vom Testverfahren m. Beispielsweise werden in strukturorientierten Testverfahren Datenflussgraphen oder Kontrollflussgraphen aufgestellt, in verhaltensorientierten Testverfahren Zustandsautomaten oder Sequenz-

diagramme. Das Aufstellen dieser verfahrenstypischen Darstellungen in der Testphase p der Iteration i erzeugt einen verfahrensabhängigen Personalaufwand:

$$K_{ipm}^{\textit{Modellbasierter Test}} = A_{ipm}^{\textit{Modellerstellung}} \cdot K^{\textit{Stundensatz eines Testers}} \tag{A.14}$$

Über Iterationen hinweg betrachtet, können Modelle wiederverwendet bzw. erweitert werden. Daher sind nicht in allen Iterationen die Kosten einer vollständigen Modellerstellung anzusetzen.

Testfallgenerierung: Die angewendeten Testverfahren bestimmen Anzahl, Länge und Komplexität der Testfälle, die in testverfahrenspezifischen Testreihen gruppiert sind. Eine Testreihe kann beispielsweise die Überdeckung einer Bedingung (z. B. MC/DC [52, 176, 241]) sicherstellen. Das Testverfahren bestimmt maßgeblich den Testfallgenerierungsaufwand [84, 224]. Die Erstellung eines MC/DC-Testfalls kann z. B. im Hinblick auf den Aufwand verschieden sein von der Testfallgenerierung einer Graphenüberdeckung [163, 206, 241].

Genau betrachtet, besitzt jeder Testfall einen spezifischen Aufwand zur Generierung. Testfallgeneratoren können diese Testfallgenerierungskosten reduzieren. Je nach Teststrategie entsteht für jeden Testfall ein Aufwand für eine manuelle oder automatisierte Testfallgenerierung. Eine algorithmische Unterscheidung zwischen automatisiert und manuell ausgeführten Generierungen ist aufgrund der Einzelbetrachtung nicht notwendig

Wird mit Projektfortschritt auf bereits in früheren Iterationen erstellte Testfälle zurückgriffen, so sind nur die Generierungskosten der jeweils neu erstellten Testfälle zu berücksichtigen. Testfälle, die in vorangegangen Iterationen erstellt wurden und wiederverwendet werden sollen, bedürfen ggf. einer Anpassung, um die Ausführbarkeit zu gewährleisten. Die Testfallgenerierungs- und -anpassungskosten für alle Testfälle t aus allen Testreihen r der Methode m in Testphase p und Iteration i berechnen sich wie folgt:

$$\begin{aligned} K_{ipm}^{\textit{Testfallgenerierung}} = & \\ & \sum_{r}^{R_{ipm}} \sum_{t}^{T_{ipmr}} \Big(A_{ipmrt}^{\textit{Manuelle oder Automatisierte Testfallerstellung}} \\ & \qquad + A_{ipmrt}^{\textit{Testfallanpassung}} \Big) \\ & \cdot K^{\textit{Stundensatz eines Testers}} \end{aligned} \tag{A.15}$$

Die ausschließliche Einbeziehung von neu entstandenem Aufwand wird in der Gleichung (A.15) durch Nutzung des neutralen Null-Elementes sichergestellt.

Testfallautomatisierung: Aus den erstellten Testfällen werden i. Allg. gemäß der Teststrategie, Testfälle zur automatisierten Testfallausführung vorbereitet. Zur exakten Quantifizierung der Testfallautomatisierung ist für jeden zu automatisierenden Testfall dessen Automatisierungsaufwand zu betrachten. In der Regel wird lediglich eine Auswahl der Testfälle eines Testverfahrens automatisiert getestet. Nur für diese Testfälle liegen von Null abweichende Werte vor.

In iterativen Entwicklungen können automatisierte Testfälle wiederverwendet werden. Werden nach Systemänderungen (iterative Weiterentwicklungen, Fehlerkorrekturen) Testfalländerungen erforderlich (s. Gleichung (A.15)), sind gewöhnlich auch die Automatisierungen der Testfälle anzupassen. Hierfür sind Kosten analog zu den Testfallanpassungskosten zu veranschlagen. Für die in der Testphase p der Iteration i eingesetzten Testverfahren m gilt:

$$
\begin{aligned}
K_{ipm}^{\textit{Testfallautomatisierung}} = & \\
& \sum_{r}^{R_{ipm}} \sum_{t}^{T_{ipmr}} \Big(A_{ipmrt}^{\textit{Vorbereitung zur automatischen Ausführung}} \\
& \qquad + A_{ipmrt}^{\textit{Anpassung der Automatisierung}} \Big) \\
& \cdot K^{\textit{Stundensatz eines Testers}}
\end{aligned}
\tag{A.16}
$$

A.2.4. Hauptkostengruppe *Testdurchführung*

Für die Berechnung der Kosten der *Testdurchführung* ($h = 5$) für das Testverfahren m in der Testphase p der Iteration i gilt:

$$
K_{ipm}^{\textit{Testdurchführung}} = K_{ipm}^{\textit{Testfallausführung}} + K_{ipm}^{\textit{Fehleridentifikation}} \tag{A.17}
$$

Testfallausführung: Der Aufwand für Testfallausführungen hängt vom Testverfahren und der Testüberdeckungsanforderung als Testfallanzahl bestimmende Faktoren ab [163, 206]. Eine formale Unterscheidung nach manuell und automatisiert ausgeführten Testfällen ist bei der exakten Berechnung nicht erforderlich, da der Aufwand eines jeden Testfalls eingeht. Die exakten Kosten der Testfallausführung für das Testverfahren m in der Testphase p der Iteration i können aus den Ausführungskosten eines jedes Testfalls t der Testreihen r ermittelt werden:

$$K_{ipm}^{\textit{Testfallausführung}} = \sum_{r}^{R_{ipm}} \sum_{t}^{T_{ipmr}} A_{ipmrt}^{\textit{Manuelle oder Automatisierte Testfallausführung}} \cdot K^{\textit{Stundensatz eines Testers}} \tag{A.18}$$

Fehleridentifikation: Die Fehleridentifikation bezieht sich auf die Identifikation von Fehlverhalten und umfasst den Personalaufwand des Abgleichs der im Test beobachteten mit den erwarteten Ergebnissen. Das Resultat dieser Aktion ist die mögliche Identifikation eines Fehlverhaltens, das prozesstechnisch eine Fehlerbeseitigung anstoßen könnte. In einer präzisen Aufwandserfassung kann jedem ausgeführten Testfall die Zeit für den Ergebnisvergleich zugeordnet werden. Die Summe repräsentiert in analoger Weise zu Gleichung (A.18) den Aufwand der Fehleridentifikation. Eine Unterscheidung zwischen automatisiert und manuell ausgeführten Vergleichen ist aufgrund der Einzelbetrachtung nicht notwendig.

$$K_{ipm}^{\textit{Fehleridentifikation}} = \sum_{r}^{R_{ipm}} \sum_{t}^{T_{ipmr}} A_{ipmrt}^{\textit{Manuelle oder Automatisierte Testauswertung}} \cdot K^{\textit{Stundensatz eines Testers}} \tag{A.19}$$

A.2.5. Hauptkostengruppe *Herstellerinduzierte Fehlerbeseitigung*

In der *Herstellerinduzierten Fehlerbeseitigung* ($h = 6$) werden die Kosten für *Fehlermanagement, Fehlerbearbeitung* und *Korrekturtest* für die Testphasen während der Entwicklung (TP_1) betrachtet.

Die exakte Berechnung der Fehlerbeseitigungskosten verwendet die konkret für jede Fehlermeldung angefallenen Kosten. Die Kosten des *Fehlermanagements* sind hierbei abhängig von der Anzahl der vorliegenden Fehlermeldungen D_{ipmr}, die Kosten der *Fehlerbearbeitung* und der *Fehlerkorrektur* von der Anzahl der akzeptierten Fehlermeldungen F_{ipmr}. Die Ausprägungen (Fälle) des Fehlverhaltens D_{ipmr} und die Fehlermeldungen F_{ipmr} stehen hierarchisch in der Testreihe. Es wird hier angenommen, dass jede akzeptierte[9] Fehlermeldung f eine Fehlerbeseitigung nach sich zieht:

[9] Wird in der Fehleranalyse eine Kosten-Nutzen-Betrachtung zur Fehlerbeseitigung durchgeführt und beschlossen, den Fehler zu beseitigen, geht die Fehlermeldung in den Zustand „akzeptiert“über (s. Kapitel 2.3). Duplikate, Falschmeldungen und für eine Beseitigung zu aufwändig eingestufte und andere abgelehnte Fehlermeldungen zählen somit nicht zu den akzeptierten Fehlermeldungen (siehe Beschreibung Lebenszyklus einer Fehlermeldung in Kapitel 2.3). Folglich wird angenommen, dass jede akzeptierte Fehlermeldung auch beseitigt wird.

$$K_{ipm}^{\textit{Herstellerinduzierte Fehlerbeseitigung}} = K_{ipm}^{\textit{Fehlermanagement}} + K_{ipm}^{\textit{Fehlerbearbeitung}} + K_{ipm}^{\textit{Korrekturtest}} \tag{A.20}$$

$$K_{ipm}^{\textit{Fehlermanagement}} = \sum_{r}^{R_{ipm}} \sum_{d}^{D_{ipmr}} K_{ipmrd}^{\textit{Fehlermanagement}} \tag{A.21}$$

$$K_{ipm}^{\textit{Fehlerbearbeitung}} = \sum_{r}^{R_{ipm}} \sum_{f}^{F_{ipmr}} K_{ipmrf}^{\textit{Fehlerbearbeitung}} \cdot K^{\textit{Stundensatz eines Testers}} \tag{A.22}$$

$$K_{ipm}^{\textit{Korrekturtest}} = \sum_{r}^{R_{ipm}} \sum_{f}^{F_{ipmr}} K_{ipmrf}^{\textit{Korrekturtest}} \tag{A.23}$$

Fehlermanagement: Die Kosten für das Management des Fehlverhaltens d durch Fehlermeldungen in einer Testreihe r sind:

$$K_{ipmrd}^{\textit{Fehlermanagement}} = A_{ipmrd}^{\textit{Datenpflege des identifizierten Fehlverhaltens in der Fehlerdatenbank}} \cdot K^{\textit{Stundensatz eines Testers}} \tag{A.24}$$

Fehlerbearbeitung: Die *Fehlerbearbeitung* für jede akzeptierte Fehlermeldung f setzt sich aus *Fehleranalyse* und *Fehlerkorrektur* zusammen. Die exakte Berechnung summiert den jeweils fehlerspezifischen Aufwand:

$$K_{ipmrf}^{\textit{Fehlerbearbeitung}} = \left(A_{ipmrf}^{\textit{Analyse eines Fehlverhaltens}} + A_{ipmrf}^{\textit{Korrektur einer Fehlerursache}} \right) \cdot K^{\textit{Stundensatz eines Testers}} \tag{A.25}$$

Korrekturtest: Der *Korrekturtest* dient der Verifizierung der Fehlerkorrektur durch Tests. Hierzu werden einerseits die ein Fehlverhalten identifizierenden Testfälle (Verifikation der Fehlerbeseitigung) und andererseits die Regressionstestfälle ausgeführt. Die Kosten des *Korrekturtests* ergeben sich wie folgt:

$$K_{ipmrf}^{\textit{Korrekturtest}} = K_{ipmrf}^{\textit{Verifikation der Fehlerbeseitigung}} + K_{ipmrf}^{\textit{Regressionstest}} \tag{A.26}$$

Verifikation der Fehlerbeseitigung: Zur Verifikation einer einzelnen Fehlerbeseitigung werden alle ein Fehlverhalten identifizierenden Testfälle u dieser Fehlerursache ausgeführt: Dies ist zunächst der ein Fehlverhalten identifizierende Testfall, der durch die betrachtete Fehlermeldung beschrieben ist. Zusätzlich werden die Testfälle aller der Fehlermeldung zugeordneten Duplikatsfehlermeldungen ausgeführt, da diese dieselbe Fehlerursache identifizierten und nach der Korrektur fehlerfrei durchlaufen müssten. Die Anzahl der ein Fehlverhalten identifizierenden Testfälle zu einer bearbeiteten Fehlermeldung f wird in Gleichung (A.27) durch U_{ipmrf} in der Summe

$$\sum_{u}^{U_{ipmrf}}$$

dargestellt.

Die für eine Testdurchführung anfallenden Kosten wurden bereits in der Hauptkostengruppe *Testdurchführung* (s. Anhang A.2.4) beschrieben. Identifiziert einer der zur Verifikation ausgeführten Testfälle ein Fehlverhalten, so ist die Fehlerbeseitigung als nicht erfolgreich anzusehen und eine erneute Fehlerbeseitigung ist durchzuführen.

Vor der Ausführung der fehleridentifizierenden Testfälle können zusätzliche Kosten zur *Testvorbereitung* entstehen, falls durch die Fehlerbeseitigung eine erneute Testfallgenerierung notwendig wird (s. Anhang A.2.3).

Für die *Verifikation der Fehlerbeseitigung* ergibt sich inkl. *Testvorbereitung* somit folgender indirekt rekursiver Berechnungsalgorithmus:

$$K_{ipmrf}^{\textit{Verifikation einer Fehlerbeseitigung}} = \sum_{u}^{U_{ipmrf}} \left(K_{ipmrfu}^{\textit{Testvorbereitung}} + K_{imprfu}^{\textit{Testdurchführung}} \right) \tag{A.27}$$

Der Berechnungsalgorithmus in Gleichung (A.27) weist eine zu Anhang A.2.4 abweichende Struktur der Laufvariablen für $K^{\textit{Testdurchführung}}$ auf (*ipmrfu* anstelle von

imp). Die Abweichung ist zu so interpretieren, dass *ipmrfu* eine Spezialisierung, also eine Teilmenge der Laufvariablen *ipm* darstellt und folglich Gleichung (A.17) zur Berechnung der *Testdurchführung* eingesetzt werden kann. Die Gleichung nutzt für die exakte Berechnung die Null-Definition (neutrales Element) für die Kosten der Testvorbereitung.

Regressionstest: Der Regressionstest zu einer Fehlerbeseitigung umfasst die Ausführung der kompletten Testreihen, die ein Fehlverhalten identifizierende Testfälle enthalten. Im Falle einer erneuten Fehleridentifikation ist eine erneute Fehlerbeseitigung durchzuführen. Im Anschluss an die erneute Fehlerdokumentation, -analyse und -korrektur folgt wiederum ein Korrekturtest mit Regressionstest. Folglich ist auch die Kostenfunktion des Regressionstests indirekt rekursiv und kann aufgrund der Hierarchie der Laufvariablen einfach durch $K_{ipmrt}^{\textit{Testdurchführung}}$ angegeben werden. Analog zur Verifikation der Fehlerbeseitigung können vor der Ausführung der Regressionstests zusätzliche Kosten zur Testvorbereitung entstehen.

Je nach dem Vorgehen in der Praxis, können die Testreihen mit einem ein Fehlverhalten identifizierenden Testfall (a) für jede Fehlermeldung oder (b) einmal nach Abschluss einer Projektphase durchgeführt werden. Die Kosten der unterschiedlichen Strategien können mit derselben Gleichung (A.28) beschrieben werden. Allerdings fallen die so beschriebenen Kosten aufgrund des unterschiedlichen Vorgehens verschieden häufig an:

$$
\begin{aligned}
K_{ipm(rf)}^{\textit{Regressionstest (für eine Fehlerbeseitigung)}} = \\
&\sum_{r}^{R_{ipm}} DV_{r(f)}^{\textit{Fehlverhalten identifizierender Testfall}} \\
&\cdot \sum_{t}^{T_{ipmr}} \left(K_{ipmrt}^{\textit{Testvorbereitung}} + K_{ipmrt}^{\textit{Testdurchführung}} \right)
\end{aligned}
\tag{A.28}
$$

Mit DV_r = 1, falls Testreihe r einen ein Fehlverhalten identifizierenden Testfall zur Fehlerursache der Fehlermeldung f enthält, sonst DV_r = 0.

Gleichung (A.28) ordnet sich auf Testverfahrensebene in die Laufzahlhierarchie ein. Für den Regressionstest am Ende einer Projektphase (Fall (b)) ist die Betrachtung auf Testverfahrensebene evident. Bei Ausführung von Regressionstest nach jeder Fehlerbeseitigung (Fall (a)) ist zu beachten, dass nicht nur die Testreihe des ein Fehlverhalten identifizierenden Testfalls ausgeführt wird, sondern auch die aller Duplikatsfehlermeldungen, die ggf. anderen Testverfahren und Testreihen zugeordnet sein können. Zur strikten Einhaltung der Laufzahlhierarchie muss daher eine Betrachtung auf Testverfahrensebene angewendet werden.

A.2.6. Qualitätskostenkategorie *Kundeninduzierte Qualitätskosten*

Die *Kundeninduzierten Qualitätskosten* setzen sich zusammen aus *Kundeninduzierten Fehlerbeseitigungskosten* und *Fehlerfolgekosten*. *Fehlerfolgekosten* werden im Qualitätskostenmodell, wie in Kapitel 4.3.2 erwähnt, nicht berücksichtigt.

Die Berechnung der *Kundeninduzierten Fehlerbeseitigungskosten* erfolgt analog zu den *Herstellerinduzierten Fehlerbeseitigungskosten*. Es werden jedoch die Testphasen TP_2, z. B. Akzeptanztest, betrachtet sowie der Hauptkostengruppenindex h = 7 verwendet. Werden vom Kunden keine das Fehlverhalten identifizierenden Testfälle und Testreihen übergeben, wird zumindest *ein* fehleridentifizierender Testfall im Zuge der Fehlerreproduktion erstellt. Ebenso liegt nicht zwingend die Angabe eines Testverfahrens vor bzw. es wurde keines eingesetzt. Da hier jeweils der Endindex 1 für erforderliche Laufzahlen gewählt werden kann, gelten ebenfalls die in Anhang A.2.5 für die Fehlerbeseitigung dargestellten Berechnungen.

A.3. Vereinfachte Berechnung der testinduzierten Qualitätskosten

Die in den vorangegangenen Unterkapiteln vorgestellten, exakten retrospektiven Berechnungen erfordern eine nicht praxiskonform detaillierte Datenbasis und beinhalten einen erheblichen Berechnungsaufwand. Daher werden im Folgenden anstelle der exakten retrospektiven Berechnungen vereinfachte Quantifizierungsalgorithmen hergeleitet, die auf stärker aggregierten Daten basieren. Das Aggregationsniveau wird anhand der entscheidenden strukturellen und kostenbestimmenden Einheiten des Testprozesses festgelegt. Der Begriff *vereinfacht* ist hier als Bezeichnung für Abstraktion bzw. als Synonym für *verallgemeinert* anzusehen.

Die Herleitung der vereinfachten Algorithmen wird hier anhand der retrospektiv vereinfachten Berechnung der Kosten der Testfallgenerierung (retrospektiv exakt in Gleichung (A.29)) erläutert.

Testfallgenerierung (retrospektiv vereinfacht): Der Algorithmus (A.29) stellt je Verfahren den Erstellungsaufwand für die Testfälle dar als ein Produkt aus Testfallanzahl und mittlerem Aufwand für die Erstellung eines Testfalls. Die getrennte Betrachtung der Verfahren wird beibehalten, da – wie bereits erwähnt – das Testverfahren maßgeblich den Testfallgenerierungsaufwand bestimmt [84, 224]. $G_{ipm}^{\textit{Automatisierte Testfallerstellung}}$ gibt den Anteil automatisiert erstellter Testfälle in der Testvorbereitung ($h = 4$) an. Der Aufwand für Testfallanpassungen wird mit Hilfe der Anzahl wiederverwendeter Testfälle, des Anteils von Testfallanpassungen $G_{ipm}^{\textit{Anpassung}}$ sowie des mittleren Aufwands für eine Anpassung dargestellt. Es ergibt sich dadurch:

$$
\begin{aligned}
K_{ipm}^{\textit{Testfallgenerierung}} = \\
&\Big(N_{ipm}^{\textit{Anzahl neuer Testfälle}} \\
&\cdot\Big(G_{ipm}^{\textit{Automatisierte Testfallerstellung}} \\
&\qquad\cdot A_{ipm}^{\textit{Mittlerer Aufwand für die automatisierte Erstellung eines Testfalls}} \\
&\quad+\Big(1 - G_{ipm}^{\textit{Automatisierte Testfallerstellung}}\Big) \\
&\qquad\cdot A_{ipm}^{\textit{Mittlerer Aufwand für die manuelle Erstellung eines Testfalls}}\Big) \\
&+N_{ipm}^{\textit{Anzahl wiederverwendeter Testfälle}} \\
&\quad\cdot G_{ipm}^{\textit{Anpassung}} \\
&\quad\cdot A_{ipm}^{\textit{Mittlerer Aufwand für die Anpassung eines Testfalls}}\Big) \\
&\cdot K^{\textit{Stundensatz eines Testers}}
\end{aligned}
\tag{A.29}
$$

Mittlerer Aufwand bei retrospektiver, vereinfachter Berechnung: Der mittlere Aufwand steht hier und im Folgenden bei retrospektiven Betrachtungen stellvertretend für den arithmetischen Mittelwert. Die Anwendung des arithmetischen Mittelwertes ermöglicht ein exaktes Berechnungsergebnis.

Liegen in der Praxis die für die retrospektiv vereinfachte Berechnung vorgesehenen Werte nicht vor, kann u. U. auf vorhandene Werte in höherer Aggregation zurückgegriffen werden. Hieraus können ebenfalls exakte Ergebnisse erzielt werden.

Weitere retrospektive, vereinfachte Algorithmen: Nach gleichen Grundgedanken lassen sich vereinfachte, exakte Algorithmen für die Testfallautomatisierung, die Testfallausführung, die Fehlererkennung und das Fehlermanagement formulieren.

A.3.1. Prospektiv geschätzte testinduzierte Qualitätskosten

Prospektive Qualitätskosten sind immer geschätzte Größen. Die Abschätzung erfolgt über die vereinfachten Algorithmen unter Verwendung der nachfolgend erläuterten Werte.

Abschätzungen für mittleren Aufwand, Anzahlen und Anteile: Bei prospektiven Abschätzungen können der mittlere Aufwand, Anzahlen und Anteile aufgrund von Verteilungsüberlegungen aus statistischen Kennzahlen (z. B. arithmetischer Mittelwert, Median) abgeleitet werden oder auf Basis von Erfahrungen geschätzt werden (zur Güte von Schätzwerten s. Kapitel 5.3).

Testfallgenerierung (prospektiv): Für die *Testfallgenerierung* ergibt sich somit nachfolgende Berechnung, die unter Verwendung von Schätzwerten auf Gleichung (A.29) aufbaut. $\widehat{G}_{ipm}$ gibt hierbei den geschätzten Anteil automatisiert erstellter

Testfälle in der Testvorbereitung ($h = 4$) an. Der Aufwand für Testfallanpassungen wird mit Hilfe der geschätzten mittleren Anzahl wiederverwendeter Testfälle $\widehat{N}_{ipm}^{\textit{Anzahl wiederverwendeter Testfälle}}$ – hier ist von den für die jeweiligen Verfahren typischen Anzahlen auszugehen –, des geschätzten mittleren Anpassungsgrad $\widehat{G}_{ipm}^{\textit{Anpassung}}$ für Testfallanpassungen sowie des mittleren Aufwands für eine Anpassung dargestellt. Der Anpassungsgrad $\widehat{G}_{ipm}^{Anpassung}$ bezeichnet den Anteil der bereits automatisierten, wiederverwendeten Testfälle, die angepasst werden müssen. Es ergibt sich dadurch:

$$
\begin{aligned}
\widehat{K}_{ipm}^{\textit{Testfallgenerierung}} = & \\
& \Big(\widehat{N}_{ipm}^{\textit{Anzahl neuer Testfälle}} \\
& \cdot \Big(\widehat{G}_{ipm} \cdot \widehat{A}_{ipm}^{\textit{Mittlerer Aufwand für die automatisierte Erstellung eines Testfalls}} \\
& \qquad + \Big(1 - \widehat{G}_{ipm} \Big) \cdot \widehat{A}_{ipm}^{\textit{Mittlerer Aufwand für die manuelle Erstellung eines Testfalls}} \Big) \\
& + \widehat{N}_{ipm}^{\textit{Anzahl wiederverwendeter Testfälle}} \\
& \qquad \cdot \widehat{G}_{ipm}^{\textit{Anpassung}} \\
& \qquad \cdot \widehat{A}_{ipm}^{\textit{Mittlerer Aufwand für die Anpassung eines Testfalls}} \Big) \\
& \cdot K^{\textit{Stundensatz eines Testers}}
\end{aligned}
\tag{A.30}
$$

A.3.2. Hauptkostengruppe *Testvorbereitung*

Testfallautomatisierung (retrospektiv vereinfacht): Die vereinfachte Berechnung in Gleichung (A.31) ermittelt die Kosten der *Testfallautomatisierung* retrospektiv vereinfacht je Testverfahren, da die Testfallgenerierungswerkzeuge üblicherweise auf Verfahrensebene arbeiten. Anstelle des testfallbezogenen Aufwands wird für Testverfahren m der Testfallautomatisierungsaufwand durch einen mittleren [10] Aufwand für einen Testfall ersetzt. Weiterhin werden der Testfallautomatisierungsgrad $G_{ipm}^{\textit{Testfallautomatisierung}}$ (mit Wert zwischen 0 und 1) des Testverfahrens und die Testfallanzahl herangezogen. Der Aufwand für die Anpassung der Automatisierung in Folge von Testfallanpassungen wird analog zu Gleichung (A.29) ermittelt. Für die Testphase p der Iteration i des betrachteten Testverfahren m gilt:

[10] für die retrospektive exakte Berechnung ist dies das arithmetische Mittel, für prospektive oder andere abschätzende Berechnungen sind dies Erfahrungswerte aus vorangegangen Projekten.

$$
\begin{aligned}
K_{ipm}^{\textit{Testfallautomatisierung}} = & \\
& \Big(N_{ipm}^{\textit{Testfallanzahl}} \cdot A_{ipm}^{\textit{Mittlerer Automatisierungsaufwand für einen Testfall}} \\
& + N_{ipm}^{\textit{Anzahl wiederverwendeter Testfälle}} \cdot G_{ipm}^{\textit{Anpassung}} \\
& \quad \cdot A_{ipm}^{\textit{Mittlerer Aufwand für die Anpassung eines Testfalls}} \Big) \\
& \cdot G_{ipm}^{\textit{Testfallautomatisierung}} \cdot K^{\textit{Stundensatz eines Testers}}
\end{aligned}
\tag{A.31}
$$

A.3.3. Hauptkostengruppe *Testdurchführung*

Testfallausführung (retrospektiv vereinfacht): Die Testautomatisierung besitzt einen großen Einfluss auf die Kosten der *Testfallausführung*. Der Aufwand wird daher differenziert betrachtet, und zwar für manuelle und automatisierte Testfallausführungen getrennt, gegliedert nach Testverfahren m in der Testphase p der Iteration i. Es fließen ein:

- Automatisierungsgrad $G_{ipm}^{\textit{Automatisierte Testausführung}}$ mit Wert zwischen 0 und 1,
- der mittlere[11] Testfallausführungsaufwand je Testverfahren (getrennt für manuelle und automatisierte Ausführung) und
- die Testfallanzahl je Verfahren (analog zu Gleichung (A.29))

$$
\begin{aligned}
K_{ipm}^{\textit{Testfallausführung}} = & \\
& N_{ipm}^{\textit{Testfallanzahl}} \\
& \cdot \Big(G_{ipm}^{\textit{Automatisierte Testfallausführung}} \\
& \quad \cdot A_{ipm}^{\textit{Mittlerer Aufwand für die automatisierte Ausführung eines Testfalls}} \\
& + \Big(1 - G_{ipm}^{\textit{Automatisierte Testfallausführung}} \Big) \\
& \quad \cdot A_{ipm}^{\textit{Mittlerer Aufwand für die manuelle Ausführung eines Testfalls}} \Big) \\
& \cdot K^{\textit{Stundensatz eines Testers}}
\end{aligned}
\tag{A.32}
$$

Fehleridentifikation (retrospektiv vereinfacht): Der Aufwand eines manuellen Vergleichs kann sich stark von dem eines automatisierten unterscheiden. Daher wird analog zu Gleichung (A.32) getrennt berechnet:

[11] Bei der retrospektiven Betrachtung wird für den mittleren Aufwand der arithmetische Mittelwert gesetzt.

$$
\begin{aligned}
K_{ipm}^{\textit{Fehleridentifikation}} = \\
& N_{ipm}^{\textit{Testfallanzahl}} \\
& \cdot \Big(G_{ipm}^{\textit{Automatisierte Testausführung}} \\
& \qquad \cdot A_{ipm}^{\textit{Mittlerer Aufwand für automatisierten Ergebnisvergleich für einen Testfall}} \\
& + \Big(1 - G_{ipm}^{\textit{Automatisierte Testausführung}} \Big) \\
& \qquad \cdot A_{ipm}^{\textit{Mittlerer Aufwand für manuellen Ergebnisvergleich für einen Testfall}} \Big) \\
& \cdot K^{\textit{Stundensatz eines Testers}}
\end{aligned}
\tag{A.33}
$$

A.3.4. Hauptkostengruppe *Herstellerinduzierte Fehlerbeseitigung*

Fehlermanagement (retrospektiv vereinfacht): Eine retrospektive, exakte Kostenbetrachtung kann auf Basis der Anzahl der in TP_1 identifizierten Fälle von Fehlverhalten und über einen mittleren Kostenaufwand für das Verwalten der dazugehörigen Fehlermeldung in der Fehlerdatenbank ermittelt werden:

$$
\begin{aligned}
K_{ipm}^{\textit{(retrospektiv) Fehlermanagement}} = \\
& N_{ipm}^{\textit{Fehlermeldungen in der Fehlerdatenbank (in } TP_1\textit{)}} \\
& \cdot A^{\textit{Mittlerer Aufwand für Datenpflege einer Fehlermeldung in der Fehlerdatenbank}} \\
& \cdot K^{\textit{Stundensatz eines Testers}}
\end{aligned}
\tag{A.34}
$$

mit

$$
N_{ipm}^{\textit{Fehlermeldungen in der Fehlerdatenbank}} = \sum_{r}^{R_{ipm}} \sum_{d}^{D_{ipmr}} 1 \tag{A.35}
$$

Fehlermanagement (prospektiv geschätzt): Die prospektive Abschätzung des Fehlermanagements geht zunächst von einem mittleren Aufwand je akzeptierter Fehlermeldungen aus. Hinzu kommt der Aufwand für das Verwalten von Duplikatsfehlermeldungen sowie für abgelehnte Fehlermeldungen. Diese erhöhen im vorliegenden Modell als Korrekturaufschläge die Anzahl der akzeptierten Fehlermeldungen, indem definiert wird, welchen Anteil Duplikatsfehlermeldungen bzw. abgelehnte Fehlermeldungen an den akzeptierten Fehlermeldungen erfahrungsgemäß haben. Wurden in abgeschlossenen Projekten beispielsweise einhundert Duplikatsfehlermeldungen pro eintausend akzeptierter Fehlermeldungen in Fehlerdatenbanken eingetragen, so wird als Korrekturaufschlag ein Duplikatsanteil (Duplikatsfehlermeldungen geteilt durch akzeptierte Fehlermeldungen) gleich 0,1 angenommen.

Entsprechend ist der Anteil der abgelehnten Fehlermeldungen abzuschätzen. Die Anteile werden durch $\widehat{G}^{\textit{Anteil der Duplikatsfehlermeldungen an den akzeptierten Fehlermeldungen}}$ und $\widehat{G}^{\textit{Anteil der abgelehnten an den akzeptierten Fehlermeldungen}}$ abgekürzt und können auch Werte größer eins annehmen. Die Anteile der Duplikatsfehlermeldungen und abgelehnten Fehlermeldungen werden summiert und mit dem Aufwand für das Verwalten von Fehlermeldungen multipliziert (zweiter Teil von Gleichung (A.36)). Es wird also vereinfachend von einer Unterscheidung des Aufwands für Duplikate und abgelehnte Fehlermeldungen abgesehen, da der Aufwand des Fehlermanagements gemessen an den Projektkosten sehr gering ist.

Dieser Aufwand kann im Gegensatz zur *retrospektiven* Quantifizierung nach (A.34) für die prospektive Quantifizierung, orientiert an den zur Verfügung stehenden Daten, als unabhängig von Iteration, Testphase und Testverfahren angenommen werden, auch wenn der Algorithmus Differenzierungsmöglichkeiten bietet. Aus dem mittleren Aufwand für die Datenpflege eines Fehlers in der Fehlerdatenbank und der Korrekturaufschläge kann definiert werden:

$$
\begin{aligned}
\widehat{K}^{\textit{Fehlermanagement (für eine Fehlerbeseitigung)}} = \\
&\cdot \widehat{A}^{\textit{Mittlerer Aufwand zur Datenpflege für eine Fehlermeldung in der Fehlerdatenbank}} \\
&\cdot K^{\textit{Stundensatz eines Testers}} \\
&\cdot \Big(1 + \widehat{G}^{\textit{Anteil der Duplikatsfehlermeldungen an den akzeptierten Fehlermeldungen}} \\
&+ \widehat{G}^{\textit{Anteil der abgelehnten an den akzeptierten Fehlermeldungen}}\Big)
\end{aligned}
\tag{A.36}
$$

Fehlerbearbeitung (prospektiv geschätzt): Die prospektive Fehlerbearbeitung ergibt sich analog zu Gleichung (A.25) aus der Summe der Abschätzungen zur Fehleranalyse und Fehlerkorrektur. Hier sieht der Algorithmus in Gleichung (A.37) vor, den Aufwand in Anhängigkeit von Fehlerklassen zu differenzieren, da eine Abhängigkeit des Fehlerbeseitigungsaufwands von der Fehlerklasse anzunehmen ist [122, 204, 252].

Die fehlerklassennormierten Kosten für die Beseitigung einer Fehlerursache berechnen sich wie folgt:

$$
\begin{aligned}
\widehat{K}_{ipm}^{\textit{Fehlerbearbeitung (für eine Fehlerbeseitigung)}} = \\
&\sum_{e}^{E_{ipm}} \Big(\widehat{G}_{ipme}^{\textit{Anteil akzeptierter Fehlermeldungen in der effektiv identifizierten Fehlerklasse e}} \\
&\cdot \Big(\widehat{A}_{ipme}^{\textit{Analyse eines Fehlverhaltens}} + \widehat{A}_{ipme}^{\textit{Korrektur einer Fehlerursache}} \Big) \Big) \\
&\cdot K^{\textit{Stundensatz eines Testers}}
\end{aligned}
\tag{A.37}
$$

mit

$$\sum_{e}^{E_{ipm}} \widehat{G}_{ipme}^{\textit{Anteil akzeptierter Fehlermeldungen in der effektiv identifizierten Fehlerklasse e}} = 1 \tag{A.38}$$

Liegen keine detaillierten Erfahrungen vor, können – wie in Kapitel 5.2.1 nachgewiesen – die Fehlerbeseitigungskosten in Abhängigkeit der Testphase der Fehleridentifikation eingesetzt werden (Gleichung (A.39)). Die Kosten für die Beseitigung einer Fehlerursache sind entsprechend Gleichung (A.20) auf Methodenebene zu betrachten. Für testphasenspezifische Fehlerbeseitigungskosten ist dies nicht zwingend notwendig. Der Anteil der akzeptierter Fehlermeldungen pro Testmethode ($\widehat{G}_{ipm}^{\textit{Anteil akzeptierter Fehlermeldungen in der Testmethode m}}$) dient hier als Hilfskonstrukt zur Eingliederung in die ipm-Hierarchie:

$$\begin{aligned}\widehat{K}_{ipm}^{\textit{Fehlerbearbeitung (für eine Fehlerbeseitigung)}} = \\ & \widehat{G}_{ipm}^{\textit{Anteil akzeptierter Fehlermeldungen in der Testmethode m}} \\ & \cdot\left(\widehat{A}_{ip}^{\textit{Analyse eines Fehlverhaltens}} + \widehat{A}_{ip}^{\textit{Korrektur einer Fehlerursache}}\right) \\ & \cdot K^{\textit{Stundensatz eines Testers}}\end{aligned} \tag{A.39}$$

mit

$$\sum_{m}^{M_{ip}} \widehat{G}_{ipm}^{\textit{Anteil akzeptierter Fehlermeldungen in der Testmethode m}} = 1 \tag{A.40}$$

Korrekturtest (prospektiv geschätzt): Der prospektive Korrekturtest ergibt sich analog zu (A.26) aus der Summe der Abschätzungen zur Verifikation der Fehlerbeseitigung und dem Regressionstest:

$$\begin{aligned}\widehat{K}_{ipm}^{\textit{Korrekturtest (für eine Fehlerbeseitigung)}} = \\ & \widehat{K}_{ipm}^{\textit{Verifikation einer Fehlerbeseitigung}} \\ & + \widehat{K}_{ipm}^{\textit{Regressionstest (für eine Fehlerbeseitigung)}}\end{aligned} \tag{A.41}$$

Verifikation der Fehlerbeseitigung (prospektiv geschätzt): Die prospektive Kostenrechnung für die Verifikation der Beseitigung einer einzelnen Fehlerursache geht von den folgenden vereinfachenden Annahmen aus:

- Die Fehlerbeseitigung ist durch die Ausführung des Fehlverhalten identifizierenden Testfalls (beschrieben durch die bearbeitete Fehlermeldung) zu verifizieren.
- Zusätzlich sind die Testfälle auszuführen, die durch die Duplikatsfehlermeldungen beschrieben werden.
- Die Anzahl der pro Fehlermeldung zu berücksichtigenden Duplikatsfehlermeldungen wird durch den über alle Fehlermeldungen ermittelten Anteil an Duplikatsfehlermeldungen abgeschätzt.
- Zur Vermeidung einer rekursiven Berechnungsstruktur wird weiterhin angenommen, dass die Fehlerkorrektur erfolgreich war und durch die Testfallausführungen kein sonstiges Fehlverhalten identifiziert wird.

Unter den genannten Annahmen ergibt sich die geschätzte Anzahl der pro Verifikation einer Fehlerbeseitigung auszuführenden Testfälle mit $1 + \widehat{G}^{\textit{Anteil Duplikatsfehlermeldungen an akzeptierten Fehlermeldungen}}$, wobei 1 für den Testfall der bearbeiteten Fehlermeldung steht und G für den Anteil der Duplikatsfehlermeldungen. Diese Testfallanzahl wird mit den durchschnittlichen Kosten der (nur bedingt notwendigen) Vorbereitung und Ausführung eines Testfalls multipliziert. Die Abschätzung der Wahrscheinlichkeit der Notwendigkeit für eine erneute Testfallvorbereitung kann fehlerklassenbasiert geschehen, wenn angenommen wird, dass Fehlerursachen innerhalb von Fehlerklassen ähnliche Fehlerbeseitigungsmaßnahmen aufweisen und so wiederum gleiche Auswirkungen auf die Vorbereitung von Testfällen haben. Für eine abschätzende Berechnungsvorschrift erscheint dies jedoch zu detailliert, weshalb von der vom Testverfahren abhängigen Wahrscheinlichkeit ausgegangen wird. Für die in Gleichung (A.41) enthaltene Kostenkomponente *Verifikation der Fehlerbeseitigung* ergibt sich somit der aus Gleichung (A.27) abgeleitete Schätzwert:

$$\begin{aligned}\widehat{K}_{ipm}^{\textit{Verifikation einer Fehlerbeseitigung}} = \\ &\left(1 + \widehat{G}^{\textit{Anteil Duplikatsfehlermeldungen an akzeptierten Fehlermeldungen}}\right) \\ &\cdot\Big(\widehat{G}_{ipm}^{\textit{Erneute Testfallgenerierung}} \cdot \widehat{K}_{ipm}^{\textit{Testvorbereitung (für eine Fehlerbeseitigung)}} \\ &+\widehat{K}_{ipm}^{\textit{Testdurchführung (für eine Fehlerbeseitigung)}}\Big)\end{aligned} \tag{A.42}$$

Die Testdurchführungskosten $\widehat{K}_{ipm}^{\textit{Testdurchführung (für eine Fehlerbeseitigung)}}$ für die Verifikation einer Fehlerbeseitigung können je nach Datenlage als projektkonstante Testdurchführungskosten entweder über

- die Kosten für die Ausführung der Testfälle je Testverfahren geteilt durch die geplante Testfallanzahl

$$\widehat{K}_{ipm}^{\textit{Testdurchführung (für eine Fehlerbeseitigung)}} = \frac{\widehat{K}^{\textit{Testfallausführung}}}{\widehat{N}^{\textit{Testfallanzahl}}} \tag{A.43}$$

- oder über einen mittleren Automatisierungsgrad $\widehat{G}^{\textit{Automatisierte Testfallausführung}}$ mit mittleren Kosten für die automatisierte und manuelle Testdurchführung eines Testfalls

$$\begin{aligned}\widehat{K}_{ipm}^{\textit{Testdurchführung (für eine Fehlerbeseitigung)}} = \\ \widehat{G}^{\textit{Automatisierte Testfallausführung}} \cdot \widehat{A}^{\textit{Automatisierte Testdurchführung}} \\ + \left(1 - \widehat{G}^{\textit{Automatisierte Testfallausführung}}\right) \cdot \widehat{A}^{\textit{Manuelle Testdurchführung}}\end{aligned} \tag{A.44}$$

abgeschätzt werden.

Regressionstests (prospektiv geschätzt): Für die Regressionstests wird zur Vermeidung einer rekursiven Berechnungsstruktur ebenfalls angenommen, dass die Fehlerkorrektur erfolgreich war. Die Kosten für den *Regressionstest* unterscheiden sich dann in der prospektiven Abschätzung zu denen der *Verifikation der Fehlerbeseitigung* ($\widehat{K}_{ipme}^{\textit{Verifikation einer Fehlerbeseitigung}}$), Gleichung (A.42)) nur in der Anzahl der auszuführenden Testfälle. Diese bestimmt sich als Summe über die Anzahl der jeweils in den Testreihen auszuführenden Testfälle und der Anzahl Testreihen, die Fehlverhalten identifizierende (Duplikats)-Testfälle enthalten. Die Anzahl der Testfälle pro Testreihe wird auf Basis des Testverfahrens bestimmt. Dies basiert auf der Annahme, dass der Umfang von Testreihen maßgeblich durch das Testverfahren bestimmt ist. Somit ergeben sich die Kosten des Regressionstests wie folgt:

$$\begin{aligned}\widehat{K}_{ipm}^{\textit{Regressionstest (für eine Fehlerbeseitigung)}} = \\ \widehat{N}_{ipm}^{\textit{Anzahl Testreihen mit fehleridentifizierenden (Duplikats)-Testfällen}} \\ \cdot \widehat{N}_{ipm}^{\textit{Anzahl Testfälle pro Testreihe}} \\ \cdot \left(\widehat{G}_{ipm}^{\textit{Erneute Testfallgenerierung}} \cdot \widehat{K}_{ipm}^{\textit{Testvorbereitung}} \right. \\ \left. + \widehat{K}_{ipm}^{\textit{Testdurchführung}}\right)\end{aligned} \tag{A.45}$$

$\widehat{N}_{ipm}^{\textit{Anzahl Testfälle pro Testreihe}}$ entspricht einem geschätzten T_{ipmr}.

$\widehat{N}^{\textit{Anzahl Testreihen mit fehleridentifizierenden (Duplikats)-Testfällen}}$ ist identisch mit dem Schätzwert für durchschnittliche Anzahl an Testreihen, aus denen die fehleridentifizierenden

(Duplikats)-Testfälle eines Fehlers stammen.

Abschätzung der Fehleranzahl im System: Von besonderem Interesse sind hierbei die erforderlichen Abschätzungen zu der Anzahl zu beseitigender Fehlerursachen. Sie wird, den Ansätzen in von Wagner [257] und Haffner [117] folgend, aus der Identifikationswahrscheinlichkeit, hier für TP_1, und der Gesamtfehleranzahl im System berechnet. In der Praxis kann die Gesamtfehleranzahl im System durch eine Fehlerprognose als Konstante *FP* abgeschätzt werden [10, 36, 106, 240, 246]. Die Verteilung der Fehlerursachen ist hier aufgrund der Kostenstruktur optimalerweise hierarchisch nach Iterationen, Phasen und Methoden abzuschätzen. Hierzu dienen die geschätzten Anteile ($\widehat{G}_{i}^{\textit{Anteil vorhandener Fehlerursachen in der Iteration}}$, $\widehat{G}_{ip}^{\textit{Anteil vorhandener Fehlerursachen in der Testphase}}$, $\widehat{G}_{ipm}^{\textit{Anteil vorhandener Fehlerursachen im Testverfahren}}$). Liegen keine detaillierten Erfahrungen vor, können vereinfachend Durchschnittswerte für die Verteilung auf die Testphasen unabhängig von Iterationen und auf die Methoden unabhängig von den übergeordneten Hierarchien eingesetzt werden. Weiterhin sind die Fehleridentifikationswahrscheinlichkeiten der eingesetzten Testverfahren m, $\widehat{W}_{m}^{\textit{Identifikationswahrscheinlichkeit des Testverfahrens}}$, abzuschätzen. Sie bestimmt die Anzahl der akzeptierten Fehlermeldungen in Fehlerdatenbanken. Für die prospektiven Kosten der Herstellerinduzierten Fehlerbeseitigungskosten ergibt sich somit:

$$
\begin{aligned}
\widehat{K}_{ipm}^{\textit{Herstellerinduzierte Fehlerbeseitigung}} = \\
&\widehat{FP}^{\textit{Fehleranzahl}} \\
&\cdot\Big(\widehat{G}_{i}^{\textit{Anteil vorhandener Fehlerursachen in der Iteration}} \\
&\cdot\widehat{G}_{ip}^{\textit{Anteil vorhandener Fehlerursachen in der Testphase}} \\
&\cdot\widehat{G}_{ipm}^{\textit{Anteil vorhandener Fehlerursachen im Testverfahren}} \\
&\cdot\widehat{W}_{m}^{\textit{Identifikationswahrscheinlichkeit des Testverfahrens}} \\
&\cdot\Big(\widehat{K}_{ipm}^{\textit{Fehlermanagement für eine Fehlerbeseitigung)}} \\
&\qquad+\widehat{K}_{ipm}^{\textit{Fehlerbearbeitung für eine Fehlerbeseitigung)}} \\
&\qquad+\widehat{K}_{ipm}^{\textit{Korrekturtest (für eine Fehlerbeseitigung)}}\Big)\Big)
\end{aligned}
\tag{A.46}
$$

Dabei gelten folgende Nebenbedingungen:

$$\sum_{i}^{I} \widehat{G}_{i}^{\text{Anteil vorhandener Fehlerursachen in der Iteration}} = 1$$

$$\sum_{p}^{P_i} \widehat{G}_{ip}^{\text{Anteil vorhandener Fehlerursachen in der Testphase}} = 1$$

$$\sum_{m}^{M_{ip}} \widehat{G}_{ipm}^{\text{Anteil vorhandener Fehlerursachen im Testverfahren}} = 1$$

B. Algorithmen zur testzielorientierten Fehlerklassengewichtung (zu Kapitel 4.4.2)

Algorithmen zur Gewichtung: Ausgangsbasis für die Gewichtung ist ein Fehlerprofil mit relativen Häufigkeiten, basierend auf der testfokusorientierten Fehlerklassifikation (s. Kapitel 4.4.1), die eine Zuordnung von Fehlerklassen zu Testverfahrenskategorien beinhaltet. Weiterhin ist ein Testziel, das priorisiert erreicht werden soll, durch *Entscheidungsaspekte* zu formulieren.

Entscheidungsaspekte: *Entscheidungsaspekte* werden als Gewichtungen der Fehlermeldungen bzw. der durch diese beschriebenen Fehlerursachen abgebildet. Entscheidungsaspekte zur Gewichtung der Fehlermeldungen können sowohl diskret in *Stufen*, wie z. B. der Entscheidungsaspekt *Fehlerpriorität* mit den drei Stufen *gering, mittel* und *hoch*, als auch kontinuierlich, wie z. B. der Entscheidungsaspekt *Fehlerbeseitigungsaufwand*, definiert sein. Unter Umständen wird es sich anbieten, kontinuierliche Entscheidungsaspekte durch Klassenbildung in diskrete Aspekte zu überführen.

Fehlergewicht einer Fehlermeldung an einem Entscheidungsaspekt in einer Fehlerklasse: Jede Fehlermeldung trägt anteilig an dem Gewicht des Entscheidungsaspektes innerhalb einer Fehlerklasse bei. Dieser Anteil wird als *Fehlergewicht* bezeichnet. Er ergibt sich für jede Fehlermeldung aus dem Wert, den die Fehlermeldung für den jeweils betrachteten Entscheidungsaspekt aufweist, z. B. aus seinem Prioritätswert.

Durchschnittliches Fehlergewicht eines Entscheidungsaspektes in einer Fehlerklasse: In der Praxis sind die zur Gewichtung benötigten Werte der Entscheidungsaspekte (z. B. der angesprochene Prioritätswert) nicht immer für alle Fehlermeldungen dokumentiert (s. Kapitel 5.1.2). Das Fehlergewicht dieser Fehlermeldungen soll dann durch ein *durchschnittliches Fehlergewicht* repräsentiert werden.

Für die Berechnung des durchschnittlichen Fehlergewichts werden Fehlermeldungen innerhalb der betrachteten Fehlerklasse k nach den Stufen s eines diskreten Entscheidungsaspektes d klassifiziert. Hier werden nur die Fehlermeldungen verwendet, für die eine Ausprägung des Entscheidungsaspektes existiert. Es ergeben sich somit Häufigkeiten je Stufe, die mit dem stufenspezifisch definierten Gewicht G_{ds} multipliziert werden. Das durchschnittliche Fehlergewicht $G_{k,d}^{\textit{Durchschnittliches Fehlergewicht}}$ ergibt

sich dann aus der Summe dieser gewichteten Häufigkeiten, dividiert durch die Anzahl der so bewerteten Fehlermeldungen. Es wird die Notation aus Tabelle B.1 verwendet:

$$G_{k,d}^{\textit{Durchschnittliches Fehlergewicht}} = \sum_{s}^{S_d} \frac{N_{k,ds} \cdot G_{ds}}{N_{k,d}} \tag{B.1}$$

Für kontinuierliche Entscheidungsaspekte bieten sich *Gewichtungsfunktionen* an, z. B. $g_e(x)=x^2$ mit x als kontinuierlicher Variable einer Fehlermeldung bzgl. des Entscheidungsaspektes *e*. Im Unterschied zu Gleichung (B.1) wird nicht eine Summe über Stufen sondern über Fehlermeldungen definiert:

$$G_{k,e}^{\textit{Durchschnittliches Fehlergewicht}} = \sum_{n}^{N_{k,e}} \frac{g_e(x_{nk,e})}{N_{k,e}} \tag{B.2}$$

Klassengewicht eines Entscheidungsaspektes: Aus den durchschnittlichen Fehlergewichten der Gleichungen (B.1) und (B.2) lassen sich leicht die Klassengewichte der Entscheidungsaspekte berechnen. Hierzu werden die durchschnittlichen Fehlergewichte mit der Fehleranzahl pro Fehlerklasse multipliziert. Für die diskreten Entscheidungsaspekte ergibt sich demnach je Entscheidungsaspekt d die Berechnung wie folgt:

$$G_{k,d} = G_{k,d}^{\textit{Durchschnittliches Fehlergewicht}} \cdot N_k \tag{B.3}$$

Das Klassengewicht eines kontinuierlichen Entscheidungsaspektes e ergibt sich analog zu Gleichung (B.3).

Normierung der Klassengewichte je Entscheidungsaspekt: Das Klassengewicht der Entscheidungsaspekte sollte unabhängig davon sein, in welchen Maßeinheiten die Entscheidungsaspekte gemessen wurden. Die Verwendung der Einheit *Stunden* oder *Tage* für den Fehlerbeseitigungsaufwand sollte z. B. das Ergebnis nicht beeinflussen. Dazu werden die Klassengewichte $G_{k,d}$ und $G_{k,e}$ normiert:

$$G_{k,d}^{\textit{Normiert}} = \frac{G_{k,d}}{G_d^{\textit{Gesamt}}} \tag{B.4}$$

mit

$$G_d^{Gesamt} = \sum_k^K G_{k,d} \tag{B.5}$$

Die Normierung für die kontinuierlichen Entscheidungsaspekte e erfolgt analog zu Gleichung (B.4).

Gewichtung von Entscheidungsaspekten im Klassengewicht (für das gewichtete Fehlerprofil): Für die praktische Anwendung kann es hierbei erforderlich sein, bestimmte Entscheidungsaspekte stärker zu gewichten. Beispielsweise könnte die Fehlerpriorität doppelt so stark gewichtet werden wie der Fehlerbeseitigungsaufwand. Hierfür fließen die Gewichtungen $G_d^{Entscheidungsaspekt}$ und $G_e^{Entscheidungsaspekt}$ in die Berechnung ein. Das Klassengewicht über alle Entscheidungsaspekte der Klasse ergibt sich entsprechend Gleichung (B.6) aus der Summe der normierten Klassengewichte der Entscheidungsaspekte, jeweils multipliziert mit den Gewichtungen der Entscheidungsaspekte:

$$\begin{aligned} G_k &= \sum_d^D \left(G_d^{Entscheidungsaspekt} \cdot G_{k,d}^{Normiert} \right) \\ &+ \sum_e^E \left(G_e^{Entscheidungsaspekt} \cdot G_{k,e}^{Normiert} \right) \end{aligned} \tag{B.6}$$

Da die Endlaufzahlen D und E unabhängig von k sind, beinhaltet der dargestellte Gewichtungsalgorithmus, dass für alle Fehlerklassen dieselben Entscheidungsaspekte mit denselben Stufen und Stufengewichten verwendet werden. Eine Erweiterung auf unterschiedliche Aspekte, Stufen und Gewichte für die Fehlerklassen ist zwar möglich, erscheint jedoch nicht sinnvoll und erschwert die Anwendung und die Überschaubarkeit der Gewichtungseinflüsse.

Notation	**Interpretation**
k = *1, ..., K*	Laufvariable für die Fehlerklassen
d = *1, ..., D*	Laufvariable für die diskreten Entscheidungsaspekte, z. B. *Fehlerpriorität*
s = *1, ...,* S_d	Laufvariable für die Stufen, z. B. *gering, mittel* und *hoch,* eines diskreten Entscheidungsaspektes d
e = *1, ..., E*	Laufvariable für die kontinuierlichen Entscheidungsaspekte, z. B. *Fehlerbeseitigungsaufwand*
G_k	Gewicht der Fehlerklasse k für das gewichtete Fehlerprofil
$G_{k,d/e}$	Gewicht des Entscheidungsaspektes d bzw. e bzgl. der Fehlerklasse k
$G_{k,d/e}^{Normiert}$	Normiertes Gewicht des Entscheidungsaspektes d bzw. e bzgl. der Fehlerklasse k
$G_{k,d/e}^{Durchschn.\ F\text{-}Gewicht}$	Durchschnittliches Gewicht der Fehlermeldung des Entscheidungsaspektes d bzw. e bzgl. einer Fehlerklasse k
G_{ds}	Gewicht für die Stufe s eines diskreten Entscheidungsaspektes d
$G_{d/e}^{Entscheidungsaspekt}$	Entsprechend der Zielsetzung definierte Gewichtung für den Entscheidungsaspekt d bzw. e
N_k	Anzahl der Fehlermeldungen einer Fehlerklasse, vor der Gewichtung gilt: $G_k = N_k$
$N_{k,d/e}$	Anzahl der bzgl. eines Entscheidungsaspektes d bzw. e bewerteten Fehlermeldungen einer Fehlerklasse k
$N_{k,ds}$	Anzahl Fehlermeldungen in der s-ten Stufe des d-ten diskreten Entscheidungsaspekts in der Fehlerklasse k mit $\sum_s^{S_d} N_{k,ds} = N_{k,d} \leq N_k$ für alle d. Insbesondere gilt $N_{k,d} \leq N_k$ da nicht zwingend alle Fehlermeldungen aufgrund fehlender Daten einer Stufe eines Entscheidungsaspektes zugeordnet werden können.
n = *1, ...,* $N_{k,e}$	Laufvariable für die anhand des kontinuierlichen Entscheidungsaspektes e auswertbaren Fehlermeldungen der Fehlerklasse k
$x_{nk,e}$	Ausprägung einer Fehlermeldung n der Fehlerklasse k bzgl. des kontinuierlichen Entscheidungsaspekts e
$g_e(x_{nk,e})$	Gewichtungsfunktion eines kontinuierlichen Entscheidungsaspektes e mit Ausprägung x der Fehlermeldung n als Eingabeparameter

Tabelle B.1.: Notation zur Berechnung eines gewichteten Fehlerprofils

C. Algorithmen zu Testautomatisierungskosten (zu Kapiteln 4.6.2 und 4.6.3)

Isolierung der die Automatisierungskosten bestimmenden Kostenelemente: Die Algorithmen zur Bestimmung der Testautomatisierungskosten sind in den Quantifizierungsvorschriften des Qualitätskostenmodells (Anhang A) enthalten. Jedoch erfolgt die Kostenberechnung dort gemäß dem chronologischen Ablauf eines Projekts in verschiedenen Hauptkostengruppen. Für die hier intendierte Break-Even-Point-Bestimmung werden die Kosten in einer Gleichung zusammengestellt. Zudem sind nur die Kostenbereiche für die in Kapitel 4.6.1 genannten Mehr- und Minderkosten einer Automatisierung erforderlich. Kostenelemente, die von der zu untersuchenden Automatisierung unabhängig sind, werden hier vereinfachend also nicht betrachtet. Dies trifft insbesondere für die Kosten der manuellen sowie der automatisierten Testfallerstellung und Testfallanpassung zu, denn unabhängig davon, ob Testfälle anschließend automatisiert werden oder nicht, sind diese zunächst in gleicher Weise manuell oder automatisiert zu erstellen und anzupassen.

Automatisierungsgrad: Der folgende Algorithmus ermittelt die Testautomatisierungskosten in Abhängigkeit vom Automatisierungsgrad G ($0 \leq G \leq 1$), der den Anteil der automatisierten Testfälle darstellt. Der Algorithmus liefert gleichermaßen die Kosten ohne Automatisierung ($G = 0$) und mit Automatisierung ($0 < G \leq 1$).

Die Anzahl der pro Iteration ausgeführten Testfälle steigt mit fortschreitenden Iterationen i. Allg. an. Der Automatisierungsgrad der durchgeführten Testfälle je Iteration bleibt i. Allg. nicht konstant. Außerdem sind unterschiedliche Automatisierungsgrade für Testfallautomatisierung und Testfallausführung anzunehmen. Der Automatisierungsgrad G ist also von der Hauptkostengruppe h und der Iteration i abhängig. Darüber hinaus findet eine testphasen- und verfahrensspezifische Betrachtung statt, was zu der Gesamtabhängigkeit G^h_{ipm} führt. G^h_{ipm} kommt vorwiegend bei abschätzenden Berechnungen Bedeutung zu.

Retrospektive, exakte Berechnung: Es werden Notation und Laufvariablen entsprechend Anhang A.1 verwendet. Damit ergibt sich die retrospektive, exakte Berechnung der Testautomatisierungskosten wie folgt:

$$
\begin{aligned}
K^{\textit{Test mit Automatisierungsgrad G}} = & \\
& \sum_{i}^{I} \sum_{p}^{P_i} \big(G^3 \\
& \cdot K^{\textit{Jahreslizenzen für Testfallautomatisierungswerkzeuge}} \\
& \cdot A^{\textit{Nutzungsdauer in Jahren, Testfallautomatisierungswerkzeuge}} \\
& + \sum_{m}^{M_{ip}^{h=4}} K_{ipm}^{\textit{Testfallautomatisierung G4}} \\
& + \sum_{m}^{M_{ip}^{h=5}} K_{ipm}^{\textit{Testdurchführung G5}} \big)
\end{aligned}
\tag{C.1}
$$

nach Notation wie in Tabelle C.1 beschrieben.

Die Automatisierungskosten der Hauptkostengruppe *Testvorbereitung* ($h = 4$) ergeben sich aus der Summe der automatisierungsbedingten Aufwandswerte. Für nicht automatisierte Testfälle sind diese Aufwandswerte mit null anzusetzen:

$$
\begin{aligned}
K_{ipm}^{\textit{Testfallautomatisierung G}} = & \\
& \sum_{r}^{R_{ipm}} \sum_{t}^{T_{ipmr}} \Big(A_{ipmrt}^{\textit{Vorbereitung des Testfalls t für automatisierten Test}} \\
& + A_{ipmrt}^{\textit{Anpassung der Automatisierung des Testfalls}} \Big) \\
& \cdot K^{\textit{Stundensatz Tester}}
\end{aligned}
\tag{C.2}
$$

Die Testfallausführungskosten der Hauptkostengruppe Testdurchführung ($h = 5$) ergeben sich wie für die Berechnung der Testautomatisierungskosten aus den Eigenschaften jedes einzelnen Testfalls, bei dem die Unterscheidung des Aufwands für manuell oder automatisch implizit gegeben ist.

$$
\begin{aligned}
K_{ipm}^{\textit{Testfalldurchführung G}} = & \\
& \sum_{r}^{R_{ipm}} \sum_{t}^{T_{ipmr}} \Big(A_{ipmrt}^{\textit{Ausführung des Testfalls t}} \\
& + A_{ipmrt}^{\textit{Fehlererkennung für Testfall t}} \Big) \\
& \cdot K^{\textit{Stundensatz Tester}}
\end{aligned}
\tag{C.3}
$$

Prospektiv abschätzende Berechnung: Für prospektive Abschätzungen ist ein vereinfachter Berechnungsalgorithmus zu verwenden, der maßgeblich durch die

i = 1, ..., I	Laufvariable für die Iteration
p =1, ..., P_i	Laufvariable für die durchzuführenden Testphasen während der Entwicklung in der Iteration *i*. Die Laufvariable *p* steht nur für die entwicklungsbegleitenden Testphasen, die in der Testphasenmenge TP_1 (Modultest, Systemtest und Integrationstests) zusammengefasst sind. Für die Phasen der kundeninduzierten Fehlerbeseitigungskosten, die in den Testphasen TP_2 (Fahrzeugtest, Akzeptanztest und Feldeinsatz) enthalten sind, wird beim Hersteller keine Testautomatisierung durchgeführt.
m = 1, ..., M_{ip}	Laufvariable für Testverfahren in der Testphase *p* der Iteration *i*
r = 1, ..., R_{ipm}	Laufvariable für erzeugte Testreihen für ein Testverfahren *m* in Testphase *p* der Iteration *i*
t = 1, ..., T_{ipmr}	Laufvariable für Testfälle der Testreihe *r* aus Testverfahren *m* in der Testphase *p* und Iteration *i*
h = 1, ..., 8	Laufvariable der Hauptkostengruppen 1 = Testmanagement 2 = Fehlervermeidung 3 = Testaufbau 4 = Testvorbereitung 5 = Testdurchführung 6 = herstellerinduzierte Fehlerbeseitigung 7 = kundeninduzierte Fehlerbeseitigung 8 = Fehlerfolgekosten
G^h_{ipm}	Automatisierungsgrad G^h in der *h*-ten Hauptkostengruppe. G^4 und G^5 sind abhängig von dem Testverfahren *m* der Testphase *p* der Iteration *i*: $G^h_{ip} \epsilon [0,1]$ als Anteile für $h \epsilon 4,5$ Die Berechnung der Testfallautomatisierungskosten basiert auf G^4_{ipm} für die Vorbereitung und $G^{4,\ Anpassung}_{ipm}$ für die Anpassung der Automatisierung, die der Testausführungskosten auf G^5_{ipm}. G^3 bestimmt die Lizenzkosten, wobei für $G^3 = 0$ (im Falle ohne Automatisierung) keine und für $G^3 = 1$ die vollen Lizenzkosten anfallen.

Tabelle C.1.: Notation der Break-Even-Point-Berechnung

Testautomatisierungsgrade $\widehat{G}^h_{ipm}$ sowie $\widehat{G}^{4,\ Anpassung}_{ipm}$ bestimmt ist (in Anlehnung an Kapitel 6.4.2). $\widehat{G}^h_{ipm}$ ist je Iteration, Testphase und Testverfahren zu verstehen als Quotient der Anzahl der zu automatisierenden Testfälle geteilt durch die Anzahl der in der jeweiligen Phase auszuführenden Testfälle. Der Anpassungsgrad $\widehat{G}^{4,\ Anpassung}_{ipm}$ bezeichnet den Anteil der bereits automatisierten, wiederverwendeten Testfälle, die angepasst werden müssen. Die Werte für $\widehat{G}^4_{ipm}$ und $\widehat{G}^5_{ipm}$ sowie $\widehat{G}^{4,\ Anpassung}_{ipm}$ werden geschätzt. Ebenso geschätzt werden iterations-, phasen- und testverfahrensabhängig

die Anzahlen der zu erstellenden, wiederverwendeten und auszuführenden Testfälle $\widehat{N}_{ipm}^{\textit{Zu erstellende Testfälle}}$, $\widehat{N}_{ipm}^{\textit{Wiederverwendete Testfälle}}$ und $\widehat{N}_{ipm}^{\textit{Auszuführende Testfälle}}$. Die Kosten für die Testfallautomatisierungswerkzeuge werden über die Nutzungsdauer berechnet. Die Gesamtkosten bestimmen sich somit prospektiv nach:

$$
\begin{aligned}
\widehat{K}^{\textit{Test mit Automatisierungsgrad G}} = \\
& G^{h=3} \cdot K^{\textit{Jahreslizenzen für Testfallautomatisierungswerkzeuge}} \\
& \cdot A^{\textit{Nutzungsdauer in Jahren, Testfallautomatisierungswerkzeuge}} \\
& + \sum_{i}^{I^{h=4}} \sum_{p}^{P_i^{h=4}} \sum_{m}^{M_{ip}^{h=4}} \widehat{K}_{ipm}^{\textit{Testfallautomatisierung G4}} \\
& + \sum_{i}^{I^{h=5}} \sum_{p}^{P_i^{h=5}} \sum_{m}^{M_{ip}^{h=5}} \widehat{K}_{ipm}^{\textit{Testdurchführung G5}}
\end{aligned}
\tag{C.4}
$$

mit

$$
\begin{aligned}
\widehat{K}_{ipm}^{\textit{Testfallautomatisierung G4}} = \\
& \Big(\widehat{N}_{ipm}^{\textit{Zu erstellende Testfälle}} \\
& \quad \cdot \widehat{A}_{ipm}^{\textit{Mittlere Vorbereitungsdauer eines Testfalls für die automatisierte Ausführung}} \\
& \quad \cdot \widehat{G}_{ipm}^{4} \\
& + \widehat{N}_{ipm}^{\textit{Wiederverwendete Testfälle}} \\
& \quad \cdot \widehat{A}_{ipm}^{\textit{Mittlere Anpassungsdauer eines automatisierten Testfalls}} \\
& \quad \cdot \widehat{G}_{ipm}^{\textit{4, Anpassung}} \Big) \\
& \cdot K^{\textit{Stundensatz Tester}}
\end{aligned}
\tag{C.5}
$$

und

$$
\begin{aligned}
\widehat{K}_{ipm}^{\textit{Testdurchführung G5}} = & \\
& \widehat{N}_{ipm}^{\textit{Auszuführende Testfälle}} \\
& \cdot \Big(\Big(\widehat{A}_{ipm}^{\textit{Mittlere Ausführungsdauer eines automatisierten Testfalls}} \\
& \qquad + \widehat{A}_{ip}^{\textit{Mittlere Dauer für automatisierte Fehlererkennung eines Testfalls}} \Big) \\
& \qquad \cdot \widehat{G}_{ipm}^{5} \\
& + \Big(\widehat{A}_{ipm}^{\textit{Mittlere Ausführungsdauer eines manuellen Testfalls}} \\
& \qquad + \widehat{A}_{ipm}^{\textit{Mittlere Dauer für manuelle Fehlererkennung eines Testfalls}} \Big) \\
& \qquad \cdot \Big(1 - \widehat{G}_{ipm}^{5} \Big) \Big) \\
& \cdot K^{\textit{Stundensatz Tester}}
\end{aligned}
\tag{C.6}
$$

Gleichung (C.5) beschreibt die geschätzten Testfallautomatisierungskosten. Anstelle des individuellen Aufwands jeden Testfalls für die Automatisierung wird von einem mittleren, geschätzten Testfallautomatisierungsaufwand ausgegangen.

Gleichung (C.6) schätzt die Kosten der Testdurchführung ab. Für den Kostenvergleich mit der rein manuellen Ausführung sind anhand des geschätzten Automatisierungsgrades die automatisiert und die verbliebenen, manuell auszuführenden Testfälle zu berücksichtigen. Auch hier wird mit gemittelten Aufwandswerten für die Testfallausführung und Fehlererkennung gerechnet. Es wird angenommen, dass die Fehlererkennung für automatisierte Testfallausführungen ebenfalls automatisiert erfolgt.

Iterationsbezogene, projektweit vereinfachende Abschätzung: In nachfolgendem Ansatz wird Gleichung (C.4) verallgemeinert, indem die Summen und Kenngrößen durch über Iterationen, Phasen und Testverfahren gemittelte Werte ersetzt werden. Dies kommt der derzeitigen Praxis entgegen, in der nicht immer iterations-, phasen- und verfahrensabhängige Kosten- und Aufwandsangaben vorliegen, auf die für die Abschätzung zur Orientierung zurückgegriffen werden kann. Für die Verallgemeinerung werden folgende Annahmen getroffen:

- Die Kosten der Testfallautomatisierung und die Kosten für die Anpassung der Automatisierung sind für alle Testfälle gleich, unabhängig von Iteration, Testphase und Testverfahren ($\widehat{K}^{\textit{Testfallautomatisierung für einen Testfall}}$, $\widehat{K}^{\textit{Anpassung der Automatisierung für einen Testfall}}$).
- Der Automatisierungsgrad für die Vorbereitung zur automatisierten Ausführung $\widehat{G}^{4}$ sowie der Anpassungsgrad als Anteil der anzupassenden automatisierten Testfälle $\widehat{G}^{4,\,\textit{Anpassung}}$ sind unabhängig von Iteration, Testphase und Testverfahren. Die Werte sind so abzuschätzen, als seien sie aus über alle Iterationen, Testphasen und Testverfahren (gewichtet) gemittelten Werten entstanden.

- Die Anzahl der erzeugten Testfälle ($\widehat{T}$) sind für das Projekt global gültig, also unabhängig von Iterationen, Testphasen und Testverfahren.
- Die Anzahl der je Iteration auszuführenden Testfälle ($\widehat{Y}$) sind identisch für alle Iterationen, mit einer projektweiten, mittleren Anzahl geschätzt.
- Der mittlere Automatisierungsgrad der auszuführenden Testfälle und der automatisierten Fehlererkennung gilt projektweit und wird geschätzt, als sei es ein über die Iterationen, Testphasen und Testverfahren (gewichtet) gemittelter Wert ($\widehat{G}^5$).

Auf Basis dieser Annahmen wird ein vereinfachender Algorithmus (s. Gleichung (C.7)) abgeleitet. Da ein Nutzen der Automatisierung mit steigenden Iterationen erreicht wird, ist der Algorithmus iterationsbezogen formuliert. Durch die vereinfachenden Annahmen können zur Schätzung der Testautomatisierungskosten für I Iterationen die Einzelsummen aus Gleichung (C.4) durch Produkte ersetzt werden:

$$
\begin{aligned}
\widehat{K}^{\textit{Test, Automatisierungsgrad G}} = \\
&\widehat{K}^{\textit{Testfallautomatisierungswerkzeuge (K=0 bei G3=0)}} \\
&+\widehat{T} \cdot \Big(\widehat{G}^4 \cdot \widehat{K}^{\textit{Testfallautomatisierung für einen Testfall}} \\
&\qquad +\widehat{G}^{\textit{4, Anpassung}} \cdot \widehat{K}^{\textit{Anpassung der Automatisierung für einen Testfall}} \Big) \\
&+\widehat{I} \cdot \widehat{Y} \\
&\cdot \Big(\widehat{G}^5 \cdot \Big(\widehat{K}^{\textit{Testfallausführung für einen automatisierten Testfall}} \\
&\qquad +\widehat{K}^{\textit{Automatisierte Fehlerkennung für einen Testfall}} \Big) \\
&+\Big(1 - \widehat{G}^5 \Big) \cdot \Big(\widehat{K}^{\textit{Testfallausführung für einen manuellen Testfall}} \\
&\qquad +\widehat{K}^{\textit{Manuelle Fehlerkennung für einen Testfall}} \Big) \Big)
\end{aligned}
\tag{C.7}
$$

Ableitung des Break-Even-Points nach Iterationen: Für den Break-Even-Point gilt:

$$
\widehat{K}^{\textit{Test ohne Automatisierung}} = \widehat{K}^{\textit{Test mit Automatisierungsgrad}\ 0<GA\leq 1}
\tag{C.8}
$$

GA ist der Anteil automatisierter Testfälle.

Wenn keine Automatisierung durchgeführt wird, entfallen die Kostenelemente $\widehat{K}^{\textit{Testfallautomatisierungswerkzeuge}}$, $K^{\textit{Testfallautomatisierung für einen Testfall}}$, $K^{\textit{Anpassung der Automatisierung für einen Testfall}}$, $K^{\textit{Testfallausführung für einen automatisierten Testfall}}$ und $K^{\textit{automatisierte Fehlerkennung für einen Testfall}}$ der Gleichung (C.7). Wird Gleichung (C.7) jeweils entsprechend in Gleichung (C.8) eingesetzt, ergibt sich::

$$
\begin{aligned}
\widehat{K}^{\textit{Test ohne Automatisierung}} = \\
& \widehat{I} \cdot \widehat{Y} \cdot \widehat{K}^{\textit{Testfallausführung für einen manuellen Testfall}} \\
& + \widehat{K}^{\textit{Manuelle Fehlerkennung für einen Testfall}} \\
& = \widehat{K}^{\textit{Testfallautomatisierungswerkzeuge}} \\
& + \widehat{T} \cdot \Big(\widehat{G}^{4} \cdot \widehat{K}^{\textit{Testfallautomatisierung für einen Testfall}} \\
& \qquad + \widehat{G}^{\textit{4, Anpassung}} \cdot \widehat{K}^{\textit{Anpassung der Automatisierung für einen Testfall}} \Big) \\
& + \widehat{I} \cdot \widehat{Y} \cdot \Big(\widehat{G}^{5} \cdot \widehat{K}^{\textit{Testfallausführung für einen automatisierten Testfall}} \\
& \qquad + \widehat{K}^{\textit{Automatisierte Fehlerkennung für einen Testfall}} \Big) \\
& + \Big(1 - \widehat{G}^{5} \cdot \widehat{K}^{\textit{Testfallausführung für einen manuellen Testfall}} \\
& \qquad + \widehat{K}^{\textit{Manuelle Fehlerkennung für einen Testfall}} \Big) \\
& = \widehat{K}^{\textit{Test mit Automatisierungsgrad}\ 0<GA\leq 1}
\end{aligned}
\tag{C.9}
$$

D. Kostenanalyse der Arbeitsschritte der Fehlerbeseitigung (zu Kapitel 5.2.2)

Die Anteile der Arbeitsschritte der Fehlerbeseitigung wurden einmal auf Basis der Mittelwerte und einmal auf Basis der Mediane berechnet. Diese arbeitsschrittbezogene Kennwertbildung ist indiziert, da nicht für jede Fehlermeldung Aufwandsangaben komplett für alle Arbeitsschritte eingetragen wurden (wie aus Tabelle D.1 Spalte *Fehleranzahl* ersichtlich). Die drei Mittelwerte (Mediane) aus den eingetragenen Aufwandsangaben für die Arbeitsschritte *Fehleranalyse, Fehlerkorrektur* und *Korrekturtest* wurden aufsummiert. Der relative Aufwand eines Arbeitschrittes ergibt sich als Anteil an der Aufwandssumme.

Da der Aufwand der Arbeitsschritte *Fehleranalyse, Fehlerkorrektur* und *Korrekturtest* nicht gleichermaßen für alle Fehlermeldungen eingetragen wurde, kann sich eine Verzerrung der Ergebnisse ergeben. So z. B. wenn bestimmte Fehlerarten vorzugsweise komplett bzw. unvollständig eingetragen werden. Zur Überprüfung der Ergebnisse wurde daher zusätzlich eine Analyse der vollständig ausgefüllten Fehlermeldungen durchgeführt (Tabelle D.2).

Projekt	Arbeitsschritt	Fehleranzahl	Relativer Aufwand auf Basis der	
			Mittelwerte	Mediane
Inf1	Fehleranalyse	420	21%	18%
	Fehlerkorrektur	408	74%	73%
	Korrekturtest	2.422	5%	9%
Inf2	Fehleranalyse	79	51%	43%
	Fehlerkorrektur	122	37%	54%
	Korrekturtest	534	12%	3%
Gewichtetes Mittel (1)	Fehleranalyse	499	27%	23%
	Fehlerkorrektur	530	66%	68%
	Korrekturtest	2.956	7%	9%
(1) Gewichtet anhand der Fehleranzahl				

Tabelle D.1.: Aufwand für Fehleranalyse, Fehlerkorrektur und Korrekturtest in Infotainment-Projekten, auf Basis von allen Fehlermeldungen

Projekt	Arbeitsschritt	Fehleranzahl	Relativer Aufwand auf Basis der	
			Mittelwerte	Mediane
Inf1	Fehleranalyse	16	22%	18%
	Fehlerkorrektur		68%	73%
	Korrekturtest		10%	9%
Inf2	Fehleranalyse	23	33%	26%
	Fehlerkorrektur		40%	53%
	Korrekturtest		27%	21%
Gewichtetes Mittel (1)	Fehleranalyse	39	28%	23%
	Fehlerkorrektur		52%	61%
	Korrekturtest		20%	16%
(1) Gewichtet anhand der Fehleranzahl				

Tabelle D.2.: Aufwand für Fehleranalyse, Fehlerkorrektur und Korrekturtest in Infotainment-Projekten, auf Basis von Fehlermeldungen mit Angaben in allen Arbeitsschritten

E. Realitätstreue von prospektiven Aufwandsschätzungen (zu Kapitel 5.3)

Datenbasis: Projekte BC2, Inf2 und M1 weisen eine für die Auswertung ausreichend breite Datenbasis auf. Somit stehen Daten aus einem Body Controller-Projekt, einem Infotainment-Projekt und einem Projekt mit beiden Systemtypen zur Verfügung. Der Aufwand für Fehlerbeseitigungsmaßnahmen wurde in den untersuchten Projekten prospektiv abgeschätzt und sollte ebenso wie der tatsächlich angefallene Aufwand in den Fehlerdatenbanken hinterlegt werden. Die Aufwandswerte wurden von Testern ohne Beschränkungen, wie Vorgabe von Intervallen oder Standardwerten, frei eingegeben. Für die Projekte BC2 und Inf2 wurden für Fehlerbeseitigungskosten unter einer Stunde in der Regel keine Aufwandsabschätzungen abgegeben.

Vorgehensweise bei der Datenanalyse: Der geschätzte Aufwand wird als *Plan-Aufwand*, der tatsächlich angefallene als *Ist-Aufwand* bezeichnet. Als Interpretationsbasis werden die Ist- und Plan-Aufwandsdaten in einem Säulendiagramm visualisiert. Aufgrund der rechtsschiefen Verteilung empfiehlt sich eine Skala mit logarithmisch äquidistanten Intervallen. Die Abweichung des Planaufwands vom Ist-Fehlerbeseitigungsaufwand wird in einer quantitativen Analyse untersucht. Hierzu werden Abschätzungsfehler als die Differenz *Schätzwert minus tatsächlichem Wert* berechnet. Negative Werte bedeuten eine Aufwandsunterschätzung und positive eine Aufwandsüberschätzung. Die Differenzen werden auf einer logarithmischen Skala grafisch dargestellt. Zusätzlich werden die Abschätzungsfehler mit statistischen Maßzahlen untersucht.

Analyseergebnisse: In allen drei Projekten ähnelt die Verteilung des Plan-Fehlerbeseitigungsaufwands dem des Ist-Fehlerbeseitigungsaufwands (s. Abbildung E.1). Das bedeutet, es liegt keine offensichtliche systematische Verzerrung durch Unter- oder Überschätzung vor.

Abbildung E.2. visualisiert die Abschätzungsfehler. Diese weisen um den Nullpunkt nahezu symmetrische Verteilungen auf und offenbaren ebenfalls keine systematische Verzerrung. Es liegt ein hoher Anteil exakter Schätzungen vor: 43% (=157/363) für Projekt BC2, 70% (=16/23) für Projekt Inf2 und 57% (=181/316) für Projekt M1. Die weitgehende Aussparung von Aufwandswerten von unter 1 Stunde bei den Projekten BC2 und Inf2 bleibt in den Klassen der Abschätzfehler sichtbar (s. Abbildung E.2).

Die Perzentile der Abschätzungsfehler (s. Tabelle E.1) weisen für alle Projekte auf die bereits erwähnten hohen Anteile an exakten Aufwandsabschätzungen hin. Die

20%- bzw. 30%-Perzentile bis hin zu den 60%-, 70%- bzw. 80%-Perzentilen sind null. Die Minima und Maxima zeigen auf, dass sowohl extreme Über- als auch Unterschätzungen vorliegen.

Projekt	Fehleranzahl	Perzentil											Mittelwert des Abschätzungsfehlers	Summe der Abschätzungsfehler	Prozentualer Abschätzungsfehler (1)
		Min	10%	20%	30%	40%	Median	60%	70%	80%	90%	Max			
Body Controller															
BC2	363	-112,0	-5,0	-2,0	0,0	0,0	0,0	0,0	1,0	3,0	7,0	70,0	0,0	0,5	0%
Infotainment															
Inf2	23	-40,0	-2,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	3,0	40,0	-0,9	-21,0	-1%
Body Controller und Infotainment															
M1	316	-80,0	-0,2	-0,1	0,0	0,0	0,0	0,0	0,0	0,1	0,2	64,0	-0,3	-81,4	-8%
$\sum$	702												-0,2	-101,9	-2%

Negative Werte bedeuten, dass der Fehlerbeseitigungsaufwand unterschätzt wurde, somit also tatsächlich ein größerer Fehlerbeseitigungsaufwand angefallen ist.
(1) Abschätzungsfehler in Prozent des gesamten Fehlerbeseitigungsaufwands

Tabelle E.1.: Perzentile und Maßzahlen der Abschätzungsfehler des Fehlerbeseitigungsaufwands in den Projekten BC2, Inf2 und M1 in Stunden

Die mittleren Abschätzungsfehler weichen in den Projekten Inf2 und M1 von null ab (siehe Tabelle E.1). Die Summe der Abschätzungsfehler gibt an, wie groß der Schätzfehler in Stunden für die hier bewerteten Fehlermeldungen war. In Projekt M1 wurden 81 Stunden mehr für die Fehlerbeseitigung benötigt als geschätzt wurde (siehe Tabelle E.1). Bezogen auf den gesamten Beseitigungsaufwand der Fehlerursachen bedeutet dies eine Unterschätzung um 8%. Bei Projekt Inf2 liegt eine Unterschätzung um 1% vor, während bei Projekt BC2 der geschätzte Gesamtaufwand genau eingehalten wurde. Die Verteilung der Perzentile deutet darauf hin, dass nur wenige Abschätzungsfehler diese Abweichungen verursachten. Eine manuell durchgeführte Analyse der jeweils größten positiven und negativen Abschätzungsfehler lässt keine Rückschlüsse darauf zu, dass Fehlerursachen mit bestimmten Eigenschaften vornehmlich diese hohen Abschätzungsfehler verursachen.

Analyse zur Qualität von Aufwandsabschätzungen in der Literatur: Zeit- und Budgetabschätzungen lassen häufig Diskrepanzen zur Realität hin erkennen: Für 67% der Entwickler sind Terminüberschreitungen *Routineereignisse* [76]. Dem Standish-Report [12] [247, 249–252] folgend, überziehen etwa 70% der Softwareprojekte entweder Zeit- oder Budgetbeschränkungen. Der Test, der einen Anteil von ca. 50% ($\pm$ 20%) des Aufwands der Softwareerstellung [8,15,199,214,215,262] einnimmt, wird hiervon nicht ausgenommen.

[12] Die Seriosität des Standish-Reports wurde in der Vergangenheit angezweifelt, da die Untersuchungsergebnisse nur zu Teilen offengelegt wurden und die durchgeführten Umfragen die Anzahl der fehlgeschlagenen Projekte überschätzen sollen [110,149]. Die Untersuchungsergebnisse decken sich dennoch mit Erfahrungen aus industriellen Projekten und der Literatur [190].

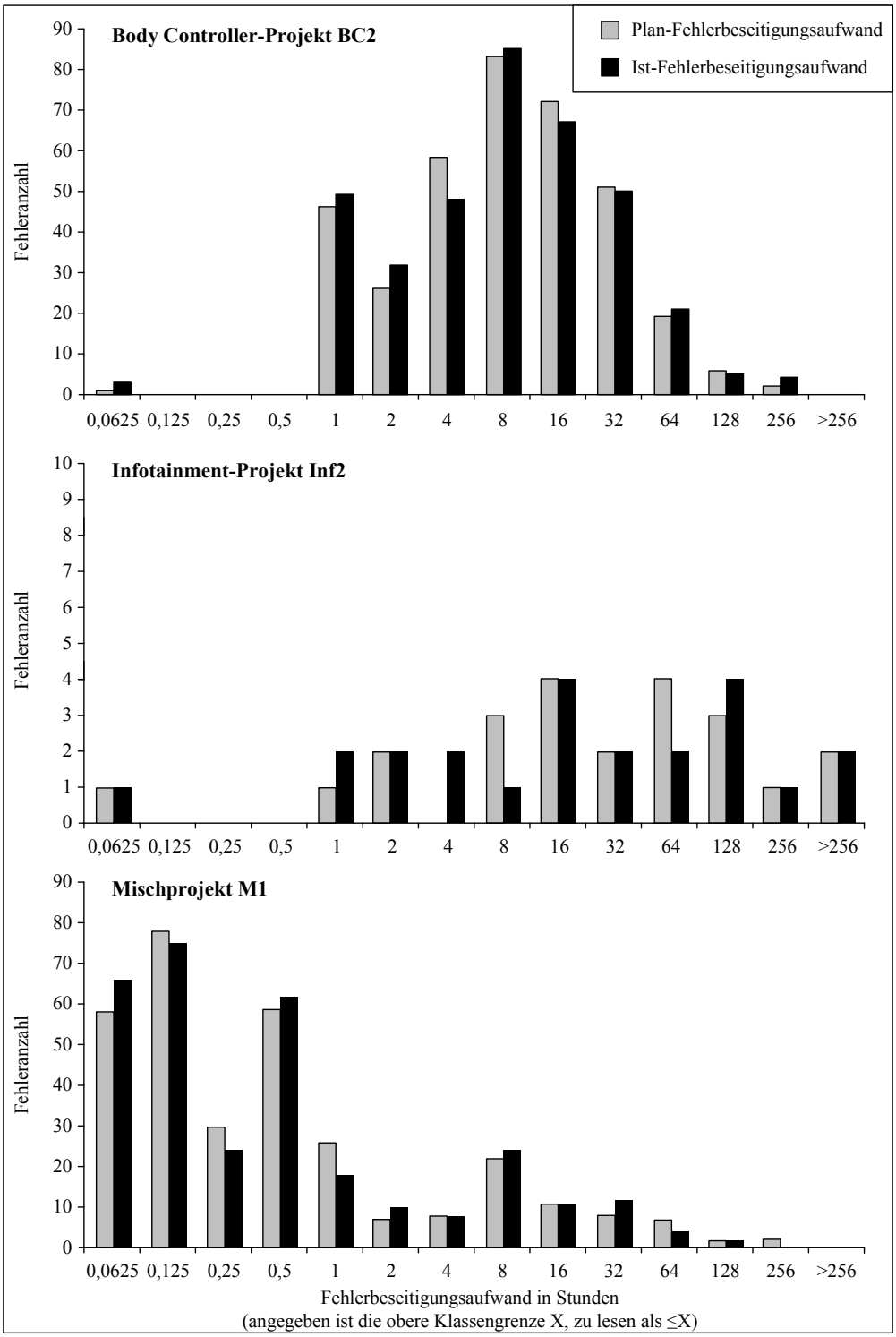


Abbildung E.1.: Verteilung des Fehlerbeseitigungsaufwands

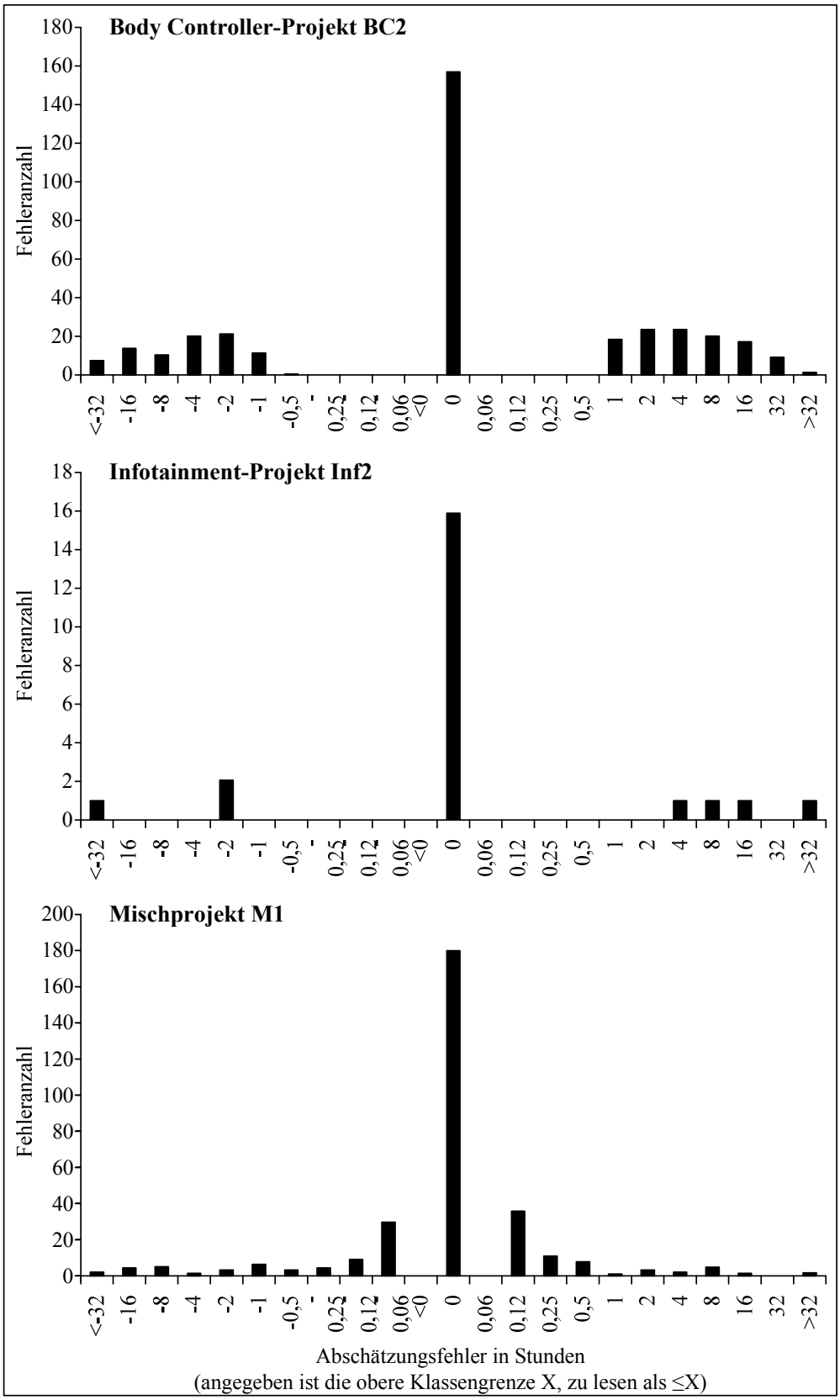


Abbildung E.2.: Verteilung der Abschätzungsfehler

F. Berechnung der Kosteneinsparpotentiale durch Fehlerfrüherkennung (zu Kapitel 6.1)

Im Folgenden wird die Berechnung der Kosteneinsparpotentiale in den Tabellen F.1 und F.2 erläutert und beispielhaft demonstriert. Kosteneinsparpotentiale ergeben sich in Bezug auf ein durchgeführtes Projekt – dadurch, dass Fehlerbeseitigungskosten mit fortschreitender Projektphase ansteigen [28] – aus der Differenz der Kosten, wenn die Fehlerbeseitigung in einem Projekt nicht in der frühestmöglichen Projektphase erfolgt ist. Die Berechnung folgt den Überlegungen in Kapitel 4.5 und zieht dazu Fehlerströme und durchschnittliche Fehlerbeseitigungskosten je Projektphase heran.

In den Tabellen F.1 und F.2 wird das *relative maximale Kosteneinsparpotential* bzgl. der Ist-Kosten mittels Gleichung (5.3, Kapitel 4.5) berechnet. Diese Kennzahl gibt an, um wie viel die gesamten Ist-Fehlerbeseitigungskosten hätten reduziert werden können, wenn alle Fehlerursachen in der Projektphase der Fehlerentstehung beseitigt worden wären: Spezifikationsfehler also während der Anforderungsdefinition, Entwurfsfehler während des Entwurfs und Implementierungsfehler während der Implementierung.

Die Ist-Fehlerbeseitigungskosten ergeben sich nach Gleichung (4.2) in Kapitel 4.5 aus der Summe der je Phase gebildeten Fehlerbeseitigungskosten, die aus dem Produkt Anzahl beseitigter Fehler in der Phase p ($N_p^{\textit{Beseitigte Fehlerursachen}}$) und den mittleren Kosten für die Beseitigung einer Fehlerursache in der Phase p ($K_p^{\textit{Mittlere Beseitigung einer Fehlerursache}}$) resultieren.

$$K^{\textit{Ist-Fehlerbeseitigung}} = \sum_{p}^{P} (N_p^{\textit{Beseitigte Fehlerursachen}} \cdot K_p^{\textit{Mittlere Beseitigung einer Fehlerursache}}) \tag{F.1}$$

Für eine vergleichende Betrachtung ist eine Normierung der Fehlerbeseitigungskosten angebracht. Daher werden statt der absoluten Anzahl von Fehlerursachen in den Phasen die relativen Fehleranteile aus den Fehlerströmen und statt der absoluten Kosten zur Beseitigung einer Fehlerursache in einer Phase die Werte des relativen Kostenwachstums benutzt.

Die Berechnung für die Aggregation *Automobil* wird hier exemplarisch demonstriert. Bei der Berechnung der Ist-Fehlerbeseitigungskosten werden die Phasen der Fehlerbeseitigung betrachtet. Die verwendeten Fehleranteile sind aus Tabelle 5.8 (s. Kapitel 5.5), das relative Wachstum aus Tabelle 5.4 (s. Kapitel 5.4.1) entnommen:

$$K^{\textit{Ist-Fehlerbeseitigung, normiert}} = 14\% \cdot 1,0 + 48\% \cdot 1,0 + 38\% \cdot 1,6 = 0,14 + 0,48 + 0,61 = 1,23$$

Die optimierten Fehlerbeseitigungskosten ergeben sich ebenfalls nach Gleichung (4.2) in Kapitel 4.5 wie folgt aus der Verteilung der Fehleranteile auf die Phasen der Fehlerentstehung (Tabelle 5.7, s. Kapitel 5.5) sowie den Literaturwerten für den relativen Kostenwachstum in den Phasen (Tabelle 3.3, s. Kapitel 3.3):

$$K^{\textit{Optimierte Fehlerbeseitigung, normiert}} = 8\% \cdot 0,2 + 13\% \cdot 0,3 + 79\% \cdot 0,5 = 0,02 + 0,04 + 0,40 = 0,45$$

Das Kosteneinsparpotential ergibt sich nach Gleichung (4.3) wie folgt:

$$\widehat{K}\textit{osteneinsparpotential}^{\textit{Optimierung}} = \widehat{K}^{\textit{Ist-Fehlerbeseitigung}} - \widehat{K}^{\textit{Optimierte Fehlerbeseitigung}} = 1,23 - 0,45 = 0,78$$

Schließlich ergibt sich das relative Kosteneinsparpotential nach Gleichung (4.4) wie folgt:

$$\textit{Relatives } \widehat{K}\textit{osteneinsparpotential}^{\textit{Optimierung}} = \frac{\widehat{K}\textit{osteneinsparpotential}^{\textit{Optimierung}}}{K^{\textit{Ist-Fehlerbeseitigung}}} = \frac{0,78}{1,23} = 0,63\ (63\%) \tag{F.2}$$

Der Wert befindet sich in Tabelle F.1, letzte Spalte der 1. Zeile, und in der 2. Spalte der Tabelle 6.1 in Kapitel 6.1.1. Die Tabellen F.1 und F.2 wurden nach dem hier beschriebenen Vorgehen erstellt. Ein Auszug aus diesen Tabellen ist in Tabelle 6.1 in Kapitel 6.1.1 gegeben.

Projekt/-gruppe	Datenart	Projektphase: Anforderungsinspektion (Fehleranteil entstanden [4])	Entwurfsinspektion (Fehleranteil entstanden [4])	Quelltextinspektion (Fehleranteil entstanden [4])	Modultest[3] (Fehleranteil beseitigt [4])	Systemtest (Fehleranteil beseitigt [4])	Fahrzeugtest/ Akzeptanztest (Fehleranteil beseitigt [4])	Relatives maximales Kosteneinsparpotential bzgl. der Ist-Kosten
Automobil	Fehleranteile	8%	13%	79%	14%	48%	38%	63%
	Relatives Kostenwachstum[5]	0,2	0,3	0,5	1,0	1,0	1,6	
Body Controller	Fehleranteile	15%	20%	65%	20%	47%	33%	74%
	Relatives Kostenwachstum[5]	0,2	0,3	0,5	1,0	1,2	2,5	
Infotainment	Fehleranteile	5%	11%	84%	10%	57%	33%	57%
	Relatives Kostenwachstum[5]	0,2	0,3	0,5	1,0	1,0	1,2	
BC1	Fehleranteile	22%	10%	68%	31%	47%	22%	68%
	Relatives Kostenwachstum[5]	0,2	0,3	0,5	1,0	0,9	2,6	
BC2	Fehleranteile	11%	27%	62%	12%	46%	42%	76%
	Relatives Kostenwachstum[5]	0,2	0,3	0,5	1,0	1,2	2,5	
BC5	Fehleranteile	17%	6%	77%	26%	74%	-	61%
	Relatives Kostenwachstum[5]	0,2	0,3	0,5	1,0 [6]	1,2 [6]	2,5 [6]	
Inf1	Fehleranteile	5%	12%	83%	10%	57%	33%	57%
	Relatives Kostenwachstum[5]	0,2	0,3	0,5	1,0	1,0	1,2	
Inf2	Fehleranteile	2%	3%	95%	7%	58%	35%	66%
	Relatives Kostenwachstum[5]	0,2	0,3	0,5	1,0	1,3	1,8	
M1	Fehleranteile	16%	14%	70%	24%	12%	64%	66%
	Relatives Kostenwachstum[5]	0,2	0,3	0,5	1,0	0,3	1,5	
Literatur ab Modultest	F[1]	20%	39%	41%	41%	45%	14%	83%
	Relatives Kostenwachstum[2]	0,2	0,3	0,5	1,0	2,0	5,5	
Literatur	Fehleranteile entstanden [1]	20%	39%	41%				73%
	Fehleranteile beseitigt [1]	4%	17%	23%	23%	25%	8%	
	Relatives Kostenwachstum[2]	0,2	0,3	0,5	1,0	2,0	5,5	

[1] Normierter Median der Fehleranteile, *„entstanden“* aus Tabelle 3.1, *„beseitigt“* (identifiziert) aus Tabelle 3.2 in Kapitel 3.2

[2] Median des relativen Kostenwachstum der Literatur nach Testphase aus Tabelle 3.3 in Kapitel 3.3

[3] Nur Daten aus einem erneuten Modultest in der Systemtestphase

[4] Gemittelte Fehleranteile, *„entstanden“* aus Tabelle 5.7, *„beseitigt“* (identifiziert) aus Tabelle 5.8 in Kapitel 5.5. Für Zeilen *„Literatur“* ist diese Tabellenkopfzeile ohne Bedeutung.

[5] Projektspezifisches, testphasenabhängiges, relatives Kostenwachstum für die Fehlerbeseitigung, normiert auf den Modultest aus Tabelle 5.4 in Kapitel 5.4.1, für erste drei Phasen Wert unbekannt, daher wird der Median der Literatur (Tabelle 3.3 in Kapitel 3.3) angenommen

[6] Wert unbekannt, daher wird der Mittelwert der untersuchten Automobilprojekte angenommen

Tabelle F.1.: Relative, maximale Kosteneinsparpotentiale der Fehlerbeseitigung, berechnet mit projekt- und testphasenspezifischen Kostenentwicklungen

Projekt/-gruppe	Datenart	Projektphase: Anforderungs-inspektion	Entwurfs-inspektion	Quelltext-inspektion	Modultest[3]	Systemtest	Fahrzeugtest/ Akzeptanztest	Relatives maximales Kosteneinsparpotential bzgl. der Ist-Kosten
		Fehleranteil entstanden [4]			Fehleranteil beseitigt [4]			
Automobil	F	8%	13%	79%	14%	48%	38%	86%
Body Controller	F	15%	20%	65%	20%	47%	33%	86%
Infotain-ment	F	5%	11%	84%	10%	57%	33%	85%
BC1	F	22%	10%	68%	31%	47%	22%	83%
BC2	F	11%	27%	62%	12%	46%	42%	88%
BC5	F	17%	6%	77%	26%	74%	0%	75%
Inf1	F	5%	12%	83%	10%	57%	33%	85%
Inf2	F	2%	3%	95%	7%	58%	35%	85%
M1	F	16%	14%	70%	24%	12%	64%	89%
Literatur ab Modultest	F[1]	20%	39%	41%	41%	45%	14%	83%
Literatur	FE[1][5]	20%	39%	41%				73%
	FB[1][5]	4%	17%	23%	23%	25%	8%	
Kosten-wachstum Literatur (Median)	K [2]	0,2	0,3	0,5	1,0	2,0	5,5	

[1] Normierter Median der Fehleranteile, *„entstanden“* aus Tabelle 3.1, *„beseitigt“* (identifiziert) aus Tabelle 3.2 in Kapitel 3.2
[2] Median des relativen Kostenwachstums der Literatur nach Testphase aus Tabelle 3.3 in Kapitel 3.3
[3] Nur Daten aus einem erneuten Modultest in der Systemtestphase
[4] Gemittelte Fehleranteile, *„entstanden“* aus Tabelle 5.7, *„beseitigt“* (identifiziert) aus Tabelle 5.8 in Kapitel 5.5. Für Zeilen *„Literatur“* ist diese Tabellenkopfzeile ohne Bedeutung.
[5] FE = Fehleranteile entstanden, FB = Fehleranteile beseitigt
F Fehleranteile
K Relatives Kostenwachstum

Tabelle F.2.: Relative, maximale Kosteneinsparpotentiale der Fehlerbeseitigung, berechnet anhand der testphasenspezifischen Kostenentwicklung der Literatur (Median)

G. Klassengewichtung über Testziel – Erläuterungen und Berechnungen (zu Kapitel 6.2)

Ausgangspunkt der Berechnungen ist die in Tabelle G.1 dargestellte ungewichtete Fehlerverteilung des durch Kunden identifizierten Fehlverhaltens.

Fehlerklassen nach Kapitel 4.4.1	Fehleranzahl	
	Absolut	Relativ
Modultest		
Bedingung	3	4%
Grenzwert	9	10%
Systemtest		
Spezifikation [(1)]	30	35%
Systemintegration	14	16%
Systemverhalten	22	25%
Robustheit	7	8%
Systemperformanz	2	2%
Summe	87	100%

[(1)] Im Rahmen des Testprozesses frühestens im Systemtest durch Testverfahren adressierbar.

Tabelle G.1.: Vom Kunden (Fahrzeughersteller) identifiziertes Fehlverhalten im Projekt BC1. Absolute und relative Verteilung der Fehlermeldungen auf die besetzten Fehlerklassen in der Reihenfolge der frühestmöglichen Fehleridentifikation im Testprozess

Erstes Testziel: Die Fehlerpriorität ist ein diskreter Entscheidungsaspekt mit den drei Stufen *gering, mittel* und *hoch*. Die drei Fehlerprioritätsstufen werden hier mit den willkürlich gewählten Stufengewichten $N_{k,gering}$ = 1, $N_{k,mittel}$ = 2, $N_{k,hoch}$ = 5 (Testziel: hohe Fehlerpriorität sollte stärkere Berücksichtigung erfahren) bewertet (s. Tabelle G.2). Die Gewichtung wird normiert, siehe letzte Spalte Tabelle G.2.

Zweites Testziel: Das Kosteneinsparpotential durch Fehlerfrüherkennung, primär ein kontinuierlicher Entscheidungsaspekt, wird aufgrund der Anwendung auf Fehlerklassen in einen diskreten Entscheidungsaspekt überführt. Die Kostenreduktion (s. Tabelle G.3) durch die Früherkennung wird je Fehlerklasse abgeschätzt aus der Differenz der Fehlerbeseitigungskosten der Ist-Testphase und der der frühestmöglichen Testphase

Besetzte Fehlerklassen k (Klassifikationsmethode nach Kapitel 4.4)	Fehlerpriorität: Anzahl: Gering $N_{k,gering}$ (*Gewicht = 1*)	Anzahl: Mittel $N_{k,mittel}$ (*Gewicht = 2*)	Anzahl: Hoch $N_{k,hoch}$ (*Gewicht = 5*)	Gewicht $G_{k,Fehlerpriorität}$ (gewichtete Summe)	Normiertes Gewicht $G^{normiert}_{k,Fehlerpriorität}$
Modultest					
Bedingung	0	3	0	6	3%
Grenzwert	1	7	1	20	11%
Systemtest					
Spezifikation(1)	12	16	2	54	29%
Systemintegration	4	8	2	30	16%
Systemverhalten	2	15	5	57	31%
Robustheit	1	5	1	16	8%
Systemperformanz	1	1	0	3	2%
Summe	21	55	11	186	100%

(1) Im Rahmen des Testprozesses frühestens im Systemtest durch Testverfahren adressierbar.

Tabelle G.2.: Gewichtung durch den Entscheidungsaspekt *Fehlerpriorität*

zur Fehleridentifikation. Die Fehlerbeseitigungskosten werden über das testphasenspezifische relative Kostenwachstum für die Fehlerbeseitigung abgeschätzt. Das absolute Gewicht einer Fehlerklasse bzgl. des fehlerklassenoptimierten Kosteneinsparpotentials bestimmt sich aus der Fehleranzahl je Klasse und der berechneten Kostenreduktion aufgrund der angenommene Früherkennung im Testprozess. Das Verfahren geht von der Annahme aus, dass *alle* Fehlerursachen in der frühestmöglichen Testphase ab Modultest fokussiert und gefunden werden. Da die Testverfahren derzeit nicht die Effektivität aufweisen bzw. keine Totalüberdeckung mit Testfällen praktiziert werden kann, werden in der Realität allerdings nicht alle Fehlerursachen identifiziert werden.

Es wird durchgängig das relative Kostenwachstum der Literatur verwendet, da die Datenbasis aus den untersuchten Projekten zu gering als Berechnungsgrundlage erscheint (s. Tabelle 3.3 in Kapitel 3.3). Die vom Kunden identifizierten Fehler werden mit den Fehlerbeseitigungskosten des Feldeinsatzes veranschlagt, da dies die einzige Projektphase mit kundeninduzierten Fehlern ist. Das Klassengewicht wird anschließend gemäß Gleichung (B.4) normiert (s. Tabelle G.4).

Berechnung des gewichteten Fehlerprofils in Abbildung 6.2 in Kapitel 6.2 mit beiden Testzielen: Das gewichtete Fehlerprofil berechnet sich pro Fehlerklasse

Besetzte Fehlerklassen *k* [1]	Testphase der frühen Fehler-identifikation	Relatives Kostenwachstum der Testphase (W) [2]	Kostenreduktion durch frühe Fehleridentifikation $(5, 5^{(3)} - W)$
Modultest			
Bedingung	Modultest	1,0	4,5
Grenzwert	Modultest	1,0	4,5
Systemtest			
Spezifikation[4]	Systemtest	2,0	3,5
Systemintegration	Systemtest	2,0	3,5
Systemverhalten	Systemtest	2,0	3,5
Systemrobustheit	Systemtest	2,0	3,5
Systemperformanz	Systemtest	2,0	3,5

[1] Klassifikationsmethode nach Kapitel 4.4
[2] Kostenreduktion als Differenz Ist-Kosten (*Feldeinsatz*) und Frühidentifikation in Testphase der Spalte 2. Wachstumsfaktor Median der Literaturwerte(s. Kapitel 3.3, Tabelle 3.3)
[3] Median der Literaturwerte (s. Kapitel 3.3, Tabelle 3.3)
[4] Im Rahmen des Testprozesses frühestens im Systemtest durch Testverfahren adressierbar.

Tabelle G.3.: Berechnung der relativen Kostenreduktion der Fehlerbeseitigungskosten je Fehlerklasse durch frühe Fehleridentifikation des durch Kunden identifizierten Fehlverhaltens

als gewichtete Summe der normierten Gewichte aus den Entscheidungsaspekten. Hierbei muss die eingangs in der Zielsetzung erwähnte höhere Gewichtung des Entscheidungsaspekts *Fehlerpriorität* berücksichtigt werden. Daher wird das normierte Gewicht der Fehlerpriorität mit $G_{Priorität} = 2$ multipliziert.

Besetzte Fehlerklassen k (2)	Kosteneinsparpotential durch frühe Fehleridentifikation			
	Fehleranzahl N_k	Kostenreduktion durch frühe Fehleridentifikation $G_k^{durchschnittliches\ Fehlergewicht}$	Gewicht $G_{k,frühe\ Fehleridentifikation}$	Normiertes Gewicht $G_{k,frühe\ Fehleridentifikation}^{normiert}$
Modultest				
Bedingung	3	4,5	14	4%
Grenzwert	9	4,5	41	13%
Systemtest				
Spezifikation (1)	30	3,5	105	33%
Systemintegration	14	3,5	49	16%
Systemverhalten	22	3,5	77	24%
Robustheit	7	3,5	25	8%
Systemperformanz	2	3,5	7	2%
Summe	87		318	100%

(1) Im Rahmen des Testprozesses frühestens im Systemtest durch Testverfahren adressierbar.
(2) Klassifikationsmethode nach Kapitel 4.4

Tabelle G.4.: Gewichtung durch den Entscheidungsaspekt *Frühere Fehleridentifikation*

H. Herleitung der Kostenelemente für testinduzierte Qualitätskosten (zu Kapitel 7.1)

H.1. Retrospektive Quantifizierung von testinduzierten Qualitätskosten

Die Kostenelemente werden zur Ist-Kostenbestimmung des Projekts BC1 wie folgt abgeschätzt:

Prüfstand (s. Gleichung (A.5), Anhang A.2.2): Die Anschaffungskosten für die Testanlagen setzen sich aus Anschaffungskosten für *Test-Racks* sowie *Switch-Boxes* zusammen, deren Kosten aus den Projektbudgetplänen entnommen werden.

Prüfling (s. Gleichung (A.6), Anhang A.2.2): Material- und Konstruktionskosten für Prüflinge (z. B. Steuergerät einer Außenlichtsteuerung für den Systemtest) sind in den Projektbudgetplänen angegeben und werden direkt übernommen.

Testfallerstellung (s. Gleichung (A.29), Anhang A.3): Die Personalkosten der Testfallerstellung lagen nicht als retrospektive Ist-Kosten vor. Daher wurden sie durch die testphasenbezogene, prospektive Abschätzung aus den Projektbudgetplänen substituiert. Die Projektbudgetpläne gehen dabei von einem durchgehenden Einsatz des auf Erfahrung basierenden, unsystematischen Testens aus. Für die Anpassung der Testfälle in Folge von Systemänderungen wurden in den Projektbudgetplänen die Kosten nicht separiert erfasst. Es wird unterstellt, dass die Kosten hierfür in der Testfallerstellung berücksichtigt sind.

Automatisierte Testfallerstellung (s. Gleichung (A.29), Anhang A.3): Testfälle wurden nicht automatisiert erstellt. Folglich fallen hierfür keine Kosten an.

Testautomatisierung (s. Gleichung (A.31), Anhang A.3.2): Die Kosten der Testautomatisierung sind in den Projektbudgetplänen über Iterationen und Testverfahren summiert angegeben und werden direkt als Summe übernommen.

Manuelle Testdurchführung (s. Gleichung (A.32), Anhang A.3.2): Die Projektbudgetpläne weisen die Personalkosten der manuellen Testfallausführung über Iterationen und Testverfahren summiert aus, jedoch nach Testphasen getrennt. Unterschieden werden die Testphasen Modultest, Systemintegrationstest sowie Systemtest. Die Kostenaufstellungen unterscheiden nicht zwischen Testfallausführung und Fehlererkennung. Daher wird in der Kostenrechnung die Summe dieser beiden Kostenelemente

betrachtet.

Automatisierte Testfallausführung (s. Gleichung (A.32), Anhang A.3.2): Nur im Systemtest mit Integrationstest wurden Testfälle automatisiert ausgeführt. Für den Systemtest liegen die Testfallausführungskosten aus den Analysen der Datenbanken in Kapitel 5.1 vor. Die Kosten für den automatisierten Systemintegrationstest liegen nicht retrospektiv vor, daher werden prospektive Kosten entsprechend der Projektbudgetpläne verwendet.

Herstellerinduzierte und kundeninduzierte Fehlerbeseitigungskosten: Die absoluten hersteller- und kundeninduzierten Fehlerbeseitigungskosten lassen sich aufgrund der Datenlage nicht unmittelbar nach Gleichung (A.46) und folgende (s. Anhang A.3.4) abschätzen, da einerseits der für die exakte Berechnung notwenige Fehlerbeseitigungsaufwand nicht lückenlos für alle Fehlermeldungen dokumentiert wurde und andererseits eine Aufschlüsselung der Kosten nach Iteration und Testverfahren nicht vorliegt. Es verbleibt eine Betrachtung der erfassten Beseitigungskosten und der Anzahl identifizierter Fehlerursachen. Die Beseitigungskosten von Fehlerursachen, die vor und nach Auslieferung an den Kunden identifiziert wurden, werden entsprechend der Modellphilosophie getrennt betrachtet. *Herstellerinduzierte Fehlerbeseitigungskosten* berücksichtigen in Kapitel 7 die Systemtestfehler samt Integrationstest und die Fälle von Fehlverhalten aus dem wiederholten Modultest. *Kundeninduzierte Fehlerbeseitigungskosten* betrachten hier das vom Fahrzeughersteller (Kunden) im Fahrzeug- und Akzeptanztest identifizierte Fehlverhalten.

Abschätzung Anzahl identifizierter Fehlerursachen: Die Anzahl akzeptierter Fehlermeldungen ab Systemtest ist mit 492 Fehlermeldungen aus Tabelle 5.3 in Kapitel 5.1.3 bekannt. Aus Tabelle 5.8 in Kapitel 5.5 ist bekannt, dass 78% (=31%+47%) der Fälle von Fehlverhalten im Systemtest (inkl. wiederholtem Modultest, der zum Systemtest zuzurechnen ist) und 22% der Fälle von Fehlverhalten im Fahrzeug- und Akzeptanztest identifiziert wurden. Eine Differenzierung nach Fahrzeug- und Akzeptanztestfehler ergibt, dass 20% der Fälle von Fehlverhalten im Fahrzeugtest und 2% im Akzeptanztest identifiziert werden. Diese Fehlerverteilung wird zur Berechnung der Ist-Kosten zugrunde gelegt.

Das in sehr geringem Umfang exemplarisch angewendete GATE- Verfahren führte zu einer früheren Fehleridentifikation. Wegen der untergeordneten Bedeutung in Bezug auf die Gesamtkosten wird dies, wie in Kapitel 7 erläutert, nicht berücksichtigt.

Abschätzung des phasenspezifischen Beseitigungsaufwands einer Fehlerursache: Tabelle 5.4 in Kapitel 5.4.1 gibt die Beseitigungskosten einer Fehlerursache im System-, Fahrzeug- und Akzeptanztest in Projekt BC1 an. Diese Werte werden entsprechend für die Berechnung übernommen.

Abschätzung des hersteller- bzw. kundeninduzierten Fehlerbeseitigungsaufwands: Tabelle 5.4 in Kapitel 5.4.1 gibt für Projekt BC1 die relativen Beseitigungskosten für eine Fehlerursache im System-, Fahrzeug- und Akzeptanztest in Bezug zum Modultest an. Je Phase lässt sich ein mittlerer phasenspezifischer Aufwand für die Beseitigung

einer einzigen Fehlerursache durch Multiplikation des relatives Kostenwachstums der betrachteten Phase mit dem durchschnittlichen Beseitigungsaufwand einer Fehlerursache im Modultest (arithm. Mittelwert ermittelt aus den Fehlerdatenbanken) berechnen. Der gesamte Fehlerbeseitigungsaufwand einer Phase ergibt sich durch Multiplikation des phasenspezifischen Aufwands für eine Fehlerursache mit der Anzahl der in dieser Phase identifizierten Fehlerursachen. Die ermittelten Aufwandsangaben werden für TP_1 (beim Hersteller: wiederholter Modultest, Systemtest) und TP_2 (beim Kunden: Fahrzeug- und Akzeptanztest) entsprechend der Modellstruktur getrennt summiert.

Kosten für Fehlerbearbeitung und Korrekturtest (s. Gleichungen (A.37) und (A.41), Anhang A.3.4):Die Kostenanteile der Kostenelemente von Fehlerbearbeitung und Korrekturtest werden anhand der in Kapitel 5.2.2 ermittelten Kostenverteilung der Projekte Inf1 und Inf2 aus dem Fehlerbeseitigungsaufwand quantifiziert (s. Tabelle D.2 in Anhang D). Aufgrund der unzureichenden Unterscheidung zwischen den Kostenelementen *Verifikation der Fehlerbeseitigung* und *Regressionstest* in den Fehlerdatenbanken der Projekte Inf1 und Inf2 werden beide Kostenelemente nur in Summe (als Kostengruppe Korrekturtest) betrachtet. Das folglich verwendete Verhältnis beträgt: 28% für Fehleranalyse, 52% für Fehlerkorrektur und 20% für Korrekturtest. Eine Übertragung des Verhältnisses von Infotainment-Projekten auf Body Controller ist aufgrund der großen Unterschiede im Fehlerbeseitigungsaufwand höchst problematisch. Jedoch dient die Aufteilung der ermittelten Fehlerbeseitigungskosten nur der Darstellung der Kostenstruktur in Tabelle 7.1 und wirkt sich nicht auf die Kostenhöhe aus.

Fehlermanagement (s. Gleichung (A.36), Anhang A.3.4):Die Projektbudgetpläne geben einen pauschalen Wert für das Fehlermanagement an und unterscheiden somit nicht zwischen Kosten der hersteller- und kundeninduzierten Fehlerbeseitigung. Daher werden die Kosten entsprechend anteilig des identifizierten Fehlverhaltens auf beide Hauptkostengruppen verteilt.

H.2. Kostenreduktion durch Einsatz des GATE-Verfahrens im Systemtest

Die Werte folgender Kostenelemente gehen gegenüber denen in Kapitel 7.1.1 modifiziert in die Berechnung ein:

Modellerstellung und Testfallgenerierung (s. Gleichungen (A.14) und (A.29), Anhang A.2.3 bzw. A.3): Der Systemtest mit dem GATE- Verfahren basiert auf Verhaltensmodellen, für deren Erstellung Kosten anfallen. Die Verhaltensmodelle wurden, wie in der exemplarischen Anwendung im Projekt BC1 durchgeführt, als Grundlage einer automatisierten Testfallgenerierung angenommen. Die Testfallgenerierungskosten des Systemstests werden durch Einsatz von Verhaltensmodellen halbiert (Erfahrungswert gemäß Kapitel 6.3.2). Der Aufwand für Modellerstellung und Testfallgenerierung können anhand der vorliegenden Daten nicht separiert werden, jedoch entfiel der Großteil der Kosten auf die Modellerstellung und nur ein geringer Anteil auf die

automatisierte Testfallgenerierung. Daher werden die Kosten hier vollständig der Modellerstellung zugeordnet.

Die Testfallgenerierungskosten aus Modul- und Systemintegrationstests bleiben konstant.

Werkzeugkosten (s. Gleichung (A.8), Anhang A.2.2): Die Testfallerstellung aus Modellen erfolgt automatisiert mit Werkzeugen. Es wird von der Projektsituation im BC1 ausgegangen: Das Systemverhalten wurde in Microsoft Office Visio [189] modelliert, da Lizenzen hierfür vorlagen, das Werkzeug bereits installiert war und zur Nutzung bereit stand. Die Lizenzkosten sind im Beratervertrag inbegriffen und werden anteilig angerechnet. Die Modellierung wurde durch Microsoft Office Visio-Vorlagen unterstützt, welche im Rahmen eines Beratungsvertrages zur Einführung des GATE-Verfahrens zur Verfügung gestellt wurden. Die Beratungskosten, inkl. der Anteile für Lizenzen, wurden den Werkzeugkosten der Testvorbereitung zugeordnet.

Fehlerbeseitigung (s. Gleichung (A.46), Anhang A.3.4): Die Anwendung des GATE-Verfahrens in der dritten Iteration des Projekts BC1 konnte die Anzahl der Fälle des vom Kunden identifizierten Fehlverhaltens, wie beschrieben, um 35% reduzieren (s. Kapitel 6.3.2). 35% der kundeninduzierten Fehlerbeseitigungskosten werden hier auf die herstellerinduzierten Fehlerbeseitigungskosten verlagert. Es wird angenommen, dass die Fehlerbeseitigung im Systemtest durchgeführt wird.

H.3. Kostenreduktion durch Automatisierungen

Quantifizierung der Kostenelemente für die Berechnung der Qualitätskosten bei Testfallautomatisierung:

Testfallgenerierung mit automatisierter Ermittlung von Testeingabedaten im Modultest (s. Gleichung (A.29), Anhang A.3): Intendiert ist eine automatisierte Testfallgenerierung und Ermittlung der Testeingabedaten im Modultest. Die notwendigen Testeingaben bestehen je Funktion aus Funktionsparametern, Rückgabewerten während der Ausführung aufgerufener Funktionen sowie Systemvariablen, die den Systemzustand beschreiben. In Projekt BC1 besaß eine Funktion durchschnittlich etwa 0,8 Parameter (insgesamt 1.982 Funktionen mit 1.517 Parametern). Im Schnitt enthielt eine Funktion 2,2 Fallunterscheidungen (4.360 Fallunterscheidungen in 1.982 Funktionen). Aufgrund dessen, dass die Anzahl der Fallunterscheidungen höher ist als die Parameteranzahl wird angenommen, dass hauptsächlich Systemvariablen und Rückgabewerte aufgerufener Funktionen den Programmablauf bestimmen. In Stichproben wurden pro Funktion durchschnittlich 3,6 kontrollflussbestimmende Funktionsrückgaben und Systemvariablen identifiziert. In der Kostenrechnung werden somit vier ($\approx$ 0,8 Parameter + 3,6 kontrollflussbestimmende Funktionsrückgaben und Systemvariablen) zu ermittelnde Eingabewerte je Funktion angenommen.

Zusätzlich sind die Rückgabewerte der zu testenden Funktionen (erwartete Ergebnisse) manuell zu ermitteln. Es wird angenommen, dass jede Funktion einen Rückgabewert besitzt. Somit ergeben sich fünf zu ermittelnde Werte pro Funktion. Wird angenommen, dass die manuelle Ermittlung eines Eingabewertes vergleichbar komplex ist wie die manuelle Ermittlung eines Rückgabewertes, so entfallen vier Fünftel des manuellen Aufwands auf die Ermittlung der Eingabewerte und ein Fünftel auf die Ermittlung der Rückgabewerte.

Ein bei einem Fahrzeugteilezulieferer entwickelter Testfallgenerator automatisierte in einem sicherheitskritischen Nicht-Automobilprojekt die Ermittlung von Testeingabedaten zur Erreichung der Zweigüberdeckung. In diesem Projekt konnten nach Aussagen der Tester etwa 85%-90% der für die Zweigüberdeckung notwendigen Testfälle mit den zugehörigen Testeingabedaten erzeugt werden. Für die Aufwandsberechnung hier wird angenommen, dass die Testeingabewertermittlung für 87,5% (Mittelwert aus 85%-90%) der Testfälle automatisierbar sei. Die resultierenden Testfälle werden als Quelltext für Modultests [121] ausgegeben, hier in der Syntax von CUnit [165] und CppUnit [171]. Die Testfälle sind somit automatisiert ausführbar.

Der Aufwand der automatisierten Wertermittlung wird exemplarisch mit null angenommen, da eine Beaufsichtigung des Programmablaufs nicht notwendig ist. Für die verbleibenden 12,5% (=100%-87,5%) der Testfälle sind die Testeingabedaten weiterhin manuell zu ermitteln. Für die Testfallgenerierung ergibt sich also eine Kostenreduktion von 70% (=(4/5)*87,5%) [13].

In Kapitel 6.4.4 wurde festgestellt, dass bei den in Projekt BC1 angewendeten Testfallgenerierungsverfahren die automatisiert erstellten Testfälle eine geringere Wahrscheinlichkeit aufwiesen, ein Fehlverhalten zu entdecken. Dies war auf die projektspezifische Teststrategie in dem Projekt zurückzuführen. Für die hier eingesetzten Verfahren wird angenommen, dass die Testautomatisierung keine Auswirkung auf die Fehleridentifikation hat. Die Anzahl der Fälle von identifiziertem Fehlverhalten sowie die Phase der Fehleridentifikation wären also als unverändert anzusehen. Da allerdings die Fehlerbeseitigung des Modultest in der Quantifizierung der Qualitätskosten nicht berücksichtig sind, ist dies für vorliegende Betrachtung ohnehin ohne Relevanz. Folglich ergibt sich keine Änderung in den Fehlerbeseitigungskosten.

Automatisierte Testautomatisierung im Modultest (s. Gleichung (A.31), Anhang A.3.2): Werden, wie bereits beschrieben, für den Modultest automatisiert Testfälle mit Eingabewerten generiert, werden die Testfälle in einem maschinenlesbaren Format ausgegeben, das unmittelbar eine automatisierte Testfallausführung ermöglicht (Unit-Tests in Form von CUnit und CppUnit [165, 171]). Somit ist ein Teil des bisher berücksichtigten Aufwands zur Testfallautomatisierung nicht mehr erforderlich. Der Quelltext der Testfälle ist lediglich um den Quelltext für die Ausgabewerte zu ergänzen, da die Ausgabewerte in der Testfallgenerierung nicht hergeleitet werden und manuell zu ermitteln sind. Für die Abschätzung der Aufwandreduktion wurden

[13] Vier Fünftel ist der Anteil des Testfallgenerierungsaufwands, der für die Ermittlung der Eingabedaten notwendig ist. 87,5 ist der Anteil der Testfälle, für die eine automatisierte Wertermittlung möglich ist.

die vorliegenden Unit-Testfälle untersucht. Es zeigte sich, dass etwa die Hälfte des Quelltextes die Strukturierung der Testfälle sicherstellt (z. B. Klassen- und Funktionsnamensräume) und automatisiert erzeugt wird. Ein Viertel des Quelltextes stellt die Einhaltung der Vorbedingungen sicher, setzt die Eingabedaten und ruft die zu testende Funktion auf. Dieser Aufwand wird im Verhältnis der automatisiert erstellten Testfälle verringert. Das verbleibende Viertel prüft die Rückgabedaten und ist weiterhin manuell zu erstellen [14]. Für 87,5% der Testfälle können Testeingabewerte in einem zur automatisierten Ausführung notwendigen Format automatisiert bereit gestellt werden (s. *Testfallgenerierung*). Dies ergibt eine Aufwandsreduktion für die Testautomatisierung von 72% (=0,5+0,25*0,875).

Testautomatisierung im Systemtest (s. Gleichung (A.31), Anhang A.3.2):Der Automatisierungsgrad des Systemtests in Projekt BC1 betrug 67% (s. Kapitel 6.4). Durch weitergehende Testfallautomatisierungen soll der Automatisierungsgrad der Systemtestfälle um 23%-Punkte auf 90% angehoben werden. Dies erscheint realistisch, da erfahrungsgemäß, bis auf einige wenige, alle Testfälle ökonomisch sinnvoll automatisierbar sind. Für die Berechnung steigen aufgrund des modellbasierten Ansatzes die Modellerstellungskosten um 34% (=23%/67%) an.

Entwicklungskosten für Treiber und Platzhalter (s. Gleichung (A.7), Anhang A.2.2): Notwendige Platzhalter des Modultests können wie beschrieben automatisiert mit Rückgabewerten generiert werden. Treiber werden in Form von Unit-Tests realisiert. Die manuelle Erstellung der Platzhalter und Treiber kann daher nahezu vollständig eingespart werden. Für die Kostenberechnung wird angenommen, dass analog zur Testeingabedatenermittlung 87,5% der Werte automatisiert berechnet werden.

Manuelle und automatisierte Testfallausführung (s. Gleichung (A.32), Anhang A.3.2): Im Modultest wurden ursprünglich keine automatisierten Testausführungen eingesetzt. Für die Testoptimierung wird angenommen, dass die Ausführung des Modultests durch die genannten Automatisierungen zu 100% automatisiert werden kann. Im Systemtest wird, wie beschrieben, anstelle des ursprünglichen Automatisierungsgrades von 67% ein Automatisierungsgrad von 90% angesetzt. Durch die Automatisierungen im Modul- und Systemtest werden die Kosten für die manuelle Testausführung reduziert und die der automatisierten Testausführung angehoben. In Summe sind die Kosten der Testfallausführung jedoch geringer, da nach Untersuchungen in Kapitel 6.4.3 eine automatisierte Testausführung 93% weniger kostet als eine manuelle Testfallausführung.

Lizenzkosten für Testfallgenerator im Modultest: Die Lizenzkosten für den firmeninternen Testfallgenerator im Modultest liegen nicht vor, da dieser nicht in diesem Projekt eingesetzt wurde. Die Höhe der dafür veranschlagten Kosten orientiert sich an den Lizenzen aus den Kapiteln 7.1.3 und 7.1.4. Die Kosten werden dem Kostenelement *Werkzeugkosten Testvorbereitung* zugeordnet.

[14] Der Umfang des zu erstellenden Quelltextes bestimmt den Aufwand der Testautomatisierung. Dieser ist unabhängig von der Testfallerstellung (Bestimmung der notwendigen Eingaben und erwarteten Ausgaben).

Lizenzkosten für Testausführungswerkzeuge: Lizenzkosten für Testausführungswerkzeuge fallen bei Verwendung der genannten Werkzeuge CUnit und CppUnit nicht an.

I. Tabellen

	Anforderungs-definition	Entwurf	Implementierung	Test
Literatur – Projekte mit zu Projekt BC1 vergleichbaren Datenerfassungsstrukturen				
Jackson, 1995 [145]	9%	21%	27%	43%
Pomberger, 1996 [213] (Sequentielle Softwareentwicklung)	25%	25%	15%	35%
Österle, 1998 (Mehrdimensionaler abgestufter Entwurf) [34]	8%	12%	33%	47%
Österle, 1998 (Composite Design) [34]	14%	9%	41%	36%
Summerville, 2005 [239]	15%	25%	20%	40%
Mittelwert	14%	25%	20%	40%
Literatur – Projekte mit unzureichenden, aggregierten oder nicht unmittelbar vergleichbaren Daten				
Wetzel [262]	k. A.	k. A.	k. A.	30%-40%
Pomberger, 1996 [213] (Entwicklung mit Prototypen)	40%	25%	10%	25%
Boehm, 1981 [28] (Implementierung inkl. Modultest)	36%		40%	24%
Madachy, 1994 [181] (Implementierung inkl. Unit-Test)	7%	39%	31%	23%
Rubin, 1995 [227] (keine Angabe zur Implementierung)	25%	26%	k. A.	49%
Jalote, 2000 [146] (Implementierung inkl. Modultest)	28%		53%	19%
Abran, 2001 [3] (keine Angabe zur Implementierung)	26%	36%	k. A.	38%
Universität Ulm, 2007 [254] (3 Studentenprojekte)	7%	15%	37%	41%
Universität Stuttgart, 2005 [253] (10 Studentenprojekte)	28%	19%	41%	13%
Universität Stuttgart, 2005 [253] (17 Studentenprojekte)	26%	19%	55%	
Projekt BC1				
Soll (laut Budgetpläne)[(1)]	7%	10%	21%	32%
Soll ohne allgemeine Projektkosten (laut Budgetpläne)	10%	14%	30%	46%

k. A. keine Angabe
(1) Auf allgemeine Projektkosten entfallen 30% der Projektkosten

Tabelle I.1.: Verteilung der Projektkosten auf die Phasen der Softwareentwicklung (Projekt BC1 und Literatur)

Testbereiche mit großem Kosteneinsparpotential (nach subjektiver Einschätzung)	Anzahl Nennungen[1]	
	Absolut	Relativ
Modultest	8	35%
Modulintegrationstest	8	35%
Systemintegrationstest	7	30%
Systemtest	11	48%
Regressionstest	6	26%
[1] Mehrfachnennung möglich		

Tabelle I.2.: Testbereiche mit großem Kosteneinsparpotential (Umfrageergebnis, n=23) [80]

H_0-Hypothese	Die Eintragung des Fehlerbeseitigungsaufwands ist von der Fehlerpriorität unabhängig
H_1-Hypothese	Die Eintragung des Fehlerbeseitigungsaufwands ist von der Prioritätseinstufung abhängig

Projekt	Anzahl Fehler-meldungen (gesamt)	Fehler mit eingetragenem Beseitigungsaufwand (prozentualer Anteil und absolut)					Chi-Quadrat-Test [2][3]	Trendtest [2][3][4]
			Prioritätsklasse					
			Gering	Mittel	Hoch	Top[5]		
BC1	887	%	3,9	3,6	2,2	n. r.	$p > 0,05$	$p > 0,05$
		abs.[1]	8/205	21/588	2/93	n. r.		
BC2	1.117	%	57,8	61,8	66,1	45,8	$p > 0,05$	$p > 0,05$
		abs.[1]	177/306	317/513	1 81/274	11/24		
Inf1	22.760	%	4,7	10,2	4,5	5,6	$p \leq 0,01$	$p \leq 0,01$
		abs.[1]	67/1423	648/6363	509/11243	210/3731		
Inf2	1.808	%	0	0,44	0,34	0	$p > 0,05$	$p > 0,05$
		abs.[1]	0/144	2/450	3/888	0/326		

[1] abs. = Anzahl Fehlermeldungen mit Beseitigungskosten/Gesamtanzahl Fehlermeldung in der Fehlerklasse
[2] $p > 0,05$ = nicht signifikant
[3] $p \leq 0,01$ = Signifikant auf dem 1%-Signifikanzniveau
[4] *Test For a Linear Trend in Proportions* [237]
[5] n. r. = Nicht relevant, Prioritätsklasse für Projekt nicht definiert

Tabelle I.3.: Chi-Quadrat- und Trendtest zur Analyse der Unabhängigkeit des Eintragens des Fehlerbeseitigungsaufwands von der Fehlerpriorität in den untersuchten Automobilprojekten

H_0-Hypothese	Zwischen den Fehlerklassen besteht kein Unterschied bzgl. des Fehlerbeseitigungsaufwands		
H_1-Hypothese	Zwischen mindestens zwei Fehlerklassen besteht ein Unterschied bzgl. des Fehlerbeseitigungsaufwands		
Fehlerklassen	Mittlerer Rang	Kenngröße	Wert
Spezifikation	8,6	Prüfgröße H	1,02
Systemintegration	7,3	Freiheitsgrad FG	2
Systemverhalten	10,1	α	5%
		$\chi^2(\alpha = 5\%, 2\ FG)$	5,99
Ergebnis	H $< \chi^2$, daher kann H_0 nicht verworfen werden		

Tabelle I.4.: Kruskal-Wallis-Test zur Überprüfung der Signifikanz der Aufwandsunterschiede zwischen Fehlerklassen

Glossar

Anforderung (engl.: *Requirement*): Bedingung oder Fähigkeit eines → Systems, um einen Vertrag, eine Norm oder eine Abmachung zu erfüllen [137].

Anforderungsdefinition: Erste → Entwicklungsphase, gefolgt von Entwurf und Implementierung. Das Ergebnis der Anforderungsdefinition ist die → Spezifikation.

Anforderungsinspektion (engl.: *Specification Review*): Manuelle Sichtung von → Anforderungen zur Identifikation von unvollständigen, missverständlichen, widersprüchlichen und redundanten Anforderungen.

Body Controller (BC), auch *Body Control Unit (BCU)* genannt: Vereinigung mehrerer Reglersteuerungsfunktionen [61], auch *Body Controller Functions (BCF),* in einer Hardware-Komponente. Body Controller können zum Beispiel folgende Funktionen steuern: Blinker, Innen-/Außenlichter, Scheibenwischer und Zentralverriegelung.

Eingebettete Systeme (engl.: *Embedded Systems*): Software-Hardware-Einheiten [7, 17, 83, 126], die integriert in Gesamtsysteme mit ihrer Umwelt über Sensoren und Aktuatoren interagieren und Überwachungs-, Steuerungs- und Regelungsaufgaben übernehmen [43].

Entwicklungsphasen: Umfassen die → Projektphasen bis SOP: in Anlehnung an das → *V-Modell* in → *Anforderungsdefinition, Entwurf, Implementierung* und → *Test* untergliedert.

Fehleranalyse: Ermitteln von Fehlereigenschaften wie → Fehlerklasse oder Priorität sowie das Identifizieren der → Fehlerursache (Fehlerlokalisierung).

Fehlerbeseitigung (engl.: *Fault Removal*): Aktivitäten zur Beseitigung einer → Fehlerursache, welche Fehlerdokumentation (Fehlermanagement), → Fehleranalyse, Fehlerkorrektur und Korrekturtest (Verifikation der → Fehlerbeseitigung und → Regressionstest) umfasst.

Fehlerbeseitigungsaufwand: Personalaufwand in Stunden zur → Fehlerbeseitigung.

Fehlerbeseitigungskosten: Monetäre Bewertung des → Fehlerbeseitigungsaufwands in einer Währungseinheit.

Fehlerdaten: Beschreibungen des identifizierten → Fehlverhaltens eines → eingebetteten Systems sowie zum Stand und Aufwand der → Fehlerbeseitigung. Werden i.

allg. in Fehlermeldungen in → Fehlerdatenbanken gespeichert.

Fehlerdatenbank: Enthält Fehlermeldungen (engl.: *Error Reports*) und Änderungswünsche (engl.: *Change Requests*). Eine Fehlermeldung besitzt i. d. R. u. a. Fehlerbeschreibungen, Informationen zum Lebenszyklus einer Fehlermeldung und Angaben zum → Fehlerbeseitigungsaufwand.

Fehleridentifikation: Identifizieren von → Fehlverhalten.

Fehlerklasse: Gruppierungen von → Fehlverhalten anhand charakteristischer Fehlereigenschaften, z. B. → Fehlerursache oder Fehlerbild. Bekannte Klassifikationen sind *Orthogonal Defect Classification Scheme (ODC)* von IBM [53], *Hewlett-Packard Scheme* [111] oder *Classification for Software Anomalie* als IEEE Standard der NIST [138].

Fehlerkorrektur: Änderung des Systems zum Zwecke der Beseitigung einer → Fehlerursache, z. B. Implementierung einer korrekten Bedingung.

Fehlerstrom, Fehlerstromdiagramm (engl.: *Fault Detection Stream*): Angabe der Anzahl (bzw. relative Anzahl) der entstandenen und identifizierten → Fehlerursachen für jede → Projektphase und → Testphase.

Fehlerursache (engl.: *Fault*): Latente, statische Abweichung des → Systems von der → Spezifikation infolge eines → Irrtums. Bei Ausführung kann ein → Fehlerzustand resultieren [169].

Fehlervermeidung: Im engeren Sinne Durchführung von Maßnahmen zur Vermeidung von Fehlern vor der Implementierung als konstruktive Qualitätssicherung (s. Kapitel 2.4.1). Im Kontext des Modells für testinduzierte Qualitätskosten wird unter diesem Begriff vereinfachend auch die frühzeitige, entwicklungsbegleitende Identifikation von → Fehlerursachen in den Projektphasen → Anforderungsdefinition, Entwurf und Implementierung subsumiert. Beispiele für Maßnahmen dieser statischen analytischen Qualitätssicherung sind → Inspektionen wie Anforderungs-, Entwurfs- und Quelltextinspektion.

Fehlerzustand (engl.: *Error*): Von der → Spezifikation abweichender Systemzustand infolge einer ausgeführten → Fehlerursache [169].

Fehlverhalten: Hier: Oberbegriff für durch dynamische Tests ausgelöste → Fehlerzustände bzw. → (System-)Versagen.

Funktionale Fehler: Das entwickelte System weist nicht die in der → Spezifikation definierten Funktionen auf, da die Funktionen entweder gänzlich fehlen oder abweichend implementiert sind. Abgrenzung zu nicht-funktionalen → Anforderungen wie Performanz oder Robustheit.

GATE-Verfahren (*Generic and Automated Test Environment*): Ein firmenspezifisches, funktionsorientiertes und verhaltensmodellbasiertes → Testverfahren, das im Systemtest Anwendung fand. Der funktionsorientierte → Test umfasst Äquivalenzklassen- und Grenzwertanalysen.

Hauptkostengruppe: Bezeichnung für mehrere, logisch zusammenhängende → Kostengruppen.

Herstellerinduzierte Qualitätskosten: → Testinduzierte Qualitätskosten, die vor der Auslieferung an den → Kunden (Fahrzeughersteller) durch den Hersteller aufzubringen sind, um eine verlangte Qualität sicherzustellen. Dies umfasst die Durchführung von → Tests sowie die Beseitigung der vom Hersteller identifizierten → Fehlerursachen.

Infotainment Systeme: Bieten Unterhaltungs-, Multimedia- und Informationsdienste in Automobilen an, z. B. Navigations-, Kommunikations-, Multimedia-, Audio- und Sound-Dienste. Das Wort setzt sich aus den englischen Begriffen *information* und *entertainment* zusammen.

Inspektion (engl.: *Review*): Manuelle, statische Prüfmethode. Eine Inspektion, z. B. Fagan-Inspektion [176], weist verschiedenen Rollen (Leser, Inspekteur, Protokollant usw.) unterschiedliche Prüfaufgaben zu. Inspektionen werden üblicherweise für → Spezifikationen, Entwürfe oder Quelltext durchgeführt.

Irrtum (engl.: *Mistake*): Fehlhandlung eines Entwicklers, die zur Entstehung einer → Fehlerursache führt [169].

Iteration: Einmaliger Durchlauf der → Projektphasen im Zuge einer Entwicklung mit mehreren Iterationen, einer sog. iterativen Entwicklung.

Iterative Entwicklung: Ein → System wird zyklisch in → Iterationen entwickelt. Innerhalb jeder Iteration werden Optimierungen vorgenommen oder auch ein Subset der → Spezifikation verfeinert und umgesetzt. In jeder Iteration können die Phasen → Anforderungsdefinition, Entwurf, Implementierung und → Test durchlaufen werden. (s. Kapitel 2.2.3).

Kategorien von Testverfahren: → Testverfahren fokussieren vorrangig jeweils → Fehlverhalten bestimmter → Fehlerklassen [26]. Testverfahren, die dieselbe Fehlerklasse vorrangig identifizieren, werden für die fehlerklassenorientierte Testverfahrensauswahl (s. Kapitel 4.4) jeweils in einer Kategorie zusammengefasst.

Korrekturtest: → Tests zur Verifikation der → Fehlerbeseitigung und → Regressionstests (→ Kostengruppe im → Kostenmodell für testinduzierte Qualitätskosten).

Kostenelement: Qualitätskostenverursachende Einheit, hier im automobilen Test, also eine kostenrelevante Einflussgröße. Variable (kleinste Einheit) im → Kostenmodell für testinduzierte Qualitätskosten.

Kostenmodell für testinduzierte Qualitätskosten: Im Rahmen der Arbeit entwickelter Algorithmus zur Berechnung und Simulation von → testinduzierten Qualitätskosten.

Kundeninduzierte Qualitätskosten: → Testinduzierte Qualitätskosten, die nach der Auslieferung an den → Kunden in Folge eines durch den Kunden identifizierten → Fehlverhaltens anfallen. Sie können sich aus → kundeninduzierten Fehlerbeseitigungskosten sowie Fehlerfolgekosten (Kosten für Reparaturen, Imageschäden usw.) zusammensetzen (s. Kapitel 4.3.2).

Kostengruppe: Bezeichnung für mehrere, logisch zusammenhängende → Kostenelemente.

Kunde: In Bezug auf den Hersteller der → eingebetteten Systeme (i. Allg. hier Fahrzeugteilezulieferer) sind dies alle Nutzer des entwickelten → Systems, z. B. Fahrzeughersteller, Fahrzeughändler und Endverbraucher.

Modellprüfung (engl.: *Model Checking*): Die Zustände eines → Systems werden analysiert und die Ausdrücke entlang aller Pfade mit Hilfe temporaler Logik verifiziert [55]. Im Falle nicht beweisbarer Aussagen wird versucht, ein Gegenbeispiel zu identifizieren [55].

Phase der Fehlerentstehung: → Projektphase, in der eine → Fehlerursache vor Beginn der → Testphase entsteht. Fehlerursachen, die durch eine → Fehlerbeseitigung entstehen, werden in diesem Zusammenhang nicht betrachtet.

Phase der Fehleridentifikation: → Projekt- bzw. → Testphase, in der die → Fehleridentifikation und die → Fehlerbeseitigung stattfinden.

Projektphase: Die Systementwicklung untergliedert sich in verschiedene Projektphasen. In dieser Arbeit wird entsprechend dem → V-Modell zwischen → Anforderungsdefinition, Entwurf, Implementierung, → Test und ggf. Feldeinsatz unterschieden. Der → Test wird in mehreren aufeinander folgenden → Testphasen durchgeführt.

Qualitätskosten: Kosten für Qualitätssicherungsmaßnahmen, die überwiegend durch Qualitätsforderungen, die auf ein fehlerfreies Produkt zielen, entstehen. Siehe auch → testinduzierte Qualitätskosten.

Referenzdaten: Kosten- oder Aufwandsdaten, die für die i. Allg. prospektive Abschätzung von → Qualitätskosten herangezogen werden, z. B. intuitive Erfahrungswerte von Entwicklern, Daten von abgeschlossenen Projekten (Referenzprojekte), Quantifizierungen aus der Literatur.

Regressionstest: Erneute Ausführung einer Testfallmenge zur Identifikation von Nebenwirkungen einer Modifikation am → System. Je nach Art der Systemmo-

difikation kann es erforderlich sein, Regressionstestfälle anzupassen. Im Rahmen dieser Arbeit wird der Begriff vorrangig in Zusammenhang mit Testwiederholungen infolge einer Fehlerkorrektur verwendet, da die Durchführung von iterationsbedingten Regressionstests im → Kostenmodell in den → Kostenelementen *Manuelle* und *Automatisierte Testfallausführung* inbegriffen sind.

Relatives Kostenwachstum: Relation der über die → Testphasen ansteigenden Kosten für die → Fehlerbeseitigung, normiert auf die Kosten des Modultests.

SOP: Nach Abschluss der Systementwicklung wird in der Automobilindustrie mit der Serienproduktion begonnen. Der Produktionsbeginn wird als *Start of Production (SOP)* bezeichnet.

Spezifikation (engl.: *Requirement Specification*): Zusammenstellung aller von einem → System zu erfüllenden → Anforderungen.

System: In dieser Arbeit werden → eingebettete Systeme mit Softwareanteil abkürzend als Systeme bezeichnet.

Systemtypen: Klassen von → eingebetteten Systemen, hier speziell → Body Controller und → Infotainment Systeme.

Systemverhaltensfehler: Fehlerhafter modulübergreifender Entwurf eines Algorithmus, fehlende modulübergreifende Algorithmusumsetzung oder missinterpretierte Systemspezifikation, die im Normalbetrieb auf Systemebene durch Beobachtung des Systemverhaltens diagnostizierbar sind.

Szenario: Eine konkrete, exemplarische Abfolge von Aktionen und zu erwartenden Reaktionen eines → Systems, diese Abfolge kann als Vorlage für → Testfälle dienen.

Test: *„Der Test ist der überprüfbare und jederzeit wiederholbare Nachweis der Korrektheit eines Softwarebausteines relativ zu vorher festgelegten Spezifikationen*“ [68]. Der Test wird vornehmlich ab Modultestphase durch testfallgesteuerte Softwareausführung durchgeführt [137]. Tester dokumentieren dabei das → Fehlverhalten häufig im Rahmen des Testmanagements in → Fehlerdatenbanken. In der Automobilindustrie werden → Systeme meistens entsprechend dem → V-Modell vom Kleinen (Funktion, Modul) ins Große (Gesamtsystem) getestet. Dabei werden verschiedene → Testphasen durchlaufen. Ziel ist die Fehleridentifikation.

Testautomatisierung: Umfasst das Vorbereiten von → Testfällen (Testfallautomatisierung) für eine automatisierte Ausführung von Testfällen (bzw. automatisierte Testfallausführung) sowie die automatisierte Ausführung an sich.

Testfall: Beschreibung zur Durchführung dynamischer Tests. Ein Testfall umfasst Vorbedingungen, Eingabesequenzen, erwartete Ausgabesequenzen und Nachbedingungen der Systemausführung. Ein Testfall dient der Identifikation von → Fehlverhalten.

Testfallautomatisierung: Eine Vorbereitung bereits erstellter → Testfälle für die automatisierte Testfallausführung.

Testfokus: Prüfziel eines → Testverfahrens, bezogen auf Systemteile (z. B. implementierte Bedingung) oder Systemeigenschaften (z. B. einzuhaltende Antwortszeit), die → Fehlerursachen enthalten können.

Testinduzierte Qualitätskosten: Umfassen die hersteller- und kundeninduzierten Qualitätskosten. → Herstellerinduzierte Qualitätskosten sollen die Qualität eines Produktes durch Tests und Fehlerbeseitigung während der Entwicklungsphasen steigern und das Auftreten von → Fehlverhalten nach Übergaben an den → Kunden vermeiden und somit die → kundeninduzierten Qualitätskosten, bestehend aus Fehlerbeseitigungskosten, nach Auslieferung minimieren.

Testkategorie: Unterteilung der → Testverfahren, z. B. in struktur- und funktionsorientiert.

Testphase: Die → Projektphase *Test* wird in dieser Arbeit in verschiedene Testphasen unterteilt: Modultest, Modulintegrationstest, Systemintegrationstest, Systemtest, Fahrzeugintegrationstest, Fahrzeugtest, Akzeptanztest. Die Integrationstests werden mitunter auch mit den jeweils zugehörigen Testphasen zusammengefasst (z. B. Systemintegrationstest zum Systemtest). Bei Kostenbetrachtungen wird der Feldeinsatz als Testphase betrachtet. Die Testphasen werden unterteilt in Testphasen TP_1 (Modultest, Systemtest und die zugehörigen Integrationstests), die beim Hersteller durchgeführt werden, und TP_2 (Fahrzeugtest, Akzeptanztest und Feldeinsatz), die in Verantwortung und auf Kosten des → Kunden durchgeführt werden. Fahrzeug- und Fahrzeugintegrationstest können projektbedingt auch beim Kunden durchgeführt werden und werden dann den TP_1 zugerechnet.

Testprozess: Abfolge von Testaktivitäten, die wenigstens aus Testvorbereitung, Testausführung, → Fehleridentifikation und Fehler- sowie Testdokumentation besteht [241].

Teststrategie: Vorgabe von Art und Umfang der in den → Testphasen anzuwendenden Kombination von → Testverfahren zur Erreichung eines definierten Testziels [241].

Testverfahren: Algorithmus zur Herleitung von → Testfällen für den dynamischen Test der Software. Hierbei werden Testeingabedaten sowie Testausgabedaten bestimmt.

Testziel: Testkriterien, die durch die Ausführung des → Testprozesses umgesetzt werden sollen.

Überdeckungskriterium: Häufig verwendetes Qualitätsmaß für → Testfälle [209]. Eine Testfallmenge wird als ausreichend bezeichnet, wenn definierte Programmstrukturen (z. B. Programmpfade, Bedingungen) geprüft werden.

Validierung: *„Der Prozess der Beurteilung eines Systems oder einer Komponente während oder am Ende des Entwicklungsprozesses, mit dem Ziel, festzustellen, ob die spezifizierten → Anforderungen erfüllt sind“* [137]. *„Bauen wir das richtige System?“* [28]

Verhaltensmodell: Formalisiertes Modell zur Beschreibung eines allgemeinen Systemverhalten im Gegensatz zu exemplarischen Szenarien. Für theoretische Grundlagen und Eigenschaften zu Modellen sei auf Broy [41], Stachowiak [243] sowie Pretschner [215] verwiesen.

Verhaltensmodellbasierte Testverfahren: → Testfälle werden aus → Verhaltensmodellen abgeleitet. Für weitere Informationen zum modellbasierten Test (engl.: *Model Based Testing*) sei auf Pretschner [215] verwiesen.

Verifikation: *„(1) der Prozess der Beurteilung eines → Systems oder einer Komponente mit dem Ziel, festzustellen, ob die Resultate einer gegebenen Entwicklungsphase den Vorgaben für diese Phase entsprechen, (2) der formale Beweis der Korrektheit eines Programms“* [137]. *„Bauen wir das System richtig?“* [28]

Versagen/Ausfall (engl.: *Failure*): Nicht-Funktionieren bzw. von der → Spezifikation abweichendes Verhalten infolge eines zuvor eingetretenen → Fehlerzustands [169]. Versagen bezeichnet eine zeitweise, Ausfall eine dauerhafte Fehlfunktion.

V-Modell: Phasenorientiertes Vorgehensmodell zur Entwicklung von → Systemen (s. Kapitel 2.2.1). Das V-Modell kann iterativ angewendet werden.

Abkürzungsverzeichnis

BC	Body Controller (hier: Steuergerät)
BCF	Body Controller-Funktion
GATE	Generic and Automated Test Environment
Inf	Infotainment
k. A.	keine Angabe
LOC	Lines of Code (KLOC = 1.000 LOC, MLOC = 1.000.000 LOC)
M	Mischprojekt mit Body Controller- und Infotainment-Komponenten
SOP	Start of Production

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] T. Abdel-Hamid und S. Madnick. *Software Project Dynamics: An Integrated Approach*. Prentice Hall, Englewood Cliffs, NJ, USA, 1991.

[2] K. Abernethy, J. Kelly, A. Sobel, J. Kiper und J. Powell. Technology Transfer Issues for Formal Methods of Software Specifications (Presentation). *Proceedings of the 13th Conference on Software Engineering Education & Training*, Seiten 23–31, 2000.

[3] A. Abran, C. Symons und S. Oligny. An Overview of COSMIC-FFP Field Trial Results. *Proceedings of the ESCOM Conference*, Seiten 2–4, 2001.

[4] M. Aichner. *Kostenbasierte Evaluierung eines statischen Analyse-Werkzeugs*. Diplomarbeit. Technische Universität München, 2006.

[5] T. Alspaugh, S. Faulk, K. Britton, R. Parker und D. Parnas. *Software Requirements for the A-7E Aircraft.* DTIC Research Report ADA255746. Naval Research Laboratory, 1992.

[6] P. Ammann und J. Offutt. Using Formal Methods to Derive Test Frames in Category-Partitiontesting. *Proceedings of the 9th Annual Conference on Computer Assurance*, Seiten 69–79, 1994.

[7] O. Anton, B. Lercher und T. Oertli. Funktionsmodellierung und Dynamikoptimierung beim mechatronischen Systementwurf. *iwb Seminarberichte*, 60:1–17, 2005.

[8] H. Balzert. *Lehrbuch der Software-Technik. 2. Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum, Akademischer Verlag, Heidelberg, 1998.

[9] L. Baresi, A. Orso und M. Pezzè. Introducing Formal Specification Methods in Industrial Practice. *Proceedings of the 19th International Conference on Software Engineering*, Seiten 56–66, 1997.

[10] V. Basili, L. Briand und W. Melo. A Validation of Object-Oriented Design Metrics as Quality Indicators. *IEEE Transactions on Software Engineering*, 22(10):751–761, 1996.

[11] V. Basili und B. Perricone. Software Errors and Complexity: An Empirical Investigation. *Communications of the ACM*, 27(1):42–52, 1984.

[12] V. Basili und R. Selby. Comparing the Effectiveness of Software Testing Strategies. *IEEE Transactions on Software Engineering*, 13(12):1278–1296, 1987.

[13] T. Bauer, F. Bohr, D. Landmann, T. Beletski, R. Eschbach und J. Poore. From Requirements to Statistical Testing of Embedded Systems. *Proceedings of the 4th International Workshop on Software Engineering for Automotive Systems*, Seite 3, 2007.

[14] A. Beer und M. Heindl. Issues in Testing Dependable Event-Based Systems at a Systems Integration Company. *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES2007)*, 178:1093–1100, 2007.

[15] B. Beizer. *Software Testing Techniques*. Van Nostrand Reinhold Co., New York, NY, USA, 1990.

[16] A. Belinfante, J. Feenstra, R. de Vries, J. Tretmans, N. Goga, L. Feijs, S. Mauw und L. Heerink. Formal Test Automation: A Simple Experiment. *International Workshop on Testing of Communicating Systems*, 12:179–196, 1999.

[17] K. Bender. *Embedded Systems - Qualitätsorientierte Entwicklung*. Springer-Verlag, New York, Secaucus, NJ, USA, 2007.

[18] G. Beneken, U. Hammerschall, M. Broy, M. Cengarle, J. Jürjens, B. Rumpe und M. Schoenmakers. Componentware - State of the Art 2003. *Background Paper for the Understanding Components Workshop*, Seiten 7–9, 2003.

[19] J. Bergmann. Test des Kontrollverhaltens eingebetteter Systeme der dezentralen Automatisierungstechnik. *Kongressband zu Kongress für industrielle Busse und Netze (iNet97)*, Seiten 114–123, 1997.

[20] E. Bernard, B. Legeard, X. Luck und F. Peureux. Generation of Test Sequences from Formal Specifications: GSM 11-11 Standard Case Study. *Software Practice and Experience*, 34(10):915–948, 2004.

[21] D. Berry. Natural Language and Requirements Engineering. *International Workshop on Requirements Engineering, Imperial College, London, UK*, 2001.

[22] D. Beyer. *Formale Verifikation von Realzeit-Systemen mittels Cottbus Timed Automata*. Verlag Mensch & Buch, Berlin, 2002.

[23] S. Biffl. *Beurteilung, Zertifizierung und Verbesserung von Prozessen*. Vorlesungsskript. Quality Software Engineering Research. Technische Universität Wien, 2004.

[24] S. Biffl. *Der Testprozess*. Vorlesungsskript. Quality Software Engineering Research. Technische Universität Wien, 2004.

[25] S. Biffl. *Testautomatisierung*. Vorlesungsskript. Quality Software Engineering Research. Technische Universität Wien, 2004.

[26] R. Binder. *Testing Object-Oriented Systems: Models, Patterns and Tools*. Addison-Wesley Verlag, Reading, USA, 1999.

[27] M. Blackburn, R. Busser und A. Nauman. Why Model-Based Test Automation is Different and What You Should Know to Get Started. *Proceedings of the International Conference on Practical Software Quality*, 2004.

[28] B. Boehm. *Software Engineering Economics*. Prentice Hall, Englewood Cliffs, NJ, USA, 1981.

[29] B. Boehm. A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes*, 11(4):14–24, 1986.

[30] B. Boehm und V. Basili. Software defect reduction top 10 list. *Computer*, 34(1):135–137, 2001.

[31] B. Boehm, L. Huang, A. Jain und R. Madachy. The ROI of Software Dependability: The iDAVE Model. *Software, IEEE*, 21(3):54–61, 2004.

[32] J. Bowen und M. Hinchey. Seven More Myths of Formal Methods. *Software, IEEE*, 12(4):34–41, 1995.

[33] J. Bowen und M. Hinchey. Ten Commandments of Formal Methods. *Computer. IEEE Computer Society, Washington D.C., USA*, 28:56–63, 1995.

[34] L. Brecht, C. Legner, S. Muschter und H. Österle. Prozeßführung mit nichtfinanziellen Führungsgrößen. Konzept und Erfahrungen. *Controlling*, 10(5):286–294, 1998.

[35] M. Breitling. *Formale Fehlermodellierung für verteilte reaktive Systeme*. Dissertation. Technische Universität München, 2001.

[36] L. Briand, S. Morasca und V. Basili. *Defining and Validating High-level Design Metrics*. Technical Report: CS-TR-3301. University of Maryland, 1994.

[37] A. Bröhl und W. Dröschel. *Das V-Modell: Der Standard für die Softwareentwicklung mit Praxisleitfaden*. R. Oldenburg Verlag, München, 1995.

[38] M. Broy, M. Jarke, M. Nagl und D. Rombach. Manifest: Strategische Bedeutung des Software Engineering in Deutschland. *Informatik-Spektrum*, 29(3):210–221, 2006.

[39] M. Broy und A. Rausch. Das neue V-Modell XT. *Informatik-Spektrum*, 28(3):220–229, 2005.

[40] M. Broy, D. Rombach, P. Stahl und M. Friedewald. *Analyse und Evaluation der Software-Entwicklung in Deutschland*. Eine Studie für das Bundesministerium für Bildung und Forschung, 2000.

[41] M. Broy und B. Rumpe. Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. *Informatik-Spektrum*, 30(1):3–18, 2007.

[42] M. Broy und T. Stauner. Requirements Engineering for Embedded Systems. *Informationstechnik und Technische Informatik*, 41:7–11, 1999.

[43] M. Broy, M. von der Beeck und I. Krüger. *SOFTBED: Problemanalyse für ein Großverbundprojekt - Systemtechnik Automobil-Software für eingebettete Systeme*. Technische Universität München, Institut für Informatik, 1998.

[44] M. Bruhn und D. Georgi. *Kosten und Nutzen des Qualitätsmanagements: Grundlagen, Methoden, Fallbeispiele*. Carl Hanser Verlag, München, 1999.

[45] F. Brunner. Steigerung der Effizienz durch Qualitätskostenanalyse. *IO Management Zeitschrift*, 60(7/8):35–38, 1991.

[46] Bundesministerium für Forschung und Technologie. *Wirtschaft, Wissenschaft und Technik*. Initiative zur Förderung der Software-Technologie. Bundesministerium für Forschung und Technologie, 1994.

[47] M. Burghardt. *Projektmanagement: Leitfaden für die Planung, Überwachung und Steuerung von Entwicklungsprojekten*. Publicis MCD Verlag, Erlangen, 1997.

[48] A. Burst, M. Wolff, M. Kühl und K. Müller-Glaser. A Rapid Prototyping Environment for the Concurrent Development of Mechatronic Systems. *Proceedings of the European Concurrent Engineering Conference (ECEC 1998)*, Seiten 237–241, 1998.

[49] R. Buschermöhle und T. Wolf. *VSEK Bedarfsanalyse Fahrzeugtechnik*. Eine Veröffentlichung des VSEK-Projekts. FKZ 01 IS C65, 2004.

[50] M. Cartwright und M. Shepperd. An Empirical Investigation of an Object-Oriented Software System. *IEEE Transactions on Software Engineering*, Seiten 786–796, 2000.

[51] Case Western Reserve University. *Uncertainties and Error Propagation*, Appendix V. Case Western Reserve University, 2004.

[52] J. Chilenski, S. Miller, B. Airplanes und W. Seattle. Applicability of Modified Condition/Decision Coverage to Softwaretesting. *Software Engineering Journal*, 9(5):193–200, 1994.

[53] R. Chillarege, I. Bhandari, J. Chaar, M. Halliday, D. Moebus, B. Ray und M. Wong. Orthogonal Defect Classification - A Concept for In-Process Measurements. *IEEE Transactions on Software Engineering*, 18(11):943–956, 1992.

[54] D. Clarke, T. Jeron, V. Rusu und E. Zinovieva. Automated Test and Oracle Generation for Smart-Card Applications. *Smart Card Programming and Security: Proceedings of International Conference on Research in Smart Cards, E-smart 2001*, Seiten 58–70, 2001.

[55] E. Clarke, O. Grumberg und D. Peled. *Model Checking*. Springer-Verlag, Berlin, 1999.

[56] D. Cohen, S. Dalal, M. Fredman und G. Patton. The AETG System: An Approach to Testing based on Combinatorial Design. *IEEE Transactions on Software Engineering*, 23(7):437–444, 1997.

[57] D. Cohen, S. Dalal, J. Parelius, G. Patton und N. Bellcore. The Combinatorial Design Approach to Automatic Test Generation. *Software, IEEE*, 13(5):83–88, 1996.

[58] J. Collofello und S. Woodfield. Evaluating the Effectiveness of Reliability-Assurance Techniques. *Journal of Systems and Software. Elsevier Science, New York, NY, USA*, 9(3):191–195, 1989.

[59] M. Conrad. *Modellbasierter Test eingebetteter Software im Automobil: Auswahl und Beschreibung von Testszenarien*. Dissertation. Deutscher Universitäts-Verlag, Wiesbaden, 2004.

[60] M. Conrad und H. Dörr. Einsatz von modellbasierten Entwicklungstechniken in sicherheitsrelevanten Anwendungen: Herausforderungen und Lösungsansätze. *Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme II*, 2006.

[61] Continental. Continental Body Controller Units. `http://www.vdo.de/generator/www/de/de/vdo/main/products_solutions/commercial_vehicles/body_controller_solutions/body_controller_solutions_de.html` (geprüft: 25.03.2010), 2009.

[62] I. Craggs, M. Sardis und T. Heuillard. AGEDIS Case Studies: Model-Based Testing in Industry. *Proceedings of the 1st European Conference on Model Driven Software Engineering, 106*, 117, 2004.

[63] P. Crosby. *Quality is Free*. Mentor Books, Denver, CO, USA, 1980.

[64] S. Dalal, A. Jain, N. Karunanithi, J. Leaton, C. Lott, G. Patton, B. Horowitz, und M. Bellcore. Model-Based Testing in Practice. *Proceedings of the 1999 International Conference on Software Engineering*, Seiten 285–294, 1999.

[65] B. Dale und J. Plunkett. *Quality Costing*. Gower Publishing, Farnham, UK, 1999.

[66] W. Damm, A. Votintseva, A. Metzner, B. Josko, T. Peikenkamp und E. Böde. Boosting Re-use of Embedded Automotive Applications Through Rich Components. *Proceedings of Foundations of Interface Technologies*, Seiten 549–554, 2005.

[67] J. Dannenberg und C. Kleinhans. The Coming Age of Collaboration in the Automotive Industry. *Mercer Management Journal*, 17:88–94, 2004.

[68] E. Denert und J. Siedersleben. *Software-Engineering: Methodische Projektabwicklung*. Springer-Verlag, Berlin, 1991.

[69] Deutsche Gesellschaft für Qualität e.V. Qualitätskosten - Rahmenempfehlungen zu ihrer Definition, Erfassung und Beurteilung. *DGQ Schrift 14-17*, 5, 1985.

[70] T. Dietmüller und I. Mühlendahl. *Ermittlung des wirtschaflichen Nutzens präventiver Qualitätsmanagement-Methoden in Serienentwicklungsprojekten*. Technischer Bericht. Technische Universität Berlin, 2007.

[71] E. Dijkstra. *Notes on Structured Programming*. Technische Hogeschool Eindhoven, Deptartment of Mathematics, 1970.

[72] DIN. *DIN 61508 (VDE 0803): Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme*. VDE Verlag, Berlin, 2008.

[73] Dr. Dobbs Portal. Getting Experienced. `http://www.ddj.com/development-tools/198000606?pgno=4` (geprüft: 25.03.2010), 2007.

[74] W. Dröschel, W. Heuser und R. Midderhoff. *Inkrementelle und objektorientierte Vorgehensweisen mit dem V-Modell 97*. R. Oldenbourg Verlag, München, 1998.

[75] J. Dushina, M. Benjamin und D. Geist. Semi-Formal Test Generation With Genevieve. *Proceedings of the 38th Conference on Design Automation*, Seiten 617–622, 2001.

[76] E. Dustin, J. Rashka und J. Paul. *Software automatisch testen: Verfahren, Handhabung und Leistung*. Springer-Verlag, Berlin, 2001.

[77] S. Easterbrook und J. Callahan. Formal Methods for Verification and Validation of Partial Specifications: A Case Study. *The Journal of Systems & Software. Elsevier Science, New York, NY, USA*, 40(3):199–210, 1998.

[78] C. Ebert und T. Liedtke. *Software Realiability Prediction for Large Systems*. Alcatel SELAG, Stuttgart, 1992.

[79] J. Ebert und G. Rothhardt. *Software-Testwirtschaftlichkeit*. TECO, Universität Karlsruhe, 2006.

[80] B. Elbel, O. Mäckel, M. Rothfelder und J.-U. Schuster. *Testen hybrider Systeme bei Siemens. Auswertung der Fragebögen aus der Umfrage im Geschäftsjahr 00/01*. Whitepaper. Siemens AG, 2001.

[81] S. Elfgen. *Konzept zur Implementierung eines Testwerkzeugs für die Automatisierung von Black-Box-Testverfahren*. GRIN Verlag, München, 2007.

[82] A. Endres. An Analysis of Errors and Their Causes in System Programs. *ACM SIGPLAN Notices. ACM New York, NY, USA*, 10(6):327–336, 1975.

[83] EQUAL. Methoden zur Unterstützung der entwicklungsbegleitenden Qualitätssicherung von eingebetteter Software (EQUAL). `http://www.embedded-quality.de/main/equal/tx_equal.htm` (geprüft: 25.03.2010), 2009.

[84] D. Ernst. *Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme*. Technischer Bericht. Universität Berlin, 1997.

[85] M. Fagan. Inspecting Software Design and Code. *Datamation*, 23(10):133–144, 1973.

[86] M. Fagan. Design and Code Inspections to Reduce Errors in Program Development. *IBM Systems Journal*, 15(3):182–211, 1976.

[87] E. Farchi, A. Hartman und S. Pinter. Using A Model-Based Test Generator to Test for Standard Conformance. *IBM Systems Journal*, 41(1):89–110, 2002.

[88] S. Faulk, L. Finneran, J. Kirby Jr, S. Shah und J. Sutton. Experience Applying the CoRE Method to the Lockheed C-130J Software Requirements. *Proceedings of the 9th Annual Conference on Computer Assurance*, Seiten 3–8, 1994.

[89] A. Feigenbaum. *Total Quality Control*. McGraw-Hill Professional, New York, USA, 1991.

[90] D. Fischer, P. Guist, A. Loibl, K. Moll und G. Wermuth. *Diagnose*. Anlage 4 zum Projektendbericht an die Bayerische Forschungsstiftung. Synergieorientierte Kooperation zwischen Automobilherstellern und Vertriebsnetz in Bayern, 1999.

[91] A. Fleischmann. *Modellbasierte Formalisierung von Anforderungen für eingebettete Systeme im Automotive-Bereich*. Dissertation. Technische Universität München, 2008.

[92] A. Fleischmann, G. Bönisch, A. Loschpe und K. Schwanitz. *Grobentwurf des Prozessmodells für das Anforderungsmanagement*. Unveröffentlichtes Ergebnisdokument des Teilprojektes TP1 im Forschungsverbund Mobilsoft. Technische Universität München, 2006.

[93] A. Fleischmann, E. Geisberger und M. Pister. *Herausforderungen für das Requirements Engineering eingebetteter Systeme*. Technischer Bericht. TUM-I0414. Technischen Universität München, 2004.

[94] FMEInfRes. *Formal Methods Application Database, a Database Containing Descriptions of Industrial Applications of Formal Methods*. Project FMEInfRes. Formal Methods Europe, 1999.

[95] L. Fournier, A. Koyfman und M. Levinger. Developing an Architecture Validation Suite. *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, 99:189–193, 1999.

[96] P. Frankl und O. Iakounenko. Further Empirical Studies of Test Effectiveness. *Proceedings of the 6th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Seiten 153–162, 1998.

[97] B. Freimut, L. Briand und F. Vollei. Determining Inspection Cost-Effectiveness by Combining Project Data and Expert Opinion. *IEEE Transactions on Software Engineering*, Seiten 1074–1092, 2005.

[98] B. Freimut, C. Denger und M. Ketterer. An Industrial Case Study of Implementing and Validating Defect Classification for Process Improvement and Quality Management. *Proceedings of the 11th IEEE International Symposium on Software Metrics*, Seite 19, 2005.

[99] B. Freimut, B. Klein, O. Laitenberger und G. Ruhe. Measurable Software Quality Improvement through Innovative Software Inspection Technologies at Allianz Life Assurance. *Proceedings of the 11th European Software Control and Metrics (ESCOM 2000)*, Seiten 345–353, 2000.

[100] M. Friske und B. Schlingloff. Abdeckungskriterien in der modellbasierten Testfallgenerierung: Stand der Technik und Perspektiven. *Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme II*, 2006.

[101] H. Fujiwara und T. Shimono. On the Acceleration of Test Generation Algorithms. *Transactions on Computers*, 100(32):1137–1144, 1983.

[102] S. Fujiwara, G. Bochmann, F. Khendek, M. Amalou und A. Ghedamsi. Test Selection based on Finite State Models. *IEEE Transactions on Software Engineering*, 17(6):591–603, 1991.

[103] H. Gaus. Die Zukunft der Automobil-Elektrik/-Elektronik. *Proceedings der Tagung Fortschritte in der Automobil-Elektronik*, 1997.

[104] W. Geiger und W. Kotte. *Handbuch Qualität - Grundlagen und Elemente des Qualitätsmanagements: Systeme - Perspektiven*. Vieweg Verlag, Wiesbaden, 2005.

[105] E. Geisberger. *Requirements Engineering eingebetteter Systeme: Ein interdisziplinärer Modellierungsansatz*. Dissertation. Technische Universität München, 2005.

[106] J. Gericke und M. Wiemann. Effort Reduction of Quality Assurance for Complex Systems through Fault Estimation. *Proceedings of the 2nd International Conference on Informatics*, Seiten 8–14, 2007.

[107] C. Ghezzi, M. Jazayeri und D. Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, PTR Upper Saddle River, NJ, USA, 2002.

[108] GI - Gesellschaft für Informatik. dpa Meldung vom 30.09.2003 zur 33. Jahrestagung der GI. `http://www.informatik2003.de/Pressemitteilungen/EinladungPKI20032.pdf` (geprüft: 25.03.2010), 2003.

[109] T. Gilb und S. Finzi. *Principles of Software Engineering Management*. Addison-Wesley Verlag, München, 1988.

[110] R. Glass. The Standish Report: Does It Really Describe a Software Crisis? *Communications of the ACM*, 49(8):15–16, 2006.

[111] R. Grady. *Practical Software Metrics for Project Management and Process Improvement*. Prentice Hall, Upper Saddle River, NJ, USA, 1992.

[112] D. Graham und M. Fewster. *Automating Software Testing*. Addison-Wesley Verlag, Reading, MA, USA, 1999.

[113] K. Grimm und M. Grochtmann. Classification Trees for Partition Testing. *Software Testing, Verification and Reliability*, 3(2):63–82, 1993.

[114] M. Grindal, B. Lindström, J. Offutt und S. Andler. An Evaluation of Combination Strategies for Test Case Selection. *Empirical Software Engineering*, 11(4):583–611, 2006.

[115] M. Grindal, J. Offutt und S. Andler. Combination Testing Strategies: A Survey. *Software Testing Verification and Reliability*, 15(3):167, 2005.

[116] C. Gühmann und J. Riese. *Testautomatisierung in der Hardware-in-the-Loop-Simulation*. IAV GmbH, 2002.

[117] A. Haffner und E. Westkämper. *Ein Modell zur Bestimmung der monetären Einsparungspotenziale bei der Durchführung einer Fehlermöglichkeits- und Einflussanalyse (FMEA)*. Jost-Jetter Verlag, Heimsheim, 2005.

[118] F. Haist und H. Fromm. *Qualität im Unternehmen: Prinzipien - Methoden - Techniken*. Carl Hanser Verlag, München, 1989.

[119] A. Hall. Seven Myths of Formal Methods. *Software, IEEE*, 7(5):11–19, 1990.

[120] A. Hall. Using Formal Methods to Develop an ATC Information System. *Software, IEEE*, 13(2):66–76, 1996.

[121] P. Hamill. *Unit Test Frameworks*. O'Reilly, Sebastopol, CA, USA, 2004.

[122] T. Hampp und M. Knauß. Eine Untersuchung über Korrekturkosten von Software-Fehlern. *Softwaretechnik-Trends*, 28:7–12, 2008.

[123] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. Shtull-Trauring und M. Trakhtenbrot. Statemate: A Working Environment for the Development of Complex Reactive Systems. *IEEE Transactions on Software Engineering*, 16(4):403–414, 1990.

[124] J. Hartmann, S. Rittmann, D. Wild und P. Scholz. Formal Incremental Requirements Specification of Service-Oriented Automotive Software Systems. *Service-Oriented System Engineering, 2006. SOSE'06. Second IEEE International Workshop*, Seiten 130–133, 2006.

[125] N. Hartmann. *Automation des Tests eingebetteter Systeme am Beispiel der Kraftfahrzeugelektronik.* Dissertation. Universität Karlsruhe, 2001.

[126] B. Heimann, W. Gerth und K. Popp. *Mechatronik*. Fachbuchverlag, Leipzig, 2003.

[127] M. Heimdahl und N. Leveson. Completeness and Consistency Analysis of State-Based Requirements. *Proceedings of the 17th International Conference on Software Engineering*, Seiten 3–14, 1995.

[128] M. Heimdahl und N. Leveson. Completeness and Consistency in Hierarchical State-Based Requirements. *IEEE Transactions on Software Engineering*, 22(6):363–377, 1996.

[129] M. Hennell, D. Hedley und I. Riddell. The LDRA Software Testbeds: Their Roles and Capabilities. *Proceedings of IEEE Soft-Fair*, 83:69–77, 1983.

[130] Hewlett-Packard GmbH. *Projektanalyse für den möglichen Einsatz von HP Quality Center*. Hewlett-Packard GmbH, 2008.

[131] J. Hopcroft und J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Verlag, Reading, MA, USA, 1979.

[132] H. Hörcher. Testautomation in der Telekommunikation. *Softwaretechnik-Trends*, 19:13–14, 1999.

[133] M. Huhn und T. Mücke. *Comparing Heuristics for Model based Testsuite Generation*. Technischer Bericht. Technische Universität Braunschweig, 2006.

[134] M. Hutchins, H. Foster, T. Goradia und T. Ostrand. Experiments of the Effectiveness of Dataflow- and Controlflow-Based Test Adequacy Criteria. *Proceedings of the 16th International Conference on Software Engineering*, Seiten 191–200, 1994.

[135] IABG Industrieanlagen-Betriebsgesellschaft mbH. V-Model XT: Part 1: Fundamentals of the V-Modell. `http://v-modell.iabg.de/index.php?option=com_docman&task=doc_download&gid=32` (geprüft: 25.03.2010). München, 2004.

[136] IBM. IBM Rational ClearQuest. `http://www-306.ibm.com/software/awdtools/clearquest/` (geprüft: 25.03.2010), 2009.

[137] IEEE. *IEEE Standard 610.12-1990. Standard Glossary of Software Engineering Terminology*. The Institute of Electrical and Electronics Engineers, 1990.

[138] IEEE. *IEEE Standard 1044-1993. Standard Classification for Software Anomalies*. The Institute of Electrical and Electronics Engineers, 1995.

[139] T. Illgen. *Entwicklung eingebetteter Systeme in der Automotive-Industrie*. Universität Leipzig. Lehrstuhl für Angewandte Telematik/E-Business. LPZ Distinguished Lecture Series. Vortrag, 2004.

[140] imbus AG. imbus TestBench. `http://www.imbus.de/produkte/testbench/index.shtml` (geprüft: 25.03.2010), 2009.

[141] is Industrie Software GmbH. *Test Center*. is Industrie Software GmbH, 2008.

[142] ISO - International Organization for Standardization. *DIN 55350 - Begriffe zum Qualitätsmanagement*. VDE Verlag, Berlin, 2004.

[143] ISO - International Organization for Standardization. *ISO 10014 - Qualitätsmanagementsysteme - Leitfaden zur Erzielung finanziellen und wirtschaftlichen Nutzens*, In Guidelines for Managing the Economics of Quality. ISO - International Organization for Standardization, 2009.

[144] ITU-TS. *ITU-TS Recommendation Z.120: Message Sequence Chart (MSC)*. ITU-TS, Geneva, 1996.

[145] M. Jackson. *Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices*. ACM Press/Addison-Wesley Verlag, New York, NY, USA, 1995.

[146] P. Jalote. *CMM in Practice: Processes for Executing Software Projects at Infosys*. Addison-Wesley Verlag, Reading, MA, USA, 2000.

[147] B. Johnson. *Design and Analysis of Fault-Tolerant Digital Systems*. Addison-Wesley Verlag, Reading, MA, USA, 1989.

[148] C. Jones. *Applied Software Measurement: Assuring Productivity and Quality*. McGraw-Hill, New York, 1991.

[149] P. Jorgensen. *Software Testing: A Craftsman's Approach*. CRC Press, 2002.

[150] S. Jungmayr. Testbarkeit in den frühen Projektphasen. *Softwaretechnik-Trends*, 21:18–19, 2001.

[151] N. Juristo, A. Moreno und S. Vegas. Reviewing 25 Years of Testing Technique Experiments. *Empirical Software Engineering*, 9(1):7–44, 2004.

[152] B. Kaiser. Integration von Sicherheits- und Zuverlässigkeitsmodellen in den Entwicklungsprozess eingebetteter Systeme. *Softwaretechnik-Trends*, 22(4), 2002.

[153] B. Kaiser. A Fault-Tree Semantics to Model Software-Controlled Systems. *Softwaretechnik-Trends*, 23(3), 2003.

[154] G. Kamiske und A. Tomys. Qualitätsmanagement verbessert den Wirkungsgrad der Produktion. *Zeitschrift für wirtschaftliche Fertigung und Automatisierung*, 88:41, 1993.

[155] S. Kan, S. Dull, D. Amundson, R. Lindner und R. Hedger. AS/400 Software Quality Management. *IBM Systems Journal*, 33(1):62–88, 1994.

[156] C. Kaner, J. Bach und B. Pettichord. *Lessons Learned in Software Testing*. John Wiley & Sons, New York, NY, USA, 2001.

[157] L. Karg und D. Voelz. Kostenaspekte bei der Modellierung von Softwarequalität. *Workshop-Band Software-Qualitätsmodellierung und -bewertung (SQMB08)*, Seite 14, 2008.

[158] J. Kelly, J. Sherif und J. Hops. An Analysis of Defect Densities Found During Software Inspections. *Journal of Systems and Software. Elsevier Science, New York, NY, USA*, 17(2):111–117, 1992.

[159] M. Kläs, T. Beletski und A. Sarishvili. *AP 3.1: Effektivität von QS-Maßnahmen. Stand der Wissenschaft*. Forschungsbericht. Fraunhofer IESE, 2007.

[160] D. Knuth. An Empirical Study of Fortran Programs. *Software-Practice and Experience*, 1:105–133, 1971.

[161] A. Koc. *Entscheidungsunterstützung zur Planung der Software-Qualitätssicherung in mechatronischen Produkten*. Dissertation. Technische Universität München, 2003.

[162] M. Koller. *Test Case Generation from Automotive UML Models*. Dissertation. Universität Passau, 2006.

[163] T. Koomen und M. Pol. *Test Process Improvement: A Practical Step-By-Step Guide to Structured Testing*. Addison-Wesley Longman Publishing, Boston, MA, USA, 1999.

[164] I. Kruger, R. Grosu, P. Scholz und M. Broy. From MSCs to Statecharts. *Proceedings of the IFIP WG10.3/WG10.5 International Workshop on Distributed and Parallel Embedded Systems*, Seiten 61–71, 1998.

[165] A. Kumar. CUnit - A Unit Testing Framework for C. `http://cunit.sourceforge.net/` (geprüft: 25.03.2010), 2009.

[166] S. Kusumoto, K. Matsumoto, T. Kikuno und K. Torii. A New Metric for Cost Effectiveness of Software Reviews. *IEICE Transactions on Information and Systems*, 75(5):674–680, 1992.

[167] T. Kutritz. *Umfassendes Qualitätsmanagement für den Bereich Elektronik im Versuchsbau der Automobilindustrie*. Dissertation. Technische Universität Berlin, 2004.

[168] K. Lamberg. *Durchgängiges, automatisiertes Testen bei der Entwicklung von Automobilelektronik*. Beitrag zur Tagung Simulation und Test in der Funktions- und Softwareentwicklung für die Automobilelektronik, 2003.

[169] J. Laprie, A. Avizienis und H. Kopetz. *Dependability: Basic Concepts and Terminology*. Springer-Verlag, New York, Secaucus, NJ, USA, 1992.

[170] P. Lee, T. Anderson, J. Laprie, A. Avizienis und H. Kopetz. *Fault Tolerance: Principles and Practice*. Springer-Verlag, New York, Secaucus, NJ, USA, 1990.

[171] B. Lepilleur. CppUnit - C++ port of JUnit. `http://sourceforge.net/projects/cppunit` (geprüft: 25.03.2010), 2009.

[172] N. Leveson, M. Heimdahl, H. Hildreth und J. Reese. Requirements Specification for Process-Control Systems. *IEEE Transactions on Software Engineering*, 20(9):684–707, 1994.

[173] M. Liegl. Mit Leirios erreichen wir eine ganz andere Größenordnung. *Computerzeitung*, 44(44):18, 2008.

[174] M. Liegl. Softwaremodelle liefern den Rohstoff für Programmtests. *Computerzeitung*, 44(44):18, 2008.

[175] P. Liggesmeyer. A Set of Complexity Metrics for Guiding the Software Test Process. *Software Quality Journal*, 4(4):257–273, 1995.

[176] P. Liggesmeyer. *Softwarequalität: Testen, Analysieren und Verifizieren von Software*. Spektrum Akademischer Verlag, Heidelberg, 2002.

[177] P. Liggesmeyer und D. Rombach. *Software Engineering eingebetteter Systeme: Grundlagen - Methodik - Anwendungen*. Spektrum Akademischer Verlag, Heidelberg, 2005.

[178] P. Liggesmeyer, M. Rothfelder, M. Rettelbach und T. Ackermann. Qualitätssicherung softwarebasierter technischer Systeme - Problembereiche und Lösungsansätze. *Informatik-Spektrum*, 21(5):249–258, 1998.

[179] M. Luisa, F. Mariangela und N. Pierluigi. Market Research for Requirements Analysis Using Linguistic Tools. *Requirements Engineering*, 9(1):40–56, 2004.

[180] R. Lutz. Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems. *Proceedings of the IEEE International Symposium on Requirements Engineering*, Seiten 35–46, 1993.

[181] R. Madachy. *A Software Project Dynamics Model for Process Cost, Schedule and Risk Assessment*. Dissertation. University of Southern California, 1994.

[182] G. Magin, J. Quade und I. Farber. *Software-Engineering in der Ausbildung am Beispiel eines Projektkurses*. Technische Universität München, 1994.

[183] P. Mandl-Striegnitz, A. Drappa und H. Lichter. Simulating Software Projects - An Approach for Teaching Project Management. *Proceedings of the INSPIRE III: Process Improvement Through Training and Education*, Seiten 87–98, 1998.

[184] W. Masing. Fehlleistungsaufwand. *Qualität und Zuverlässigkeit*, 12:11, 1988.

[185] T. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, 1976.

[186] S. McConnell. *Code Complete*. Microsoft Press, Redmond, WA, USA, 2004.

[187] Method Park Software AG. *Ein Netzwerk umfassender Kompetenzen*. Method Park Software AG, 2007.

[188] C. Michael, R. Jones, R. Res und V. Sterling. On the Uniformity of Error Propagation in Software. *Proceedings of the 12th Annual Conference on Computer Assurance*, Seiten 68–76, 1997.

[189] Microsoft Corporation. Office Visio 2007. `http://office.microsoft.com/de-de/visio/FX100487861031.aspx` (geprüft: 25.03.2010), 2007.

[190] K. Moll, M. Broy, M. Pizka, T. Seifert, K. Bergner und A. Rausch. Erfolgreiches Management von Softwareprojekten. *Informatik-Spektrum*, 27(5):419–432, 2004.

[191] K.-H. Möller. *Ausgangsdaten für Qualitätsmetriken - Eine Fundgrube für Analysen. In C. Ebert und R. Dumke (Hrsg.). Software-Metriken in der Praxis* Springer Verlag, Berlin.

[192] K.-H. Möller. Metrikeinsatz in der Softwareentwicklung. *HMD-Theorie und Praxis der WIN 163*, 92:17–30, 2002.

[193] K.-H. Möller und D. Paulish. *Software Metrics: A Practitioners Guide to Improved Product Development*. Chapman & Hall Verlag, London, UK, 1993.

[194] S. Montenegro. *Sichere und fehlertolerante Steuerungen*. Carl Hanser Verlag, München, 1999.

[195] S. Morasca und S. Serra-Capizzano. On the Analytical Comparison of Testing Techniques. *ACM SIGSOFT Software Engineering Notes*, 29(4):154–164, 2004.

[196] Mozilla Foundation. Bugzilla. `http://www.bugzilla.org` (geprüft: 25.03.2010).

[197] U. Müller, T. Wiegmann und O. Avci. *State of the Practice der Prüf- und Testprozesse in der Softwareentwicklung. Ergebnisse einer empirischen Untersuchung bei deutschen Softwareunternehmen*. Technischer Bericht. Universität Köln, 1998.

[198] L. Murray, D. Carrington, I. MacColl, J. McDonald und P. Strooper. *Formal Derivation of Finite State Machines for Class Testing*, In Lecture Notes in Computer Science, Seiten 42–59. Springer Verlag, Berlin, 1998.

[199] G. Myers. *The Art of Software Testing*. Wiley-Interscience Verlag, Malden, MA, USA, 1979.

[200] T. Nakajo und H. Kume. A Case History Analysis of Software Error Cause-Effect Relationships. *IEEE Transactions on Software Engineering*, 17(8):830–838, 1991.

[201] T. Neustupny, J. Hiemer und O. Fricke. *Validation of a Requirements Specification*. Fraunhofer FIRST, Berlin, 1997.

[202] OMG. Unified Modeling Language (UML), version 2.1.2. `http://www.omg.org/technology/documents/formal/uml.htm` (geprüft: 05.09.2008) (geprüft: 25.03.2010), 2008.

[203] R. Otterbach. Effiziente Funktions- und Software-Entwicklung für mechatronische Systeme. *VDI Berichte*, 1842:119–126, 2004.

[204] J. Overbeck. Objektorientierter Integrationstest und Wiederverwendbarkeit. *5. Treffen des Arbeitskreises Testen, Analysieren und Verifizieren von Software, Gesellschaft für Informatik*, 1994.

[205] A. Mas Y Parareda. BMW: IT-Client auf Rädern. *IT-Szene München*, 01:1, 2007.

[206] W. Perry. *Effective Methods for Software Testing*. Wiley-QED Publishing, Somerset, NJ, USA, 1995.

[207] C. Pfaller. Requirements-Based Test Case Specification by Using Information from Model Construction. *Proceedings of the 3rd International Workshop on Automation of Software Test*, Seiten 7–16, 2008.

[208] C. Pfaller und M. Pister. *Combining Structural and Functional Test Case Generation, In Lecture Notes in Computer Science*, Seiten 229–241. Springer Verlag, Berlin.

[209] C. Pfaller, S. Wagner, J. Gericke und M. Wiemann. Multi-Dimensional Measures for Test Case Quality. *Proceedings of the 1st Software Testing Benchmark Workshop (TESTBENCH'08)*, 2008.

[210] J. Philipps, A. Pretschner, O. Slotosch, E. Aiglstorfer, S. Kriebel und K. Scholl. *Model-Based Test Case Generation for Smart Cards*, In Electronic Notes in Theoretical Computer Science, Volume 80, Seiten 170–184. Elsevier, London, 2003.

[211] G. Pinter und I. Majzik. Program Code Generation based on UML Statechart Models. *Periodica Polytechnica, Electrical Engineering*, 47(3):187–204, 2003.

[212] M. Pol, T. Koomen und A. Spillner. *Management und Optimierung des Testprozesses: Ein praktischer Leitfaden für erfolgreiches Testen von Software mit TPI und TMAP*. dpunkt Verlag, Heidelberg, 2000.

[213] G. Pomberger und G. Blaschek. *Software Engineering: Prototyping und objektorientierte Software-Entwicklung*. Carl Hanser Verlag, München, 1996.

[214] A. Pretschner. *Zum modellbasierten funktionalen Test reaktiver Systeme*. Dissertation. Technische Universtität München, 2003.

[215] A. Pretschner. Zur Kosteneffektivität des modellbasierten Testens. *Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme*, 2:85–94, 2006.

[216] A. Pretschner, M. Broy, I. Kruger und T. Stauner. Software Engineering for Automotive Systems: A Roadmap. *Proceedings of the 2007 Future of Software Engineering (FOSE'07)*, Seiten 55–71, 2007.

[217] A. Pretschner und J. Philipps. *Methodological Issues in Model-Based Testing*, In Lecture Notes in Computer Science, Seiten 281–291. Springer Verlag, Berlin, 2005.

[218] A. Pretschner, W. Prenninger, S. Wagner, C. Kühnel, M. Baumgartner, B. Sostawa, R. Zölch und T. Stauner. One Evaluation of Model-Based Testing and its Automation. *Proceedings of the 27th International Conference on Software Engineering*, 15(21):392–401, 2005.

[219] A. Rausch, M. Broy, K. Bergner, R. Höhn und S. Höppner. *Das V-Modell XT: Grundlagen, Methodik und Anwendungen*. Springer-Verlag, Heidelberg, 2007.

[220] A. Rausch und S. Höppner. *V-Modell XT eine Einführung in Softwarequalitätsmanagement*. Logos-Verlag, Berlin, 2005.

[221] M. Recknagel und C. Rupp. Messbare Qualität in Anforderungsdokumenten. *Vertikal*, 2:12, 2006.

[222] B. Reddy. *Functional Analysis and Boundary-Value Problems: An Introductory Treatment*. John Wiley & Sons, New York, NY, USA, 1987.

[223] H. Remus. Integrated Software Validation in the View of Inspections/Reviews. *Proceedings of a Symposium on Software Validation: Inspection - Testing - Verification - Alternatives*, Seiten 57–64, 1984.

[224] A. Reuys, E. Kamsties, K. Pohl und S. Reis. Szenariobasierter Systemtest von Software-Produktfamilien. *Informatik-Forschung und Entwicklung*, 20(1):33–44, 2005.

[225] D. Rico. *ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers*. J. Ross Publishing, Fort Lauderdale, FL, USA, 2004.

[226] R. Rubey, L. Browning und A. Roberts. Cost Effectiveness of Software Quality Assurance. *Proceedings of the IEEE 1989 National Aerospace and Electronics Conference*, Seiten 1614–1620, 1989.

[227] H. Rubin. *Worldwide Benchmark Project Report*. Rubin Systems, Mettmenstetten, CH, 1995.

[228] R. Rust, A. Zahorik und T. Keiningham. *Return on Quality: Measuring the Financial Impact of Your Company's Quest for Quality*. Heinemann Asia, Kuala Lumpur, MAL, 1994.

[229] B. Schätz, A. Fleischmann, E. Geisberger und M. Pister. Modellbasierte Anforderungsentwicklung mit AutoRAID. *Proceeding of the Object-Oriented Software-Engineering (OOSE). Net.ObjectDays-Conference, 2005.*

[230] J. Schäuffele und T. Zurawka. Automotive Software Engineering. *Vieweg Verlag, Wiesbaden, 2006.*

[231] H. Schlingloff, M. Conrad, H. Dörr und C. Sühl. Modellbasierte Steuergerätesoftwareentwicklung für den Automobilbereich. Proceedings of Automotive-Safety & Security*, 6(7):51–64, 2004.*

[232] U. Schöning. Logik für Informatiker. *Spektrum Akademischer Verlag, Heidelberg, 2000.*

[233] F. Schweiggert. Return-on-Investment bei Prozessverbesserungsmaßnahmen in der Softwareentwicklung. *Technischer Bericht. Universität Ulm SAI, 1999.*

[234] T. Schwinn. Effiziente Software-Inspektion durch ein Rahmenwerk zur antizipativen Berücksichtigung des Return-on-Investment. *Shaker Publishing Verlag, Aachen, 2003.*

[235] J. Shen und J. Abraham. An RTL Abstraction Technique for Processor Microarchitecture Validation and Test Generation. Journal of Electronic Testing*, 16(1):67–81, 2000.*

[236] T. Simpson. Product Platform Design and Customization: Status and Promise. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, *18(01):3–20, 2005.*

[237] G. Snedecor und W. Cochran. Statistical Methods. *Iowa State Press, Ames, IA, USA, 1989.*

[238] H. Sneed und A. Merey. Automated Software Quality Assurance. IEEE Transactions on Software Engineering, *Seiten 909–916, 1985.*

[239] I. Sommerville. Software Engineering. *Addison-Wesley Verlag, Harlow, UK., 2001.*

[240] Q. Song, M. Shepperd, M. Cartwright und C. Mair. Software Defect Association Mining and Defect Correction Effort Prediction. IEEE Transactions on Software Engineering, *32:69–82, 2006.*

[241] A. Spillner und T. Linz. Basiswissen Softwaretest. *dpunkt Verlag, Heidelberg, 2004.*

[242] G. Spur. Fabrikbetrieb: Handbuch der Fertigungstechnik. *Carl Hanser Verlag, München, 1994.*

[243] H. Stachowiak. Allgemeine Modelltheorie. *Springer-Verlag, Berlin, 1973.*

[244] G. Tassey. The Economic Impacts of Inadequate Infrastructure for Software Testing, Final Report. *National Institute of Standards and Technology, 2002.*

[245] Telelogic. Telelogic Synergy. `http://www.telelogic.com/products/synergy/index.cfm` *(inzwischen von IBM aufgekauft, daher Weiterleitung zu IBM. Geprüft: 25.03.2010), 2009.*

[246] G. Thaller. Software-Qualität. *Sybex-Verlag, Köln, 1990.*

[247] The Standish Group International. The Chaos Report. *Standish Group, West Yarmouth, MA, 1994.*

[248] The Standish Group International. The Chaos Report. *Standish Group, West Yarmouth, MA, 1995.*

[249] The Standish Group International. The Chaos Report. *Standish Group, West Yarmouth, MA, 1996.*

[250] The Standish Group International. The Chaos Report. *Standish Group, West Yarmouth, MA, 1998.*

[251] The Standish Group International. The Chaos Report. *Standish Group, West Yarmouth, MA, 2000.*

[252] The Standish Group International. The Chaos Report. *Standish Group, West Yarmouth, MA, 2004.*

[253] Universität Stuttgart, Abteilung Software Engineering. Metriken aus abgeschlossenen Studienprojekten. `http://www.informatik.uni-stuttgart.de/iste/se/people/hampp/stuprometrics/` *(geprüft: 25.03.2010), 2005.*

[254] Universität Ulm. Implementierung einer Türsteuerung in Java. *Technischer Bericht. Studentenprojekt an der Universität Ulm, 2001.*

[255] S. Vilkomir und J. Bowen. Reinforced Condition/Decision Coverage (RC/DC): A New Criterion for Software Testing*, In Lecture Notes in Computer Science, Seiten 291–308. Springer Verlag, Berlin, 2002.*

[256] E. Vitrián. Beitrag zur Ermittlung von Kosten und Nutzen der präventiven Qualitätsmethoden QFD und FMEA. *Dissertation. Technische Universität Berlin., 2004.*

[257] S. Wagner. Cost-Optimisation of Analytical Software Quality Assurance*. Dissertation. Technische Universität München, 2007.*

[258] E. Wallmüller. Risikomanagement für IT- und Software-Projekte. *Carl Hanser Verlag, München, 2004.*

[259] M. Wegener. Werkzeugunterstützte Testfallermittlung für den funktionalen Test mit dem Klassifikationsbaum-Editor CTE. Proceedings der GI-Fachtagung Softwaretechnik*, Seiten 95–102, 1993.*

[260] C. Weiss, R. Premraj, T. Zimmermann und A. Zeller. How Long Will It Take to Fix This Bug. Proceedings of the 4th International Workshop on Mining Software Repositories*, 2007.*

[261] M. Wendehals. Kostenorientiertes Qualitätscontrolling, Planung – Steuerung – Beurteilung*. Dissertation. Universtität Paderborn, 2000.*

[262] M. Wetzel und K. Siegwart. Empirische Untersuchung zur analytischen Qualitätssicherung in der Industrie. Softwaretechnik-Trends*, 25:6–11, 2005.*

[263] E. Weyuker und B. Jeng. Analyzing Partition Testing Strategies. IEEE Transactions on Software Engineering*, 17(7):703–711, 1991.*

[264] R. Wieringa und E. Dubois. Integrating Semi-Formal and Formal Software Specification Techniques. Information Systems*, 23(3-4):159–178, 1998.*

[265] H. Wildemann. Kosten- und Leistungsbeurteilung von Qualitätssicherungssystemen. Zeitschrift für Betriebswirtschaft*, 62(7):761–782, 1992.*

[266] A. Williams. Determination of Test Configurations for Pair-Wise Interaction Coverage. Proceedings of the 13th International Conference on the Testing of Communicating Systems (TestCom 2000)*, Seiten 59–74, 2000.*

[267] A. Williams und R. Probert. A Practical Strategy for Testing Pair-Wise Coverage of Network Interfaces. Proceedings of the 7th International Symposium on Software Reliability Engineering (ISSRE'96)*, Seite 246, 1996.*

[268] R. Willis, R. Rova, M. Scott, M. Johnson, J. Ryskowski, J. Moon, K. Shumate, und T. Winfield. Hughes Aircrafts Widespread Deployment of a Continuously Improving Software Process. *Technical Report CMU/SEI-98-TR-006. Carnegie Mellon University, 2002.*

[269] T. Yamaura. How to Design Practical Test Cases. Software, IEEE*, 15(6):30–36, 1998.*

[270] J. Yoo, T. Kim, S. Cha, J. Lee und H. Seong Son. A Formal Software Requirements Specification Method for Digital Nuclear Plant Protection Systems. The Journal of Systems & Software*, 74(1):73–83, 2005.*

[271] A. Zeller. Why Programs Fail: A Guide to Systematic Debugging. *Morgan Kaufmann Pub, San Francisco, CA, USA, 2009.*

[272] D. Ziegenbein, P. Braun, U. Freund, A. Bauer, J. Romberg und B. Schatz. AutoMoDe-Model-Based Development of Automotive Software. Proceedings of the Conference Design, Automation and Test in Europe*, Seiten 171–176, 2005.*