Optimierte Fehlerfindung im Funktionstest durch automatisierte Analyse von Testprotokollen

Jörg Gericke, Matthias Wiemann

Siemens CT PP 6
Otto-Hahn-Ring 6
81739 München
joerg.gericke@siemens.com
matthias.wiemann.ext@siemens.com

Abstract: Mit dem Ziel, den Aufwand für die Fehlerbeseitigung in der Softwareentwicklung effektiv zu verringern, wird für den Funktionstest eine Testmethodik vorgestellt, die nach Automatisierung der Testfallgenerierung und Testfallreduktion mit ausgewählten und bewährten Verfahren ein neu entwickeltes Diagnoseverfahren anschließt. Dieses ermittelt aus den bis dahin vorliegenden Testergebnissen Kenngrößen, die es ermöglichen, den weiteren Testaufwand effektiv zu reduzieren, indem die benötigte Zeit zum Lokalisieren von beobachteten Fehlern verkürzt wird. Dabei interpretiert das automatisierbare Diagnoseverfahren Testausführungsprotokolle und identifiziert fehleranfällige Äquivalenzklassen. Die Ergebnisse sind zudem mit semantischen Informationen angereichert, was die Interpretation durch den Programmierer vereinfacht.

1 Einleitung

In der Softwareentwicklung für eingebettete Systeme entfallen bis zu 50% (±20%) [Pr03, Be83] der Entwicklungskosten auf die Fehlerbeseitigung, wobei der Begriff Fehlerbeseitigung das Erstellen und Ausführen von Testfällen sowie das Lokalisieren und Beheben der Fehlerursachen umfasst. In der Praxis ist das vollständige Testen zeit- und kostenintensiv, weshalb Testverfahren mit dem Ziel der Zeit- und Kostenoptimierung entwickelt wurden und werden. Der Aufwand zum Erzeugen von Testsuiten kann durch automatisierbare Verfahren drastisch verringert werden. Hierzu zählen beispielsweise kombinatorische Ansätze wie Orthogonal Arrays (OA) [WP96, Ma85], Combinatorial Auction Test Suite (CATS) [Sh94], Automatic Efficient Test Generator (AETG) [CDP96, BJE94], Covering Arrays (CA) [Wi00] und Base Choice (BC) [AO94]. Alle genannten Verfahren haben neben der Automatisierbarkeit den Vorteil, dass eine im Vergleich zum vollständigen Test reduzierte Testfallmenge erzeugt wird, die trotz geringer Testzahlen gute Ergebnisse im Hinblick auf das Identifizieren von Fehlern aufweist [CDP96, GOA04, BPP92].

Das Lokalisieren der gefundenen Fehler in Funktionstests, zu denen keine strukturellen Informationen vorliegen, ist derzeit recht zeitaufwendig. Es stellt eine herausfordernde

Aufgabe dar, die zahlreichen Testfälle zu untersuchen, um die Fehler verursachenden Parameterwerte oder Fehler verursachenden Wertekombinationen verschiedener Parameter zu identifizieren. Das im Folgenden beschriebene, neue und automatisierbare Diagnoseverfahren für den Funktionstest identifiziert die kritischen, Fehler verursachenden Parameterbereiche und führt damit zur Kostenreduzierung. Weiterhin lassen sich die Quellen beobachteter Fehler schneller lokalisieren, da Testfälle, die zu fehlerhaften Ergebnissen führen, automatisch mit semantischen Informationen angereichert werden. Dies minimiert den Zeitaufwand für das Lokalisieren der Fehlerquellen und somit auch die Kosten.

2 Automatisierbares Diagnoseverfahren für den Funktionstest

Das neue Diagnoseverfahren wird in eine Testmethodik eingebettet, bei der sich die einzelnen Schritte optimal ergänzen. Zunächst werden mit bewährten Verfahren die zu testenden Parameterwerte bzw. deren Äquivalenzklassen strukturiert identifiziert (s. Kapitel 2.1), dann Testfälle automatisiert generiert und reduziert (s. Kapitel 2.2). Anschließend werden die resultierenden Testfallausführungen mit dem neuen Diagnoseverfahren interpretiert (s. Kapitel 2.3), um die kritischen, fehlerhafte Ausführungen produzierenden Äquivalenzklassen schnell zu ausfindig zu machen. Die vorgestellte Testmethodik baut somit optimal auf bewährte Methodiken auf und bietet weiterführende Unterstützung an.

2.1 Äquivalenzklassenbildung zur Testfallreduktion

Speziell im Funktionstest lassen Parameter mit vielen Parameterwerten die Testfallanzahl des vollständigen Tests schnell unhandhabbar anwachsen. Im ersten Schritt der Testmethodik wird deshalb eine Reduktion der Anzahl der Testfälle durch die Klassifizierung von Parameterwerten - eine anerkannte und vielfach eingesetzte Verfahrensweise [GOA04] - erreicht bzw. eine weitere, spätere Reduktion vorbereitet. Hierbei werden die Ausprägungen der Parameter in Äquivalenzklassen unterteilt, wobei angenommen wird, dass alle Werte einer Äquivalenzklasse in der Testausführung ein gleiches Verhalten produzieren. Folglich muss pro Äquivalenzklasse nur ein exemplarischer Wert getestet werden. Somit führen Klassifikationen durch Identifikation von Äquivalenzklassen zur einer erleichterten Reduktion der großen Zahl möglicher Parameterwerte auf wenige „interessante“[1] Werte. Dies verringert gleichzeitig drastisch die Anzahl der möglichen Kombinationen der Werte der Parametern, und somit die Anzahl der kombinatorisch möglichen Testfälle [GW06].

Ein bewährtes und weit verbreitetes Verfahren zum Klassifizieren von Parameterwerten ist die Klassifikationsbaummethode (Classification Tree Method, CTM) [GG93]. Durch Anwendung der CTM werden häufig übersehene, interessante Parameterwerte identifi-

[1] Obwohl der Begriff „interessant“ ungenau erscheint, wird er in mehreren Publikationen verwendet als Attribut für Werte, die ein Qualitätssicherer bzw. Tester für testenswert erachtet.

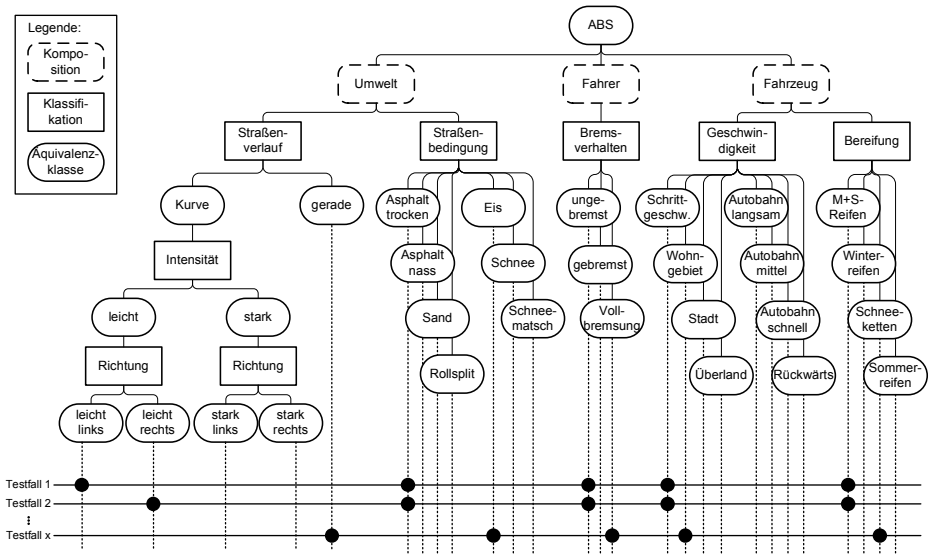

Abbildung 1. Klassifikationsbaum

ziert, sowie Abhängigkeiten zwischen Parametern erkannt [GG93]. Dieses manuelle Verfahren hinterlegt Äquivalenzklassen und Klassifikationen formalisiert, so dass anschließende Testschritte die erstellten Klassifikationsbäume automatisiert auslesen und verwenden können.

Zur Veranschaulichung der Testmethodik wird ein durchgehend verwendetes Beispiel eingeführt: Getestet werden soll eine Steuerung für ein Anti-Blockier-System (ABS) eines Kraftfahrzeuges. Die Funktion besitzt die fünf Parameter „Straßenverlauf“, „Straßenbedingung“, „Bremsverhalten“, „Geschwindigkeit“ und „Bereifung“ mit jeweils unterschiedlichen Werteausprägungen. Mit Hilfe der Klassifikationsbaummethode werden Äquivalenzklassen identifiziert (s. Abbildung 1): {gerade, leicht links, stark links, leicht rechts, stark rechts} für „Straßenverlauf“, {Asphalt trocken, Asphalt nass, Sand, Rollsplitt, Eis, Schnee, Schneematsch} für „Straßenbedingung“, {ungebremst, gebremst, Vollbremsung} für „Bremsverhalten“, {Schrittgeschwindigkeit, Wohngebiet, Stadt, Überland, Autobahn langsam, Autobahn mittel, Autobahn schnell, Rückwärts} für „Geschwindigkeit“ und {Sommerreifen, M+S-Reifen, Winterreifen, Schneeketten} für „Bereifung“. Aus den Äquivalenzklassen werden willkürlich „interessante“ Wertebelegungen der Parameter für den Test bestimmt, beispielsweise könnte für den Parameter „Geschwindigkeit“ aus der Äquivalenzklasse „Wohngebiet“ der konkrete Wert „30 km/h“ gewählt werden.

2.2 Automatisierte Testfallgenerierung und -reduktion

Auf der Basis eines formalen Klassifikationsbaumes lassen sich Testfälle automatisiert generieren. Jedoch ist es bereits in einfachen Beispielen unmöglich, alle Kombinationen

der Äquivalenzklassen aller Parameter (N-wise coverage) zu testen [GW06], da die Testfallanzahl exponentiell zur Parameteranzahl wächst. Zusätzlich zur Beschränkung auf „interessante“ Äquivalenzklassen muss die Testfallanzahl weiter systematisch reduziert werden.

Mit Hilfe des Robust Testings (Orthogonal Arrays) [WP96, MM94] kann die Testfallanzahl deterministisch reduziert werden, wobei eine paarweise Überdeckung (*pair-wise coverage*) der Äquivalenzklassen sichergestellt wird. Robust Testing reduziert das exponentielle Wachstum der Testanzahl in Bezug zur Parameteranzahl auf ein quadratisches Wachstum. Eventuelle Abhängigkeiten zwischen Parametern und deren Äquivalenzklassen müssen entsprechend [GW06] und [WP96] aufgelöst werden.

Robust Testing basiert auf der empirisch nachgewiesenen [KWG06, WK01] Annahme, dass die meisten Fehler durch einen Parameterwert alleine oder durch das Zusammenspiel von Werten jeweils zweier Parameter produziert werden. Robust Testing garantiert durch die paarweise Überdeckung das Finden sämtlicher Fehler der beiden involvierten Äquivalenzklassen. Fehler, die aus dem Zusammenspiel von drei oder mehr Parametern entstehen, treten weniger häufig auf und werden mit dem Verfahren nicht zwingend gefunden. Untersuchungen [TL02, BPP92, DJK99] haben ergeben, dass Testfälle einer paarweisen Überdeckung zwischen 50% und 70% aller Fehler finden. Im medizinischen Umfeld liegt der Anteil sogar bei über 90% [WK01]. Untersuchungen von Cohen et al. [CDP96] belegen, dass trotz extremer Reduzierung hohe Implementationsüberdeckungen erzielt werden: 93% Block-Überdeckung, 83% Entscheidungs-Überdeckung, 76% Überdeckung nach computational use (c-use) und 73% Überdeckung nach predicate use (p-use). Nach Brownlie et al. [GOA04, BPP92] wurden im Vergleich zu „konventionellen“ Tests 22% mehr Fehler in der Hälfte der Zeit gefunden.

Auf das oben gegebene Beispiel wird der von Williams und Probert [WP96] ausführlich beschriebene Robust Testing Algorithmus unter Berücksichtigung der in [WP96] beschriebenen Optimierungen angewendet. Robust Testing garantiert unter Einbeziehung aller Parameter bereits mit 56 Testfällen eine paarweise Überdeckung. Ein vollständiger Test aus dem Produkt der Anzahlen der Werteausprägungen der Parameter „Straßenverlauf“, „Straßenbedingung“, „Bremsverhalten“, „Geschwindigkeit“ und „Bereifung“ würde 5×7×3×8×4=3360 Testfälle erzeugen. Die Testfallmenge ist somit auf 2% reduziert worden. Tabelle 1 zeigt exemplarisch die ersten sechs generierten Testfälle. Vom Testfallgenerator erzeugte „don’t care“-Werte erlauben die Benutzung beliebiger Äquivalenzklassen, da sie zur Erlangung der paarweisen Überdeckung nicht relevant sind.

Nr.	**Straßenverlauf**	**Straßenbedingung**	**Bremsverhalten**	**Geschwindigkeit**	**Bereifung**
Testfall 1	gerade	Asphalt trocken	ungebremst	Schrittgeschwindigkeit	Sommerreifen
Testfall 2	leicht links	Asphalt nass	gebremst	Schrittgeschwindigkeit	M+S-Reifen
Testfall 3	stark links	Sand	Vollbremsung	Schrittgeschwindigkeit	Winterreifen
Testfall 4	leicht rechts	Rollsplitt	*don't care*	Schrittgeschwindigkeit	Schneeketten
Testfall 5	stark rechts	Eis	*don't care*	Schrittgeschwindigkeit	*don't care*
Testfall 6	*don't care*	Schnee	*don't care*	Schrittgeschwindigkeit	*don't care*
...	...	...	...	...	...

Tabelle 1. Automatisiert generierte Testfälle des Robust Testing Algorithmus

Den Abschluss des zweiten Schrittes bildet die Durchführung der ermittelten Testfälle. Die Ergebnisse liegen anschließend im Testprotokoll mit der Bewertung der Tests vor.

2.3 Automatisierbare Diagnose der Testergebnisse

Bisher wurden im ersten Schritt Äquivalenzklassen der Parameter identifiziert. Anschließend wurden im zweiten Schritt Testfälle systematisch generiert, reduziert und ausgeführt. Im dritten Schritt wird auf das resultierende Testprotokoll, das für jeden Testfall das Ergebnis wiedergibt, das neue Diagnoseverfahren angewendet. Hierdurch können die diagnostizierten Fehler mit einem deutlich geringeren Aufwand als durch manuelle Sichtung aller Testfälle lokalisiert werden.

2.3.1 Diagnose auf Äquivalenzklassenebene

Baudry et al. [BFT06] haben ein Diagnoseverfahren mit der Berechnung von Kennzahlen zur Identifikation kritischer Quelltextzeilen vorgestellt. Baudry et al. Identifizieren anhand der Parameter bzw. Parameterkombinationen eines jeden Testfalls die ausgeführten Quelltextzeilen und ermitteln daraus für jeden Testfall Kennzahlen. Dies stellt einen erheblichen Aufwand dar, der zudem testfallabhängig, d.h. also spezifisch zu ermitteln ist. Dieses Verfahren wurde erfolgreich in Beispielen für den Test mit strukturellen Informationen angewandt.

Ziel des hier vorgestellten Diagnoseverfahrens ist es, eine automatisierbare Bewertung der Testergebnisse mit generellerer Anwendbarkeit **ohne** Wissen um strukturelle Informationen zu ermöglichen. Inspiriert durch das Verfahren von Baudry et al. werden zwar ebenfalls Kennzahlen zur Identifikation von fehleranfälligen Äquivalenzklassen der Parameter berechnet, jedoch kommt dieser Ansatz ohne den kritisierten Mehraufwand zur Identifikation ausgeführter Quelltextzeilen aus. Anstelle von Quelltextzeilen wertet das neue Diagnoseverfahren das aus der Testausführung resultierende Testprotokoll mit Hilfe von einfach automatisierbaren Algorithmen aus. Das nach der Testausführung vorliegende Testprotokoll beinhaltet die Testergebnisse (Verdict mit den Werten FAIL und PASS) jeder Testfallausführung. Für jede Äquivalenzklasse *e* werden die absoluten Anzahlen positiver (ohne Fehler) und negativer (mit Fehler) Testergebnisse ermittelt. Daraus wird die relative Beteiligung der Testergebnisse mit beobachteten Fehlern an der Gesamtanzahl der Testergebnisse mit der Beteiligung der Äquivalenzklasse *e* entsprechend Formel 1 berechnet. Dieser Wert wird hier als Kritikalität von *e* eingeführt. Ein hoher Kritikalitätswert bedeutet eine hohe Wahrscheinlichkeit für die alleinige Verantwortlichkeit der Werte der Äquivalenzklasse *e* am Misslingen von Testfallausführungen.

$$\textit{Kritikalität (e)} = \frac{\textit{Anzahl der negativen Testergebnisse mit Beteiligung der Äquivalenzklasse e}}{\textit{Anzahl aller Testergebnisse mit Beteiligung der Äquivalenzklasse e}}$$

Formel 1. Definition des Kritikalitätswertes einer Äquivalenzklasse *e*

Der Kritikalitätswert einer Äquivalenzklasse wird auf einen Rangwert aller Äquivalenzklassen eines Parameters in der Weise abgebildet, dass steigende Werte fallende Ränge ergeben. Niedrige Ränge weisen somit auf eine höhere Wahrscheinlichkeit hin, dass

diese Äquivalenzklasse eines Parameters unabhängig von Äquivalenzklassen anderer Parameter Fehler produziert. Die ermittelte Rangfolge ist nicht zwingend eindeutig, da verschiedene Äquivalenzklassen aufgrund gleicher Kennzahlen einen gleichen Rang zugewiesen bekommen.

Die Anzahl der mit einer bestimmten Äquivalenzklasse *e* durchgeführten Testfälle ist eine wichtige Kennzahl in der Beurteilung der Verlässlichkeit der Kritikalität von *e*, da Aussagen über eine Äquivalenzklasse, die nur in wenige Testfälle involviert war, weniger verlässlich ist. Ein simples Maß für die Verlässlichkeit ist die relative Anzahl der Testfälle unter Benutzung der betrachteten Äquivalenzklasse *e* zur Anzahl aller ausgeführten Testfälle (s. Formel 2).

$$\textit{Verlässlichkeit (e)} = \frac{\textit{Anzahl der Testergebnisse mit Beteiligung der Äquivalenzklasse e}}{\textit{Anzahl aller Testergebnisse}}$$

Formel 2. Definition des Verlässlichkeitswertes einer Äquivalenzklasse *e*

Äquivalenzklassen von Parametern mit wenigen Äquivalenzklassen werden also zwangsläufig intensiver getestet als Äquivalenzklassen von Parametern mit vielen Äquivalenzklassen. Allgemein bestimmt der Parameter mit der zweitgrößten Anzahl von Äquivalenzklassen die minimale Anzahl der Testfälle für jede Äquivalenzklasse, da jede Äquivalenzklasse $e_{i(k)}$ des Parameters p_k mit allen Äquivalenzklassen $e_{i(j)}$ anderer Parameter p_j (mit $k \neq j$) mindestens einmal überdeckt wird. In dem gewählten Beispiel wird also jede Äquivalenzklasse mindestens sieben Mal getestet, da der Parameter „Straßenbedingung“ mit sieben Äquivalenzklassen die zweithöchste Anzahl von Äquivalenzklassen besitzt. Die Äquivalenzklassen des Parameters „Bremsverhalten“ werden ca. $56/3 \approx 19$ Mal getestet, wenn die drei Äquivalenzklassen gleichmäßig mit Hilfe der „don’t-care“-Werte auf die ausgeführten 56 Testfälle aufgeteilt werden.

Für eine gleichmäßige Verlässlichkeit der Kritikalitäten der Äquivalenzklassen sollte jede Äquivalenzklasse $e_{i(k)}$ eines Parameters p_k annähernd gleich oft getestet werden [BFT06], was durch die Anwendung von Robust Testing als Testfallgenerator bestmöglich realisiert wird, da sämtliche Äquivalenzklassen $e_{i(k)}$ eines Parameters p_k gleich oft benutzt werden. „Don’t care“-Werte (s. Tabelle 1) sollten also so belegt werden, dass eine möglichst gleichhäufige Verwendung aller Äquivalenzklassen $e_{i(k)}$ erzielt wird.

In der Weiterführung des Beispiels werden im Zuge des Diagnoseverfahrens für jede Äquivalenzklasse der Parameter die oben beschriebenen Kennzahlen für Kritikalität und Verlässlichkeit aus dem Testprotokoll der Testfälle aus Tabelle 1 ermittelt und in Kennzahlentabellen zusammengestellt (in Tabelle 2 exemplarisch mit den Kennzahlen der Äquivalenzklasse „Straßenverlauf“). Die Kennzahlentabelle (s. Tabelle 2) beinhaltet zunächst die absoluten Anzahlen der positiven und negativen Testergebnisse. So wurden beispielsweise 10 der 11 Testfälle mit Benutzung der Äquivalenzklasse „leicht links“ erfolgreich ausgeführt, ein Testfall endete mit einem Fehler. Die Berechnung des Kritikalitätswertes ergibt, dass 9% (1/11=0,09) der durchgeführten Testfälle unter Benutzung der Äquivalenzklasse „leicht links“ nicht erfolgreich waren. Aufgrund dieses relativ geringen Kritikalitätswertes kann die Äquivalenzklasse „leicht links“ als am wenigsten kritisch eingestuft werden und erhält den in diesem Beispiel höchsten Rang 4, ebenso die Äquivalenzklasse „leicht rechts“. Die Äquivalenzklassen „gerade“, „stark rechts“ und

„stark links“ mit jeweils geringeren Rängen sind demzufolge als fehleranfälliger zu bewerten. Zur Beurteilung der Aussagekraft wird der Verlässlichkeitswert berechnet. Sämtliche Äquivalenzklassen des Parameters „Straßenverlauf“ haben aufgrund des vorangegangenen Robust Testing Algorithmus idealer Weise annähernd den gleichen Verlässlichkeitswert von 20% (11/56≈0,20).

Parameter "Straßenverlauf"						
Äquivalenz-klasse	**# PASS**	**# FAIL**	**# Gesamt**	**Kritikalität**	**Verlässlichkeit**	**Rang**
gerade	10	2	12	17%	21%	3
leicht links	10	1	11	9%	20%	4
leicht rechts	10	1	11	9%	20%	4
stark rechts	3	8	11	73%	20%	1
stark links	5	6	11	55%	20%	2
Summe	39	17			100%	
	56		von 56 Testfällen			

Tabelle 2. Beispiel einer Kennzahlentabelle für den Parameter „Straßenverlauf“

Der Kritikalitätswert kann zwischen 0 und 100 liegen. Auch wenn eine Äquivalenzklasse *e* bereits ohne Zusammenwirken mit anderen Parametern zu Fehlern führt, müssen nicht zwingend alle Testfallausführungen mit der entsprechenden Äquivalenzklasse *e* fehlschlagen, der Kritikalitätswert kann also durchaus kleiner 100% sein. Das Erzeugen von Fehlern durch die Äquivalenzklasse *e* kann durch die Kombination mit Äquivalenzklassen $e_{i(k)}$ anderer Parameter p_k „verdeckt“ werden. Beispielsweise könnte die Äquivalenzklasse „ungebremst“ des Parameters „Bremsverhalten“ bewirken, dass eine für einen bestimmten „Straßenverlauf“ an sich fehlerhafte Anweisung nicht ausgeführt wird. Analog kann ein Fehler ausschließlich durch die Kombination mehrerer Äquivalenzklassen verschiedener Parameter eintreten. Dies hat zur Folge, dass die Kennzahlentabelle Kritikalitätswerte von mehr als 0% verzeichnet.

Die Diagnosekennzahlen sagen nur etwas über die Kritikalität isoliert betrachteter Äquivalenzklassen aus. Entsteht der Fehler durch die genannten Seiteneffekte (verdecken und kombinieren) wird das nicht in den Diagnosekennzahlen der einzelnen Äquivalenzklassen sichtbar. Vielmehr wird der Kritikalitätswert einer Äquivalenzklasse sogar durch Äquivalenzklassen anderer Parameter verfälscht. Da derartige Seiteneffekte relativ selten sind - besonders im Vergleich zur Gesamtzahl der ausgeführten Testfälle - wird die Aussagequalität im Allgemeinen nur vernachlässigbar verfälscht [KWG04, WK01, GOA04]. Um Seiteneffekte trotzdem genauer zu untersuchen, werden Kennzahlentabellen für Kombinationen von Äquivalenzklassen aufgestellt. Wird die Anzahl der Testfälle, wie im vorgestellten Fall, durch paarweise Überdeckung reduziert, wird eine ausgewogene Testfallzahl für Äquivalenzklassen erzielt und somit eine gute Basis für diese weitergehende Analyse geschaffen. Die Kombinationen lassen sich hierarchisch anhand der Anzahl beteiligter Parameter ordnen. Zunächst werden für Äquivalenzklassen mit als nicht eindeutig eingestuften Kritikalitätswerten (z.B. zwischen 15% und 85%) Kennzahlentabellen für Kombinationen von Äquivalenzklassen von zwei Parametern aufgestellt. Tabelle 3 zeigt exemplarisch die Kombination der Äquivalenzklasse „Straßenverlauf gerade“ mit den Äquivalenzklassen des Parameters „Bremsverhalten“. Die Äquivalenzklassenkombination „gerade & Vollbremsung“ erzeugt in 67% der ausgeführten Testfäl-

le Fehler, wohingegen die restlichen Kombinationen fehlerfrei getestet wurden. Daraus folgt, dass die Äquivalenzklassenkombination „gerade & Vollbremsung“ als relativ kritisch einzustufen ist. Jedoch sollten aufgrund des Kritikalitätswertes von 67% auch Kombinationen von drei Parametern betrachtet werden. Wenn beispielsweise eine Kritikalität von 100% in einer Kombination mit einem dritten Parameter gefunden wird, ist dies eine wesentliche Information zur Fehlerfindung.

Deutlich zu erkennen ist der in dieser Detaillierungsstufe geringe Verlässlichkeitswert von ca. 5%, der bedeutet, dass nur 5% der ausgeführten Testfälle zur Berechnung des Kritikalitätswertes herangezogen wurden. In Tabelle 3 werden beispielsweise insgesamt lediglich 8 (14%) der 56 ausgeführten Testfällen ausgewertet. Ganz allgemein senkt die Betrachtung von Kombinationen aus drei und mehr Parametern den Verlässlichkeitswert weiter, was in dieser Situation eher für die Interpretation auf Grundlage einer groben Unterscheidung nach unkritisch (0%) und kritisch (100%) spricht.

Parameterkombination "Straßenverlauf" und "Bremsverhalten"

Äquivalenz-klasse	# PASS	# FAIL	# Gesamt	Kritikalität	Verlässlichkeit	Rang
gerade & Vollbremsung	1	2	3	67%	5%	1
gerade & gebremst	3	0	3	100%	5%	2
gerade & ungebremst	2	0	2	100%	4%	2
Summe	6	2			14%	
	8		von 56 Testfällen			

Tabelle 3. Kennzahlentabelle für Parameterkombination

Die Rangfolgen der Kennzahlentabellen müssen nicht, wie auch bereits in Tabelle 2 ersichtlich, eindeutig sein. So sind in diesem Beispiel die Äquivalenzklassen „leicht links“ und „leicht rechts“ auf den gleichen Rang eingestuft worden. Generell sollten solche Uneindeutigkeiten verhindert oder minimiert werden, um die Aussagequalität der Rangliste zu erhöhen. Baudry et. al. bezeichnen eine Menge von Tabellenzeilen mit gleichem Rang als Dynamic Basic Block (DBB) [BFT06]. Optimalerweise repräsentiert eine Zeile der Kennzahlentabelle jeweils eine eigene DBB, da somit ein Maximum von Rängen erzielt wird. Die Zahl und Größe der DBBs lässt sich durch die Wahl der Testfälle beeinflussen [BFT06]. Die Anwendung der Orthogonal Arrays zum Erreichen einer paarweisen Überdeckung erlaubt jedoch ein solches Optimieren der Testfälle nur in geringem Umfang, da eine Veränderung der Parameterwerte zum Verlust des Überdeckungskriteriums führen kann. Lediglich „don’t-care“-Werte sollten zur Optimierung der Anzahl der DBBs benutzt werden. Um die Anzahl der DBBs dennoch zu maximieren, ist ein zusätzliches Erzeugen von Testfällen durch den Übergang zu einer 3-fachen Überdeckung mit mehr generierten Testfällen möglich.

2.3.3 Unterstützende Anreicherung mit Semantik

Zu den Äquivalenzklassen können wie erwähnt aus dem Klassifikationsbaum semantische Informationen automatisiert bereitgestellt werden. Nachdem die kritischen Äquivalenzklassen identifiziert sind, ist es sinnvoll, die vorliegenden semantischen Informationen zur Lokalisierung der Fehler heranzuziehen.

Während der Implementation hat der Programmierer ein mentales Modell der zu imple-

mentierenden Funktion entwickelt [Pr03]. Kann der Programmierer fehlerhafte Stellen im mentalen Modell lokalisieren, kann er auch fehlerhafte Stellen im Quelltext schneller identifizieren. Dies wird durch Angabe von semantischen Informationen in Form von die Fehlerquelle klassifizierenden Schlagworten unterstützt. Die notwendigen semantischen Informationen können automatisch aus dem im ersten Schritt der Testmethodik beschriebenen Klassifikationsbaum generiert werden. Dazu muss der Klassifikationsbaum geparst werden, indem die Äquivalenzklassen und Klassifikationen betrachtet werden, die auf den Wegen von den Blättern – den Äquivalenzklassen der benutzten Parameterwerten – zur Wurzel des Klassifikationsbaumes liegen. Beispielsweise lässt sich aus Tabelle 3 folgende semantische Information ermittelt werden: Beim „Bremsverhalten“ „Vollbremsung“ und „Straßenverlauf“ „gerade“ treten in zwei Drittel der Testfälle Fehler auf. Eine Abhängigkeit zu mindestens einem der Faktoren „Straßenbedingung“, „Geschwindigkeit“ oder „Bereifung“ ist anzunehmen und sollte (sofern möglich) weiter untersucht werden. Dahingegen erscheint ein „gerader“ „Straßenverlauf“ beim „Bremsverhalten“ „gebremst“ oder „ungebremst“ unkritisch zu sein. Auch durch andere Einflussfaktoren wie „Straßenbedingung“, „Geschwindigkeit“ oder „Bereifung“ scheinen unter diesen Bedingungen keinen Fehler hervorzurufen.

3 Zusammenfassung und Ausblick

Der Kosten- und Zeitaufwand des Funktionstests einer softwaregestützten Entwicklung ist effektiv zu reduzieren. Hierfür existieren bereits bewährte Verfahren zur automatisierten Testfallgenerierung, Reduktion der Testfälle und Ansätze zum Lokalisieren beobachteter Fehler auf Codeebene. Die hier vorgestellte Testmethodik nutzt die erprobten Komponenten in allen drei Bereichen als Basis und schließt für den Funktionstest ein Diagnoseverfahren zur weiteren Verfahrensoptimierung an, das nicht auf strukturelle Informationen angewiesen ist, bzw. nicht darauf basiert. Der Fokus liegt auf der neuartigen automatisierbaren Methode zur Lokalisierung der Fehler im Funktionstest auf der Basis von Testausführungsprotokollen. Anhand der Protokolle der ausgeführten Testfälle werden Kennzahlen berechnet, mit denen besonders kritische Äquivalenzklassen identifiziert werden. Die Äquivalenzklassen werden mit einfachen semantischen Informationen angereichert, so dass Programmierer beobachtete Fehler schneller eingrenzen können. Umfangreichere und eine optimierte Zusammenstellung der semantischen Informationen wäre zukünftig wünschenswert und ist ein Gebiet der aktuellen Weiterentwicklung.

Die Aussagekraft der im Diagnoseverfahren berechneten Kennzahlen ist gekoppelt an die zugrunde liegende Datenbasis, die mit steigender Zahl der ausgeführten Testfälle an Solidität gewinnt. Somit ist das Verfahren erst zu einem späten Zeitpunkt des Testens sinnvoll nutzbar. Wird die Testfallausführung jedoch aufgrund von beobachteten Fehlern vorzeitig abgebrochen, so können berechnete Kennzahlen als wenig gesichert gelten. Deshalb empfiehlt es sich, die Diagnose erst nach Ausführung aller Testfälle durchzuführen.

Das beschriebene Verfahren wurde in eine industrielle generische automatisierte Testfallumgebung (engl.: Generic Automated Test-Environment, GATE) eingebunden und in

Projekten erfolgreich eingesetzt. Im Systemtest eingebetteter Systeme wurden durch Anwendung der hier beschriebenen Teststrategie mindestens zwei Entwicklungszyklen eingespart. Weiterhin hat sich während der Projektarbeit herausgestellt, dass semantische Informationen schon aus der Anforderungsphase automatisiert übernommen und in das Diagnoseverfahren eingeflochten werden könnten. Hierzu wird als nächster Schritt ein semantischer Anforderungsparser entwickelt und in das System integriert.

Literaturverweise

[AO94] P.E. Ammann und A.J. Offutt. Using formal methods to derive test frames in category-partition testing. In Proceedings of the 9th COMPASS'94. Gaithersburg MD, pages 69-80. IEEE Computer Society Press. June 1994.

[Be83] B. Beizer. Software Testing Techniques. Van Nostrand Reinhold. New York, NY. 1983.

[BFT06] B. Baudry, F. Fleurey und Y. Le Traon. Improving Test Suites for Efficient Fault Localization. 2006.

[BJE94] K. Burroughs, A. Jain und R.L. Erickson. Improved quality of protocol testing through techniques of experimental design. In Proceedings of Supercomm/ICC'94. May 1-5. New Orleans, Louisiana, USA. pages 745-752. IEEE. May 1994.

[BPP92] R. Brownlie, J. Prowse und M.S. Phadke. Robust Testing of AT&T PMX/StarMAIL using OATS. AT&T Technical Journal, pages 41-47. May/June 1992.

[CDP96] D.M. Cohen, S.R. Dalal, J. Parelius und G.C. Patton. The Combinatorial Design Approach to Automatic Test Generation. IEEE Software. pages 83-88. September 1996.

[DJK99] S.R. Dalal, A. Jain, N. Karunanithi, J.M. Leaton, C.M. Lott, G.C. Patton und B.M. Horowitz. Model-Based Testing in Practice. In Proceedings of ICSE'99. 1999.

[GG93] M. Grochtmann, K. Grimm. Classification trees for partition testing. In Software Testing Verification & Reliability. Wiley. Vol. 3, no. 2, pp. 63-82. 1993

[GOA04] M. Grindal, J. Offutt und S. F. Andler. Combination Testing Strategies: A Survey. 2004.

[GW06] J. Gericke und M. Wiemann. A Method for Generating a Minimal Functional Set of Test-Cases for Software-Intensive Systems. In Proceedings of SERP06: 106-112. 2006.

[KW04] D.R. Kuhn, D.R. Wallace und A.M. Gallo. Software Fault Interactions and Implications for Software Testing. IEEE TSE. 30(40):1-4 (2004). 2004.

[Ma85] R. Mandl. Orthogonal Latin Squares: An application of experiment design to compiler testing. Communications of the ACM, 28(10):1054-1058. October 1985.

[MM94] R. McDaniel und J. D. McGregor. Testing the Polymorphic Interactions Between Classes. TR94-103. Clemson University. 1994.

[Pr03] A. Pretschner. Zum modellbasierten funktionalen Test reaktiver Systeme. Dissertation. TU München. 2003.

[Sh94] G. Sherwood. Effective testing of factor combinations. In Proceedings of STAR94. Washington DC. Software Quality Engineering. May 1994.

[TL02] K.C. Tai und Y. Lei. A Test Generation Strategy for Pairwise Testing. IEEE Trans. Software Eng. vol. 28, no. 1, 109-11 (January 2002). 2002.

[Wi00] A.W. Williams. Determination of test configurations for pair-wise interaction coverage. In Proceedings of TestCom 2000. Ottawa, Canada, August 2000. pages 59-74, 2000.

[WK01] D.R. Wallace und D.R. Kuhn. Failure Modes in Medical Device Software: an Analysis of 15 Years of Recall Data. IJRQSE, Vol. 8. No. 4. 2001.

[WP96] A.W. Williams und R.L. Probert. A practical strategy for testing pair-wise coverage of network interfaces. In Proceedings of ISSRE96. White Plains, New York, USA, Oct 30 - Nov 2. 1996. Nov 1996.